

---

# **OpenCOBOL FAQ Documentation**

*Release 1.1.11*

**Brian Tiffin**

February 25, 2013



# CONTENTS

<b>1</b>	<b>1.1</b>	<b>What is OpenCOBOL?</b>	<b>5</b>
<b>2</b>	<b>1.2</b>	<b>What is COBOL?</b>	<b>7</b>
<b>3</b>	<b>1.3</b>	<b>How is OpenCOBOL licensed?</b>	<b>9</b>
<b>4</b>	<b>1.4</b>	<b>What platforms are supported by OpenCOBOL?</b>	<b>11</b>
<b>5</b>	<b>1.5</b>	<b>Are there pre-built OpenCOBOL packages</b>	<b>13</b>
	5.1	1.5.1 kiska.net repository . . . . .	13
	5.2	1.5.2 sourceforge . . . . .	13
<b>6</b>	<b>1.6</b>	<b>What is the most recent version of OpenCOBOL?</b>	<b>15</b>
<b>7</b>	<b>1.7</b>	<b>How complete is OpenCOBOL?</b>	<b>17</b>
<b>8</b>	<b>1.8</b>	<b>Will I be amazed by OpenCOBOL?</b>	<b>19</b>
<b>9</b>	<b>1.9</b>	<b>Who do I thank for OpenCOBOL?</b>	<b>21</b>
<b>10</b>	<b>1.10</b>	<b>Does OpenCOBOL include a Test Suite?</b>	<b>23</b>
<b>11</b>	<b>1.11</b>	<b>Does OpenCOBOL pass the NIST Test Suite?</b>	<b>25</b>
<b>12</b>	<b>1.12</b>	<b>What about OpenCOBOL and benchmarks?</b>	<b>27</b>
	12.1	1.12.1 telco billing . . . . .	27
<b>13</b>	<b>1.13</b>	<b>Can OpenCOBOL be used for CGI?</b>	<b>29</b>
<b>14</b>	<b>1.14</b>	<b>Does OpenCOBOL support a GUI?</b>	<b>31</b>
	14.1	1.14.1 GTK . . . . .	31
	14.2	1.14.2 Tcl/Tk . . . . .	31
	14.3	1.14.3 Vala, WebKit . . . . .	31
<b>15</b>	<b>1.15</b>	<b>Does OpenCOBOL have an IDE?</b>	<b>33</b>
<b>16</b>	<b>1.16</b>	<b>Can OpenCOBOL be used for production applications?</b>	<b>35</b>
	16.1	1.16.1 Nagasaki Prefecture . . . . .	35
	16.2	1.16.2 Stories from Currey Adkins . . . . .	35
	16.3	1.16.3 Public Accounting . . . . .	36

<b>17</b>	<b>1.17</b>	<b>Where can I get more information about COBOL?</b>	<b>39</b>
<b>18</b>	<b>1.18</b>	<b>Where can I get more information about OpenCOBOL?</b>	<b>41</b>
	18.1	1.18.1 The OpenCOBOL Programmer's Guide . . . . .	41
<b>19</b>	<b>1.19</b>	<b>Can I help out with the OpenCOBOL project?</b>	<b>43</b>
	19.1	1.19.1 Translation Efforts . . . . .	43
<b>20</b>	<b>1.20</b>	<b>Is there an OpenCOBOL mailing list?</b>	<b>45</b>
<b>21</b>	<b>1.21</b>	<b>Where can I find more information about COBOL standards?</b>	<b>47</b>
<b>22</b>	<b>1.22</b>	<b>Can I see the OpenCOBOL source codes?</b>	<b>49</b>
	22.1	1.22.1 What was used to color the source code listings? . . . . .	49
<b>23</b>	<b>1.23</b>	<b>What happened to opencobol.org?</b>	<b>51</b>
<b>24</b>	<b>1.24</b>	<b>What is COBOL in Latin?</b>	<b>53</b>
<b>25</b>	<b>1.25</b>	<b>Do you know any good jokes?</b>	<b>55</b>
	25.1	1.25.1 A 5-7-5 haiku? . . . . .	56
<b>26</b>	<b>2</b>	<b>History</b>	<b>57</b>
	26.1	2.1 What is the history of COBOL? . . . . .	57
	26.2	2.2 What are the Official COBOL Standards? . . . . .	58
	26.3	2.3 What is the development history of OpenCOBOL? . . . . .	58
	26.4	2.4 What is the current version of OpenCOBOL? . . . . .	59
<b>27</b>	<b>3</b>	<b>Using OpenCOBOL</b>	<b>61</b>
	27.1	3.1 How do I install OpenCOBOL? . . . . .	61
	27.2	3.2 What are the configure options available for building OpenCOBOL? . . . . .	64
	27.3	3.3 Does OpenCOBOL have any other dependencies? . . . . .	66
	27.4	3.4 How does the OpenCOBOL compiler work? . . . . .	66
	27.5	3.5 What is cobc? . . . . .	74
	27.6	3.6 What is cobcrun? . . . . .	74
	27.7	3.7 What is cob-config? . . . . .	75
	27.8	3.8 What compiler options are supported? . . . . .	75
	27.9	3.9 What dialects are supported by OpenCOBOL? . . . . .	77
	27.10	3.10 What extensions are used if cobc is called with/without "-ext" for COPY . . . . .	77
	27.11	3.11 What are the OpenCOBOL compile time configuration files? . . . . .	77
	27.12	3.12 Does OpenCOBOL work with make? . . . . .	80
	27.13	3.13 Do you have a reasonable source code skeleton for OpenCOBOL? . . . . .	81
	27.14	3.14 Can OpenCOBOL be used to write command line stdin, stdout filters? . . . . .	84
	27.15	3.15 How do you print to printers with OpenCOBOL? . . . . .	86
	27.16	3.16 Can I run background processes using OpenCOBOL? . . . . .	94
	27.17	3.17 Is there OpenCOBOL API documentation? . . . . .	94
	27.18	3.18 How do I use LD_RUN_PATH with OpenCOBOL? . . . . .	95
	27.19	3.19 What GNU build tool options are available when building OpenCOBOL? . . . . .	95
<b>28</b>	<b>4</b>	<b>Reserved Words</b>	<b>99</b>
	28.1	4.1 What are the OpenCOBOL RESERVED WORDS? . . . . .	99
	28.2	4.2 Does OpenCOBOL implement any Intrinsic FUNCTIONS? . . . . .	217
	28.3	4.3 Can you clarify the use of FUNCTION in OpenCOBOL? . . . . .	247
	28.4	4.4 What is the difference between the LENGTH verb and FUNCTION LENGTH? . . . . .	247
	28.5	4.5 What STOCK CALL LIBRARY does OpenCOBOL offer? . . . . .	247
	28.6	4.6 What are the XF4, XF5, and X91 routines? . . . . .	261

28.7	4.7	What is CBL_OC_NANOSLEEP OpenCOBOL library routine?	262
28.8	4.8	How do you use C\$JUSTIFY?	262
28.9	4.9	What compiler directives are supported by OpenCOBOL?	264
<b>29</b>	<b>5</b>	<b>Features and extensions</b>	<b>267</b>
29.1	5.1	How do I use OpenCOBOL for CGI?	269
29.2	5.2	What is ocdoc?	273
29.3	5.3	What is CBL_OC_DUMP?	283
29.4	5.4	Does OpenCOBOL support any SQL databases?	293
29.5	5.5	Does OpenCOBOL support ISAM?	296
29.6	5.6	Does OpenCOBOL support modules?	302
29.7	5.7	What is COB_PRE_LOAD?	302
29.8	5.8	What is the OpenCOBOL LINKAGE SECTION for?	302
29.9	5.9	What does the -fstatic-linkage OpenCOBOL compiler option do?	303
29.10	5.10	Does OpenCOBOL support Message Queues?	303
29.11	5.11	Can OpenCOBOL interface with Lua?	313
29.12	5.12	Can OpenCOBOL use ECMAScript?	321
29.13	5.13	Can OpenCOBOL use JavaScript?	321
29.14	5.14	Can OpenCOBOL interface with Scheme?	328
29.15	5.15	Can OpenCOBOL interface with Tcl/Tk?	331
29.16	5.16	Can OpenCOBOL interface with Falcon PL?	344
29.17	5.17	Can OpenCOBOL interface with Ada?	344
29.18	5.18	Can OpenCOBOL interface with Vala?	346
29.19	5.19	Can OpenCOBOL interface with S-Lang?	354
29.20	5.20	Can the GNAT Programming Studio be used with OpenCOBOL?	358
29.21	5.21	Does OpenCOBOL support SCREEN SECTION?	364
29.22	5.22	What are the OpenCOBOL SCREEN SECTION colour values?	365
29.23	5.23	Does OpenCOBOL support CRT STATUS?	365
29.24	5.24	What is CobCurses?	366
29.25	5.25	What is CobXRef?	367
29.26	5.26	Does OpenCOBOL implement Report Writer?	373
29.27	5.27	Does OpenCOBOL implement LINAGE?	373
29.28	5.28	Can I use ctags with OpenCOBOL?	373
29.29	5.29	What about debugging OpenCOBOL programs?	374
29.30	5.30	Is there a C interface to OpenCOBOL?	376
29.31	5.31	What are some idioms for dealing with C char * data from OpenCOBOL?	377
29.32	5.32	Does OpenCOBOL support COPY includes?	378
29.33	5.33	Does OpenCOBOL support WHEN-COMPILED?	378
29.34	5.34	What is PI in OpenCOBOL?	378
29.35	5.35	Does OpenCOBOL support the Object features of the 2002 standard?	378
29.36	5.36	Does OpenCOBOL implement PICTURE 78?	379
29.37	5.37	Does OpenCOBOL implement CONSTANT?	379
29.38	5.38	What source formats are accepted by OpenCOBOL?	379
29.39	5.39	Does OpenCOBOL support continuation lines?	379
29.40	5.40	Does OpenCOBOL support string concatenation?	380
29.41	5.41	Does OpenCOBOL support D indicator debug lines?	381
29.42	5.42	Does OpenCOBOL support mixed case source code?	381
29.43	5.43	What is the shortest OpenCOBOL program?	382
29.44	5.44	What is the shortest Hello World program in OpenCOBOL?	383
29.45	5.45	How do I get those nifty sequential sequence numbers in a source file?	383
29.46	5.46	Is there a way to count trailing spaces in data fields using OpenCOBOL?	384
29.47	5.47	Is there a way to left justify an edited numeric field?	385
29.48	5.48	Is there a way to determine when OpenCOBOL is running ASCII or EBCDIC?	385
29.49	5.49	Is there a way to determine when OpenCOBOL is running on 32 or 64 bits?	386

29.50	5.50	Does OpenCOBOL support recursion?	386
29.51	5.51	Does OpenCOBOL capture arithmetic overflow?	387
29.52	5.52	Can OpenCOBOL be used for plotting?	389
29.53	5.53	Does OpenCOBOL support the GIMP ToolKit, GTK+?	393
29.54	5.54	What is ocsort?	406
29.55	5.55	When is Easter?	407
29.56	5.56	Does Vim support OpenCOBOL?	411
29.57	5.57	What is w3m?	412
29.58	5.58	What is COB_LIBRARY_PATH?	414
29.59	5.59	Can OpenCOBOL interface with Rexx?	414
29.60	5.60	Does OpenCOBOL support table SEARCH and SORT?	420
29.61	5.61	Can OpenCOBOL handle named pipes?	424
29.62	5.62	Can OpenCOBOL interface with ROOT/CINT?	426
29.63	5.63	Can OpenCOBOL be used to serve HTTP?	429
29.64	5.64	Is there a good SCM tool for OpenCOBOL?	432
29.65	5.65	Does OpenCOBOL interface with FORTRAN?	435
29.66	5.66	Does OpenCOBOL interface with APL?	436
29.67	5.67	Does OpenCOBOL interface with J?	436
29.68	5.68	What is COBOLUnit?	438
29.69	5.69	Can OpenCOBOL interface with Gambas?	439
29.70	5.70	Does OpenCOBOL work with LLVM?	440
<b>30</b>	<b>6</b>	<b>Notes</b>	<b>443</b>
30.1	6.1	big-endian	443
30.2	6.2	little-endian	444
30.3	6.3	ASCII	444
30.4	6.4	currency symbol	444
30.5	6.5	DSO	444
30.6	6.6	errno	444
30.7	6.7	gdb	445
30.8	6.8	GMP	445
30.9	6.9	ISAM	445
30.10	6.10	line sequential	446
30.11	6.11	APT	446
30.12	6.12	ROBODoc Support	446
30.13	6.13	cobol.vim	457
30.14	6.14	make check listing	462
30.15	6.15	ABI	472
30.16	6.16	Tectonics	472
30.17	6.17	Setting Locale	472
30.18	6.18	GNU	473
30.19	6.19	Performing FOREVER?	473
30.20	6.20	POSIX	482
<b>31</b>	<b>7</b>	<b>Authors</b>	<b>483</b>
<b>32</b>	<b>8</b>	<b>Maintainers and Contributors</b>	<b>485</b>
<b>33</b>	<b>9</b>	<b>ChangeLog</b>	<b>487</b>
<b>34</b>		<b>Indices and tables</b>	<b>491</b>
		<b>Bibliography</b>	<b>493</b>



# OpenCOBOL

## Status

This is a 1.1 release candidate of the OpenCOBOL FAQ. Sourced at [ocfaq.rst](#). Courtesy of ReStructuredText, Sphinx and Pygments.

[ocfaq.pdf](#) is also available, using **rst2latex** and then **pdflatex**.

This FAQ is more than a FAQ and less than a FAQ. Someday that will change and this document will be split into an OpenCOBOL manual and a simplified Frequently Asked Questions file.

Website favicon by Mark James, [help.png](#) from the FAMFAMFAM Silk icon set. <http://creativecommons.org/licenses/by/2.5/>

“COBOL Warriors” image Copyright © 2008 Robert Saczkowski. Banner courtesy of the GIMP, Copyright © 2008-2013 Brian Tiffin and both are licensed under Creative Commons Attribution-Share Alike 2.0 Generic License <http://creativecommons.org/licenses/by-sa/2.0/>

## Authors

Brian Tiffin [[btiffin](#)]

Answers, quotes and contributions:

John Ellis [[jrls\\_swla](#)], Vincent Coen, Jim Currey, Bill Klein [[wmklein](#)],  
Ganymede, Simon Sobisch [[human](#)], Rildo Pragana, Sergey Kashyryin,  
Federico Priolo, Frank Swarbrick, Angus, DamonH, Parhs, Gerald Chudyk

Compiler by:

**Roger While** [[Roger](#)],  
Keisuke Nishida [[Keisuke](#)],  
(with the invaluable assistance of many others)

Special credits to

**Gary Cutler** author of the [OpenCOBOL Programmers Guide](#)  
Joseph James Frantz for hosting and advocacy [[aoirthoir](#)]

**Version** 1.1rc13, February 25th, 2013 (now in Sphinx)

**Status** close to complete

**Copyright** Copyright © 2008-2013 Brian Tiffin

**ChangeLog** [ChangeLog](#)

---

**Note:** Regarding COBOL Standards, Official COBOL Standards: There are many references to **standards** in this document. Very few of them are *technically* correct references. Apologies to all the hard working men and women of the technical committees for this unintentional slight. For specific details on what wordings should be used please see [What are the Official COBOL Standards?](#)

---

## OpenCOBOL

### Contents

- 1 OpenCOBOL FAQ
  - 1.1 What is OpenCOBOL?
  - 1.2 What is COBOL?
  - 1.3 How is OpenCOBOL licensed?
  - 1.4 What platforms are supported by OpenCOBOL?
  - 1.5 Are there pre-built OpenCOBOL packages
  - 1.6 What is the most recent version of OpenCOBOL?
  - 1.7 How complete is OpenCOBOL?
  - 1.8 Will I be amazed by OpenCOBOL?
  - 1.9 Who do I thank for OpenCOBOL?
  - 1.10 Does OpenCOBOL include a Test Suite?
  - 1.11 Does OpenCOBOL pass the NIST Test Suite?
  - 1.12 What about OpenCOBOL and benchmarks?
  - 1.13 Can OpenCOBOL be used for CGI?
  - 1.14 Does OpenCOBOL support a GUI?
  - 1.15 Does OpenCOBOL have an IDE?
  - 1.16 Can OpenCOBOL be used for production applications?
  - 1.17 Where can I get more information about COBOL?
  - 1.18 Where can I get more information about OpenCOBOL?
  - 1.19 Can I help out with the OpenCOBOL project?
  - 1.20 Is there an OpenCOBOL mailing list?
  - 1.21 Where can I find more information about COBOL standards?
  - 1.22 Can I see the OpenCOBOL source codes?
  - 1.23 What happened to opencobol.org?
  - 1.24 What is COBOL in Latin?
  - 1.25 Do you know any good jokes?
- 2 History
  - 2.1 What is the history of COBOL?
  - 2.2 What are the Official COBOL Standards?
  - 2.3 What is the development history of OpenCOBOL?
  - 2.4 What is the current version of OpenCOBOL?
- 3 Using OpenCOBOL
  - 3.1 How do I install OpenCOBOL?
  - 3.2 What are the configure options available for building OpenCOBOL?
  - 3.3 Does OpenCOBOL have any other dependencies?
  - 3.4 How does the OpenCOBOL compiler work?
  - 3.5 What is cobc?
  - 3.6 What is cobcrun?
  - 3.7 What is cob-config?
  - 3.8 What compiler options are supported?
  - 3.9 What dialects are supported by OpenCOBOL?
  - 3.10 What extensions are used if cobc is called with/without “-ext” for COPY
  - 3.11 What are the OpenCOBOL compile time configuration files?
  - 3.12 Does OpenCOBOL work with make?
  - 3.13 Do you have a reasonable source code skeleton for OpenCOBOL?
  - 3.14 Can OpenCOBOL be used to write command line stdin, stdout filters?
  - 3.15 How do you print to printers with OpenCOBOL?
  - 3.16 Can I run background processes using OpenCOBOL?
  - 3.17 Is there OpenCOBOL API documentation?
  - 3.18 How do I use LD\_RUN\_PATH with OpenCOBOL?
  - 3.19 What GNU build tool options are available when building OpenCOBOL?
- 4 Reserved Words
  - 4.1 What are the OpenCOBOL RESERVED WORDS?
  - 4.2 Does OpenCOBOL implement any Intrinsic FUNCTIONS?
  - 4.3 Can you clarify the use of FUNCTION in OpenCOBOL?
  - 4.4 What is the difference between the LENGTH verb and FUNCTION LENGTH?



OpenCOBOL FAQ



## 1.1 WHAT IS OPENCOBOL?

[OpenCOBOL](#) is an open-source [COBOL](#) compiler. OpenCOBOL implements a substantial part of the [COBOL 85](#) and [COBOL 2002](#) standards, as well as many extensions of the existent [COBOL](#) compilers.

OpenCOBOL translates [COBOL](#) into [C](#) and compiles the translated code using the native [C](#) compiler. You can build your [COBOL](#) programs on various platforms, including [Unix/Linux](#), [Mac OS X](#), and [Microsoft Windows](#).

The most excellent [OpenCOBOL Programmer's Guide](#) can be found at [OpenCOBOL Programmers Guide](#).



## 1.2 WHAT IS COBOL?

**COBOL** is an acronym for **CO**mmun **B**usiness **O**riented **L**anguage. This author has always thought of it as “Common Business” Oriented more than Common “Business Oriented”, but that emphasis is perhaps up to the reader’s point of view.



## 1.3 HOW IS OPENCOBOL LICENSED?

The compiler is licensed under [GNU General Public License](#).

The run-time library is licensed under [GNU Lesser General Public License](#).

All source codes are copyright by the respective authors.

What that means, roughly, is:

You are allowed to write OpenCOBOL programs that use the libcob run time library however you like. Closed, proprietary, commercial use is allowed and encouraged. You can ship programs in binary form as you wish.

Modifications to the compiler itself, MUST provide access to source code and be licensed under the GNU GPL. This ensures that no one is allowed to call modified sources their own, nor deny anyone else the chance to copy and re-distribute the compiler source code, including your local changes.

Please note: any version of the compiler that is configured to use Berkeley DB beyond version 1.85 must abide by the Oracle license and sources of the COBOL programs that use libdb must be shipped with any binaries.

OpenCOBOL is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.





## 1.4 WHAT PLATFORMS ARE SUPPORTED BY OPENCOBOL?

OpenCOBOL 1.0 the current official release version, hosted on SourceForge.net, compiles on:

- All 32-bit MS Windows (95/98/NT/2000/XP)
- All POSIX (Linux/BSD/UNIX-like OSes)
- OS/X

OpenCOBOL 1.1, has been built on

- MS Windows native
- MS Windows with Cygwin
- GNU/Linux
- POSIX Systems including OpenSolaris
- OS/X
- AS/400
- HP Integrity HPUX 11.23
- RS600 AIX 5
- 390 Mainframe z/OS OMVS/USS
- others



## 1.5 ARE THERE PRE-BUILT OPENCOBOL PACKAGES

Yes. [Debian APT](#), and RPM packages exist. Packages for NetBSD. Many. Google *opencobol packages* for any late breaking news.

A Debian Advanced Package Tool binary package exists for OpenCOBOL 1.0 as **open-cobol** and lists dependencies of

- `libc6 (>= 2.7-1)`,
- `libcob1`,
- `libcob1-dev (= 1.0-1)`,
- `libdb4.5 (>= 4.5.20-3)`,
- `libdb4.5-dev`,
- `libgmp3-dev`,
- `libgmp3c2`,
- `libltdl3-dev`,
- `libncurses5 (>= 5.6+20071006-3)`

Thanks to the gracious efforts of Bart Martens, bartm on Debian's .org domain.

### 5.1 1.5.1 [kiska.net](#) repository

Also check out [kiska.net](#) for binary builds on various platforms. Thanks to Sergey Kashyrin.

### 5.2 1.5.2 [sourceforge](#)

There are OpenCOBOL links at <http://cobol.sourceforge.net>

In particular, <http://sourceforge.net/projects/cobol/files/open-cobol/> can come in handy, with sources and MinGW binaries at a minimum. Maybe more as time goes on.



## 1.6 WHAT IS THE MOST RECENT VERSION OF OPENCOBOL?

See [What is the current version of OpenCOBOL?](#)



## 1.7 HOW COMPLETE IS OPENCOBOL?

OpenCOBOL 1.0 implements a substantial portion of COBOL 85, supports many of the advances and clarifications of COBOL 2002, and includes many extensions in common use from Micro Focus COBOL, ACUCOBOL and other existent compilers.

OpenCOBOL 1.1 implements a more substantial portion of the COBOL 85 Dialect, COBOL 2002 and a growing number of vendor extensions. Some proposed COBOL 20xx features have also been implemented. Compatibility support includes:

- MF for Micro Focus
- IBM for IBM compatibility
- MVS
- BS2000

OpenCOBOL also includes some advanced features allowing source code such as

```
CALL "cfunction" USING BY REFERENCE ADDRESS OF VAR-IN-LINKAGE-SECTION.
```

Passing the equivalent of char\*\*, pointer to pointer to char. Just as a small example of the level of coverage and flexibility provided by OpenCOBOL.

```
DISPLAY
  FUNCTION UPPER-CASE (
    FUNCTION SUBSTITUTE (
      "This is the original string.";
      "original"; "new"; "string"; "text"
    )
  )
END-DISPLAY
```

To allow for substitution of mixed length strings, something not normally so easy in COBOL. The above will output:

```
THIS IS THE NEW TEXT.
```

---

**Note:** While OpenCOBOL can be held to a high standard of quality and robustness, the authors *DO NOT* claim it to be a “Standard Conforming” implementation of COBOL.

---





## 1.8 WILL I BE AMAZED BY OPENCOBOL?

This author believes so. For an open source implementation of COBOL, OpenCOBOL may surprise you in the depth and breadth of its COBOL feature support, usability and robustness.

COBOL has historically been very secretive and low key. Its domain of use being very secretive and low key. COBOL programmers rarely work on systems that would allow for open internet chat regarding details, let alone *existence*. It is a tribute to the professionalism of these programmers that most people rarely, if ever, hear the name COBOL, a programming language with billions of lines of source code compiled and in production around the world over half a century.

OpenCOBOL is poised to change that historic trend, and allow for the long overdue sharing of wisdom that legions of COBOL developers have accumulated over 50 years of success and failure. The OpenCOBOL conversation may be more [POSIX](#) than mainframe, but there is now room to share, critique and pass on the hard lessons learned from critical systems computing. Given that millions of COBOL programmers kept billions of lines of COBOL source out of the press, surely some of the wisdom can be passed on in a way that keeps all the secrets secret while curious developers are exposed to COBOL outside the vaults.



## 1.9 WHO DO I THANK FOR OPENCOBOL?

Many people. In particular Keisuke Nishida and Roger While.

See the THANKS file in the source code archive for more names of people that have worked on the OpenCOBOL project. Roger points out that the list is woefully incomplete. To quote:

```
The OC project would not have been where it is today without the
significant/enormous help from many-many persons. The THANKS
file does not even do justice to this.
```



## 1.10 DOES OPENCOBOL INCLUDE A TEST SUITE?

Why yes it does. 74 syntax tests, 170 coverage tests, and 16 data representation tests in the February 2009 pre-release. 88 syntax, 253 coverage, and 22 data tests in a 2010 cut.

From the development tarball:

```
$ make check
```

will evaluate and report on the test suite. See [make check listing](#) for a current output listing of a test run.



## 1.11 DOES OPENCOBOL PASS THE NIST TEST SUITE?

The National Institute of Standards and Technology, NIST, maintains a COBOL 85 implementation verification suite of tests. An archive of the tests can be found at

[http://www.itl.nist.gov/div897/ctg/cobol\\_form.htm](http://www.itl.nist.gov/div897/ctg/cobol_form.htm)

OpenCOBOL passes many of the tests included in the NIST sponsored COBOL 85 test suite. While it passes over 9000 of the tests, OpenCOBOL does not claim conformance to any level of COBOL *Standard*.

Instructions for use of the NIST suite is included in the build archive under:

```
tests/cobol85/README
```

Basically, it is a simple **uncompress** and **make** then sit back and relax. The scripts run OpenCOBOL over some 374 programs/modules and includes thousands of test passes.

```
Test Modules  
-----
```

```
Core tests:
```

```
NC - COBOL nucleus tests  
SM - COPY sentence tests  
IC - CALL sentence tests
```

```
File I-O tests:
```

```
SQ - Sequential file I-O tests  
RL - Relative file I-O tests  
IX - Indexed file I-O tests  
ST - SORT sentence tests
```

```
Advanced facilities:
```

```
IF - Intrinsic Function tests
```

With the addition of GLOBAL support, the OpenCOBOL 1.1 pre-release fails none of the attempted tests.

The summary.log from a run in February 2009:

```
----- Directory Information -----      --- Total Tests Information ---  
Module Programs Executed Error Crash      Pass Fail Deleted Inspect Total  
-----  
NC                92          92          0          0      4363   0          6          11  4380
```

SM	15	15	0	0	290	0	3	1	294
IC	24	24	0	0	246	0	4	0	250
SQ	81	81	0	0	512	0	6	81	599
RL	32	32	0	0	1827	0	5	0	1832
IX	39	39	0	0	507	0	1	0	508
ST	39	39	0	0	278	0	0	0	278
SG	5	5	0	0	193	0	0	0	193
OB	5	5	0	0	16	0	0	0	16
IF	42	42	0	0	732	0	0	0	732
-----									
Total	374	374	0	0	8964	0	25	93	9082



## 1.12 WHAT ABOUT OPENCOBOL AND BENCHMARKS?

COBOL has a legacy dating back to 1959. Many features of the COBOL standard provide defaults more suitable to mainframe architecture than the personal computer a 3rd millennium OpenCOBOL developer will likely be using.

OpenCOBOL, by default, generates code optimized for **big-endian** hardware. Fairly dramatic speed improvements on Intel architecture can come from simple **USAGE IS COMPUTATIONAL-5** clauses in the DATA DIVISION.

### 12.1 1.12.1 telco billing

There is a benchmark posted at <http://speleotrove.com/decimal/telco.html> and thanks to Bill Klein [wmklein], there is a COBOL entry. From the source code at <http://home.comcast.net/~wmklein/DOX/TELCO.txt> you should only have to modify

```
Input-Output Section.  
File-Control.  
  Select InFile Assign to  
    "C:\expon180.1e6".  
  Select OutFile Assign to  
    "C:\TELCO.TXT"  
    Line  
    Sequential.
```

to point to the correct filename for your local copy of the benchmark million entry file and a suitable OutFile name for a clean compile and run.

In summary, the benchmark reads a large input file containing a suitably distributed list of telephone call durations (each in seconds). For each call, a charging rate is chosen and the price calculated and rounded to hundreths. One or two taxes are applied (depending on the type of call) and the total cost is converted to a character string and written to an output file. Running totals of the total cost and taxes are kept; these are displayed at the end of the benchmark for verification.

A run on an older pentium 4 and the million number file gave:

```
$ echo 'N' | time ./telco  
Enter 'N' to skip calculations:  
0.46user 1.08system 0:01.61elapsed 96%CPU (0avgtext+0avgdata 0maxresident)k  
0inputs+134776outputs (0major+345minor)pagefaults 0swaps  
$ echo '' | time ./telco  
Enter 'N' to skip calculations:  
11.37user 1.41system 0:12.95elapsed 98%CPU (0avgtext+0avgdata 0maxresident)k
```

24inputs+134776outputs (0major+360minor)pagefaults 0swaps

\$ tail TELCO.TXT

35	D		0.31	0.02	0.01		0.34
193	D		1.73	0.11	0.05		1.89
792	L		1.03	0.06			1.09
661	D		5.91	0.39	0.20		6.50
44	L		0.06	0.00			0.06
262	L		0.34	0.02			0.36
-----							
Totals:			922,067.11	57,628.30	25,042.17		1,004,737.58
Start-Time:09:37:23.93							
End-Time:09:37:36.83							

A more recent 1.1 pre-release, on a dual quad-core Xeon box running Linux SLES 10 64-bit:

35	D		0.31	0.02	0.01		0.34
193	D		1.73	0.11	0.05		1.89
792	L		1.03	0.06			1.09
661	D		5.91	0.39	0.20		6.50
44	L		0.06	0.00			0.06
262	L		0.34	0.02			0.36
-----							
Totals:			922,067.11	57,628.30	25,042.17		1,004,737.58
Start-Time:21:40:48.52							
End-Time:21:40:51.92							

3.4 seconds cache-hot. Not bad.

## 1.13 CAN OPENCOBOL BE USED FOR CGI?

Yes. Through standard IO redirection and the extended **ACCEPT ... FROM ENVIRONMENT ...** feature, OpenCOBOL is more than capable of supporting advanced Common Gateway Interface programming. See [How do I use OpenCOBOL for CGI?](#) for a sample *Hello Web* program.

For those developers looking to serve OpenCOBOL applications on hosted systems and no super user privileges, see [How do I use LD\\_RUN\\_PATH with OpenCOBOL?](#) for some pointers.



# 1.14 DOES OPENCOBOL SUPPORT A GUI?

Yes, but not out of the box. There is not currently (*February 2013*) anything that ships with the product.

Third party extensions for Tcl/Tk and bindings for GTK+ do allow for graphical user interfaces. See [Does OpenCOBOL support the GIMP ToolKit, GTK+?](#) and [Can OpenCOBOL interface with Tcl/Tk?](#).

## 14.1 1.14.1 GTK

The expectation is that GTK+ will be completely bound as a callable interface. That is currently (*February 2013*) not the case, with perhaps 2% of the GTK+ functionality wrapped (but with that 2%, fully functional graphical interfaces are possible).

## 14.2 1.14.2 Tcl/Tk

The Tcl/Tk engine is already quite complete but does place most of the burden of GUI development squarely on the Tk side.

## 14.3 1.14.3 Vala, WebKit

Vala will also open up a quick path to GUI development with OpenCOBOL. There is already an embedded web browser using the Vala bindings to WebKit. See [Can OpenCOBOL interface with Vala?](#) for a lot more details.



## 1.15 DOES OPENCOBOL HAVE AN IDE?

Yes and no. There is no IDE that ships with the product. The add1tocobol team is currently (*February 2013*) at work creating extensions for the GNAT Programming Studio. This is working out quite nicely and will likely be the IDE of choice for the add1tocobol OpenCOBOL developers.

See [Can the GNAT Programming Studio be used with OpenCOBOL?](#) for more information.

There is also the Eclipse IDE and a major project for integrating COBOL but this will not be OpenCOBOL specific.

Many text editors have systems in place for invoking compilers. SciTE, Crimson Editor, vi and emacs to name but a few of the hundreds that support edit/compile/test development cycles.

See [Does OpenCOBOL work with make?](#) for some information on command line compile assistance.





## 1.16 CAN OPENCOBOL BE USED FOR PRODUCTION APPLICATIONS?

Depends. OpenCOBOL is still in active development. Feature coverage is growing, and while the current implementation offers great coverage, applicability to any given situation would need to be analyzed and risks evaluated before commitment to production use.

The licensing allows for commercial use, but OpenCOBOL also ships with notice of indemnity, meaning that there are no guarantees when using OpenCOBOL, directly or indirectly.

There may be a time when commercial support of OpenCOBOL is offered, but at the time of writing no known offering exists.

*Search google just in case!*

And yes, OpenCOBOL is used in production environments.

From [Roger]:

```
Incidentally, OC has been (and still is) used in production
environments since 2005.
(This includes projects that I personally worked on plus other
  projects reported to me; these worldwide)
```

```
The OC project would not have been where it is today without the
significant/enormous help from many-many persons. The THANKS
file does not even do justice to this.
```

### 16.1 1.16.1 Nagasaki Prefecture

Reported on [opencobol.org](http://opencobol.org), The Nagasaki Prefecture, population 1.44 million and 30,000 civil employees is using OpenCOBOL in support of its payroll management system. A team of 3 ported and maintain a suite of 200 COBOL programs, mingled with Perl and specialized reporting modules, running on Nec PX9000 big iron and Xeon servers.

### 16.2 1.16.2 Stories from Currey Adkins

Another post from [opencobol.org](http://opencobol.org) in April 2009, *reprinted with permission*.

OpenCOBOL viability

```
For those concerned about the viability of OpenCOBOL in a production
```

environment, I offer our situation as an example.

We started loading OpenCOBOL to a Debian (Etch) Parisc box in mid March. With some valuable help from this forum we were up and running in a few days.

We then explored the CGI capabilities and moved our home-brewed CGI handler (written in HP3000 Cobol) over. We ended up changing only a few lines.

As Marcr's post indicates, we found a MySQL wrapper and made some minor changes to it.

Starting the second week in April we were in full development of new systems for commercial use.

Please accept our congratulations to the community and our gratitude for the help from the forum.

jimc

**Another reference by Jim, some 6 months later in February 2010, which seems to be enough time for any rose-coloured glass effect to have worn off if it was going to.**

For our part, the answer is yes.

You may want to read an earlier thread about this. Search on OpenCOBOL viability.

Having worked with Cobol since the 1960's, my mindset is that no conversion is automatic.

In our case we are not converting from a specific dialect like MF, but instead are either writing entirely new systems or are changing features (making them web based for example) in older systems.

There are some identified failures in OpenCOBOL execution that have been discussed in this forum. We have found them to be inconsequential and simply work around them. Then again I do not remember working with a bug-free compiler.

Our environment is Debian Linux, OpenCOBOL 1.1, MySQL, ISAM (the one provided with the 1.1 prerelease), HTML (via CGI) and a new PreProcessor to relieve the tedium of writing SQL statements.

If you have some "nay sayers" in your organization and would like some support I will be happy to speak with them.

jimc

I hope people don't mind a little advertising in this FAQ, but Jim has done a lot for OpenCOBOL and his company is a community minded company. <http://www.curreyadkins.com>

## 16.3 1.16.3 Public Accounting

Another from opencobol.org

As part of an initial study of COBOL compilers for finding an alternative to that of MicroFocus, OpenCobol was selected to develop a model for the compilation of a public accounting package (1.5 million lines).

The model had to validate this choice, including with the use of sequential indexed files, with OpenCobol version 0.33 and small adjustments to the COBOL code (mainly using reserved keywords and keywords not implemented).

After the functional qualification of this model, the software is in production since July, 2011 under Linux RedHat Enterprise Linux 4 Advanced Server 32-bit virtualized environment VMWARE ESX - 4 GB of RAM - processor dual AMD Opteron 6176 (tm).

The software package is deployed for 650 users whose 150 connected simultaneously, at the peaks of activity and in comparison with the previous platform on AIX 4.3 and MicroFocus, performance gain is in a report, at best, 1-10 (batch of exploitation of entrustment), at worst, 1 to 4 (batch of recalculation).

With the rise of the package version, a functional validation is in progress since September 2011 with OpenCobol version 1.1 under Linux RedHat Enterprise Linux 5 Advanced Server 64-bit and dual Quad-Core AMD Opteron 8356 (tm) processor. No loss of performance related to the new version of OpenCobol (but related to the package of 10% to 20% loss) after campaign in the two environments.



## 1.17 WHERE CAN I GET MORE INFORMATION ABOUT COBOL?

The [COBOL FAQ](#) by William M Klein is a great place to start.

A google of the search words “COBOL” or “OpenCOBOL” are bound to lead to enough days worth of reading of in-depth articles, opinions and technical information to satisfy the greatest of curiosities.

The [COBUG](#) site *COBOL User Groups* is also a wonderful resource for OpenCOBOL developers.

*This is highly subject to change*, but currently (*February 2013*) a Draft of 20xx is available at <http://www.cobolstandard.info/j4/index.htm> and in particular <http://www.cobolstandard.info/j4/files/std.zip>

---

**Note:** While OpenCOBOL can be held to a high standard of quality and robustness, the authors *DO NOT* claim it to be a “Standard Conforming” implementation of COBOL.

---



## 1.18 WHERE CAN I GET MORE INFORMATION ABOUT OPENCOBOL?

The [opencobol.org](http://opencobol.org) website is probably the best place to find out more about the OpenCOBOL system. [add1tocobol.com](http://add1tocobol.com) is a place to find out about a few of the fan initiatives. (An older archive has been stashed at <http://oldsite.add1tocobol.com>)

### 18.1 1.18.1 The OpenCOBOL Programmer's Guide

A very well written and masterful OpenCOBOL reference and COBOL development guide. By Gary Cutler, [OpenCOBOL Programmers Guide](#).





## 1.19 CAN I HELP OUT WITH THE OPENCOBOL PROJECT?

Absolutely. Visit the [opencobol.org](http://opencobol.org) website and either post a message asking what needs to be done, or perhaps join the development mailing list to find out the current state of development. See [Is there an OpenCOBOL mailing list?](#) for some details. OpenCOBOL is a GPL licensed open source project and while [\[Roger\]](#) is the lead developer he is quite open to code submissions. Having a central point of development allows for consistency and the very high level of quality control enjoyed by OpenCOBOL users.

### 19.1 1.19.1 Translation Efforts

A new project has started to see native language support in the **cobc** compile and run-time systems. Please see [http://www.opencobol.org/modules/newbb/viewtopic.php?topic\\_id=1127&forum=1](http://www.opencobol.org/modules/newbb/viewtopic.php?topic_id=1127&forum=1) for details if you think you can help.

Hi folks!

We're starting to translate upcoming versions into different languages. The necessary code changes for OC 2.0 were already done. Now we need translators.

Before posting every stuff here I want to gather the translators here. Who is able and willing to translate the strings (currently 667) into what language(s) [or has somebody who does this]?

From the last discussions I remember people wanting to do this for French, Italian, Spanish, German but I don't remember who exactly said that he/she will help. We already have a Japanese translation, but that needs an heavy update.

...



## 1.20 IS THERE AN OPENCOBOL MAILING LIST?

Yes. Visit [opencobol.org](http://opencobol.org) for details. The OpenCOBOL development mailing list is graciously hosted by SourceForge. The ML archive is available at [http://sourceforge.net/mailarchive/forum.php?forum\\_name=open-cobol-list](http://sourceforge.net/mailarchive/forum.php?forum_name=open-cobol-list) and once you have subscribed, the list will accept messages at the open-cobol-list email destination at lists.sourceforge.net.



## 1.21 WHERE CAN I FIND MORE INFORMATION ABOUT COBOL STANDARDS?

The COBOL 85 standard is documented in

- ANSI X3.23-1985
- ISO 1989-1985
- ANSI X3.23a-1989
- ANSI X3.23b-1993

*This is highly subject to change, but currently (February 2013) a Draft of 20xx is available at <http://www.cobolstandard.info/j4/index.htm> and in particular <http://www.cobolstandard.info/j4/files/std.zip>*

---

**Note:** While OpenCOBOL can be held to a high standard of quality and robustness, the authors *DO NOT* claim it to be a “Standard Conforming” implementation of COBOL.

---



## 1.22 CAN I SEE THE OPENCOBOL SOURCE CODES?

Absolutely. Being an open source system, all sources that are used to build the compiler are available and free.

The *opencobol.org* site has links to release and pre-release archives. Most distributions of GNU/Linux will also have source code bundles. For example

```
$ apt-get source open-cobol
```

on Debian GNU/Linux will retrieve the most recent released package sources.

A **ROBODoc** experimental project to document the source codes is hosted at [ocrobo](#). See [ROBODoc Support](#) for a sample configuration file.

### 22.1 1.22.1 What was used to color the source code listings?

I wrote a Pygments lexer, mushed it into a local copy of [Pygments](#) and then call a `rst2html-pygments.py` program. Requires a fair amount of mucking about. See [ReStructuredText](#) and [Pygments](#) for some details.

**As of January 2013, the COBOL lexer is in mainline Pygments. No more mucking about required.**





## 1.23 WHAT HAPPENED TO OPENCOBOL.ORG?

Due to robot spam, new registrations on [opencobol.org](http://opencobol.org) were disabled in 2012.

The active site is now hosted by SourceForge, at

<https://sourceforge.net/projects/open-cobol/>



## 1.24 WHAT IS COBOL IN LATIN?

I came up with Publicus Negotiatio Cursus Lingua, and then smarter people suggested:

- negotium Orientatur lingua plebeius
- generalis negotium pertineo lingua
- de communi codice pro calculorum negotii
- codex communis pro calculorum negotii

I like the last one. ccpcn, pronounce that as kick-pickin'.

Thanks to Ray, Paul, and Daniel on LinkedIn.



## 1.25 DO YOU KNOW ANY GOOD JOKES?

Maybe.

- A computer without COBOL and Fortran is like a piece of chocolate cake without ketchup or mustard.

*John Krueger*

- A determined coder can write COBOL programs in any language.

*Author: unknown*

- Rumour has it that the object oriented specification for COBOL was code named

*ADD 1 TO COBOL GIVING COBOL.*

*Author: unknown*

A less verbose, more concise version; *very unCOBOL that*

*ADD 1 TO COBOL.*

*Thanks to aoirthoir*

And, just because;

*ADD 1 TO COBOL GIVING OpenCOBOL*

- A common disrespect of COBOL joke is that the acronym stands for:  
Completely Obsolete Business Oriented Language.

*Author unkown*

We know better. The reality is:

Can't Obsolesce Because Of Legacy. *And why would you want to?*

*Brian Tiffin*

- COBOL

Certainly Old But Often Limber.

*Brian Tiffin*

- Ruby on Rails? Don't forget COBOL ON COGS.

<http://www.coboloncogs.org/INDEX.HTM>

- Eat COBOL, 200 billion lines can't be wrong.

*Brian Tiffin*

- What did COBOL yell to the escaping thief?

**STOP RUN RETURNING NOW.**

*Brian Tiffin*

- A COBOL programmer's husband asks, "*Honey can you go to the store and get some milk. And if they have eggs, get a dozen.*" After twenty minutes she returns and flops 12 bags of milk on the table. He looks at her curiously, "*Honey, why did you do that?*" She responds flatly, "**They had eggs.**"

*Author unknown*

- What did COBOL reply to the executive? *Yes, I can*

**PERFORM JUMPS THRU HOOPS.**

*Brian Tiffin*

- What did OpenCOBOL reply to the executive? *Sir, I can*

**PERFORM JUMPS THRU FLAMING-HOOPS UNTIL HELL-FREEZES-OVER.**

*And being COBOL, I have to show you how little code it takes:*

```
identification division.
program-id. freeze.

data division.
working-storage section.
01 hell                pic 9.
   88 hell-freezes-over value 1.

procedure division.
perform jumps thru flaming-hoops until hell-freezes-over.
stop run.

jumps.
flaming-hoops.
divide 1 by 0 giving hell.
```

- Wrote COBOL all morning, all afternoon and into the night. *Another carpe, diem'ed.*

*Brian Tiffin, ripped from a meme, then farberized*

- Ok, sorry for the lame. Here is a link to some actual humour; Bob the Dinosaur, thanks to Scott Adams.

<http://dilbert.com/strips/comic/1997-11-04/>

## 25.1 1.25.1 A 5-7-5 haiku?

How about a 5-7-5 haiku?

```
program-id. one.
procedure division. add
1 to return-code.
```

*\*btiffin\**

Compiles to a program that fails when run. Fails as poetry, fails as code. Your welcome.

## 2 HISTORY

### History

- 2.1 What is the history of COBOL?
- 2.2 What are the Official COBOL Standards?
- 2.3 What is the development history of OpenCOBOL?
- 2.4 What is the current version of OpenCOBOL?

History

### 26.1 2.1 What is the history of COBOL?

Starting in 1959, a committee was formed under the sponsorship of the United States Department of Defense to recommend a short range option regarding business computing. The Conference on Data System Languages (CODASYL) led by Joe Wegstein of National Bureau of Standards (now National Institute of Standards and Technology) developed a new language, and created the first standardized business computer programming language.

The COmmon Business Oriented Language acronym was announced on September 18th, 1959.

Late in 1960, *essentially* the same COBOL program ran on two different hardware platforms, and stakeholders espied the potential for fulfilling the objective of industry wide, compatible business systems.

[Admiral Grace Hopper](#) is affectionately referred to as the *mother of the COBOL language* as she and her previous work with FLOW-MATIC greatly influenced the specifications of the first COBOL.

Standards have been published for:

- COBOL-68
- COBOL-74
- COBOL-85
- COBOL-2002
- Draft work for COBOL-20xx is currently (*February 2013*) underway

and these roughly correspond to the year they were produced. Note the y2k flavour of four digit naming occurred after the millennium change.

Estimates vary, but it is entirely reasonable to believe that of the some 300,000,000,000 (three hundred thousand million) lines of computer source code in production as of 1995, 200,000,000,000 (two hundred thousand million) lines were COBOL. A full 2/3rds of the world's source code at the time.

See the Wikipedia entry for [COBOL](#) for a lot more details.

## 26.2 2.2 What are the Official COBOL Standards?

Many thanks to William Klein, [\[wmklein\]](#) for details on what wordings are to be used when referencing COBOL Standards:

There are several references to "COBOL 85" and these are often distinguished from "Intrinsic Functions".

The official (but really obscure) term that should be used is "Amended Third Standard COBOL". The "clearer" (and IMHO better) term that should be used is something like

- "'85 Standard COBOL with its amendments"

By 1991 (actually 1993 for ISO rather than ANSI) there was no such thing as "just '85 Standard COBOL". The only recognized Standard was the "base" document (X3.23-1985) ALONG with its two amendments

- Intrinsic Functions Module Amendment
- Corrections Amendment

An interesting related fact is that the "Intrinsic Functions Module" was OPTIONAL in the ANSI and ISO COBOL Standards but was REQUIRED (at the HIGH level) for FIPS COBOL. As the "certification tests" were aimed at getting US government contracts, most vendors (who were still doing certification) actually treated Intrinsic Functions required not optional for "High-level" certification. (They were NOT included in the FIPS intermediate certification process).

Bottom-Line:

Although some intrinsic functions were added in the '02 Standard (and more are included in the draft revision), it is not proper (in my opinion) to distinguish between supporting the '85 Standard and supporting intrinsic functions.

P.S. The corrections amendment did make some technical changes but all of these were included in the '02 Standard. Therefore, hopefully, what it did won't impact OpenCOBOL much.

---

**Note:** While OpenCOBOL can be held to a high standard of quality and robustness, the authors *DO NOT* claim it to be a "Standard Conforming" implementation of COBOL.

---

## 26.3 2.3 What is the development history of OpenCOBOL?

OpenCOBOL was initially developed by Keisuke Nishida [\[Keisuke\]](#) from experience working on TinyCOBOL originally developed by Rildo Pragana.

The first public release was version 0.9.0 on January 25th, 2002.



Development continued apace, with version 0.30 released by Keisuke on August 8th, 2004.

Roger While [Roger] then took up the role as lead developer on October 30th, 2004.

**Version 0.31** was released February 1st, 2005.

**Version 0.32** was released May 12th, 2005.

**Version 0.33** started on May 13th, 2005.

**Version 1.0** was released on December 27th, 2007.

**Version 1.1** was released on SourceForge on May 4th, 2012.

**Version 1.1CE** went into active development on May 4th, 2012.

## 26.4 2.4 What is the current version of OpenCOBOL?

OpenCOBOL 1.0 was released December 27th, 2007 by Roger While [Roger].

The decision to go 1.0 from the 0.33 version followed many incremental enhancements from 2005 through till late in 2007.

OpenCOBOL 1.1 pre-release became active on December 27th, 2007 and is currently (*February 2013*) in active development. The pre-release source tar can be found at [OpenCOBOL 1.1](#) with installer instructions at [OpenCOBOL Install](#) and in the INSTALLING text file of the sources.

The 1.1 pre-release of February 2009 was tagged as release on SourceForge in May of 2012. The 1.1 community edition is in active development at <http://sourceforge.net/projects/open-cobol>

After a download and extract from <http://sourceforge.net/projects/open-cobol/files/latest/download?source=files>

```
$ ./configure
$ make
$ make check
$ sudo make install
```

will place a new set of binaries rooted off **/usr/local**

Be sure to see [What are the configure options available for building OpenCOBOL?](#) for all the available options for building from sources.

### 26.4.1 2.4.1 occurlrefresh

If you build a pre-release OC1.1, you will be able to compile the **occurlrefresh.cbl** (with **occurlsym.cpy**) application and an early **occurl.c** libCURL wrapper that allows file transfers off the Internet. **occurlrefresh** includes default filenames for retrieving the most recent pre-release source archive and only updates the local copy if there has been a newer upstream release.

Thanks to [aoirthoir] for hosting these; currently (*February 2013*) at

- [occurlrefresh.cbl](#)
- [occurlsym.cpy](#)
- [occurl.c](#)

and then simply

```
$ ./occurlrefresh
```

to download any new development archives. libCURL tests the modification timestamps, so this procedure is very resource efficient, only pulling from the server if there is something new. A **-b** option is accepted that will spawn off **tar**, **configure** and **make** pass to compile a fresh copy. **-b** does not do an install, you'll still have to do that manually after verifying that everything is ok.

## 3 USING OPENCOBOL

### Using OpenCOBOL

- 3.1 How do I install OpenCOBOL?
- 3.2 What are the configure options available for building OpenCOBOL?
- 3.3 Does OpenCOBOL have any other dependencies?
- 3.4 How does the OpenCOBOL compiler work?
- 3.5 What is cobc?
- 3.6 What is cobcrun?
- 3.7 What is cob-config?
- 3.8 What compiler options are supported?
- 3.9 What dialects are supported by OpenCOBOL?
- 3.10 What extensions are used if cobc is called with/without “-ext” for COPY
- 3.11 What are the OpenCOBOL compile time configuration files?
- 3.12 Does OpenCOBOL work with make?
- 3.13 Do you have a reasonable source code skeleton for OpenCOBOL?
- 3.14 Can OpenCOBOL be used to write command line stdin, stdout filters?
- 3.15 How do you print to printers with OpenCOBOL?
- 3.16 Can I run background processes using OpenCOBOL?
- 3.17 Is there OpenCOBOL API documentation?
- 3.18 How do I use LD\_RUN\_PATH with OpenCOBOL?
- 3.19 What GNU build tool options are available when building OpenCOBOL?

Using OpenCOBOL

### 27.1 3.1 How do I install OpenCOBOL?

Installation instructions can be found at [OpenCOBOL Install](#).

#### 27.1.1 3.1.1 From source with GNU/Linux

```
$ wget http://sourceforge.net/projects/open-cobol/files/open-cobol/1.1/open-cobol-1.1.tar.gz
$ tar xvf open-cobol-1.1.tar.gz
$ cd open-cobol-1.1
$ ./configure
$ make
```

```
$ make check
$ sudo make install
$ sudo ldconfig
```

## 27.1.2 3.1.2 Debian

The Debian binary package makes installing OpenCOBOL 1.0 a snap. From **root** or using **sudo**

```
$ apt-get install open-cobol
```

## 27.1.3 3.1.3 Fedora

From the main Fedora repositories

```
$ yum install open-cobol
```

## 27.1.4 3.1.4 Windows

Build from sources under Cygwin or MinGW. Follow the instructions from the site listed above, or read the OC\_GettingStarted\_Windows document by [wmklein] available online at

- [http://opencobol.add1tocobol.com/oc\\_gettingstarted\\_windows.html](http://opencobol.add1tocobol.com/oc_gettingstarted_windows.html)
- [http://opencobol.add1tocobol.com/OC\\_GettingStarted\\_Windows.pdf](http://opencobol.add1tocobol.com/OC_GettingStarted_Windows.pdf)

Also see [What is the current version of OpenCOBOL?](#).

## 27.1.5 3.1.5 Macintosh

From Ganymede on [opencobol.org](http://opencobol.org)

### HOWTO: Installing OpenCOBOL 1.0.0 (with BerkeleyDB) under Mac OS 10.5.x-10.6.x

On Mac OS X 10.5.x/10.6.x, I have successfully managed to compile and install OpenCOBOL 1.0.0 (including libdb linking), and am now happily compiling production systems with it. It's not *entirely* straightforward, as it involves installing GMP via MacPorts -- the *only way* that GMP will install properly because of some eccentricities in Apple's Xcode development tools (particularly with relation to c99 in gcc), unless you are willing to patch things by hand. In addition, the earlier BerkeleyDB versions (the 4.x.x ones available via MacPorts) cause some strange ioctl errors at runtime under Mac OS X Leopard and Snow Leopard when attempting certain types of ORGANIZATION IS INDEXED operations; precisely what conditions causes this I am yet to fully ascertain. The upshot of it is that in order to compile and run a complete OpenCOBOL 1.0.0 installation on Leopard and Snow Leopard, one has to 1) install GMP via MacPorts; but 2) compile and install a recent version of BerkeleyDB natively.

Probably at some point, I'm going to package this into a pretty-pretty precompiled .app and .dmg along with a rudimentary Cocoa compiler interface. Until then, however -- my COBOL on Mac comrades! -- please do the following:

```
-- INSTALLATION STEPS (Tested on both 10.5.x and 10.6.x) --
1) Download an appropriate MacPorts distribution for your OS:
<http://distfiles.macports.org/MacPorts/>
```

If you want to use the installer:

\* For 10.5.x: MacPorts-1.8.0-10.5-Leopard.dmg

\* For 10.6.x: MacPorts-1.8.0-10.6-SnowLeopard.dmg

From source, MacPorts-1.8.0.tar.gz is confirmed to work on both versions.

NB: Make sure PATH is properly set by install in your active user's ~/.profile.

2) Update MacPorts: `sudo port -d selfupdate`

3) Install GMP with MacPorts: `sudo port install gmp`

4) Download the Oracle Berkeley DB 5.0.21 (or later) .tar.gz source:

<<http://www.oracle.com/technology/products/berkeley-db/db/index.html>>

5) Untar, cd to the Berkeley DB source folder, then:

```
cd /build_unix
```

6) Do the following to configure, make and install Berkeley DB:

```
../dist/configure
```

```
make
```

```
sudo make install
```

7) Download and untar OpenCOBOL 1.0.0, cd to directory

8) Run ./configure, setting CPPFLAGS and LDFLAGS as below (CHANGING ANY

VERSION-SPECIFIC PATHS TO WHAT YOU JUST INSTALLED) as follows:

```
./configure
```

```
CPPFLAGS="-I/opt/local/var/macports/software/gmp/5.0.1_0/opt/local/include/
```

```
-I/usr/local/BerkeleyDB.5.0/include/"
```

```
LDFLAGS="-L/opt/local/var/macports/software/gmp/5.0.1_0/opt/local/lib
```

```
-L/usr/local/BerkeleyDB.5.0/lib/"
```

9) Make and install:

```
make
```

```
sudo make install
```

10) Et voila! Try exiting the directory and invoking `cobc`.

-- YOU SHOULD THEN BE ABLE TO DO SOMETHING LIKE THIS: --

```
phrygia.ganymede-labs.com:bottles ganymede$ sw_vers
```

```
ProductName: Mac OS X
```

```
ProductVersion: 10.5.6
```

```
BuildVersion: 9G55
```

```
phrygia.ganymede-labs.com:bottles ganymede$ cobc -V
```

```
cobc (OpenCOBOL) 1.0.0
```

```
Copyright (C) 2001-2007 Keisuke Nishida
```

```
Copyright (C) 2007 Roger While
```

```
phrygia.ganymede-labs.com:bottles ganymede$ cobc -v -x bottles.cbl
```

```
preprocessing bottles.cbl into
```

```
/var/folders/KI/KI15WC0KGMmrvv0980RztgU+++TI/-Tmp-//cob75450_0.cob translating
```

```
/var/folders/KI/KI15WC0KGMmrvv0980RztgU+++TI/-Tmp-//cob75450_0.cob into
```

```
/var/folders/KI/KI15WC0KGMmrvv0980RztgU+++TI/-Tmp-//cob75450_0.c
```

```
gcc -pipe -c -I/usr/local/include
```

```
-I/opt/local/var/macports/software/gmp/5.0.1_0/opt/local/include/
```

```
-I/usr/local/BerkeleyDB.5.0/include/ -I/usr/local/include -O2 -Wno-unused
```

```
-fsigned-char -Wno-pointer-sign -o
```

```
/var/folders/KI/KI15WC0KGMmrvv0980RztgU+++TI/-Tmp-//cob75450_0.o
```

```
/var/folders/KI/KI15WC0KGMmrvv0980RztgU+++TI/-Tmp-//cob75450_0.c gcc -pipe
```

```
-L/opt/local/var/macports/software/gmp/5.0.1_0/opt/local/lib
```

```
-L/usr/local/BerkeleyDB.5.0/lib/ -o bottles
```

```
/var/folders/KI/KI15WC0KGMmrvv0980RztgU+++TI/-Tmp-//cob75450_0.o
```

```
-L/opt/local/var/macports/software/gmp/5.0.1_0/opt/local/lib
```

```
-L/usr/local/BerkeleyDB.5.0/lib/ -L/usr/local/lib -lcob -lm -lgmp
```

```
-L/usr/local/lib -lintl -liconv -lc -R/usr/local/lib -lncurses -ldb
```

```
With lots of sloppy LINKAGE SECTION kisses,  
-- Ganymede
```

## 27.2 3.2 What are the configure options available for building OpenCOBOL?

*configure* is a defacto standard development tool for POSIX compliant operating systems, in particular GNU/Linux. It examines the current environment and creates a Makefile suitable for the target computer and the package being built.

For OpenCOBOL, the *configure* script accepts **-help** as a command line option to display all of the available configuration choices.

'configure' configures OpenCOBOL 1.1 to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as VAR=VALUE. See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:

```
-h, --help                display this help and exit  
  --help=short            display options specific to this package  
  --help=recursive       display the short help of all the included packages  
-V, --version            display version information and exit  
---quiet, --silent      do not print 'checking...' messages  
  --cache-file=FILE     cache test results in FILE [disabled]  
-C, --config-cache      alias for '--cache-file=config.cache'  
-n, --no-create         do not create output files  
  --srcdir=DIR          find the sources in DIR [configure dir or `..']
```

Installation directories:

```
--prefix=PREFIX          install architecture-independent files in PREFIX  
                        [/usr/local]  
--exec-prefix=EPREFIX    install architecture-dependent files in EPREFIX  
                        [PREFIX]
```

By default, 'make install' will install all the files in '/usr/local/bin', '/usr/local/lib' etc. You can specify an installation prefix other than '/usr/local' using '--prefix', for instance '--prefix=\$HOME'.

For better control, use the options below.

Fine tuning of the installation directories:

```
--bindir=DIR             user executables [EPREFIX/bin]  
--sbindir=DIR           system admin executables [EPREFIX/sbin]  
--libexecdir=DIR        program executables [EPREFIX/libexec]  
--datadir=DIR           read-only architecture-independent data [PREFIX/share]  
--sysconfdir=DIR        read-only single-machine data [PREFIX/etc]  
--sharedstatedir=DIR    modifiable architecture-independent data [PREFIX/com]  
--localstatedir=DIR     modifiable single-machine data [PREFIX/var]  
--libdir=DIR            object code libraries [EPREFIX/lib]  
--includedir=DIR        C header files [PREFIX/include]
```

```
--oldincludedir=DIR    C header files for non-gcc [/usr/include]
--infodir=DIR          info documentation [PREFIX/info]
--mandir=DIR           man documentation [PREFIX/man]
```

Program names:

```
--program-prefix=PREFIX    prepend PREFIX to installed program names
--program-suffix=SUFFIX    append SUFFIX to installed program names
--program-transform-name=PROGRAM  run sed PROGRAM on installed program names
```

System types:

```
--build=BUILD    configure for building on BUILD [guessed]
--host=HOST      cross-compile to build programs to run on HOST [BUILD]
```

Optional Features:

```
--disable-FEATURE    do not include FEATURE (same as --enable-FEATURE=no)
--enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
--enable-maintainer-mode  enable make rules and dependencies not useful
                        (and sometimes confusing) to the casual installer
--disable-dependency-tracking  speeds up one-time build
--enable-dependency-tracking  do not reject slow dependency extractors
--enable-experimental    (OpenCOBOL) enable experimental code (Developers only!)
--enable-param-check     (OpenCOBOL) enable CALL parameter checking
--enable-shared[=PKGS]  build shared libraries [default=yes]
--enable-static[=PKGS]  build static libraries [default=yes]
--enable-fast-install[=PKGS]
                        optimize for fast installation [default=yes]
--disable-libtool-lock   avoid locking (might break parallel builds)
--disable-rpath          do not hardcode runtime library paths
--disable-nls            do not use Native Language Support
```

Optional Packages:

```
--with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
--without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
--with-cc=<cc>          (OpenCOBOL) specify the C compiler used by cobc
--with-seqra-extfh      (OpenCOBOL) Use external SEQ/RAN file handler
--with-cisam            (OpenCOBOL) Use CISAM for ISAM I/O
--with-disam            (OpenCOBOL) Use DISAM for ISAM I/O
--with-vbisam           (OpenCOBOL) Use VBISAM for ISAM I/O
--with-index-extfh      (OpenCOBOL) Use external ISAM file handler
--with-dbl              (OpenCOBOL) use Berkeley DB 1.85 (libdb-1.85)
--with-db               (OpenCOBOL) use Berkeley DB 3.0 or later (libdb) (default)
--with-lfs64            (OpenCOBOL) use large file system for file I/O (default)
--with-dl               (OpenCOBOL) use system dynamic loader (default)
--with-patch-level      (OpenCOBOL) define a patch level (default 0)
--with-varse            (OpenCOBOL) define variable sequential format (default 0)
--with-gnu-ld           assume the C compiler uses GNU ld [default=no]
--with-pic              try to use only PIC/non-PIC objects [default=use
                        both]
--with-tags[=TAGS]     include additional configurations [automatic]
--with-gnu-ld           assume the C compiler uses GNU ld default=no
--with-libiconv-prefix[=DIR]  search for libiconv in DIR/include and DIR/lib
--without-libiconv-prefix  don't search for libiconv in includedir and libdir
--with-libintl-prefix[=DIR]  search for libintl in DIR/include and DIR/lib
--without-libintl-prefix  don't search for libintl in includedir and libdir
```

Some influential environment variables:

CC	C compiler command
CFLAGS	C compiler flags
LDFLAGS	linker flags, e.g. -L<lib dir> if you have libraries in a nonstandard directory <lib dir>
CPPFLAGS	C/C++ preprocessor flags, e.g. -I<include dir> if you have headers in a nonstandard directory <include dir>
CPP	C preprocessor
CXXCPP	C++ preprocessor

Use these variables to override the choices made by 'configure' or to help it to find libraries and programs with nonstandard names/locations.

Report bugs to <open-cobol-list@lists.sourceforge.net>.

## 27.3 3.3 Does OpenCOBOL have any other dependencies?

OpenCOBOL relies on a native C compiler with [POSIX](#) compatibility. GCC being a freely available compiler collection supported by most operating systems currently (*February 2013*) in use.

OpenCOBOL requires the following external libraries to be installed:

**GNU MP (libgmp) 4.1.2 or later** libgmp is used to implement decimal arithmetic. GNU MP is licensed under GNU Lesser General Public License.

**GNU Libtool (libltdl)** libltdl is used to implement dynamic CALL statements. GNU Libtool is licensed under GNU Lesser General Public License.

NOTE - Libtool is not required for Linux and Windows (including MinGW and Cygwin)

The following libraries are optional:

**Berkeley DB (libdb) 1.85 or later** libdb can be used to implement indexed file I/O and SORT/MERGE. Berkeley DB is licensed under the original BSD License (1.85) or their own open-source license (2.x or later). Note that, as of 2.x, if you linked your software with Berkeley DB, you must distribute the source code of your software along with your software, or you have to pay royalty to Oracle Corporation. For more information about Oracle Berkeley DB dual licensing go to : [Oracle / Embedded / Oracle Berkeley DB](#)

**Ncurses (libncurses) 5.2 or later** libncurses can be used to implement SCREEN SECTION. Ncurses is licensed under a BSD-style license.

## 27.4 3.4 How does the OpenCOBOL compiler work?

OpenCOBOL is a multi-stage command line driven compiler. Command line options control what stages are performed during processing.

1. Preprocess
2. Translate
3. Compile
4. Assemble
5. Link
6. Build



OpenCOBOL produces intermediate C source code that is then passed to a configured C compiler and other tools. the GNU C compiler, **gcc** being a standard.

The main tool, **cobc**, by default, produces modules, linkable shared object files.

### 27.4.1 3.4.1 Example of OpenCOBOL stages

Documenting the output of the various stages of OpenCOBOL compilation.

### 27.4.2 3.4.2 Original source code;

```
$ cat hello.cob

000100* HELLO.COB OpenCOBOL FAQ example
000200 IDENTIFICATION DIVISION.
000300 PROGRAM-ID. hello.
000400 PROCEDURE DIVISION.
000500     DISPLAY "Hello World!".
000600     STOP RUN.
```

### 27.4.3 3.4.3 Preprocess

```
$ cobc -E hello.cob
```

Preprocess only pass. One operation of the preprocessor is to convert FIXED format to FREE format. **COPY** includes are also read in along with **REPLACE** substitution. The above command displayed:

```
# 1 "hello.cob"

IDENTIFICATION DIVISION.
PROGRAM-ID. hello.
PROCEDURE DIVISION.
  DISPLAY "Hello World!".
  STOP RUN.
```

to standard out.

### 27.4.4 3.4.4 Translate

```
$ cobc -C hello.cob
```

Translate only; preprocesses and then translates the COBOL sources into C. You can examine these files to get a good sense of how the OpenCOBOL environment interacts with the native C facilities. OpenCOBOL 1.1 produced **hello.c.h** and **hello.c**.

### 27.4.5 3.4.5 hello.c.h

```
/* Generated by          cobc 1.1.0 */
/* Generated from       hello.cob */
/* Generated at         Oct 04 2008 00:19:36 EDT */
/* OpenCOBOL build date Oct 01 2008 22:15:19 */
/* OpenCOBOL package date Oct 01 2008 16:31:26 CEST */
```

```
/* Compile command          cbbc -C hello.cob */

/* PROGRAM-ID : hello */

static unsigned char b_5[4] __attribute__((aligned));      /* COB-CRT-STATUS */
static unsigned char b_1[4] __attribute__((aligned));      /* RETURN-CODE */
static unsigned char b_2[4] __attribute__((aligned));      /* SORT-RETURN */
static unsigned char b_3[4] __attribute__((aligned));      /* NUMBER-OF-CALL-PARAMETERS */

/* attributes */
static cob_field_attr a_1 = {16, 4, 0, 0, NULL};
static cob_field_attr a_2 = {33, 0, 0, 0, NULL};

/* fields */
static cob_field f_5      = {4, b_5, &a_1};                /* COB-CRT-STATUS */

/* constants */
static cob_field c_1      = {12, (unsigned char *)"Hello World!", &a_2};

/* ----- */
```

## 27.4.6 3.4.6 hello.c

```
/* Generated by          cbbc 1.1.0 */
/* Generated from        hello.cob */
/* Generated at          Oct 04 2008 00:19:36 EDT */
/* OpenCOBOL build date  Oct 01 2008 22:15:19 */
/* OpenCOBOL package date Oct 01 2008 16:31:26 CEST */
/* Compile command       cbbc -C hello.cob */

#define __USE_STRING_INLINES 1
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <libcob.h>

#define COB_SOURCE_FILE      "hello.cob"
#define COB_PACKAGE_VERSION "1.1"
#define COB_PATCH_LEVEL     0

/* function prototypes */
static int hello_ (const int);

int hello (void);

/* functions */

int
hello ()
{
    return hello_ (0);
}

/* end functions */
```

```
static int
hello_ (const int entry)
{

#include "hello.c.h" /* local variables */

static int initialized = 0;
static cob_field *cob_user_parameters[COB_MAX_FIELD_PARAMS];
static cob_module module = { NULL, NULL, &f_5, NULL, cob_user_parameters, 0, '.', '$', ',', 1, 1, 1

/* perform frame stack */
int frame_index;
struct frame {
    int perform_through;
    void *return_address;
} frame_stack[255];

/* Start of function code */

if (unlikely(entry < 0)) {
    if (!initialized) {
        return 0;
    }
    initialized = 0;
    return 0;
}

module.next = cob_current_module;
cob_current_module = &module;

if (unlikely(initialized == 0))
{
    if (!cob_initialized) {
        cob_fatal_error (COB_FERROR_INITIALIZED);
    }
    cob_check_version (COB_SOURCE_FILE, COB_PACKAGE_VERSION, COB_PATCH_LEVEL);
    if (module.next)
        cob_set_cancel ((const char *)"hello", (void *)hello, (void *)hello_);
    (*(int *) (b_1)) = 0;
    (*(int *) (b_2)) = 0;
    (*(int *) (b_3)) = 0;
    memset (b_5, 48, 4);

    initialized = 1;
}

/* initialize frame stack */
frame_index = 0;
frame_stack[0].perform_through = -1;

/* initialize number of call params */
(*(int *) (b_3)) = cob_call_params;
cob_save_call_params = cob_call_params;

goto l_2;
```

```
/* PROCEDURE DIVISION */

/* hello: */

1_2;;

/* MAIN SECTION: */

/* MAIN PARAGRAPH: */

/* hello.cob:5: DISPLAY */
{
  cob_new_display (0, 1, 1, &c_1);
}
/* hello.cob:6: STOP */
{
  cob_stop_run ((*int *) (b_1));
}

cob_current_module = cob_current_module->next;
return ((*int *) (b_1));
}

/* end function stuff */
```

## 27.4.7 3.4.7 Generate assembler

Using the `-S` switch asks **cobc** to ask the C compiler tool chain to not process farther than the assembler code generation phase.

```
$ cobc -S hello.cob
```

## 27.4.8 3.4.8 hello.s

```
.file "cob9141_0.c"
.text
.globl hello
.type hello, @function
hello:
  pushl  %ebp
  movl   %esp, %ebp
  subl   $8, %esp
  movl   $0, (%esp)
  call   hello_
  leave
  ret
.size   hello, .-hello
.data
.align 4
.type   module.5786, @object
.size   module.5786, 28
module.5786:
  .long  0
```

```

.long    0
.long    f_5.5782
.long    0
.long    cob_user_parameters.5785
.byte    0
.byte    46
.byte    36
.byte    44
.byte    1
.byte    1
.byte    1
.byte    0
.local   cob_user_parameters.5785
.comm    cob_user_parameters.5785,256,32
.local   initialized.5784
.comm    initialized.5784,4,4
.section          .rodata
.LC0:
.string  "Hello World!"
.data
.align  4
.type    c_1.5783, @object
.size    c_1.5783, 12
c_1.5783:
.long    12
.long    .LC0
.long    a_2.5781
.align  4
.type    f_5.5782, @object
.size    f_5.5782, 12
f_5.5782:
.long    4
.long    b_5.5776
.long    a_1.5780
.align  4
.type    a_2.5781, @object
.size    a_2.5781, 8
a_2.5781:
.byte    33
.byte    0
.byte    0
.byte    0
.long    0
.align  4
.type    a_1.5780, @object
.size    a_1.5780, 8
a_1.5780:
.byte    16
.byte    4
.byte    0
.byte    0
.long    0
.local   b_3.5779
.comm    b_3.5779,4,16
.local   b_2.5778
.comm    b_2.5778,4,16
.local   b_1.5777
.comm    b_1.5777,4,16

```

```
.local b_5.5776
.comm b_5.5776,4,16
.section .rodata
.LC1:
.string "1.1"
.LC2:
.string "hello.cob"
.LC3:
.string "hello"
.text
.type hello_, @function
hello_:
    pushl   %ebp
    movl   %esp, %ebp
    subl   $2072, %esp
    movl   8(%ebp), %eax
    shrl   $31, %eax
    testl  %eax, %eax
    je     .L4
    movl   initialized.5784, %eax
    testl  %eax, %eax
    jne   .L5
    movl   $0, -2052(%ebp)
    jmp   .L6
.L5:
    movl   $0, initialized.5784
    movl   $0, -2052(%ebp)
    jmp   .L6
.L4:
    movl   cob_current_module, %eax
    movl   %eax, module.5786
    movl   $module.5786, cob_current_module
    movl   initialized.5784, %eax
    testl  %eax, %eax
    sete   %al
    movzbl %al, %eax
    testl  %eax, %eax
    je     .L7
    movl   cob_initialized, %eax
    testl  %eax, %eax
    jne   .L8
    movl   $0, (%esp)
    call   cob_fatal_error
.L8:
    movl   $0, 8(%esp)
    movl   $.LC1, 4(%esp)
    movl   $.LC2, (%esp)
    call   cob_check_version
    movl   module.5786, %eax
    testl  %eax, %eax
    je     .L9
    movl   $hello_, 8(%esp)
    movl   $hello, 4(%esp)
    movl   $.LC3, (%esp)
    call   cob_set_cancel
.L9:
    movl   $b_1.5777, %eax
    movl   $0, (%eax)
```

```

movl    $b_2.5778, %eax
movl    $0, (%eax)
movl    $b_3.5779, %eax
movl    $0, (%eax)
movl    $4, 8(%esp)
movl    $48, 4(%esp)
movl    $b_5.5776, (%esp)
call    memset
movl    $1, initialized.5784
.L7:
movl    $0, -4(%ebp)
movl    $-1, -2044(%ebp)
movl    $b_3.5779, %edx
movl    cob_call_params, %eax
movl    %eax, (%edx)
movl    cob_call_params, %eax
movl    %eax, cob_save_call_params
.L10:
movl    $c_1.5783, 12(%esp)
movl    $1, 8(%esp)
movl    $1, 4(%esp)
movl    $0, (%esp)
call    cob_new_display
movl    $b_1.5777, %eax
movl    (%eax), %eax
movl    %eax, (%esp)
call    cob_stop_run
.L6:
movl    -2052(%ebp), %eax
leave
ret
.size   hello_, .-hello_
.ident  "GCC: (Debian 4.3.1-9) 4.3.1"
.section        .note.GNU-stack,"",@progbits

```

Produces **hello.s**.

### 27.4.9 3.4.9 Produce object code

```
$ cobc -c hello.cob
```

Compile and assemble, do not link. Produces **hello.o**.

### 27.4.10 3.4.10 Build modules

```
$ cobc -m hello.cob
```

Build dynamically loadable module. This is the *default behaviour*. This example produces **hello.so** or **hello.dll**.

```
$ cobc -b hello.cob
```

will do the same thing, but in this case, the extended Build is the same as the single Module build with **-m**. **-b** will build a dynamically loadable module that includes all of the entry points created from multiple command line inputs. It's fun; you can mix **.cob**, **.c**, and **-l** libs and OpenCOBOL does the right thing glueing it all together. **-b** Build is suited to Programming In The Large and using **cobcrun**.

### 27.4.11 3.4.11 Module run

```
$ cobcrun hello
Hello World!
```

Will scan the DSO hello.so, and then link, load, and execute hello.

### 27.4.12 3.4.12 Create executable

```
$ cobc -x hello.cob
```

Create an executable program. This examples produces **hello** or **hello.exe**.

**This is important.** *cobc* produces a *Dynamic Shared Object* by default. To create executables, you need to use **-x**.

```
$ ./hello
Hello World!
```

OpenCOBOL also supports features for multiple source, multiple language programming, detailed in the FAQ at [Does OpenCOBOL support modules?](#).

### 27.4.13 3.4.13 sizes for hello on Fedora 16

The directory after using the various *cobc* options:

```
-rwxrwxr-x. 1 btiffin btiffin 9730 Apr 22 00:25 hello
-rw-rw-r--. 1 btiffin btiffin 2253 Apr 22 00:26 hello.c
-rw-rw-r--. 1 btiffin btiffin 835 Apr 22 00:26 hello.c.h
-rw-rw-r--. 1 btiffin btiffin 391 Apr 22 00:26 hello.c.l.h
-rw-rw-r--. 1 btiffin btiffin 181 Apr 22 00:24 hello.cob
-rw-rw-r--. 1 btiffin btiffin 3288 Apr 22 00:24 hello.o
-rw-rw-r--. 1 btiffin btiffin 2577 Apr 22 00:26 hello.s
-rwxrwxr-x. 1 btiffin btiffin 9334 Apr 22 00:27 hello.so
```

## 27.5 3.5 What is cobc?

**cobc** is the OpenCOBOL compiler. It processes source code into object, library or executable code.

See [What compiler options are supported?](#) for more information.

## 27.6 3.6 What is cobcrun?

**cobcrun** is the OpenCOBOL driver program that allows the execution of programs stored in OpenCOBOL modules.

The **cobc** compiler, by default, produces modules (the *-m* option). These modules are linkable dynamic shared objects (DSO). Using GNU/Linux for example

```
$ cobc -x hello.cob
$ ./hello
Hello World!
$ cobc hello.cob
$ cobcrun hello
Hello World!
```



The **cobc -x hello.cob** built an executable binary called hello. The **cobc hello.cob** produced a DSO hello.so, and **cobcrun** resolves the entry point and executes the code, right from the DSO.

**cobcrun** is the compiler author's preferred way to manage OpenCOBOL development. It alleviates knowing which source file needs **-x** while encouraging proper modular programming, a mainstay of OpenCOBOL.

## 27.7 3.7 What is cob-config?

**cob-config** is a program that can be used to find the C compiler flags and libraries required for compiling. Using GNU/Linux for example

```
$ cob-config
Usage: cob-config [OPTIONS]
Options:
    [--prefix[=DIR]]
    [--exec-prefix[=DIR]]
    [--version]
    [--libs]
    [--cflags]
$ cob-config --libs
-L/usr/local/lib -lcob -lm -lgmp -lncurses -ldb
$ cob-config --cflags
-I/usr/local/include
```

You may need to use these features during mixed source language development, usually by back-ticking the command output inline with other **gcc** commands.

## 27.8 3.8 What compiler options are supported?

The OpenCOBOL system strives to follow standards, yet also remain a viable compiler option for the many billions of existing lines of COBOL sources, by supporting many existing extensions to the COBOL language. Many details of the compile can be controlled with command line options. Please also see [What are the OpenCOBOL compile time configuration files?](#) for more details on this finely tuned control.

```
$ cobc -V
cobc (OpenCOBOL) 1.1.0
Copyright (C) 2001-2008 Keisuke Nishida / Roger While
Built Oct 29 2008 16:32:02
Packaged Oct 28 2008 19:05:45 CET

$ cobc --help
Usage: cobc [options] file...

Options:
  --help                Display this message
  --version, -V         Display compiler version
  -v                    Display the programs invoked by the compiler
  -x                    Build an executable program
  -m                    Build a dynamically loadable module (default)
  -std=<dialect>        Compile for a specific dialect :
                        cobol2002  Cobol 2002
                        cobol85    Cobol 85
                        ibm         IBM Compatible
                        mvs         MVS Compatible
```

```

        bs2000      BS2000 Compatible
        mf          Micro Focus Compatible
        default     When not specified
See config/default.conf and config/*.conf
-free            Use free source format
-fixed           Use fixed source format (default)
-O, -O2, -Os    Enable optimization
-g              Produce debugging information in the output
-debug          Enable all run-time error checking
-o <file>       Place the output into <file>
-b              Combine all input files into a single
                dynamically loadable module
-E              Preprocess only; do not compile, assemble or link
-C              Translation only; convert COBOL to C
-S              Compile only; output assembly file
-c              Compile and assemble, but do not link
-t <file>       Generate and place a program listing into <file>
-I <directory> Add <directory> to copy/include search path
-L <directory> Add <directory> to library search path
-l <lib>        Link the library <lib>
-D <define>     Pass <define> to the C compiler
-conf=<file>    User defined dialect configuration - See -std=
--list-reserved Display reserved words
--list-intrinsics Display intrinsic functions
--list-mnemonics Display mnemonic names
-save-temps(=<dir>) Save intermediate files (default current directory)
-MT <target>    Set target file used in dependency list
-MF <file>      Place dependency list into <file>
-ext <extension> Add default file extension

-W              Enable ALL warnings
-Wall           Enable all warnings except as noted below
-Wobsolete      Warn if obsolete features are used
-Warchaic       Warn if archaic features are used
-Wredefinition  Warn incompatible redefinition of data items
-Wconstant      Warn inconsistent constant
-Wparentheses   Warn lack of parentheses around AND within OR
-Wstrict-typing Warn type mismatch strictly
-Wimplicit-define Warn implicitly defined data items
-Wcall-params   Warn non 01/77 items for CALL params (NOT set with -Wall)
-Wcolumn-overflow Warn text after column 72, FIXED format (NOT set with -Wall)
-Wterminator    Warn lack of scope terminator END-XXX (NOT set with -Wall)
-Wtruncate      Warn possible field truncation (NOT set with -Wall)
-Wlinkage       Warn dangling LINKAGE items (NOT set with -Wall)
-Wunreachable   Warn unreachable statements (NOT set with -Wall)

-ftrace         Generate trace code (Executed SECTION/PARAGRAPH)
-ftraceall      Generate trace code (Executed SECTION/PARAGRAPH/STATEMENTS)
-fsyntax-only   Syntax error checking only; don't emit any output
-fdebugging-line Enable debugging lines ('D' in indicator column)
-fsource-location Generate source location code (Turned on by -debug or -g)
-fimplicit-init Do automatic initialization of the Cobol runtime system
-fsign-ascii    Numeric display sign ASCII (Default on ASCII machines)
-fsign-ebcdic   Numeric display sign EBCDIC (Default on EBCDIC machines)
-fstack-check   PERFORM stack checking (Turned on by -debug or -g)
-ffold-copy-lower Fold COPY subject to lower case (Default no transformation)
-ffold-copy-upper Fold COPY subject to upper case (Default no transformation)
-fnotrunc       Do not truncate binary fields according to PICTURE

```

-ffunctions-all	Allow use of intrinsic functions without FUNCTION keyword
-fmfcomment	'*' or '/' in column 1 treated as comment (FIXED only)
-fnnull-param	Pass extra NULL terminating pointers on CALL statements

## 27.9 3.9 What dialects are supported by OpenCOBOL?

Using the `std=<dialect>` compiler option, OpenCOBOL can be configured to compile using specific historical COBOL compiler features and quirks.

Supported dialects include:

- default
- cobol85
- cobol2002
- ibm
- mvs
- mf
- bs2000

For details on what options and switches are used to support these dialect compilers, see the `config/` directory of your OpenCOBOL installation. For Debian GNU/Linux, that will be `/usr/share/open-cobol/config/` if you used APT to install an OpenCOBOL package or `/usr/local/share/open-cobol/config/` after a build from the source archive.

For example: the `bs2000.conf` file restricts data representations to 2, 4 or 8 byte binary while `mf.conf` allows data representations from 1 thru 8 bytes. `cobol85.conf` allows debugging lines, `cobol2002.conf` configures the compiler to warn that this feature is obsolete.

## 27.10 3.10 What extensions are used if cobc is called with/without “-ext” for COPY

From Roger on [opencobol.org](http://opencobol.org)

In the following order -  
 COPY, CBL, COB, cpy, cbl, cob and finally with no extension.

User specified extensions (in the order as per command line) are inspected  
 PRIOR to the above defaults.

ie. They take precedence.

## 27.11 3.11 What are the OpenCOBOL compile time configuration files?

To assist in the support of the various existent COBOL compilers, OpenCOBOL reads configuration files controlling various aspects of a compile pass.

Each supported dialect will also have a *.conf* file in the **config/** sub-directory of its installation. For Debian GNU/Linux, these will be in **/usr/share/open-cobol/config/** or **/usr/local/share/open-cobol/config** under default package and default *make* conditions.

For example, the default configuration, *default.conf* is:

```
# COBOL compiler configuration                                -*- sh -*-

# Value: any string
name: "OpenCOBOL"

# Value: int
tab-width: 8
text-column: 72

# Value: 'cobol2002', 'mf', 'ibm'
#
assign-clause: mf

# If yes, file names are resolved at run time using environment variables.
# For example, given ASSIGN TO "DATAFILE", the actual file name will be
# 1. the value of environment variable 'DD_DATAFILE' or
# 2. the value of environment variable 'dd_DATAFILE' or
# 3. the value of environment variable 'DATAFILE' or
# 4. the literal "DATAFILE"
# If no, the value of the assign clause is the file name.
#
# Value: 'yes', 'no'
filename-mapping: yes

# Value: 'yes', 'no'
pretty-display: yes

# Value: 'yes', 'no'
auto-initialize: yes

# Value: 'yes', 'no'
complex-odo: no

# Value: 'yes', 'no'
indirect-redefines: no

# Value:          signed  unsigned  bytes
#                - - - - -  - - - - -  - - - -
# '2-4-8'         1 - 4          5 - 9          10 - 18         2
#                5 - 9          10 - 18         15 - 21         4
#                10 - 18         15 - 21         20 - 26         8
#
# '1-2-4-8'       1 - 2          3 - 4          5 - 9          10 - 18         1
#                3 - 4          5 - 9          7 - 11         15 - 21         2
#                5 - 9          7 - 11         9 - 13         15 - 21         4
#                7 - 9          9 - 13         11 - 15         15 - 21         8
#
# '1--8'          1 - 2          3 - 4          5 - 6          7 - 9          10 - 11         12 - 14         1
#                3 - 4          5 - 6          7 - 8          9 - 11         12 - 13         14 - 16         2
#                5 - 6          7 - 8          9 - 10         11 - 12         13 - 14         15 - 16         3
#                7 - 9          9 - 11         11 - 13         13 - 15         15 - 17         17 - 19         4
#                10 - 11         12 - 13         14 - 15         16 - 17         18 - 19         20 - 21         5
#                12 - 14         14 - 16         16 - 18         18 - 20         20 - 22         22 - 24         6
```

```

#           15 - 16   15 - 16       7
#           17 - 18   17 - 18       8
binary-size: 1-2-4-8

# Value: 'yes', 'no'
binary-truncate: yes

# Value: 'native', 'big-endian'
binary-byteorder: big-endian

# Value: 'yes', 'no'
larger-redefines-ok: no

# Value: 'yes', 'no'
relaxed-syntax-check: no

# Perform type OSVS - If yes, the exit point of any currently executing perform
# is recognized if reached.
# Value: 'yes', 'no'
perform-osvs: no

# If yes, non-parameter linkage-section items remain allocated
# between invocations.
# Value: 'yes', 'no'
sticky-linkage: no

# If yes, allow non-matching level numbers
# Value: 'yes', 'no'
relax-level-hierarchy: no

# not-reserved:
# Value: Word to be taken out of the reserved words list
# (case independent)

# Dialect features
# Value: 'ok', 'archaic', 'obsolete', 'skip', 'ignore', 'unconformable'
author-paragraph:           obsolete
memory-size-clause:         obsolete
multiple-file-tape-clause:  obsolete
label-records-clause:       obsolete
value-of-clause:            obsolete
data-records-clause:        obsolete
top-level-occurs-clause:    skip
synchronized-clause:       ok
goto-statement-without-name: obsolete
stop-literal-statement:     obsolete
debugging-line:             obsolete
padding-character-clause:   obsolete
next-sentence-phrase:       archaic
eject-statement:            skip
entry-statement:            obsolete
move-noninteger-to-alphanumeric: error
odo-without-to:             ok

```

## 27.12 3.12 Does OpenCOBOL work with make?

Absolutely. Very well.

A sample **makefile**

```
# OpenCOBOL rules

COBCWARN = -W

# create an executable
%: %.cob
    cobc $(COBCWARN) -x $^ -o $@

# create a dynamic module
%.so: %.cob
    cobc $(COBCWARN) -m $^ -o $@

# create a linkable object
%.o: %.cob
    cobc $(COBCWARN) -c $^ -o $@

# generate C code
%.c: %.cob
    cobc $(COBCWARN) -C $^

# generate assembly
%.s: %.cob
    cobc $(COBCWARN) -S $^

# generate intermediate suitable for cobxref
%.i: %.cob
    [ -d tmps ] || mkdir tmps
    cobc $(COBCWARN) --save-temps=tmps -c $^

# hack extension; create an executable; if errors, call vim in quickfix
%.q: %.cob
    cobc $(COBCWARN) -x $^ 2>errors.err || vi -q

# hack extension; make binary; capture warnings, call vim quickfix
%.qw: %.cob
    cobc $(COBCWARN) -x $^ 2>errors.err ; vi -q

# run ocdoc to get documentation
%.html: %.cob
    ./ocdoc $^ $*.rst $*.html $*.css

# run cobxref and get a cross reference listing (leaves tmps dir around)
%.lst: %.cob
    [ -d tmps ] || mkdir tmps
    cobc $(COBCWARN) --save-temps=tmps -c $^ -o tmps/$*.o && ~/writing/add1/tools/cobxref/cobxref t

# tectonics for occurlrefresh
occurlrefresh: occurl.c occurlsym.cpy occurlrefresh.cbl
    cobc -c -Wall occurl.c
    cobc -x -lcurl occurlrefresh.cbl occurl.o
```

And now to compile a small program called **program.cob**, just use

```

$ make program      # for executables
$ make program.o   # for object files
$ make program.so  # for shared library
$ make program.q   # create an executable and call vi in quickfix mode

```

The last rule, *occurlrefresh* is an example of how a multi-part project can be supported. Simply type

```
$ make occurlrefresh
```

and make will check the timestamps for *occurl.c*, *occurlsym.cpy* and *occurlrefresh.cbl* and then build up the executable if any of those files have changed compared to timestamp of the binary.

See *Tectonics* for another word to describe building code.

## 27.13 3.13 Do you have a reasonable source code skeleton for OpenCOBOL?

Maybe. Style is a very personal developer choice. OpenCOBOL pays homage to this freedom of choice.

Here is the FIXED form header that this author uses. It includes **ocdoc** lines.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*><* =====
*><*
*><* =====
*><* :Author:
*><* :Date:
*><* :Purpose:
*><* :Tectonics: cobc
*> *****
  identification division.
  program-id. .

  environment division.
  configuration section.

  input-output section.
  file-control.
*>  select
*>  assign to
*>  organization is
*>  .

  data division.
  file section.
*>fd .
*>  01 .

  working-storage section.
  local-storage section.
  linkage section.
  screen section.

*> *****
  procedure division.

```

```
goback.  
end program .  
*><*  
*><* Last Update: dd-Mmm-yyyy
```

Fill in the *program-id* and *end program* to compile. Fill in the ocdoc title for generating documentation. See [What is ocdoc?](#) for more information on (*one method of*) inline documentation.

Here are some templates that can cut and pasted.

Fixed form in lowercase

```
OCOBOL >>SOURCE FORMAT IS FIXED  
*> *****  
*> Author:  
*> Date:  
*> Purpose:  
*> Tectonics: cobb  
*> *****  
identification division.  
program-id. .  
  
environment division.  
configuration section.  
  
input-output section.  
*> file-control.  
*>     select  
*>     assign to  
*>     organization is  
*>     .  
  
data division.  
*> file section.  
*> fd .  
*>     01 .  
  
working-storage section.  
  
local-storage section.  
  
linkage section.  
  
screen section.  
  
*> *****  
procedure division.  
  
goback.  
end program .
```

Fixed form in UPPERCASE

```
OCOBOL >>SOURCE FORMAT IS FIXED  
*****  
* Author:  
* Date:  
* Purpose:  
* Tectonics: cobb  
*****
```



```
IDENTIFICATION DIVISION.
PROGRAM-ID. .
```

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
```

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT
    ASSIGN TO
    ORGANIZATION IS
    .
```

```
DATA DIVISION.
FILE SECTION.
FD .
    01 .
```

```
WORKING-STORAGE SECTION.
```

```
LOCAL-STORAGE SECTION.
```

```
LINKAGE SECTION.
```

```
SCREEN SECTION.
```

```
*****
PROCEDURE DIVISION.
```

```
GOBACK.
END PROGRAM .
```

The OCOBOL “sequence number” can safely be removed. It is there to ensure proper alignment in the browser.

FREE FORM can be compiled with **cobc -free** or use the supported compiler directive:

```
>>SOURCE FORMAT IS FREE
```

the above line must start in column 7 unless **cobc -free** is used.

```
*> ** >>SOURCE FORMAT IS FREE
*> *****
*> Author:
*> Date:
*> Purpose:
*> Tectonics: cobc -free
*> *****
identification division.
program-id. .

environment division.
configuration section.

input-output section.
file-control.
    select
        assign to
        organization is
    .
```

```
data division.
file section.
fd .
    01 .

working-storage section.

local-storage section.

linkage section.

screen section.

procedure division.

goback.
end program .
```

These files can be downloaded from

- headfix.cob
- headfixupper.cob
- headfree.cob

---

**Note:** There are tricks to ensure that FIXED FORMAT source code can be compiled in a FREE FORMAT mode. That includes using free form end of line comments, no sequence numbers, free form DEBUG line directives with the >>D starting in column 5 (so the D ends up in column 7).

---

## 27.14 3.14 Can OpenCOBOL be used to write command line stdin, stdout filters?

Absolutely. It comes down to SELECT name ASSIGN TO KEYBOARD for standard input, and SELECT name ASSIGN TO DISPLAY for standard out.

Below is a skeleton that can be used to write various filters. These programs can be used as command line pipes, or with redirections.

```
$ cat datafile | filter
$ filter <inputfile >outputfile
```

**filter.cob.** You'll want to change the 01-transform paragraph to do all the processing of each record. This skeleton simply copies stdin to stdout, *with a limit of 32K records* so that may need to be changed as well or tests made to ensure the default LINE SEQUENTIAL mode of KEYBOARD and DISPLAY are appropriate for the task at hand.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*><* =====
*><* filter
*><* =====
*><* :Author:    Brian Tiffin
*><* :Date:      20090207
*><* :Purpose:   Standard IO filters
*><* :Tectonics: cobb -x filter.cob
```

```

*> *****
identification division.
program-id. filter.

environment division.
configuration section.

input-output section.
file-control.
    select standard-input assign to keyboard.
    select standard-output assign to display.

data division.
file section.
fd standard-input.
    01 stdin-record      pic x(32768).
fd standard-output.
    01 stdout-record    pic x(32768).

working-storage section.
01 file-status          pic x value space.
   88 end-of-file       value high-value
      when set to false is low-value.

*> *****
procedure division.
main section.
00-main.

perform 01-open

perform 01-read

perform
    until end-of-file
        perform 01-transform
        perform 01-write
        perform 01-read
end-perform
.

00-leave.
perform 01-close
.

goback.
*> end main

support section.
01-open.
open input standard-input
open output standard-output
.

01-read.
read standard-input
    at end set end-of-file to true
end-read

```

```
.
*> All changes here
01-transform.
  move stdin-record to stdout-record
.
*>

01-write.
  write stdout-record end-write
.

01-close.
  close standard-input
  close standard-output
.

end program filter.
*><*
*><* Last Update: dd-Mmm-yyyy
```

## 27.15 3.15 How do you print to printers with OpenCOBOL?

OpenCOBOL and COBOL in general does not directly support printers. That role is delegated to the operating system. Having said that, there are a few ways to get data to a printer.

### 27.15.1 3.15.1 printing with standard out

Writing directly to standard out, as explained in [Can OpenCOBOL be used to write command line stdin, stdout filters?](#) and then simply piping to **lpd** should usually suffice to get text to your printer.

```
$ ./cobprog | lp
$ ./yearend | lp -d $PRESIDENTSPRINTER
```

Don't try the above with the DISPLAY verb; use WRITE TO stdout, with stdout selected and assigned to the **DISPLAY** name.

### 27.15.2 3.15.2 calling the system print

Files can be routed to the printer from a running program with sequences such as

```
CALL "SYSTEM"
  USING "lp os-specific-path-to-file"
  RETURNING status
END-CALL
```

### 27.15.3 3.15.3 print control library calls

And then we open up the field of callable libraries for print support. Below is some template code for sending files to a local CUPS install.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian
*> Date:        10-Aug-2009
*> Purpose:     CUPS quick print
*> Tectonics:   cobb -lcups -x cupscob.cob
*> *****
  identification division.
  program-id. cupscob.

  data division.
  working-storage section.
01 result          usage binary-long.
01 cupsError       usage binary-long.
01 msgPointer      usage pointer.
01 msgBuffer       pic x(1024) based.
01 msgDisplay      pic x(132).

*> *****
  procedure division.
  call "cupsPrintFile"
    using
      "cupsQueue" & x"00"
      "filename.prn" & x"00"
      "OpenCOBOL CUPS interface" & x"00"
    by value 0
    by reference NULL
  returning result
  end-call

  if result equals zero
    call "cupsLastError" returning cupsError end-call
    display "Err: " cupsError end-display

    call "cupsLastErrorString" returning msgPointer end-call
    set address of msgBuffer to msgPointer
    string
      msgBuffer delimited by x"00"
      into msgDisplay
    end-string
    display function trim(msgDisplay) end-display
  else
    display "Job: " result end-display
  end-if

  goback.
  end program cupscob.

```

#### 27.15.4 3.15.4 print to PDF with CUPS

As it turns out, the above code snippet can be used to print directly to a PDF defined cups-pdf printer. By

```
$ apt-get install cups cups-pdf
```

under Debian, you can then

```
call "cupsPrintFile"  
  using  
    "PDFer"      & x"00"  
    "cupscob.cob" & x"00"  
    "cupscob.pdf" & x"00"  
    by value 0  
    by reference NULL  
  returning result  
end-call
```

assuming **PDFer** is a Class or printer with a PDF member. A PDF version of the text in **cupscob.cob** will be placed in **~/PDF/** as **cupscob.pdf**.

Roger While added this wisdom:

Check if your particular distro has cups-pdf in its repository. (eg. Using Yast with Suse).  
If yes, install from there.  
If no, use one of the RPM finders on the web to find a version for your distro.  
eg. [www.rpmfind.com](http://www.rpmfind.com)

The installation of cups-pdf should automatically set up a dummy printer with the name "cups-pdf".  
So you do not actually need to define a class.  
You can print directly to "cups-pdf".  
(Check defined printers with eg. "lpstat -t")

The output file location is dependent on the cups-pdf configuration file normally located at `/etc/cups/cups-pdf.conf`.  
So, eg. on my box the location is defined thus -  
Out `${HOME}/Documents/PDFs`

The code with a little more documentation, in case it turns out to be useful.

```
call "cupsPrintFile"          *> requires -lcups  
  using  
    "cups-pdf"      & x"00"          *> printer class  
    "cupscob.cob"  & x"00"          *> input filename  
    "cupscob.pdf"  & x"00"          *> title  
    by value 0      *> num_options  
    by reference NULL  *> options struct <*>  
  returning result  
  on exception  
    display "hint: use -lcups for cupsPrintFile" end-display  
end-call
```

### 27.15.5 3.15.5 Jim Currey's prtcb1

Jim kindly donated this snippet. One of his earliest efforts establishing a base of OpenCOBOL resources. `prtcb1` produces source code listing with results piped to a printer.

**A few customizations.** This version requires a change to a filename for printer control, location of copybooks, and possible changes to the system `lp` command line.

Stash a print setup string in the file so named. The program prompts for input, output and printer.

Jim pointed out that this was early attempts with OpenCOBOL as a tool to support better in house development, *and was nice enough to let me reprint it.*

## OCOBOL IDENTIFICATION DIVISION.

PROGRAM-ID. PRTCBL.

\*AUTHOR. J C CURREY.

```

*****
*          PRINTS A COBOL SOURCE FILE WITH IT'S COPY BOOKS          *
*                                                                    *
*          VERSION 001--ORIGINAL VERSION                            *
*              3/26/2009--J C CURREY                                *
*                                                                    *
*          002--ADDS .CPY (CAPS) IF .cpy FAILS TO FIND             *
*              FILE AND EXPANDS INPUT TO 132 CHARACTERS*          *
*              4/09/2009--J C CURREY                                *
*                                                                    *
*          003--ADDS NOLIST AND LIST SUPPORT (NOTE NOT             *
*              SUPPORTED BY OPENCOBOL COMPILER)                    *
*              **NOLIST IN COL 7-14 TURNS OFF LISTING              *
*              **LIST IN COL 7-12 TURNS ON LISTING                 *
*              4/22/2009--J C CURREY                                *
*                                                                    *
*          004--ADDS SUPPORT FOR /testing-set-1/copybooks          *
*              Copybooks are searched for first in the              *
*              local directory and if not found, then in           *
*              /testing-set-1/copybooks                             *
*              5/7/2009--J C CURREY                                *
*                                                                    *
*          005--CORRECTS MISSING LINE ISSUE ON PAGE BREAKS*      *
*              IN THE COPY FILE PRINTING SECTION.                  *
*              1285451--SANDY DOSS                                  *
*              06/19/2009--JEREMY MONTOYA                          *
*                                                                    *
*          006--USES EXTERNAL PCL CODE FILE TO INSERT PCL          *
*              CODE INTO PRINT FILE FOR FORMATTING.                *
*              1330505--JIM CURREY                                  *
*              12/14/2009--PETE MCTHOMPSON                         *
*****

```

## ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

```

121409 SELECT FORMAT-FILE ASSIGN TO WS-NAME-FORMAT-FILE
121409 ORGANIZATION IS LINE SEQUENTIAL.
SELECT PRINT-FILE ASSIGN TO WS-NAME-PRINT-FILE
ORGANIZATION IS LINE SEQUENTIAL.
SELECT INPUT-FILE ASSIGN TO WS-NAME-INPUT-FILE
ORGANIZATION IS LINE SEQUENTIAL
FILE STATUS IS WS-INPUT-FILE-STATUS.
SELECT COPY-FILE ASSIGN TO WS-NAME-COPY-FILE
ORGANIZATION IS LINE SEQUENTIAL
FILE STATUS IS WS-COPY-FILE-STATUS.

```

DATA DIVISION.

FILE SECTION.

```

*
FD PRINT-FILE.
121409 01 FORMAT-LINE PIC X(140) .
01 PRINT-LINE.
05 OR-LINE-NUMBER PIC Z(6) .
05 OR-FILLER-1 PIC XX.
05 OR-TEXT PIC X(132) .
121409*

```

```

121409 FD  FORMAT-FILE.
121409 01  FORMAT-RECORD          PIC X(140) .
*
FD  INPUT-FILE.
01  INPUT-RECORD.
    05  IR-BUFFER              PIC X(132) .

FD  COPY-FILE.
01  COPY-RECORD.
    05  CR-BUFFER              PIC X(132) .
**NOLIST
* THIS IS ANOTHER LINE
**LIST
*
WORKING-STORAGE SECTION.
*****
*  CONSTANTS, COUNTERS AND WORK AREAS  *
*****
01  WS-NAME-PROGRAM          PIC X(12) VALUE
121409                               "prtcbl  006".
01  WS-NO-PARAGRAPH          PIC S9(4) COMP.
01  WS-I                      PIC S9(4) COMP.
01  WS-J                      PIC S9(4) COMP.
01  WS-K                      PIC S9(4) COMP.
01  WS-NAME-PRINT-FILE       PIC X(64) VALUE SPACES.
01  WS-NAME-INPUT-FILE       PIC X(64) VALUE SPACES.
01  WS-INPUT-FILE-STATUS     PIC XX VALUE "00".
050709 01  WS-NAME-COPY-FILE   PIC X(128) VALUE SPACES.
050709 01  WS-HOLD-NAME-COPY-FILE PIC X(128) VALUE SPACES.
121409 01  WS-NAME-FORMAT-FILE  PIC X(128) VALUE SPACES.
01  WS-COPY-FILE-STATUS      PIC XX VALUE "00".
01  WS-LINE-PRINTER-NAME     PIC X(16) VALUE SPACES.
01  WS-LINE-NUMBER           PIC S9(6) COMP
                               VALUE ZERO.
01  WS-PAGE-LINE-COUNTER     PIC S9(4) COMP
                               VALUE 999.
01  WS-PAGE-NUMBER           PIC S9(4) COMP
                               VALUE ZERO.
01  WS-PRINT-COMMAND         PIC X(128) .
*
01  WS-ESCAPE-CHARACTER      PIC X VALUE X"1B".
*
01  WS-HEADING-LINE          PIC X(132) .
01  WS-CURRENT-DATE          PIC X(21) .
01  WS-ED4S                  PIC ZZZZ-.
042209 01  WS-SWITCH-PRINT    PIC X VALUE SPACE.
*****
*                               PROCEDURE DIVISION                               *
*****
PROCEDURE DIVISION.
0000-MAIN SECTION.
    PERFORM 1000-INITIALIZATION THRU 1990-EXIT.
    PERFORM 2000-PROCESS THRU 2990-EXIT.
    PERFORM 9000-END-OF-PROGRAM THRU 9990-EXIT.
    STOP RUN.
*****
*                               INITIALIZATION                               *
*****

```



```

1000-INITIALIZATION.
    MOVE 1000 TO WS-NO-PARAGRAPH.
    DISPLAY "I) ", WS-NAME-PROGRAM, " BEGINNING AT--"
        FUNCTION CURRENT-DATE.
1002-GET-INPUT-FILE.
    DISPLAY "A) ENTER INPUT-FILE NAME " WITH NO ADVANCING.
    ACCEPT WS-NAME-INPUT-FILE.
    OPEN INPUT INPUT-FILE.
    IF WS-INPUT-FILE-STATUS IS EQUAL TO 35
        DISPLAY "W) INPUT FILE NOT FOUND"
        GO TO 1002-GET-INPUT-FILE.
    DISPLAY "A) ENTER PRINT-FILE (WORK FILE) NAME "
        WITH NO ADVANCING.
    ACCEPT WS-NAME-PRINT-FILE.
    DISPLAY "A) ENTER PRINTER NAME " WITH NO ADVANCING.
    ACCEPT WS-LINE-PRINTER-NAME.
    OPEN OUTPUT PRINT-FILE.
121409 MOVE "laserjet_113D.txt" TO WS-NAME-FORMAT-FILE.
121409 OPEN INPUT FORMAT-FILE.
121409 1010-OUTPUT-PCL-CODES.
121409 READ FORMAT-FILE NEXT RECORD AT END GO TO 1020-FORMAT-EOF.
121409 MOVE FORMAT-RECORD TO FORMAT-LINE.
121409 WRITE FORMAT-LINE.
121409 GO TO 1010-OUTPUT-PCL-CODES.
121409 1020-FORMAT-EOF.
121409 CLOSE FORMAT-FILE.
1990-EXIT.
    EXIT.
*****
*                               *
*                               *
*****
2000-PROCESS.
    MOVE 2000 TO WS-NO-PARAGRAPH.
    READ INPUT-FILE NEXT RECORD AT END GO TO 2990-EXIT.
    ADD 1 TO WS-LINE-NUMBER.
    IF WS-PAGE-LINE-COUNTER IS GREATER THAN 112
        PERFORM 2800-HEADINGS THRU 2890-EXIT.
    MOVE WS-LINE-NUMBER TO OR-LINE-NUMBER.
    MOVE SPACES TO OR-FILLER-1.
    MOVE INPUT-RECORD TO OR-TEXT.
042209 IF IR-BUFFER (7:6) IS EQUAL TO "**LIST"
042209     MOVE "Y" TO WS-SWITCH-PRINT.
042209 IF WS-SWITCH-PRINT IS EQUAL TO "N"
042209     THEN NEXT SENTENCE
042209     ELSE WRITE PRINT-LINE
042209     ADD 1 TO WS-PAGE-LINE-COUNTER.
042209 IF IR-BUFFER (7:8) IS EQUAL TO "**NOLIST"
042209     MOVE "N" TO WS-SWITCH-PRINT.
    IF IR-BUFFER (7:1) IS EQUAL TO "*" GO TO 2000-PROCESS.
    MOVE 1 TO WS-I.
2010-COMPARE-LOOP.
    IF IR-BUFFER (WS-I:2) IS EQUAL TO ">" GO TO 2090-ENDER.
    IF IR-BUFFER (WS-I:6) IS EQUAL TO " COPY " GO TO 2020-COPY.
    ADD 1 TO WS-I.
    IF WS-I IS LESS THAN 73 GO TO 2010-COMPARE-LOOP.
    GO TO 2000-PROCESS.
2020-COPY.
    SUBTRACT 1 FROM WS-LINE-NUMBER.

```

```

ADD 6 TO WS-I.
MOVE 1 TO WS-J.
MOVE SPACES TO WS-NAME-COPY-FILE.
2022-MOVE-LOOP.
IF IR-BUFFER (WS-I:1) IS EQUAL TO SPACE
GO TO 2030-OPEN-COPYFILE.
IF IR-BUFFER (WS-I:1) IS EQUAL TO "."
MOVE ".cpy" to WS-NAME-COPY-FILE (WS-J:4)
GO TO 2030-OPEN-COPYFILE.
MOVE IR-BUFFER (WS-I:1) TO WS-NAME-COPY-FILE (WS-J:1).
ADD 1 TO WS-I, WS-J.
IF WS-I IS GREATER THAN 73
OR WS-J IS GREATER THAN 64
THEN MOVE "**PROBLEM WITH.COPY STATEMENT ABOVE**"
TO OR-TEXT
WRITE PRINT-LINE
ADD 1 TO WS-PAGE-LINE-COUNTER
GO TO 2000-PROCESS.
GO TO 2022-MOVE-LOOP.
2030-OPEN-COPYFILE.
OPEN INPUT COPY-FILE.
IF WS-COPY-FILE-STATUS IS NOT EQUAL TO "00"
040909 MOVE ".CPY" TO WS-NAME-COPY-FILE (WS-J:4)
040909 OPEN INPUT COPY-FILE
040909 IF WS-COPY-FILE-STATUS IS NOT EQUAL TO "00"
050709 MOVE WS-NAME-COPY-FILE TO WS-HOLD-NAME-COPY-FILE
050709 STRING "/testing-set-1/copybooks/"
050709 WS-HOLD-NAME-COPY-FILE
050709 INTO WS-NAME-COPY-FILE
* DISPLAY "D) AT.COPY FILE OPEN NAME=", WS-NAME-COPY-FILE, "\"
050709 OPEN INPUT COPY-FILE
050709 IF WS-COPY-FILE-STATUS IS NOT EQUAL TO "00"
050709 ADD 25 TO WS-J
050709 MOVE ".cpy" TO WS-NAME-COPY-FILE (WS-J:4)
* DISPLAY "D) AT.COPY FILE OPEN NAME=", WS-NAME-COPY-FILE, "\"
050709 OPEN INPUT COPY-FILE
050709 IF WS-COPY-FILE-STATUS IS NOT EQUAL TO "00"
050709 MOVE "***COPY FILE ABOVE NOT FOUND***" TO OR-TEXT
050709 WRITE PRINT-LINE
050709 ADD 1 TO WS-LINE-NUMBER
050709 ADD 1 TO WS-PAGE-LINE-COUNTER
050709 GO TO 2000-PROCESS
050709 END-IF
050709 END-IF
040909 END-IF
040909 END-IF.
2032-PRINT-LOOP.
READ COPY-FILE NEXT RECORD AT END GO TO 2039-EOF.
ADD 1 TO WS-LINE-NUMBER.
061909* MOVE WS-LINE-NUMBER TO OR-LINE-NUMBER.
061909* MOVE SPACES TO OR-FILLER-1.
061909* MOVE COPY-RECORD TO OR-TEXT.
IF WS-PAGE-LINE-COUNTER IS GREATER THAN 112
PERFORM 2800-HEADINGS THRU 2890-EXIT.
061909 MOVE WS-LINE-NUMBER TO OR-LINE-NUMBER.
061909 MOVE SPACES TO OR-FILLER-1.
061909 MOVE COPY-RECORD TO OR-TEXT.
042209 IF CR-BUFFER (7:6) IS EQUAL TO "**LIST"

```

```

042209     MOVE "Y" TO WS-SWITCH-PRINT.
042209     IF WS-SWITCH-PRINT IS EQUAL TO "N"
042209     THEN NEXT SENTENCE
042209     ELSE WRITE PRINT-LINE
042209     ADD 1 TO WS-PAGE-LINE-COUNTER.
042209     IF CR-BUFFER (7:8) IS EQUAL TO "**NOLIST"
042209     MOVE "N" TO WS-SWITCH-PRINT.
GO TO 2032-PRINT-LOOP.
2039-EOF.
CLOSE COPY-FILE.
042209     MOVE "Y" TO WS-SWITCH-PRINT.
2090-ENDER.
GO TO 2000-PROCESS.
*
* PAGE HEADINGS
*
2800-HEADINGS.
INITIALIZE PRINT-LINE.
ADD 1 TO WS-PAGE-NUMBER.
MOVE FUNCTION CURRENT-DATE TO WS-CURRENT-DATE.
MOVE WS-NAME-INPUT-FILE TO PRINT-LINE.
MOVE WS-PAGE-NUMBER TO WS-ED4S.
MOVE "PAGE" TO PRINT-LINE (66:4).
MOVE WS-ED4S TO PRINT-LINE (71:4).
MOVE WS-CURRENT-DATE (5:2) TO PRINT-LINE (80:2).
MOVE "/" TO PRINT-LINE (82:1).
MOVE WS-CURRENT-DATE (7:2) TO PRINT-LINE (83:2).
MOVE "/" TO PRINT-LINE (85:1).
MOVE WS-CURRENT-DATE (1:4) TO PRINT-LINE (86:4).
MOVE WS-CURRENT-DATE (9:2) TO PRINT-LINE (92:2).
MOVE ":" TO PRINT-LINE (94:1).
MOVE WS-CURRENT-DATE (11:2) TO PRINT-LINE (95:2).
MOVE ":" TO PRINT-LINE (97:1).
MOVE WS-CURRENT-DATE (13:2) TO PRINT-LINE (98:2).
IF WS-PAGE-NUMBER IS EQUAL TO 1
THEN WRITE PRINT-LINE
ELSE WRITE PRINT-LINE AFTER ADVANCING PAGE.
INITIALIZE PRINT-LINE.
WRITE PRINT-LINE.
MOVE 4 TO WS-PAGE-LINE-COUNTER.
2890-EXIT.
EXIT.
*
* END OF JOB
*
2990-EXIT.
EXIT.
*****
* TERMINATION *
*****
9000-END-OF-PROGRAM.
MOVE 9000 TO WS-NO-PARAGRAPH.
CLOSE INPUT-FILE.
CLOSE PRINT-FILE.
121409* STRING "lp -d " DELIMITED BY SIZE,
121409* WS-LINE-PRINTER-NAME DELIMITED BY SIZE,
121409* "-o sides=two-sided-long-edge " DELIMITED BY SIZE,
121409* "-o lpi=11 -o cpi=18 -o page-left=34 " DELIMITED BY SIZE,

```

```
121409*      WS-NAME-PRINT-FILE DELIMITED BY SIZE
121409*      INTO WS-PRINT-COMMAND.
STRING "lp -d " DELIMITED BY SIZE,
      WS-LINE-PRINTER-NAME DELIMITED BY SIZE,
      "-o raw " DELIMITED BY SIZE,
      WS-NAME-PRINT-FILE DELIMITED BY SIZE
      INTO WS-PRINT-COMMAND.
CALL "SYSTEM" USING WS-PRINT-COMMAND.
DISPLAY "I) " WS-NAME-PROGRAM " COMPLETED NORMALLY AT--"
      FUNCTION CURRENT-DATE.
9990-EXIT.
EXIT.
```

## 27.16 3.16 Can I run background processes using OpenCOBOL?

Absolutely. Using the CALL “SYSTEM” service. Some care must be shown to properly detach the input output handles, and to instruct the processes to ignore hangup signals along with the “run in a background subshell” control.

```
CALL "SYSTEM"
  USING
    "nohup whatever 0</dev/null 1>mystdout 2>mystderr &"
  RETURNING result
END-CALL
```

runs **whatever** in the background, detaches stdin, sends standard output to the file **mystdout** and standard error to **mystderr**.

*The above example is for POSIX\_shell operating systems. As always, the commands sent through SYSTEM are VERY operating system dependent.*

## 27.17 3.17 Is there OpenCOBOL API documentation?

Absolutely. Sort of. And it’s beautiful, complete and awe inspiring.

Dimitri van Heesch’s 1.7.4 release of Doxygen, <http://www.doxygen.org> was used to produce <http://opencobol.add1tocobol.com/doxy/> and along with Gary’s OCic.cbl <http://opencobol.add1tocobol.com/doxyapp/> to highlight the absolutely beautiful compiler and application documentation available for OpenCOBOL now. These pages were produced with very little effort with only a few small tweaks to the Doxygen generated Doxyfile (to turn on all files, and to generate call graphs). The sample pass produces a 1400 page beauty of a reference manual in PDF generated from the Doxygen LaTeX output. *2950 pages for the sample application run.*

OpenCOBOL ships as a developer tarball and Doxygen was let loose on the source tree after a ./configure and make pass. When the -C output of Gary Cutler’s OCic.clb was placed into the tree, the output includes the call graphs that exercise some of the OpenCOBOL runtime library. This application level documentation is world class.

Regarding the above “sort of”. This was a near effortless use of Doxygen. OpenCOBOL was not touched and the sources have no explicit Doxygen tags. It also excludes many of the automake, libtool, bison and flex source files. Even still, beautiful. The compiler API is now an easy grok, and application level documentation (doxyapp using OCic.cbl as a sample) should satisfy the world’s most ruthless code auditor and meticulous development team lead.

See [http://opencobol.add1tocobol.com/doxy/d2/dd4/structcb\\_\\_field.html](http://opencobol.add1tocobol.com/doxy/d2/dd4/structcb__field.html) for a tantalizing sample of cb\_field collaboration diagram and completeness of source code coverage. See [http://opencobol.add1tocobol.com/doxyapp/d4/da8/OCic\\_8c.html](http://opencobol.add1tocobol.com/doxyapp/d4/da8/OCic_8c.html) for a view of how Doxygen handles the application level documentation. All for free.

## 27.18 3.18 How do I use LD\_RUN\_PATH with OpenCOBOL?

LD\_RUN\_PATH can be a saving grace for developers that want to build OpenCOBOL on hosted environments. LD\_RUN\_PATH is similar to LD\_LIBRARY\_PATH but builds the shared library path into **cobc** and then all of the binaries *compiled* with **cobc**. That means you can cherry pick the link loader paths when you build OpenCOBOL in a way that can add support for unsupported host features.

If you want a recent version of **cobc** on your hosting service, but don't have root permissions, you can build it into one of your own directories then `EXPORT LD_RUN_PATH=mylibdir BEFORE you ./configure ; make ; make install` your OpenCOBOL. All compiles with **cobc** will now include `mylibdir` during compiles, and better yet, the binaries produced will also include `mylibdir` in the search path at runtime.

If you don't have `RECORD_PATH` in your **cobc** then you can simply compile with

```
LD_RUN_PATH=mylibdir cobc -x nextbigthing.cob
```

to achieve similar results.

With the CGI interface, see [How do I use OpenCOBOL for CGI?](#), you can now build up a complete web side solution using OpenCOBOL with little worry about being stuck on link library dependencies or running scripts to setup any path variables before safely using your `cgi-bin` binaries.

LD\_RUN\_PATH is magical. It also avoids many security problems that can occur if you rely on LD\_LIBRARY\_PATH user environment settings. Your **cobc** will have **your** search path and not some `/home/badusers` trickery settings as LD\_RUN\_PATH searches come BEFORE LD\_LIBRARY\_PATH. Relying on LD\_LIBRARY\_PATH is deemed a **Don't do** by some experts. LD\_RUN\_PATH is a much safer bet.

## 27.19 3.19 What GNU build tool options are available when building OpenCOBOL?

The sources for the OpenCOBOL compiler follows GNU standards whenever possible. This includes being built around the GNU build system.

### 27.19.1 3.19.1 Basics

From an end-user perspective, what this means is that the source code distributions follow these basic steps:

```
tar xvf open-cobol-1.1.tar.gz
cd open-cobol-1.1
./configure
make
make check
sudo make install
sudo ldconfig
```

But that is just scratching the surface of the possibilities. See [What are the configure options available for building OpenCOBOL?](#) for the first steps with `./configure`.

### 27.19.2 3.19.2 Out of tree builds

Next up, OpenCOBOL fully supports out-of-source-tree builds.

From Roger:

I mentioned in the past the preferred way of doing a configure/build ie. Out-of-source-tree build.

eg.

We have OC 2.0 in /home/open-cobol-2.0

We want to test -

OC with BDB

OC with vbisam

OC without db (ISAM)

```
mkdir /home/oc20110710bdb
cd /home/oc20110710bdb
/home/open-cobol-2.0/configure --enable-debug
make
make check
cd tests
cd cobol85
# <Get newcob.val - per README>
make test
```

```
mkdir /home/oc20110710vbisam
cd /home/oc20110710vbisam
/home/open-cobol-2.0/configure --enable-debug --with-vbisam
make
make check
cd tests
cd cobol85
# <Get newcob.val - per README>
make test
```

```
mkdir /home/oc20110710nodb
cd /home/oc20110710nodb
/home/open-cobol-2.0/configure --enable-debug --without-db
make
make check
cd tests
cd cobol85
# <Get newcob.val - per README>
make test
```

For the last example both the OC and ANSI85 tests have been adjusted to cater for lack of ISAM functionality.

To set your current environment to compile/execute from any of the above (ie. without doing a "make install" from any directory), then either "source" or execute as part of current environment (with . ) the following files from the build directory -  
tests/atconfig  
tests/atlocal

(Note in that order)

So eg.

```
./home/oc20110710vbisam/tests/atconfig
```

```
./home/oc20110710vbisam/tests/atlocal
```

will set compiler/runtime to this environment in the current shell.

Note that both the OC tests and the ANSI85 tests do this internally (Fairly obvious otherwise we would not be testing the right thing).

Of course, from any of the above example directories you can do a final "make install".

### 27.19.3 3.19.3 Autotest options

By developing the OpenCOBOL system around the GNU build tools, developers receive a great many options *for free*. **make check** can include TESTSUITEFLAGS.

The TESTSUITEFLAGS allows for options that include:

- `make check TESTSUITEFLAGS="--list"` to list the available tests and descriptions
- `--verbose` to show a little more information during the tests
- `--jobs=n` to run n tests in parallel. On multi core systems, the speed up is fairly dramatic. For 425 tests, normally 1 minute 22 seconds, `--jobs=4` ran in 36 seconds (on a small little AMD Athlon(tm) II X2 215 Processor). The more cores, the more dramatic the improvement.





## 4 RESERVED WORDS

.. sidebar:: COBOL Reserved Words

- 4.1 What are the OpenCOBOL RESERVED WORDS?
- 4.2 Does OpenCOBOL implement any Intrinsic FUNCTIONs?
- 4.3 Can you clarify the use of FUNCTION in OpenCOBOL?
- 4.4 What is the difference between the LENGTH verb and FUNCTION LENGTH?
- 4.5 What STOCK CALL LIBRARY does OpenCOBOL offer?
- 4.6 What are the XF4, XF5, and X91 routines?
- 4.7 What is CBL\_OC\_NANOSLEEP OpenCOBOL library routine?
- 4.8 How do you use C\$JUSTIFY?
- 4.9 What compiler directives are supported by OpenCOBOL?

COBOL Reserved Words

### 28.1 4.1 What are the OpenCOBOL RESERVED WORDS?

COBOL is a reserved word rich language. The OpenCOBOL compiler recognizes:

## Reserved Words

- 4.1.1 ACCEPT
- 4.1.2 ACCESS
- 4.1.3 ACTIVE-CLASS
- 4.1.4 ADD
- 4.1.5 ADDRESS
- 4.1.6 ADVANCING
- 4.1.7 AFTER
- 4.1.8 ALIGNED
- 4.1.9 ALL
- 4.1.10 ALLOCATE
- 4.1.11 ALPHABET
- 4.1.12 ALPHABETIC
- 4.1.13 ALPHABETIC-LOWER
- 4.1.14 ALPHABETIC-UPPER
- 4.1.15 ALPHANUMERIC
- 4.1.16 ALPHANUMERIC-EDITED
- 4.1.17 ALSO
- 4.1.18 ALTER
- 4.1.19 ALTERNATE
- 4.1.20 AND
- 4.1.21 ANY
- 4.1.22 ANYCASE
- 4.1.23 ARE
- 4.1.24 AREA
- 4.1.25 AREAS
- 4.1.26 ARGUMENT-NUMBER
- 4.1.27 ARGUMENT-VALUE
- 4.1.28 ARITHMETIC
- 4.1.29 AS
- 4.1.30 ASCENDING
- 4.1.31 ASSIGN
- 4.1.32 AT
- 4.1.33 ATTRIBUTE
- 4.1.34 AUTO
- 4.1.35 AUTO-SKIP
- 4.1.36 AUTOMATIC
- 4.1.37 AUTOTERMINATE
- 4.1.38 B-AND
- 4.1.39 B-NOT
- 4.1.40 B-OR
- 4.1.41 B-XOR
- 4.1.42 BACKGROUND-COLOR
- 4.1.43 BASED
- 4.1.44 BEEP
- 4.1.45 BEFORE
- 4.1.46 BELL
- 4.1.47 BINARY
- 4.1.48 BINARY-C-LONG
- 4.1.49 BINARY-CHAR
- 4.1.50 BINARY-DOUBLE
- 4.1.51 BINARY-LONG
- 4.1.52 BINARY-SHORT
- 4.1.53 BIT

- 4.1.54 BLANK
- 4.1.55 BLINK
- 4.1.56 BLOCK
- 4.1.57 BOOLEAN
- 4.1.58 BOTTOM

## Reserved Words

514 words in OC 1.1, 136 of which are marked not yet implemented. 378 functional reserved words, as of August 2008.

### 28.1.1 4.1.1 ACCEPT

Makes data available from the keyboard or operating system to named data items.

```
ACCEPT variable FROM CONSOLE.
```

```
ACCEPT variable FROM ENVIRONMENT "path".
ACCEPT variable FROM COMMAND LINE.
```

```
ACCEPT variable AT 0101.
ACCEPT screen-variable.
```

```
ACCEPT today FROM DATE.
ACCEPT today FROM DATE YYYYMMDD.
```

### 28.1.2 4.1.2 ACCESS

Defines a file's access mode. One of DYNAMIC, RANDOM, or SEQUENTIAL.

```
SELECT filename
  ASSIGN TO "filename.dat"
  ACCESS MODE IS RANDOM
  RELATIVE KEY IS keyfield.
```

### 28.1.3 4.1.3 ACTIVE-CLASS

Not yet implemented. Object COBOL feature.

### 28.1.4 4.1.4 ADD

Sums two or more numerics, with an eye toward financial precision and error detection.

```
ADD 1 TO cobol GIVING OpenCOBOL END-ADD.
```

```
ADD
  a b c d f g h i j k l m n o p q r s t u v w x y z
  GIVING total-of
  ON SIZE ERROR
    PERFORM log-problem
  NOT ON SIZE ERROR
    PERFORM graph-result
END-ADD
```

### 28.1.5 4.1.5 ADDRESS

Allows program access to memory address reference and, under controlled conditions, assignment.

```
SET pointer-variable TO ADDRESS OF linkage-store.  
SET ADDRESS OF based-var TO ADDRESS OF working-var
```

## 28.1.6 4.1.6 ADVANCING

Programmer control of newline output and paging.

```
DISPLAY "Legend: " WITH NO ADVANCING END-DISPLAY.  
WRITE printrecord AFTER ADVANCING PAGE END-WRITE.
```

## 28.1.7 4.1.7 AFTER

Nested PERFORM clause and can influence when loop conditional testing occurs.

```
PERFORM  
  WITH TEST AFTER  
  VARYING variable FROM 1 BY 1  
    UNTIL variable > 10  
  AFTER inner FROM 1 BY 1  
    UNTIL inner > 4  
    DISPLAY variable ", " inner END-DISPLAY  
END-PERFORM.
```

Will display 55 lines of output. 1 to 11 and 1 to 5. Removing the *WITH TEST AFTER* clause would cause 40 lines of output. 1 to 10 and 1 to 4.

## 28.1.8 4.1.8 ALIGNED

Not yet implemented feature that will influence the internal alignment of not yet implemented USAGE BIT fields.

## 28.1.9 4.1.9 ALL

A multipurpose reserved in context word.

```
INSPECT variable REPLACING ALL "123" WITH "456".  
MOVE ALL QUOTES TO var.
```

## 28.1.10 4.1.10 ALLOCATE

Allocates actual working storage for a BASED element.

```
ALLOCATE based-var INITIALIZED RETURNING pointer-var.
```

## 28.1.11 4.1.11 ALPHABET

```
* Set up for a mixed case SORT COLLATING SEQUENCE IS  
CONFIGURATION SECTION.  
SPECIAL-NAMES.  
  ALPHABET name IS "AaBbCcDdEe..".
```

## 28.1.12 4.1.12 ALPHABETIC

One of the OpenCOBOL data class (*category*) tests.

```

IF variable IS ALPHABETIC
    DISPLAY "alphabetic" END-DISPLAY
END-IF

```

## 28.1.13 4.1.13 ALPHABETIC-LOWER

One of the OpenCOBOL data class (*category*) tests.

```

IF variable IS ALPHABETIC-LOWER
    DISPLAY "alphabetic-lower" END-DISPLAY
END-IF

```

## 28.1.14 4.1.14 ALPHABETIC-UPPER

One of the OpenCOBOL data class (*category*) tests.

```

DISPLAY variable "alphabetic-upper " WITH NO ADVANCING
IF variable IS ALPHABETIC-UPPER
    DISPLAY "true" END-DISPLAY
ELSE
    DISPLAY "false" END-DISPLAY
END-IF

```

## 28.1.15 4.1.15 ALPHANUMERIC

```

INITIALIZE data-record REPLACING ALPHANUMERIC BY literal-value

```

## 28.1.16 4.1.16 ALPHANUMERIC-EDITED

```

INITIALIZE data-record
    REPLACING ALPHANUMERIC-EDITED BY identifier-1

```

## 28.1.17 4.1.17 ALSO

A powerful, multiple conditional expression feature of `EVALUATE`.

```

EVALUATE variable ALSO second-test
    WHEN "A" ALSO 1 THRU 5 PERFORM first-case
    WHEN "A" ALSO 6           PERFORM second-case
    WHEN "A" ALSO 7 THRU 9 PERFORM third-case
    WHEN OTHER                PERFORM invalid-case
END-EVALUATE

```

## 28.1.18 4.1.18 ALTER

Obsolete and unsupported verb that altered the jump target for GO TO statements.

Yeah, just don't.

*Rumour is, 1.1 may support this verb, to increase support for legacy code, and NOT as homage to a good idea. But to be honest, I do look forward to seeing the first OpenCOBOL Flying Spaghetti Monster for the giggles of righteous indignation.*

## 28.1.19 4.1.19 ALTERNATE

Defines an ALTERNATE key for ISAM data structures.

```
SELECT file
  ASSIGN TO filename
  ACCESS MODE IS RANDOM
  RECORD KEY IS key-field
  ALTERNATE KEY IS alt-key WITH DUPLICATES.
```

## 28.1.20 4.1.20 AND

COBOL rules of precedence are; NOT, AND, OR.

```
IF field = "A" AND num = 3
  DISPLAY "got 3" END-DISPLAY
END-IF
```

COBOL also allows abbreviated combined relational conditions.

```
IF NOT (a NOT > b AND c AND NOT d)
  code
END-IF
```

is equivalent to

```
IF NOT (((a NOT > b) AND (a NOT > c)) AND (NOT (a NOT > d)))
  code
END-IF
```

## 28.1.21 4.1.21 ANY

Allows for any value is TRUE in an EVALUATE statement.

```
EVALUATE TRUE ALSO TRUE
  WHEN a > 3 ALSO ANY      *> b can be any value **
  PERFORM a-4-b-any
  WHEN a = 3 ALSO b = 1
  PERFORM a-3-b-1
END-EVALUATE
```

## 28.1.22 4.1.22 ANYCASE

Not yet implemented. Will allow case insensitive match of currency symbols with FUNCTION NUMVAL-C.

### 28.1.23 4.1.23 ARE

Allows for multiple conditional VALUES.

```

01 cond-1    PIC X.
   88 first-truth    VALUES ARE "A" "B" "C".
   88 second-truth  VALUES ARE "X" "Y" "Z".

```

### 28.1.24 4.1.24 AREA

Controls SORT, MERGE and RECORD data definitions.

```

I-O-CONTROL.
    SAME RECORD AREA FOR file1, file2.

```

### 28.1.25 4.1.25 AREAS

Plural readability option for AREA

```

SAME RECORD AREAS

```

### 28.1.26 4.1.26 ARGUMENT-NUMBER

Holds the number of OS parsed command line arguments, and can act as the explicit index when retrieving ARGUMENT-VALUE data. ARGUMENT-NUMBER can be used in ACCEPT FROM and DISPLAY UPON expressions.

```

ACCEPT command-line-argument-count FROM ARGUMENT-NUMBER END-ACCEPT

DISPLAY 2 UPON ARGUMENT-NUMBER END-DISPLAY
ACCEPT indexed-command-line-argument FROM ARGUMENT-VALUE END-ACCEPT

```

See COMMAND-LINE for more information on the unparsed command invocation string.

### 28.1.27 4.1.27 ARGUMENT-VALUE

Returns the next command line argument. This post from John on [opencobol.org](http://opencobol.org) is an excellent idiom for parsing command line arguments without too much worry as to the order.

```

>>source format is free
*>*****
*> Author:   jrls (John Ellis)
*> Date:     Nov-2008
*> Purpose:  command line processing
*>*****
identification division.
program-id. cmdline.
data division.
*>
working-storage section.
*>*****
01 argv          pic x(100) value spaces.
   88 recv       value "-r", "--recv".

```

```
88 email                value "-e", "--email".
88 delivered            value "-d", "--delivered".
01 cmdstatus            pic x    value spaces.
88 lastcmd              value "1".
01 reptinfo.
05 rept-recv           pic x(30) value spaces.
05 rept-howsent        pic x(10) value spaces.
*>
procedure division.
0000-start.
*>
    perform until lastcmd
        move low-values      to argv
        accept argv         from argument-value
        if argv > low-values
            perform 0100-process-arguments
        else
            move "1"         to cmdstatus
        end-if
    end-perform
    display reptinfo.
    stop run.
*>
0100-process-arguments.
*>
    evaluate true
        when recv
            if rept-recv = spaces
                accept rept-recv from argument-value
            else
                display "duplicate " argv
            end-if
        when email
            move "email"      to rept-howsent
        when delivered
            move "delivered"  to rept-howsent
        when other display "invalid switch: " argv
    end-evaluate.
```

Example run:

```
./cmdline --recv "john ellis" -e -f
invalid switch: -f
john ellis                email
```

## 28.1.28 4.1.28 ARITHMETIC

Not yet implemented feature of the not yet implemented `OPTIONS` paragraph of the `IDENTIFICATION DIVISION`.

## 28.1.29 4.1.29 AS

`PROGRAM-ID.` `program-name` `AS` `literal`.



### 28.1.30 4.1.30 ASCENDING

COBOL table suport.

```
01 CLUBTABLE.
   05 MEMBER-DATA OCCURS 1 TO 6000000000 TIMES
      DEPENDING ON PEOPLE
      ASCENDING KEY IS HOURS-DONATED.
```

### 28.1.31 4.1.31 ASSIGN

Assign a name to a file or other external resource.

```
SELECT input-file
ASSIGN TO "filename.ext"
```

The actual filename used is dependent on a configuration setting. Under default configuration settings, **filename-mapping** is set to **yes**.

See [What are the OpenCOBOL compile time configuration files?](#) for details.

```
# If yes, file names are resolved at run time using
# environment variables.
# For example, given ASSIGN TO "DATAFILE", the actual
# file name will be
# 1. the value of environment variable 'DD_DATAFILE' or
# 2. the value of environment variable 'dd_DATAFILE' or
# 3. the value of environment variable 'DATAFILE' or
# 4. the literal "DATAFILE"
# If no, the value of the assign clause is the file name.
#
# Value: 'yes', 'no'
filename-mapping: yes
```

So, under GNU/Linux, bash shell

```
$ export DD_DATAFILE='/tmp/opencobol.dat'
$ ./myprog
```

the program will find the data in **/tmp/opencobol.dat**

```
$ export DD_DATAFILE='/tmp/other.dat'
$ ./myprog
```

this run of the same program will find the data in **/tmp/other.dat**

As shown in the sample .conf comments, the order of environment variable lookup proceeds through three environment variables before using a literal as the filename.

- DD\_DATAFILE
- dd\_DATAFILE
- DATAFILE
- and finally “DATAFILE”

where DATAFILE is the **name** used in

```
ASSIGN TO name
```

and can be any valid COBOL identifier, or string leading to a valid operating system filename, and is not limited to *DATAFILE*.

### 28.1.32 4.1.32 AT

Controls position of ACCEPT and DISPLAY screen oriented verbs.

```
*> Display at line 1, column 4 <*  
DISPLAY "Name:" AT 0104 END-DISPLAY  
*> Accept starting at line 1, column 10 for length of field <*  
ACCEPT name-var AT 0110 END-ACCEPT
```

### 28.1.33 4.1.33 ATTRIBUTE

Not yet implemented, but when it is, it will allow

```
SET screen-name ATTRIBUTE BLINK OFF
```

### 28.1.34 4.1.34 AUTO

Automatic cursor flow to next field in screen section.

### 28.1.35 4.1.35 AUTO-SKIP

Alias for AUTO

### 28.1.36 4.1.36 AUTOMATIC

LOCK MODE IS AUTOMATIC. See [MANUAL](#) and [EXCLUSIVE](#) for more LOCK options.

### 28.1.37 4.1.37 AUTOTERMINATE

Alias for AUTO

### 28.1.38 4.1.38 B-AND

Not yet implemented BIT field operation. See [What STOCK CALL LIBRARY does OpenCOBOL offer? CBL\\_AND](#) for alternatives allowing bitwise operations.

### 28.1.39 4.1.39 B-NOT

Not yet implemented BIT field operation. See [What STOCK CALL LIBRARY does OpenCOBOL offer? CBL\\_NOT](#) for alternatives allowing bitwise operations.

## 28.1.40 4.1.40 B-OR

Not yet implemented BIT field operation. See [What STOCK CALL LIBRARY does OpenCOBOL offer?](#) **CBL\_OR** for alternatives allowing bitwise operations.

For example:

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin
*> Date:      20110626
*> Purpose:   Demonstrate alternative for B-OR
*> Tectonics: ccbc -x bits
*> *****
identification division.
program-id. bits.

data division.
working-storage section.
01 s1 pic 999 usage comp-5.
01 t2 pic 999 usage comp-5.
01 len pic 9.
01 result usage binary-long.

*> *****
procedure division.
move 2 to s1
move 4 to t2
move 1 to len

*> CBL_OR takes source, target and length value 2 OR 4 is 6.  **
call "CBL_OR" using s1 t2 by value len returning result end-call
display s1 space t2 space len space result end-display

goback.
end program bits.
```

giving:

```
$ ccbc -x bits.cob
$ ./bits
002 006 1 +0000000000
```

## 28.1.41 4.1.41 B-XOR

Not yet implemented BIT field operation. See [What STOCK CALL LIBRARY does OpenCOBOL offer?](#) **CBL\_XOR** for alternatives allowing bitwise operations.

## 28.1.42 4.1.42 BACKGROUND-COLOR

```
05 BLANK SCREEN BACKGROUND-COLOR 7 FOREGROUND-COLOR 0.
```

## 28.1.43 4.1.43 BASED

```
01 based-var PIC X(80) BASED.
```

A sample posted by [human]

```
OCOBOL*-----
IDENTIFICATION DIVISION.
PROGRAM-ID. 'MEMALL'.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES. DECIMAL-POINT IS COMMA.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
DATA DIVISION.
FILE SECTION.
*
WORKING-STORAGE SECTION.
*
77 mychar          pic x.
01 REC-TEST BASED.
   03 REC-TEST-PART1 PIC X(5500000) .
   03 REC-TEST-PART2 PIC X(0100000) .
   03 REC-TEST-PART3 PIC X(1200000) .
   03 REC-TEST-PART4 PIC X(1200000) .
   03 REC-TEST-PART5 PIC X(1700000) .
*-----
LINKAGE SECTION.
*-----
PROCEDURE DIVISION.
declaratives.
end declaratives.
*-----
main section.
00.
   FREE ADDRESS OF REC-TEST
   display 'MEMALL loaded and REC-TEST FREED before ALLOCATE'
   accept  mychar
*
   IF ADDRESS OF REC-TEST = NULL
      display 'REC-TEST was not allocated before'
   ELSE
      display 'REC-TEST was allocated before'
   END-IF
   accept  mychar
*
   ALLOCATE REC-TEST
   move all '9' to REC-TEST
   display 'REC-TEST allocated and filled with '
           REC-TEST (1:9)
   end-display
   accept  mychar
*
   IF ADDRESS OF REC-TEST = NULL
      display 'REC-TEST was not allocated before'
      ALLOCATE REC-TEST
      display 'REC-TEST allocated again, filled with '
             REC-TEST (1:9)
      end-display
```

```

ELSE
    display 'REC-TEST was allocated before'
END-IF
accept mychar
*
*
FREE ADDRESS OF REC-TEST
display 'REC-TEST FREEd'
accept mychar
*
stop run
*
continue.
ex. exit program.
*-----
*--- End of program MEMALL -----

```

### 28.1.44 4.1.44 BEEP

Ring the terminal bell during `DISPLAY` output. Alias for `BELL`

```
DISPLAY "Beeeeeep" LINE 3 COLUMN 1 WITH BEEP END-DISPLAY.
```

### 28.1.45 4.1.45 BEFORE

Sets up a `PERFORM` loop to test the conditional `BEFORE` execution of the loop body. See `AFTER` for the alternative. `BEFORE` is the default.

```

MOVE 1 TO counter
PERFORM WITH TEST BEFORE
    UNTIL counter IS GREATER THAN OR EQUAL TO limiter
        CALL "subprogram" USING counter RETURNING result END-CALL
        MOVE result TO answers(counter)
        ADD 1 TO counter END-ADD
END-PERFORM

```

Also used with the `WRITE` verb.

```

WRITE record-name
    BEFORE ADVANCING some-number LINES

```

And to control how the `INSPECT` verb goes about its job.

```

INSPECT character-var TALLYING
    the-count FOR ALL "tests" BEFORE "prefix"

```

And not currently (*February 2013*) supported, in the declaratives for `REPORT SECTION` control.

```

USE BEFORE REPORTING
...

```

### 28.1.46 4.1.46 BELL

Ring the terminal bell during `DISPLAY` output. Alias for `BEEP`

```
DISPLAY "Beeeeeep" LINE 3 COLUMN 1 WITH BELL END-DISPLAY.
```

### 28.1.47 4.1.47 BINARY

```
01 result PIC S9(8) USAGE BINARY
```

### 28.1.48 4.1.48 BINARY-C-LONG

With OpenCOBOL's tight integration with the C Application Binary Interface the compiler authors have built in support that guarantees a native system *C long* value being the same bit size between COBOL and C modules. This increases coverage of the plethora of open C library functions that can be directly used with the `CALL` verb. Including cases where callback functions that require *long* stack parameters (that can't as easily be wrapped in thin C code layers) can now be used more effectively and safely.

### 28.1.49 4.1.49 BINARY-CHAR

Defines an 8 bit usage item.

### 28.1.50 4.1.50 BINARY-DOUBLE

Defines a 64 bit usage item.

### 28.1.51 4.1.51 BINARY-LONG

32 bit native `USAGE` modifier. Equivalent to `S9(8)`.

### 28.1.52 4.1.52 BINARY-SHORT

16 bit native `USAGE`. Equivalent to `S9(5)`.

### 28.1.53 4.1.53 BIT

Not yet implemented. See [What STOCK CALL LIBRARY does OpenCOBOL offer?](#) for alternatives allowing bitwise operations.

### 28.1.54 4.1.54 BLANK

```
05 BLANK SCREEN BACKGROUND-COLOR 7 FOREGROUND-COLOR 0.
```

### 28.1.55 4.1.55 BLINK

Aaaaaah, my eyes!!

## 28.1.56 4.1.56 BLOCK

```
FD file-name
  BLOCK CONTAINS 1 TO n RECORDS
```

## 28.1.57 4.1.57 BOOLEAN

As yet unsupported modifier.

## 28.1.58 4.1.58 BOTTOM

A LINAGE setting.

```
FD mini-report
  linage is 16 lines
  with footing at 15
  lines at top 2
  lines at bottom 2.
```

## 28.1.59 4.1.59 BY

```
PERFORM the-procedure
  VARYING step-counter FROM 1 BY step-size
  UNTIL step-counter > counter-limit
```

## 28.1.60 4.1.60 BYTE-LENGTH

Human incisors average about 16mm.

More to the point, the BYTE-LENGTH returns the length, in bytes, of a data item. See [FUNCTION BYTE-LENGTH](#)

## 28.1.61 4.1.61 CALL

The OpenCOBOL CALL verb accepts literal or identifier stored names when resolving the transfer address. The USING phrase allows argument passing and OpenCOBOL includes internal rules for the data representation of the call stack entities that depend on the COBOL PICTURE and USAGE clauses. Return values are captured with RETURNING identifier. See [What STOCK CALL LIBRARY does OpenCOBOL offer?](#).

For more information see [http://www.opencobol.org/modules/bwiki/index.php?cmd=read&page=UserManual%2F2\\_3#content\\_1\\_0](http://www.opencobol.org/modules/bwiki/index.php?cmd=read&page=UserManual%2F2_3#content_1_0)

CALL is the verb that opens up access to the plethora of C based ABI libraries. A **plethora**, and the standard C library is accessible without explicit linkage as a bonus.

One item of note is C pointers. Especially those passed around as handles. When calling a C routine that returns a handle, the RETURNING identifier will receive a C pointer. To use that handle in later CALLs, the argument from COBOL should usually be passed BY VALUE. This passes the C pointer, not the address of the COBOL identifier as the default BY REFERENCE argument handling would do.

Below is a sample that allows fairly carefree use of CBL\_OC\_DUMP during development. ON EXCEPTION CONTINUE.

```

OCOBOL*>>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20110701
*> Purpose:     Try C library formatted printing, and CALL exception
*> Tectonics:   ccbc -x callon.cob
*>              or ccbc -x callon.cob CBL_OC_DUMP.cob
*> *****
identification division.
program-id. callon.

data division.
working-storage section.
01 result      usage binary-long.

01 pie         usage float-short.
01 stuff      pic x(12) value 'abcdefghijkl'.

*> *****
procedure division.
move 3.141592654 to pie

*> Get a dump of the memory at pie, but don't stop if not linked
call "CBL_OC_DUMP" using pie 4 on exception continue end-call

*> Call C's printf, abort if not available
call static "printf" using
    "float-short: %10.8f" & x"0a00"
    by value pie
    returning result
end-call
display pie space length of pie space result end-display

*> Get a dump of the memory used by stuff, don't stop if no link
call "CBL_OC_DUMP" using stuff 12 on exception continue end-call

*> Get a dump of the memory used by stuff, abort if not linked < *
call "CBL_OC_DUMP" using stuff 12 end-call

goback.
end program callon.

```

See *What is CBL\_OC\_DUMP?* for details of the subprogram.

A runtime session shows:

```

$ ccbc -x callon.cob
$ ./callon
float-short: 3.14159274
3.1415927 4 +0000000024
libcob: Cannot find module 'CBL_OC_DUMP'
$ ccbc -x callon.cob CBL_OC_DUMP.cob
$ ./callon

```

```

Offset  HEX-- -- -- -5 -- -- -- -- 10 -- -- -- -- 15 --    CHARS----1----5--
000000  db 0f 49 40                                     ..I@.....

```

```

float-short: 3.14159274
3.1415927 4 +0000000024

```



```
Offset  HEX-- -- -- -5 -- -- -- -- 10 -- -- -- -- 15 --  CHARS-----1-----5-
000000  61 62 63 64 65 66 67 68 69 6a 6b 6c                      abcdefghijkl....
```

```
Offset  HEX-- -- -- -5 -- -- -- -- 10 -- -- -- -- 15 --  CHARS-----1-----5-
000000  61 62 63 64 65 66 67 68 69 6a 6b 6c                      abcdefghijkl....
```

So, the first CALL to CBL\_OC\_DUMP doesn't 'fail' as the ON EXCEPTION CONTINUE traps the condition and lets the program carry on without a dump displayed. The last CALL does abend the program with 'Cannot find module' when CBL\_OC\_DUMP is not compiled in.

### 28.1.62 4.1.62 CANCEL

Virtual cancel of a module is supported. Physical cancel support is on the development schedule.

### 28.1.63 4.1.63 CD

A control clause of the as yet unsupported COMMUNICATION DIVISION.

### 28.1.64 4.1.64 CENTER

An as yet unsupported keyword.

### 28.1.65 4.1.65 CF

Shortform for CONTROL FOOTING, a clause used in REPORT SECTION.

### 28.1.66 4.1.66 CH

Shortform for CONTROL HEADING, a clause used in PAGE descriptors in the REPORT SECTION.

### 28.1.67 4.1.67 CHAIN

Invokes a subprogram, with no return of control implied. The chained program unit virtually becomes the main program within the run unit.

### 28.1.68 4.1.68 CHAINING

Passes procedure division data through WORKING-STORAGE and can be used for shell command line arguments as well, as in CALL "myprog" USING string END-CALL.

from opencobol.org by human

```
WORKING-STORAGE SECTION.
```

```
  01 cmd-argument.
  02 some-text pic x(256).
```

```
procedure division Chaining cmd-argument.
```

```
display 'You wrote:'
      '>' function trim(some-text) ''
      'from shell command line'
end-display
```

## 28.1.69 4.1.69 CHARACTER

### PADDING CHARACTER IS

A soon to be obsolete feature.

## 28.1.70 4.1.70 CHARACTERS

A multi use keyword.

Used in SPECIAL-NAMES

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20101031
*> Purpose:     Try out SYMBOLIC CHARACTERS
*> Tectonics:   cobc -x figurative.cob
*> Rave:        OpenCOBOL is stone cold cool
*> *****
identification division.
program-id. figurative.

environment division.
configuration section.
special-names.
      symbolic characters TAB is 10
                          LF is 11
                          CMA is 45.

data division.
working-storage section.
01 a-comma pic x(1) value ", ".
01 lots-of-commas pic x(20).

*> *****
procedure division.
display
      "thing" TAB "tabbed thing" LF
      "and" TAB "another tabbed thing" LF
      "other" CMA " things"
end-display

move a-comma to lots-of-commas
display "MOVE a-comma : " lots-of-commas end-display

move CMA to lots-of-commas
display "MOVE symbolic: " lots-of-commas end-display

goback.
end program figurative.
```

Output:

```
$ cobc -x figuratives.cob
$ ./figuratives
thing    tabbed thing
and      another tabbed thing
other,  things
MOVE a-comma : ,
MOVE symbolic: ,,,,,,,,,,,,,,,,,,,,,,
```

Used in INSPECT

```
INSPECT str TALLYING tal FOR CHARACTERS
```

Used in a File Description FD

```
FD file-name
   BLOCK CONTAINS integer-1 TO integer-2 CHARACTERS
   RECORD IS VARYING IN SIZE FROM integer-5 TO integer-6 CHARACTERS
   DEPENDING ON identifier-1.
```

## 28.1.71 4.1.71 CLASS

Used to create alphabets in SPECIAL-NAMES.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
CLASS octals IS '0' THRU '7'.
```

...

```
PROCEDURE DIVISION.
IF user-value IS NOT octals
   DISPLAY "Sorry, not a valid octal number" END-DISPLAY
ELSE
   DISPLAY user-value END-DISPLAY
END-IF
```

## 28.1.72 4.1.72 CLASS-ID

An as yet unsupported Object COBOL class identifier clause.

## 28.1.73 4.1.73 CLASSIFICATION

An as yet unsupported source code internationalization clause.

## 28.1.74 4.1.74 CLOSE

Close an open file. OpenCOBOL will implicitly close all open resources at termination of a run unit and will display a warning message stating so, and the danger of potentially unsafe termination.

```
CLOSE input-file
```

### 28.1.75 4.1.75 CODE

A syntactically recognized, but as yet unsupported clause of a report descriptor, `RD`.

### 28.1.76 4.1.76 CODE-SET

An as yet unsupported data internationalization clause.

### 28.1.77 4.1.77 COL

Alias for `COLUMNS`.

### 28.1.78 4.1.78 COLLATING

Allows definition within a program unit of a character set.

```
OBJECT-COMPUTER. name.  
PROGRAM COLLATING SEQUENCE IS alphabet-1.
```

### 28.1.79 4.1.79 COLS

Alias for `COLUMNS`.

### 28.1.80 4.1.80 COLUMN

1. A recognized but unsupported REPORT SECTION `RD` descriptor clause.
2. Also used for positional DISPLAY and ACCEPT, which implicitly uses SCREEN SECTION style screen IO.

```
DISPLAY var-1 LINE 1 COLUMN 23 END-DISPLAY
```

### 28.1.81 4.1.81 COLUMNS

A recognized but as yet unsupported `RD` clause.

### 28.1.82 4.1.82 COMMA

A `SPECIAL-NAMES` clause supporting commas in numeric values versus the default period decimal point. COBOL was way ahead of the internationalization curve, *and this feature has caused compiler writers no little grief in its time, a challenge they rise to and deal with for the world's benefit.*

```
DECIMAL POINT IS COMMA
```

### 28.1.83 4.1.83 COMMAND-LINE

Provides access to command line arguments.

```
ACCEPT the-args FROM COMMAND-LINE END-ACCEPT
```

### 28.1.84 4.1.84 COMMIT

Flushes ALL current locks, synching file I/O buffers. OpenCOBOL supports safe transactional processing with ROLL-BACK capabilities. *Assuming the ISAM handler configured when building the compiler can support LOCK\_*

### 28.1.85 4.1.85 COMMON

```
PROGRAM-ID. CBL_OC_PROGRAM IS COMMON PROGRAM.
```

Ensures a nested sub-program is also available to other nested sub-programs with a program unit heirarchy.

### 28.1.86 4.1.86 COMMUNICATION

currently (*February 2013*) unsupported DIVISION, but see [Does OpenCOBOL support Message Queues?](#) for an alternative.

### 28.1.87 4.1.87 COMP

See COMPUTATIONAL

### 28.1.88 4.1.88 COMP-1

See COMPUTATIONAL-1

### 28.1.89 4.1.89 COMP-2

See COMPUTATIONAL-2

### 28.1.90 4.1.90 COMP-3

See COMPUTATIONAL-3

### 28.1.91 4.1.91 COMP-4

See COMPUTATIONAL-4

### 28.1.92 4.1.92 COMP-5

See COMPUTATIONAL-5

### 28.1.93 4.1.93 COMP-X

See COMPUTATIONAL-X

### 28.1.94 4.1.94 COMPUTATIONAL

Implementors choice; OpenCOBOL is a big-endian default. With most Intel personal computers and operating systems like GNU/Linux, COMPUTATIONAL-5 will run faster.

### 28.1.95 4.1.95 COMPUTATIONAL-1

Single precision float. Equivalent to FLOAT-SHORT.

### 28.1.96 4.1.96 COMPUTATIONAL-2

Double precision float. Equivalent to FLOAT-LONG.

### 28.1.97 4.1.97 COMPUTATIONAL-3

Equivalent to PACKED DECIMAL. Packed decimal is two digits per byte, always sign extended and influenced by a .conf setting *binary-size* COMPUTATIONAL-6 is UNSIGNED PACKED.

### 28.1.98 4.1.98 COMPUTATIONAL-4

Equivalent to BINARY.

### 28.1.99 4.1.99 COMPUTATIONAL-5

Native form.

### 28.1.100 4.1.100 COMPUTATIONAL-6

Unsigned packed decimal form, see COMPUTATIONAL-3.

### 28.1.101 4.1.101 COMPUTATIONAL-X

Native form.

### 28.1.102 4.1.102 COMPUTE

Computational arithmetic.

```
COMPUTE circular-area = radius ** 2 * FUNCTION PI END-COMPUTE
```

OpenCOBOL supports the normal gamut of arithmetic expressions.

- Add +

- Subtract -
- Multiply \*
- Divide /
- Raise to power \*\*

Order of precedence rules apply.

1. unary minus, unary plus
2. exponentiation
3. multiplication, division
4. addition, subtraction

### Spaces and expressions

Due to COBOL allowing *dash* in user names, care must be taken to properly space arithmetic expressions.

Some examples of seemingly ambiguous and potentially dangerous code

```
OCOBOL*> *****
identification division.
program-id. computing.

data division.
working-storage section.
01 answer pic s9(8).
01 var    pic s9(8).

*> *****
procedure division.
compute answer = 3*var-1 end-compute

goback.
end program computing.
```

That is NOT three times var *minus one*, OpenCOBOL will complain.

```
$ cobc -x computing.cob
computing.cob:18: Error: 'var-1' is not defined
```

whew, saved!

```
OCOBOL*> *****
identification division.
program-id. computing.

data division.
working-storage section.
01 answer pic s9(8).
01 var    pic s9(8).
01 var-1  pic s9(8).

*> *****
procedure division.
compute answer = 3*var-1 end-compute

goback.
end program computing.
```

With the above source, the compile will succeed.

```
$ cobc -x computing.cob
```

OpenCOBOL will (properly, according to standard) compile this as **three times var-1**. Not saved, if you meant 3 times var minus 1.

*OpenCOBOL programmers are strongly encouraged to use full spacing inside COMPUTE statements.*

```
OCOBOL*> *****
  identification division.
  program-id. computing.

  data division.
  working-storage section.
  01 answer pic s9(8).
  01 var    pic s9(8).
  01 var-1  pic s9(8).

*> *****
  procedure division.
  compute
    answer = 3 * var - 1
    on size error
      display "Problem, call the ghost busters" end-display
    not on size error
      display "All good, answer is trustworthy" end-display
  end-compute

  goback.
  end program computing.
```

COMPUTE supports ON SIZE ERROR, NOT ON SIZE ERROR imperatives for safety, and the ROUNDED modifier for bankers.

### 28.1.103 4.1.103 CONDITION

As yet unsupported USE AFTER EXCEPTION CONDITION clause.

### 28.1.104 4.1.104 CONFIGURATION

A SECTION of the ENVIRONMENT DIVISION. Holds paragraphs for

- SOURCE-COMPUTER
- OBJECT-COMPUTER
- REPOSITORY
- SPECIAL-NAMES

### 28.1.105 4.1.105 CONSTANT

An extension allowing constant definitions

```
01 enumerated-value CONSTANT AS 500.
```



## 28.1.106 4.1.106 CONTAINS

An FD clause:

```
FD a-file RECORD CONTAINS 80 CHARACTERS.
```

## 28.1.107 4.1.107 CONTENT

A CALL clause that controls how arguments are passed and expected.

```
CALL "subprog" USING BY CONTENT alpha-var.
```

alpha-var will not be modifiable by subprog as a copy is passed.

See REFERENCE and VALUE for the other supported CALL argument control.

## 28.1.108 4.1.108 CONTINUE

A placeholder, no operation verb.

```
if action-flag = "C" or "R" or "U" or "D"  
  continue  
else  
  display "invalid action-code" end-display  
end-if
```

## 28.1.109 4.1.109 CONTROL

As yet unsupported REPORT SECTION clause for setting control break data fields.

## 28.1.110 4.1.110 CONTROLS

As yet unsupported REPORT SECTION clause for setting control break data fields.

## 28.1.111 4.1.111 CONVERTING

A clause of the INSPECT verb.

```
INSPECT X CONVERTING "012345678" TO "999999999".
```

## 28.1.112 4.1.112 COPY

The COBOL include pre-processor verb. Also see REPLACE and Does OpenCOBOL support COPY includes?.

## 28.1.113 4.1.113 CORR

Alias for CORRESPONDING.

## 28.1.114 4.1.114 CORRESPONDING

Move any and all sub fields with matching names within records.

```
01 bin-record.  
   05 first-will usage binary-short.  
   05 second-will usage binary-long.  
   05 this-wont-move usage binary-long.  
   05 third-will usage binary-short.  
01 num-record.  
   05 first-will pic 999.  
   05 second-will pic s9(9).  
   05 third-will pic 999.  
   05 this-doesnt-match pic s9(9).  
  
move corresponding bin-record to num-record  
display  
   first-will in num-record  
   second-will in num-record  
   third-will in num-record  
end-display
```

## 28.1.115 4.1.115 COUNT

Sets the count of characters set in an UNSTRING substring.

From the OpenCOBOL Programmer's Guide's UNSTRING entry.

```
UNSTRING Input-Address  
  DELIMITED BY ", " OR "/"  
  INTO  
    Street-Address DELIMITER D1 COUNT C1  
    Apt-Number DELIMITER D2 COUNT C2  
    City DELIMITER D3 COUNT C3  
    State DELIMITER D4 COUNT C4  
    Zip-Code DELIMITER D5 COUNT C5  
END-UNSTRING
```

## 28.1.116 4.1.116 CRT

```
SPECIAL-NAMES.  
  CONSOLE IS CRT  
  CRT STATUS is identifier-1.
```

CONSOLE IS CRT allows "CRT" and "CONSOLE" to be used interchangeably on DISPLAY but this is a default for newer OpenCOBOL implementations.

CRT STATUS IS establishes a PIC 9(4) field for screen ACCEPT status codes. There is also an implicit COB-CRT-STATUS register defined for all programs, that will be used if no explicit field is established.

## 28.1.117 4.1.117 CURRENCY

```
SPECIAL-NAMES.  
  CURRENCY SIGN IS literal-1.
```

Default currency sign is the dollar sign "\$".

## 28.1.118 4.1.118 CURSOR

Tracks the line/column location of screen ACCEPT.

**SPECIAL-NAMES.**

```
CURSOR IS identifier-2.
```

identifier-2 is to be declared as PIC 9(4) or 9(6). If 4, the field is LLCC. With 9(6) it is LLLCCC where L is line and C is column, zero relative.

## 28.1.119 4.1.119 CYCLE

A clause that causes EXIT PERFORM to return to the top of a loop. See [FOREVER](#) for an example.

## 28.1.120 4.1.120 DATA

A magical [DIVISION](#). One of COBOL's major strength is the rules surrounding the DATA DIVISION and pictorial record definitions.

## 28.1.121 4.1.121 DATA-POINTER

An as yet unsupported Object COBOL feature.

## 28.1.122 4.1.122 DATE

An [ACCEPT](#) source. 6 digit and 8 digit Gregorian dates.

1. ACCEPT ident-1 FROM DATE
2. ACCEPT ident-2 FROM DATE YYYYMMDD

**identification division.**

```
program-id. dates.
```

**data division.**

**working-storage section.**

```
01 date-2nd
   03 date-yy   pic 9(2).
   03 date-mm   pic 9(2).
   03 date-dd   pic 9(2).
01 date-3rd
   03 date-yyyy pic 9(4).
   03 date-mm   pic 9(2).
   03 date-dd   pic 9(2).
```

**procedure division.**

```
accept date-2nd from date end-accept
```

```
*> Just before the 3rd millennium, programmers admitted   <*>
*> that 2 digit year storage was a bad idea and ambiguous <*>
accept date-3rd from date yyyyymmdd end-accept
```

```
display date-2nd space date-3rd end-display
```

```
goback.
end program dates.
```

```
./dates
110701 20110701
```

## 28.1.123 4.1.123 DAY

An **ACCEPT** source. Access the current date in Julian form. Returns yyddd and yyyyddd formats.

1. ACCEPT ident-1 FROM DAY
2. ACCEPT ident-2 FROM DAY YYYYDDD

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        2011182 (July 01)
*> Purpose:     Accept from day in Julian form
*> Tectonics:   cobc -x days.cob
*> *****
identification division.
program-id. days.

data division.
working-storage section.
01 julian-2nd.
   03 julian-yy   pic 9(2).
   03 julian-days pic 9(3).
01 julian-3rd.
   03 julian-yyyy pic 9(4).
   03 julian-days pic 9(3).

procedure division.
accept julian-2nd from day end-accept

*> Just before the 3rd millennium, programmers admitted   <*>
*> that 2 digit year storage was a bad idea and ambiguous  <*>
accept julian-3rd from day yyyyddd end-accept

display julian-2nd space julian-3rd end-display

goback.
end program days.
```

```
$ make days
cobc -W -x days.cob -o days
$ ./days
11182 2011182
```

## 28.1.124 4.1.124 DAY-OF-WEEK

An **ACCEPT** source. Single digit day of week. 1 for Monday, 7 for Sunday.

```
accept the-day from day-of-week
```

## 28.1.125 4.1.125 DE

Report Writer shortcut for DETAIL. Recognized, but not yet implemented. This author found this type of shortcut very unCOBOL, until trying to layout a report, when it made a lot more practical sense *in FIXED form COBOL*.

## 28.1.126 4.1.126 DEBUGGING

A SOURCE-COMPUTER clause and DECLARATIVE phrase.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER mine
    WITH DEBUGGING MODE.
```

DEBUGGING MODE can also be toggled on with the *-fdebugging-line* cobc option, and will compile in 'D' lines.

```
PROCEDURE DIVISION.
DECLARATIVES.
decl-debug section.
    USE FOR DEBUGGING ON ALL PROCEDURES
decl-paragraph.
    DISPLAY "Why is this happening to me?" END-DISPLAY
END DECLARATIVES.
```

USE FOR DEBUGGING sets up a section that is executed when the named section is entered. Powerful. It can also name a file, and the debug section is evaluated after open, close, read, start etc. Identifiers can be also be named and the debug section will trigger when referenced (usually after).

## 28.1.127 4.1.127 DECIMAL-POINT

Allows internationalization for number formatting. In particular

```
IDENTIFICATION DIVISION.
PROGRAM-ID. 'MEMALL'.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES. DECIMAL-POINT IS COMMA.
```

will cause OpenCOBOL to interpret numeric literals along the lines of 123,45 as one hundred twenty three and forty five one hundredths.

DECIMAL-POINT IS COMMA, while world friendly, can be the cause of ambiguous parsing and care must be taken by developers that use comma to separate parameters to FUNCTIONS.

## 28.1.128 4.1.128 DECLARATIVES

An imperative entry that can control exception handling of file operations and turn on debug entry points.

```
procedure division.
declaratives.
handle-errors section.
    use after standard error procedure on filename-1.
handle-error.
    display "Something bad happened with " filename-1 end-display.
.
helpful-debug section.
```

```
    use for debugging on main-file.  
help-me.  
    display "Just touched " main-file end-display.  
.   
end declaratives.
```

## 28.1.129 4.1.129 DEFAULT

A multi-use clause used in

- CALL ... SIZE IS DEFAULT
- ENTRY ... SIZE IS DEFAULT
- INITIALIZE ... WITH ... THEN TO DEFAULT

## 28.1.130 4.1.130 DELETE

Allows removal of records from RELATIVE and INDEXED files.

```
DELETE filename-1 RECORD  
  INVALID KEY  
    DISPLAY "no delete" END-DISPLAY  
  NOT INVALID KEY  
    DISPLAY "record removed" END-DISPLAY  
END-DELETE
```

### 4.1.130.1 OC 2.0

Allows file deletes.

```
DELETE FILE  
  filename-1 filename-2 filename-3  
END-DELETE
```

## 28.1.131 4.1.131 DELIMITED

A fairly powerful keyword used with the STRING and UNSTRING verbs. Accepts literals and the BY SIZE modifier.

```
STRING null-terminated  
  DELIMITED BY LOW-VALUE  
  INTO no-zero  
END-STRING
```

## 28.1.132 4.1.132 DELIMITER

Tracks which delimiter was used for a substring in an UNSTRING operation.

From Gary's OCic.cbl

```
UNSTRING Expand-Code-Rec
  DELIMITED BY ". " OR " "
  INTO SPI-Current-Token
  DELIMITER IN Delim
  WITH POINTER Src-Ptr
END-UNSTRING
```

### 28.1.133 4.1.133 DEPENDING

Sets a control identifier for variable OCCURS table definitions.

```
01 TABLE-DATA.
  05 TABLE-ELEMENTS
    OCCURS 1 TO 100 TIMES DEPENDING ON crowd-size
    INDEXED BY cursor-var.
  10 field-1 PIC X.
```

### 28.1.134 4.1.134 DESCENDING

Controls a descending sort and/or retrieval order, with

- SORT filename ON DESCENDING KEY alt-key
- OCCURS 1 TO max-size TIMES DESCENDING KEY key-for-table

### 28.1.135 4.1.135 DESTINATION

Currently unsupported data descriptor. Part of VALIDATE.

### 28.1.136 4.1.136 DETAIL

A recognized but currently unsupported report descriptor detail line control clause.

### 28.1.137 4.1.137 DISABLE

An unsupported COMMUNICATION SECTION control verb.

### 28.1.138 4.1.138 DISK

A SELECT devicename phrase.

```
ASSIGN TO DISK USING dataname
```

Alternative spelling of **DISC** is allowed.

### 28.1.139 4.1.139 DISPLAY

A general purpose output verb.

- prints values to default console or other device

- set the current `ARGUMENT-NUMBER` influencing subsequent access `ACCEPT FROM ARGUMENT-VALUE` statements
- specify explicit `COMMAND-LINE` influencing subsequent access with `ACCEPT FROM COMMAND-LINE`, but not `ARGUMENT-VALUE` access
- sets environment variables, as part of a two step process. (Use the more concise `SET ENVIRONMENT` instead)
  1. `DISPLAY "envname" UPON ENVIRONMENT-NAME`
  2. `DISPLAY "envname-value" UPON ENVIRONMENT-VALUE`

```
DISPLAY "First value: " a-variable " and another string" END-DISPLAY
```

```
DISPLAY "1" 23 "4" END-DISPLAY
```

The setting of environment variables does not influence the owning process shell.

```
DISPLAY "ENVNAME" UPON ENVIRONMENT-NAME END-DISPLAY
DISPLAY "COBOL value" UPON ENVIRONMENT-VALUE
    ON EXCEPTION stop run
    NOT ON EXCEPTION continue
END-DISPLAY
CALL "SYSTEM" USING "echo $ENVNAME"
```

gives:

```
$ ENVNAME="parent shell value"
$ ./disps
COBOL value
$ echo $ENVNAME
parent shell value
```

### 28.1.140 4.1.140 DIVIDE

Highly precise arithmetic. Supports various forms:

- `DIVIDE INTO`
- `DIVIDE INTO GIVING`
- `DIVIDE BY GIVING`
- `DIVIDE INTO REMAINDER`
- `DIVIDE BY REMAINDER`

For example:

```
DIVIDE dividend BY divisor GIVING answer ROUNDED REMAINDER r
    ON SIZE ERROR
        PERFORM log-division-error
        SET division-error TO TRUE
    NOT ON SIZE ERROR
        SET division-error TO FALSE
END-DIVIDE
```

The 20xx draft standard requires conforming implementations to use 1,000 digits of precision for intermediate results. There will be no rounding errors when properly calculating financials in a COBOL program.



### 28.1.141 4.1.141 DIVISION

Ahh, sub-divisions. I think my favourite is the DATA DIVISION. It gives COBOL a distinctive and delicious flavour in a picturesque codescape.

Divisions must be specified in the order below within each source program unit.

1. IDENTIFICATION DIVISION.
2. ENVIRONMENT DIVISION.
3. DATA DIVISION.
4. PROCEDURE DIVISION.

A handy mnemonic may be “*I Enter Data Properly*”.

OpenCOBOL is flexible enough to compile files with only a PROCEDURE DIVISION, and even then it really only needs a PROGRAM-ID. See [What is the shortest OpenCOBOL program?](#) for an example.

### 28.1.142 4.1.142 DOWN

Allows decrement of an index control or pointer variable.

```
SET ind-1 DOWN BY 2
```

Also used for SCREEN SECTION scroll control.

```
SCROLL DOWN 5 LINES
```

### 28.1.143 4.1.143 DUPLICATES

Allows duplicate keys in indexed files.

```
SELECT filename
  ALTERNATE RECORD KEY IS altkey WITH DUPLICATES
```

Also for SORT control.

```
SORT filename ON DESCENDING KEY keyfield
  WITH DUPLICATES IN ORDER
  USING sort-in GIVING sort-out.
```

### 28.1.144 4.1.144 DYNAMIC

A file access mode allowing runtime control over SEQUENTIAL and RANDOM access for INDEXED and RELATIVE ORGANIZATION.

```
SELECT filename
  ORGANIZATION IS RELATIVE
  ACCESS MODE IS DYNAMIC
```

### 28.1.145 4.1.145 EBCDIC

Extended Binary Coded Decimal Interchange Code.

A character encoding common to mainframe systems, therefore COBOL, therefore OpenCOBOL. Different than ASCII and OpenCOBOL supports both through efficient mappings. See <http://en.wikipedia.org/wiki/EBCDIC> for more info.

ASCII to EBCDIC conversion the OpenCOBOL way

```
SPECIAL-NAMES.  
ALPHABET ALPHA IS NATIVE.  
ALPHABET BETA IS EBCDIC.  
  
PROCEDURE DIVISION.  
INSPECT variable CONVERTING ALPHA TO BETA
```

### 28.1.146 4.1.146 EC

An unsupported shortform for USE AFTER EXCEPTION CONDITION

### 28.1.147 4.1.147 EGI

An unsupported COMMUNICATION SECTION word.

### 28.1.148 4.1.148 ELSE

Alternate conditional branch point.

```
IF AGE IS ZERO  
    DISPLAY "Cigar time" END-DISPLAY  
ELSE  
    DISPLAY "What is it with kids anyway?" END-DISPLAY  
END-IF
```

For multi branch conditionals, see EVALUATE.

### 28.1.149 4.1.149 EMI

An unsupported COMMUNICATION SECTION word.

### 28.1.150 4.1.150 ENABLE

An unsupported COMMUNICATION SECTION control verb.

### 28.1.151 4.1.151 END

Ends things. Programs, declaratives, functions.

### 28.1.152 4.1.152 END-ACCEPT

Explicit terminator for ACCEPT.

### **28.1.153 4.1.153 END-ADD**

Explicit terminator for `ADD`.

### **28.1.154 4.1.154 END-CALL**

Explicit terminator for `CALL`.

### **28.1.155 4.1.155 END-COMPUTE**

Explicit terminator for `COMPUTE`.

### **28.1.156 4.1.156 END-DELETE**

Explicit terminator for `DELETE`.

### **28.1.157 4.1.157 END-DISPLAY**

Explicit terminator for `DISPLAY`.

### **28.1.158 4.1.158 END-DIVIDE**

Explicit terminator for `DIVIDE`.

### **28.1.159 4.1.159 END-EVALUATE**

Explicit terminator for `EVALUATE`.

### **28.1.160 4.1.160 END-IF**

Explicit terminator for `IF`.

### **28.1.161 4.1.161 END-MULTIPLY**

Explicit terminator for `MULTIPLY`.

### **28.1.162 4.1.162 END-OF-PAGE**

A `LINAGE` phrase used by `WRITE` controlling end of page imperative clause.

### **28.1.163 4.1.163 END-PERFORM**

Explicit terminator for `PERFORM`.

#### **28.1.164 4.1.164 END-READ**

Explicit terminator for `READ`.

#### **28.1.165 4.1.165 END-RECEIVE**

Explicit terminator for `RECEIVE`.

#### **28.1.166 4.1.166 END-RETURN**

Explicit terminator for `RETURN`.

#### **28.1.167 4.1.167 END-REWRITE**

Explicit terminator for `REWRITE`.

#### **28.1.168 4.1.168 END-SEARCH**

Explicit terminator for `SEARCH`.

#### **28.1.169 4.1.169 END-START**

Explicit terminator for `START`.

#### **28.1.170 4.1.170 END-STRING**

Explicit terminator for `STRING`.

#### **28.1.171 4.1.171 END-SUBTRACT**

Explicit terminator for `SUBTRACT`.

#### **28.1.172 4.1.172 END-UNSTRING**

Explicit terminator for `UNSTRING`.

#### **28.1.173 4.1.173 END-WRITE**

Explicit terminator for `WRITE`.

#### **28.1.174 4.1.174 ENTRY**

Always for `CALL` entry points without being fully specified sub-programs. Great for defining callbacks required by many GUI frameworks.

See [Does OpenCOBOL support the GIMP ToolKit, GTK+?](#) for an example.

**28.1.175 4.1.175 ENTRY-CONVENTION**

An as yet unsupported clause.

**28.1.176 4.1.176 ENVIRONMENT**

Divisional name. And allows access to operating system environment variables. OpenCOBOL supports

- CONFIGURATION SECTION
- INPUT-OUTPUT SECTION

within the ENVIROMENT DIVISION.

Also a context sensitive keyword for access to the process environment variables.

- SET ENVIRONMENT “env-var” TO value
- ACCEPT var FROM ENVIRONMENT “env-var” END-ACCEPT

**28.1.177 4.1.177 ENVIRONMENT-NAME**

Provides access to the running process environment variables.

**28.1.178 4.1.178 ENVIRONMENT-VALUE**

Provides access to the running process environment variables.

**28.1.179 4.1.179 EO**

An unsupported shortform for USE AFTER EXCEPTION OBJECT

**28.1.180 4.1.180 EOL**

ERASE to End Of Line.

**28.1.181 4.1.181 EOP**

LINAGE clause shortform for END-OF-PAGE.

**28.1.182 4.1.182 EOS**

ERASE to End Of Screen.

**28.1.183 4.1.183 EQUAL**

Conditional expression to compare two data items for equality.

## 28.1.184 4.1.184 EQUALS

Conditional expression to compare two data items for equality.

## 28.1.185 4.1.185 ERASE

A screen section data attribute clause that can control which portions of the screen are cleared during `DISPLAY`, and `ACCEPT`.

```
01 form-record.  
  02 first-field PIC xxx  
     USING identifier-1  
     ERASE EOL.
```

## 28.1.186 4.1.186 ERROR

A `DECLARATIVES` clause that can control error handling.

```
USE AFTER STANDARD ERROR PROCEDURE ON filename-1
```

Program return control.

```
STOP RUN WITH ERROR STATUS stat-var.
```

## 28.1.187 4.1.187 ESCAPE

Programmer access to escape key value during `ACCEPT`.

```
ACCEPT identifier FROM ESCAPE KEY END-ACCEPT
```

Data type is 9(4).

## 28.1.188 4.1.188 ESI

Unsupported `COMMUNICATION SECTION` control.

## 28.1.189 4.1.189 EVALUATE

A very powerful and concise selection construct.

```
EVALUATE a ALSO b ALSO TRUE  
  WHEN 1 ALSO 1 THRU 9 ALSO c EQUAL 1 PERFORM all-life  
  WHEN 2 ALSO 1 THRU 9 ALSO c EQUAL 2 PERFORM life  
  WHEN 3 THRU 9 ALSO 1 ALSO c EQUAL 9 PERFORM disability  
  WHEN OTHER PERFORM invalid  
END-EVALUATE
```

## 28.1.190 4.1.190 EXCEPTION

Allow detection of `CALL` problem.

**CALL** "CBL\_OC\_DUMP" **ON EXCEPTION CONTINUE END-CALL**

### 28.1.191 4.1.191 EXCEPTION-OBJECT

Unsupport object COBOL data item reference.

### 28.1.192 4.1.192 EXCLUSIVE

Mode control for file locks.

### 28.1.193 4.1.193 EXIT

OpenCOBOL supports

- EXIT
- EXIT PROGRAM
- EXIT PERFORM [CYCLE]
- EXIT SECTION
- EXIT PARAGRAPH

Controls flow of the program. EXIT PERFORM CYCLE causes an inline perform to return control to the **VARYING**, **UNTIL** or **TIMES** clause, testing the conditional to see if another cycle is required. EXIT PERFORM without the CYCLE option causes flow to continue passed the end of the current PERFORM loop.

### 28.1.194 4.1.194 EXPANDS

Unsupported COMMUNICATION SECTION control.

### 28.1.195 4.1.195 EXTEND

Open a resource in an append mode.

### 28.1.196 4.1.196 EXTERNAL

Clause to specify external data item, file connection and program unit.

```
77 shared-var PIC S9(4) IS EXTERNAL AS 'shared_var' .
```

### 28.1.197 4.1.197 FACTORY

An unsupported object COBOL keyword.

## 28.1.198 4.1.198 FALSE

Logical false and conditional set condition.

```
01 record-1          pic 9.
   88 conditional-1 values 1,2,3 when set to false is 0.

set conditional-1 to true
display record-1 end-display

set conditional-1 to false
display record-1 end-display

if conditional-1
    display "BAD" end-display
end-if
```

Runs as:

```
$ ./conditionals
1
0
```

## 28.1.199 4.1.199 FD

The record side of the COBOL file system. The File Descriptor.

```
FD filename-sample
   RECORD IS VARYING IN SIZE FROM 1 TO 32768 CHARACTERS
   DEPENDING ON record-size-sample.
```

## 28.1.200 4.1.200 FILE

Files.

```
USE AFTER EXCEPTION FILE filename-maybe
```

## 28.1.201 4.1.201 FILE-CONTROL

Files. The paragraph in the INPUT-OUTPUT section, in the ENVIRONMENT division. It's verbose, a little voodooey, and worth it.

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    select optional data-file assign to file-name
        organization is line sequential
        file status is data-file-status.

    select mini-report assign to "mini-report".
```

## 28.1.202 4.1.202 FILE-ID

Files.





## 28.1.208 4.1.208 FLOAT-SHORT

OpenCOBOL supports short floating point.

## 28.1.209 4.1.209 FOOTING

A well supported `LINAGE` clause.

## 28.1.210 4.1.210 FOR

Recognized but unsupported Report Writer clause.

## 28.1.211 4.1.211 FOREGROUND-COLOR

Screen section foreground color control. See [What are the OpenCOBOL SCREEN SECTION colour values?](#)

## 28.1.212 4.1.212 FOREVER

Provides for infinite loops. Use `EXIT PERFORM` or `EXIT PERFORM CYCLE` to control program flow.

```
identification division.
program-id. foreverloop.

data division.
working-storage section.
01 cobol    pic 9 value 0.
01 c       pic 9 value 1.
01 fortran pic 9 value 2.

procedure division.

perform forever
  add 1 to cobol
  display "cobol at " cobol end-display

  if cobol greater than fortran
    exit perform
  end-if

  if cobol greater than c
    exit perform cycle
  end-if

  display "cobol still creeping up on c" end-display
end-perform

display "cobol surpassed c and fortran" end-display

goback.
end program foreverloop.
```

Which produces:

```
$ cobc -free -x foreverloop.cob
$ ./foreverloop
cobol at 1
cobol still creeping up on c
cobol at 2
cobol at 3
cobol surpassed c and fortran
```

I asked on [opencobol.org](http://opencobol.org) for some input, and an interesting conversation ensued. I've included the forum thread archive, nearly in its entirety, to give a sense of various programmer styles and group thought processing. See [Performing FOREVER?](#).

### 28.1.213 4.1.213 FORMAT

Source format directive.

```
123456 >>SOURCE FORMAT IS FIXED
```

### 28.1.214 4.1.214 FREE

Properly cleans up `ALLOCATE` allotted memory, and source format directive.

```
>>SOURCE FORMAT IS FREE

01 var PIC X(1024) BASED.

ALLOCATE var
CALL "buffer-thing" USING BY REFERENCE var END-CALL
MOVE var TO working-store
FREE var
```

### 28.1.215 4.1.215 FROM

```
ACCEPT var FROM ENVIRONMENT "path"
  ON EXCEPTION
    DISPLAY "No path" END-DISPLAY
  NOT ON EXCEPTION
    DISPLAY var END-DISPLAY
END-ACCEPT
```

### 28.1.216 4.1.216 FULL

A screen section screen item control operator, requesting the normal terminator be ignored until the field is completely full or completely empty.

### 28.1.217 4.1.217 FUNCTION

Allows use of the many OpenCOBOL supported intrinsic functions.

```
DISPLAY FUNCTION TRIM(" trim off leading spaces" LEADING) END-DISPLAY.
```

See [Does OpenCOBOL implement any Intrinsic FUNCTIONS?](#) for details.

### 28.1.218 4.1.218 FUNCTION-ID

Not yet implemented, but it will allow for user defined FUNCTION.

### 28.1.219 4.1.219 GENERATE

Not yet implemented beyond simple parsing REPORT writer feature.

### 28.1.220 4.1.220 GET

Unsupported.

### 28.1.221 4.1.221 GIVING

Destination control for computations, and return value clause.

```
ADD 1 TO cobol GIVING OpenCOBOL.
```

### 28.1.222 4.1.222 GLOBAL

A global name is accessible to all contained programs.

### 28.1.223 4.1.223 GO

GO TO is your friend. Edsger was wrong. Transfer control to a named paragraph or section. See [ALTER](#) for details of monster goto power.

### 28.1.224 4.1.224 GOBACK

A return. This will work correctly for all cases. A return to the operating system or a return to a called program.

```
GOBACK.
```

### 28.1.225 4.1.225 GREATER

COBOL conditional expression, IF A GREATER THAN B, See [LESS](#)

### 28.1.226 4.1.226 GROUP

Recognized but unsupported Report Writer clauses.

### 28.1.227 4.1.227 GROUP-USAGE

An unsupported BIT clause.

### 28.1.228 4.1.228 HEADING

Recognized but unsupported Report Writer clauses.

### 28.1.229 4.1.229 HIGH-VALUE

A figurative ALPHABETIC constant, being the highest character value in the COLLATING sequence. It's invalid to MOVE HIGH-VALUE to a NUMERIC field.

### 28.1.230 4.1.230 HIGH-VALUES

Plural of HIGH-VALUE.

### 28.1.231 4.1.231 HIGHLIGHT

Screen control for field intensity.

### 28.1.232 4.1.232 I-O

An OPEN mode allowing for both read and write.

### 28.1.233 4.1.233 I-O-CONTROL

A paragraph in the INPUT-OUTPUT section, allowing sharing memory areas for different files.

```
ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
I-O-CONTROL.  
    SAME RECORD AREA FOR filename-1 filename-2.
```

### 28.1.234 4.1.234 ID

Shortform for IDENTIFICATION.

### 28.1.235 4.1.235 IDENTIFICATION

The initial division for OpenCOBOL programs.

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. sample.
```

Many historical paragraphs from the IDENTIFICATION DIVISION have been deemed obsolete. OpenCOBOL will treat these as comment paragraphs. Including

- AUTHOR
- DATE-WRITTEN
- DATE-MODIFIED
- DATE-COMPILED

- INSTALLATION
- REMARKS
- SECURITY

### 28.1.236 4.1.236 IF

Conditional branching. In COBOL, conditionals are quite powerful and there are many conditional expressions allowed with concise shortcuts.

```
IF A = 1 OR 2
    MOVE 1 TO B
END-IF
```

### 28.1.237 4.1.237 IGNORING

```
READ filename-1 INTO identifier-1 IGNORING LOCK END-READ
```

### 28.1.238 4.1.238 IMPLEMENTS

Unsupported Object COBOL expression.

### 28.1.239 4.1.239 IN

A data structure reference and name conflict resolution qualifier.

```
MOVE "abc" TO field IN the-record IN the-structure
```

Synonym for OF

### 28.1.240 4.1.240 INDEX

```
01 cursor-var USAGE INDEX.
```

```
SET cursor-var UP BY 1.
```

### 28.1.241 4.1.241 INDEXED

An ISAM file organization.

```
environment division.
input-output section.
file-control.
    select optional indexing
    assign to "indexing.dat"
    organization is indexed
    access mode is dynamic
    record key is keyfield of indexing-record
    alternate record key is splitkey of indexing-record
    with duplicates
.
```

Sets an indexing control identifier for OCCURS data arrays.

```
01 TABLE-DATA.
   05 TABLE-ELEMENTS
      OCCURS 1 TO 100 TIMES DEPENDING ON crowd-size
      INDEXED BY cursor-var.
   10 field-1 PIC X.
```

### 28.1.242 4.1.242 INDICATE

GROUP INDICATE is an as yet unsupported REPORT SECTION RD clause that specifies that printable item is output only on the first occurrence of its report group for that INITIATE, control break, or page advance.

### 28.1.243 4.1.243 INHERITS

An unsupported Object COBOL clause.

### 28.1.244 4.1.244 INITIAL

A modifier for the PROGRAM-ID clause, that causes the entire DATA DIVISION to be set to an initial state each time the subprogram is executed by CALL.

```
ocobol >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20111226
*> Purpose:     Small sample of INITIAL procedure division clause
*> Tectonics:   cobc -x -w -g -debug initialclause.cob
*> *****
identification division.
program-id. initialclause.

*> -----
procedure division.
call "with-initial" end-call
call "without-initial" end-call
call "with-initial" end-call
call "without-initial" end-call
call "without-initial" end-call
goback.
end program initialclause.

*> -----
*> -----
identification division.
program-id. with-initial is initial.

data division.
working-storage section.
01 the-value pic 99 value 42.

*> -----
procedure division.
display "Inside with-initial with      : " the-value end-display
```

```
multiply the-value by 2 giving the-value
  on size error
    display "size overflow" end-display
end-multiply
goback.
end program with-initial.

*> -----**
*> -----**
  identification division.
  program-id. without-initial.

  data division.
  working-storage section.
  01 the-value pic 99 value 42.

*> -----**
  procedure division.
  display "Inside without-initial with: " the-value end-display
  multiply the-value by 2 giving the-value
    on size error
      display "size overflow" end-display
  end-multiply
  goback.
  end program without-initial.
```

Gives:

```
[btiffin@home cobol]$ ./initialclause
Inside with-initial with : 42
Inside without-initial with: 42
Inside with-initial with : 42
Inside without-initial with: 84
size overflow
Inside without-initial with: 84
size overflow
```

INITIAL sets the-value to 42 upon each and every entry, without-initial multiplies through 42, 84, 168 (or would have).

## 28.1.245 4.1.245 INITIALIZE

A sample of the INITIALIZE verb posted to [opencobol.org](http://opencobol.org) by human

```
OCOBOL*-----**
  IDENTIFICATION DIVISION.
  PROGRAM-ID. 'INITTEST'.
  ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  SPECIAL-NAMES. DECIMAL-POINT IS COMMA.
  INPUT-OUTPUT SECTION.
  DATA DIVISION.
*
  WORKING-STORAGE SECTION.
*
  77 mychar      pic x.
  77 mynumeric   pic 9.
```



```

01 REC-TEST BASED.
   03 REC-TEST-PART1 PIC X(10) value all '9'.
   03 REC-TEST-PART2 PIC X(10) value all 'A'.
01 fillertest.
   03 fillertest-1 PIC 9(10) value 2222222222.
   03 filler      PIC X      value '|'.
   03 fillertest-2 PIC X(10) value all 'A'.
   03 filler      PIC 9(03)  value 111.
   03 filler      PIC X      value '.'.
-----
LINKAGE SECTION.
-----
PROCEDURE DIVISION.
-----
Main section.
00.
*
  display 'fillertest '
         'on start:'
  end-display
  display fillertest
  end-display
  accept mychar
*
  initialize fillertest
  display 'fillertest '
         'after initialize:'
  end-display
  display fillertest
  end-display
  accept mychar
*
  initialize fillertest replacing numeric by 9
  display 'fillertest '
         'after initialize replacing numeric by 9:'
  end-display
  display fillertest
  end-display
  accept mychar
*
  initialize fillertest replacing alphanumeric by 'X'
  display 'fillertest '
         'after initialize replacing alphanumeric by "X":'
  end-display
  display fillertest
  end-display
  accept mychar
*
  initialize fillertest replacing alphanumeric by all 'X'
  display 'fillertest '
         'after initialize replacing alphanumeric by all "X":'
  end-display
  display fillertest
  end-display
  accept mychar
*
  initialize fillertest with filler
  display 'fillertest '

```

```
        'after initialize with filler:'
end-display
display fillertest
end-display
accept mychar
*
initialize fillertest all to value
display 'fillertest '
        'after initialize all to value:'
end-display
display fillertest
end-display
accept mychar
*
ALLOCATE REC-TEST
display 'REC-TEST after allocating:'
end-display
display REC-TEST
end-display
accept mychar
*
initialize REC-TEST all to value
display 'REC-TEST after initalize all to value:'
end-display
display REC-TEST
end-display
accept mychar
*
stop run
*
continue.
ex. exit program.
-----
*--- End of program INITTEST -----
```

#### Outputs:

```
fillertest on start:
2222222222|AAAAAAAAAA111.
fillertest after initialize:
0000000000|          111.
fillertest after initialize replacing numeric by 9:
0000000009|          111.
fillertest after initialize replacing alphanumeric by "X":
0000000009|X          111.
fillertest after initialize replacing alphanumeric by all "X":
0000000009|XXXXXXXXXX111.
fillertest after initialize with filler:
0000000000          000
fillertest after initialize all to value:
2222222222|AAAAAAAAAA111.
REC-TEST after allocating:

REC-TEST after initalize all to value:
9999999999AAAAAAAAAA
```

## 28.1.246 4.1.246 INITIALIZED

A modifier for the `ALLOCATE` verb, filling the target with a default value.

```

77 based-var    PIC X(9)  BASED VALUE "ALLOCATED".
77 pointer-var  USAGE POINTER.

ALLOCATE based-var
DISPLAY ":" based-var ":" END-DISPLAY
FREE based-var
ALLOCATE based-var INITIALIZED RETURNING pointer-var
DISPLAY ":" based-var ":" END-DISPLAY

```

displays:

```

:      :
:ALLOCATED:

```

## 28.1.247 4.1.247 INITIATE

Initialize internal storage for named REPORT SECTION entries.

Not currently (*February 2013*) supported.

## 28.1.248 4.1.248 INPUT

A mode of the `OPEN` verb for file access.

```
OPEN INPUT file
```

A `SORT` clause allowing programmer controlled input read passes where sortable records are passed to the sort algorithm using `RELEASE`.

```

procedure division.
sort sort-work
    on descending key work-rec
    collating sequence is mixed
    input procedure is sort-transform
    output procedure is output-uppercase.

```

```

display sort-return end-display.
goback.

```

## 28.1.249 4.1.249 INPUT-OUTPUT

A section in the ENVIRONMENT DIVISION of a COBOL source file containing FILE and I-O control paragraphs.

```

environment division.
input-output section.
file-control.
    select htmlfile
    assign to filename
    organization is record sequential.

```

OpenCOBOL supports

- FILE-CONTROL

- I-O-CONTROL

paragraphs within the INPUT-OUTPUT SECTION.

### 28.1.250 4.1.250 INSPECT

Provides very powerful parsing and replacement to COBOL and OpenCOBOL supports the full gamet of options.

```
ocobol identification division.
  program-id. inspecting.

  data division.
  working-storage section.
01 ORIGINAL          pic XXXX/XX/XXBXX/XX/XXXXXXXX/XX.
01 DATEREC           pic XXXX/XX/XXBXX/XX/XXXXXXXX/XX.

  procedure division.

  move function when-compiled to DATEREC ORIGINAL

  INSPECT DATEREC REPLACING ALL "/" BY ":" AFTER INITIAL SPACE

  display
    "Intrinsic function WHEN-COMPILED " ORIGINAL
  end-display
  display
    " after INSPECT REPLACING          " DATEREC
  end-display

  goback.
  end program inspecting.
```

Example output:

```
Intrinsic function WHEN-COMPILED 2010/03/25 23/05/0900-04/00
after INSPECT REPLACING          2010/03/25 23:05:0900-04:00
```

### 28.1.251 4.1.251 INTERFACE

Unsupported.

### 28.1.252 4.1.252 INTERFACE-ID

An unsupported Object COBOL clause in the IDENTIFICATION division.

### 28.1.253 4.1.253 INTO

Division.

```
DIVIDE A INTO B GIVING C.
```

### 28.1.254 4.1.254 INTRINSIC

Used in REPOSITORY to allow the optional use of "FUNCTION" keyword.

```
environment division.
configuration section.
repository.
    function all intrinsic.
```

The source unit will now allow for program lines such as

```
move trim(" abc") to dest
move function trim(" abc") to dest
```

to compile the same code.

### 28.1.255 4.1.255 INVALID

Key exception imperative phrase.

```
READ filename-1
    INVALID KEY
        DISPLAY "Bad key"
    NOT INVALID KEY
        DISPLAY "Good read"
END-READ
```

### 28.1.256 4.1.256 INVOKE

Unsupported Object COBOL method call.

### 28.1.257 4.1.257 IS

Readability word. A IS LESS THAN B is equivalent to A LESS B.

### 28.1.258 4.1.258 JUST

Alias for JUSTIFIED.

### 28.1.259 4.1.259 JUSTIFIED

Tweaks storage rules in wierd JUST ways, lessening the voodoo behind MOVE instructions, *he said, sarcastically.*

```
77 str1 pic x(40) justified right.
```

### 28.1.260 4.1.260 KEY

Multi use, always means key:

- RELATIVE KEY IS
- ALTERNATE RECORD KEY IS
- NOT INVALID KEY
- SORT filename ON DESCENDING KEY keyfield
- START indexing KEY IS LESS THAN keyfield

### 28.1.261 4.1.261 KEYBOARD

A special value for Standard Input

```
file-control.  
  select cgi-in  
  assign to keyboard.
```

### 28.1.262 4.1.262 LABEL

A record label. As with most record labels, falling into disuse.

### 28.1.263 4.1.263 LAST

Used in START to prepare a read of the last record. A recognized but unsupported Report Writer clause.

```
START filename-1 LAST  
  INVALID KEY  
    MOVE ZERO TO record-count  
    >>D DISPLAY "No last record for " filename-1 END-DISPLAY  
END-START
```

### 28.1.264 4.1.264 LC\_ALL

A reserved but unsupported category group. See [Setting Locale](#). OpenCOBOL is 'locale' aware, but it is currently more *external* than in COBOL source. For now, it is safest to assume `LC_ALL=C`, but this can be configured differently when OpenCOBOL is built.

### 28.1.265 4.1.265 LC\_COLLATE

A reserved but unsupported category name. Will be used with SET.

### 28.1.266 4.1.266 LC\_CTYPE

A reserved but unsupported Locale category name. Will be used with SET.

### 28.1.267 4.1.267 LC\_MESSAGES

A reserved but unsupported category name. See [Setting Locale](#). OpenCOBOL is 'locale' aware, but it is currently more *external* than in COBOL source.

OpenCOBOL 2.0 extends locale support to the compiler messages.

```
$ export LC_MESSAGES=es_ES
$ cobc -x fdfgffd.cob
cobc: fdfgffd.cob: No existe el fichero o el directorio
```

### 28.1.268 4.1.268 LC\_MONETARY

A reserved but unsupported Locale category name. Will be used with SET.

### 28.1.269 4.1.269 LC\_NUMERIC

A reserved but unsupported Locale category name. Will be used with SET.

### 28.1.270 4.1.270 LC\_TIME

A reserved but unsupported Locale category name. Will be used with SET.

### 28.1.271 4.1.271 LEADING

Multipurpose.

```
DISPLAY FUNCTION TRIM(var-1 LEADING) END-DISPLAY
```

```
INSPECT FUNCTION REVERSE(TEST-CASE)
  TALLYING B-COUNT
  FOR LEADING ' '.
DISPLAY B-COUNT.
```

```
INSPECT X REPLACING LEADING ZEROS BY SPACES.
```

as well as use in the COBOL preprocessor:

```
COPY "copy.inc"
  REPLACING LEADING ==TEST== BY ==FIRST==
  LEADING ==NORM== BY ==SECOND==.
```

### 28.1.272 4.1.272 LEFT

SYNCHRONIZED control.

### 28.1.273 4.1.273 LENGTH

A 'cell-count' length. Not always the same as BYTE-LENGTH.

### 28.1.274 4.1.274 LESS

A comparison operation.

```
IF requested LESS THAN OR EQUAL TO balance
    PERFORM transfer
ELSE
    PERFORM reject
END-IF
```

## 28.1.275 4.1.275 LIMIT

Recognized but unsupported Report Writer clause.

## 28.1.276 4.1.276 LIMITS

Recognized but unsupported Report Writer clause.

## 28.1.277 4.1.277 LINAGE

LINAGE is a *SPECIAL-REGISTER* supported by OpenCOBOL. A counter is maintained for file WRITE and can be used for paging *and other* control.

```
COBOL *****
* Example of LINAGE File Descriptor
* Author: Brian Tiffin
* Date: 10-July-2008
* Tectonics: $ cocb -x linage.cob
*           $ ./linage <filename ["linage.cob"]>
*           $ cat -n mini-report
*****
IDENTIFICATION DIVISION.
PROGRAM-ID. linage-demo.

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    select optional data-file assign to file-name
           organization is line sequential
           file status is data-file-status.
    select mini-report assign to "mini-report".

DATA DIVISION.
FILE SECTION.
FD data-file.
01 data-record.
   88 endofdata           value high-values.
   02 data-line           pic x(80).
FD mini-report
   linage is 16 lines
       with footing at 15
       lines at top 2
       lines at bottom 2.
01 report-line           pic x(80).

WORKING-STORAGE SECTION.
01 command-arguments     pic x(1024).
01 file-name              pic x(160).
```



```

01 data-file-status      pic 99.
01 lc                   pic 99.
01 report-line-blank.
  02 filler              pic x(18) value all "*".
  02 filler              pic x(05) value spaces.
  02 filler              pic x(34)
    VALUE "THIS PAGE INTENTIONALLY LEFT BLANK".
  02 filler              pic x(05) value spaces.
  02 filler              pic x(18) value all "*".
01 report-line-data.
  02 body-tag            pic 9(6).
  02 line-3              pic x(74).
01 report-line-header.
  02 filler              pic x(6) VALUE "PAGE: ".
  02 page-no             pic 9999.
  02 filler              pic x(24).
  02 filler              pic x(5) VALUE " LC: ".
  02 header-tag         pic 9(6).
  02 filler              pic x(23).
  02 filler              pic x(6) VALUE "DATE: ".
  02 page-date          pic x(6).

01 page-count           pic 9999.

```

#### PROCEDURE DIVISION.

```

accept command-arguments from command-line end-accept.
string
  command-arguments delimited by space
  into file-name
end-string.
if file-name equal spaces
  move "linage.cob" to file-name
end-if.

open input data-file.
read data-file
  at end
    display
      "File: " function trim(file-name) " open error"
    end-display
  go to early-exit
end-read.

open output mini-report.

write report-line
  from report-line-blank
end-write.

move 1 to page-count.
accept page-date from date end-accept.
move page-count to page-no.
write report-line
  from report-line-header
  after advancing page
end-write.

```

```

perform readwrite-loop until endofdata.

display
  "Normal termination, file name: "
  function trim(file-name)
  " ending status: "
  data-file-status
end-display.
close mini-report.

* Goto considered harmful? Bah! :)
early-exit.
close data-file.
exit program.
stop run.

*****
readwrite-loop.
move data-record to report-line-data
move lineage-counter to body-tag
write report-line from report-line-data
end-of-page
  add 1 to page-count end-add
  move page-count to page-no
  move lineage-counter to header-tag
  write report-line from report-line-header
  after advancing page
  end-write
end-write
read data-file
  at end set endofdata to true
end-read
.

*****
* Commentary
* LINAGE is set at a 20 line logical page
* 16 body lines
* 2 top lines
* A footer line at 15 (inside the body count)
* 2 bottom lines
* Build with:
* $ cobc -x -Wall -Wtruncate lineage.cob
* Evaluate with:
* $ ./linage
* This will read in lineage.cob and produce a useless mini-report
* $ cat -n mini-report
*****
END PROGRAM lineage-demo.

```

## Using

```
$ ./linage except.cob
```

Produces a *mini-report* of:

```
*****          THIS PAGE INTENTIONALLY LEFT BLANK          *****
```

```
PAGE: 0001                                LC: 000000                                DATE: 090206
000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID. MINIPROG.
000003 ENVIRONMENT DIVISION.
000004 CONFIGURATION SECTION.
000005 SOURCE-COMPUTER. LINUX.
000006 OBJECT-COMPUTER. LINUX.
000007 SPECIAL-NAMES.
000008 INPUT-OUTPUT SECTION.
000009 FILE-CONTROL.
000010 SELECT PRINTFILE ASSIGN TO "XXRXWXX"
000011 FILE STATUS RXWSTAT.
000012 DATA DIVISION.
000013 FILE SECTION.
000014 FD PRINTFILE.
```

```
PAGE: 0002                                LC: 000015                                DATE: 090206
000001 01 PRINTREC PIC X(132).
000002 WORKING-STORAGE SECTION.
000003 01 RXWSTAT PIC XX.
000004 01 str      pic x(4).
000005 PROCEDURE DIVISION.
000006 A00-MAIN SECTION.
000007 001-MAIN-PROCEDURE.
000008 OPEN INPUT PRINTFILE.
000009 DISPLAY "File Status: " RXWSTAT.
000010 DISPLAY "EXCEPTION-FILE: " FUNCTION EXCEPTION-FILE.
000011 DISPLAY "Return Length: "
000012 FUNCTION LENGTH (FUNCTION EXCEPTION-FILE).
000013 DISPLAY "EXCEPTION-STATUS: " FUNCTION EXCEPTION-STATUS.
000014 DISPLAY "EXCEPTION-STATEMENT: " FUNCTION EXCEPTION-STATEMENT.
```

```
PAGE: 0003                                LC: 000015                                DATE: 090206
000001 STRING "TOOLONG" DELIMITED SIZE INTO RXWSTAT.
000002 DISPLAY "EXCEPTION-STATUS: " FUNCTION EXCEPTION-STATUS.
```

```
000003 DISPLAY "EXCEPTION-STATEMENT: " FUNCTION EXCEPTION-STATEMENT.
000004 DISPLAY "EXCEPTION-LOCATION: " FUNCTION EXCEPTION-LOCATION.
000005 STOP RUN.
```

See *except.cob* under the [FUNCTION EXCEPTION-STATUS](#) entry.

### 28.1.278 4.1.278 LINAGE-COUNTER

An internal OpenCOBOL noun, or *Special Register*. Value is readonly and is maintained during WRITES to files that have a [LINAGE](#) clause. Useful for quick reports and logical page layouts.

### 28.1.279 4.1.279 LINE

LINE SEQUENTIAL files. Screen section line control.

### 28.1.280 4.1.280 LINE-COUNTER

Special register for the unsupported Report Writer.

### 28.1.281 4.1.281 LINES

Screen section line control, screen occurs control and area scrolling.

### 28.1.282 4.1.282 LINKAGE

A [SECTION](#) in the [DATA DIVISION](#). Used for call frame data handling when the current run unit may not be in charge of the location of working storage. Defaults to uninitialized references which must be set with [USING](#) in a [CALL](#) or explicitly with [SET ADDRESS](#). References without initialization will cause an addressing segfault.

### 28.1.283 4.1.283 LOCAL-STORAGE

A [SECTION](#) in the [DATA DIVISION](#). Data defined in local storage will be local to the running module and re-entrant within subprogram call trees.

### 28.1.284 4.1.284 LOCALE

Unsupported in OpenCOBOL 1.1pre-rel. Support added in 2.0

A [SPECIAL-NAMES](#) entry giving OpenCOBOL an international flair.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    LOCALE spanish IS 'ES_es'.
```

## 28.1.285 4.1.285 LOCK

Record management.

```
SELECT filename-1 ASSIGN TO 'master.dat' LOCK MODE IS MANUAL.
```

## 28.1.286 4.1.286 LOW-VALUE

A figurative ALPHABETIC constant, being the lowest character value in the COLLATING sequence.

```
MOVE LOW-VALUE TO alphanumeric-1.

IF alphabetic-1 EQUALS LOW-VALUE
    DISPLAY "Failed validation" END-DISPLAY
END-IF.
```

It's invalid to MOVE LOW-VALUE to a numeric field.

## 28.1.287 4.1.287 LOW-VALUES

A pluralized form of LOW-VALUE. Equivalent.

```
MOVE LOW-VALUES TO alphanumeric-1.
```

## 28.1.288 4.1.288 LOWLIGHT

A screen attribute for DISPLAY and SCREEN SECTION fields.

```
SCREEN SECTION.
01 example.
    05 FILLER
        LINE 1 COLUMN 10
        VALUE IS "Example:"
        LOWLIGHT.
```

Will display the *Example:* legend in a dimmed video if supported with the current terminal settings.

## 28.1.289 4.1.289 MANUAL

LOCK MODE IS MANUAL WITH LOCK ON MULTIPLE RECORDS. See AUTOMATIC and EXCLUSIVE for more LOCK options.

## 28.1.290 4.1.290 MEMORY

An OBJECT-COMPUTER clause.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
OBJECT-COMPUTER.
MEMORY SIZE IS 8 CHARACTERS.
```

## 28.1.291 4.1.291 MERGE

Combines two or more identically sequenced files on a set of specified keys.

```
MERGE sort-file
  ON DESCENDING KEY key-field-1
  WITH DUPLICATES IN ORDER
  COLLATING SEQUENCE IS user-alphabet
  USING filename-1 filename-2
  GIVING filename-3
```

## 28.1.292 4.1.292 MESSAGE

Unsupported Communication Section clause.

## 28.1.293 4.1.293 METHOD

Unsupported Object COBOL feature.

## 28.1.294 4.1.294 METHOD-ID

Unsupported Object COBOL feature.

## 28.1.295 4.1.295 MINUS

Screen section relative line and column control.

```
05 some-field pic x(16)
   line number is plus 1
   column number is minus 8
```

## 28.1.296 4.1.296 MODE

Locking mode. See MANUAL, AUTOMATIC, EXCLUSIVE.

## 28.1.297 4.1.297 MOVE

A workhorse of the COBOL paradigm. MOVE is highly flexible, intelligent, safe and sometimes perplexing data movement verb.

```
01 alphanum-3          PIC XXX.
01 num2                PIC 99.
```

```
MOVE "ABCDEFGH" TO xvar3
DISPLAY xvar3 END-DISPLAY
```

```
MOVE 12345 TO num2
DISPLAY num2 END-DISPLAY
```

displays:

ABC  
45

Note the 45, MOVE uses a right to left rule when moving numerics.

Groups can be moved with

```
MOVE CORRESPONDING ident-1 TO ident-2
```

in which case only the group items of the same name will be transferred from the ident-1 group to the ident-2 fields.

### 28.1.298 4.1.298 MULTIPLE

```
LOCK MODE IS MANUAL WITH LOCK ON MULTIPLE RECORDS.
```

### 28.1.299 4.1.299 MULTIPLY

A mathematic operation.

```
MULTIPLY var-1 BY var-2 GIVING var-3  
    ON SIZE ERROR  
        SET invalid-result TO TRUE  
END-MULTIPLY
```

### 28.1.300 4.1.300 NATIONAL

NATIONAL character usage. Not yet supported. OpenCOBOL does support PICTURE N.

### 28.1.301 4.1.301 NATIONAL-EDITED

Category.

### 28.1.302 4.1.302 NATIVE

Alphabet.

### 28.1.303 4.1.303 NEGATIVE

Conditional expression.

```
IF a IS NEGATIVE  
    SET in-the-red TO TRUE  
END-IF
```

### 28.1.304 4.1.304 NESTED

An unsupported program-prototype CALL clause.

### 28.1.305 4.1.305 NEXT

With `READ`, to read the next record, possibly by `KEY`. Also an obsolete control flow verb.

```
READ index-sequential-file NEXT RECORD INTO ident-1

IF condition-1
    NEXT SENTENCE
ELSE
    PERFORM do-something.
```

### 28.1.306 4.1.306 NO

Specify `NO` locks, `NO` sharing, `NO` rewind.

```
CLOSE filename-1 WITH NO REWIND

READ file-1 WITH NO LOCK
```

### 28.1.307 4.1.307 NONE

Unsupported `DEFAULT IS NONE`.

### 28.1.308 4.1.308 NORMAL

Program return control

```
STOP RUN WITH NORMAL STATUS status-val
```

See `ERROR`

### 28.1.309 4.1.309 NOT

Conditional negation. See `AND`, `OR`. Also used in operational declaratives such as `NOT ON SIZE ERROR`, *in which case the operation succeeded without overflowing the receiving data field.*

```
IF NOT testing
    CALL "thing"
        NOT ON EXCEPTION
            DISPLAY "Linkage to thing, OK" END-DISPLAY
    END-CALL
END-IF
```

### 28.1.310 4.1.310 NULL

Void. A zero address pointer. A symbolic literal.

```
CALL "thing" RETURNING NULL END-CALL

SET ADDRESS OF ptr TO NULL

IF ptr EQUAL NULL
```



```
    DISPLAY "ptr not valid" END-DISPLAY
END-IF
```

```
MOVE CONCATENATE (TRIM(cbl-string TRAILING) NULL) TO c-string
```

### 28.1.311 4.1.311 NULLS

Plural of NULL.

```
MOVE ALL NULLS TO var
```

### 28.1.312 4.1.312 NUMBER

Screen section LINE COLUMN control.

```
05 some-field pic x(16) LINE NUMBER 5.
```

### 28.1.313 4.1.313 NUMBERS

Plural of NUMBER.

### 28.1.314 4.1.314 NUMERIC

Category.

### 28.1.315 4.1.315 NUMERIC-EDITED

Category.

### 28.1.316 4.1.316 OBJECT

Unsupported Object COBOL feature.

### 28.1.317 4.1.317 OBJECT-COMPUTER

Environment division, configuration section run-time machine paragraph.

OpenCOBOL supports

```
OCOBOL identification division.
    program-id. runtime-computer.

    environment division.
    configuration section.
    object-computer.
        memory size is 8 characters
        program collating sequence is bigiron-alphabet
        segment-limit is 64
        character classification is spanish-locale.
```

```
repository.  
  function all intrinsic.  
special-names.  
  alphabet bigiron-alphabet is ebcdic  
  symbolic characters BS is 9  
                      TAB is 10  
                      LF is 11  
                      NEWLINE is 11  
                      CMA is 45  
  locale spanish-locale is "es_ES".
```

### 28.1.318 4.1.318 OBJECT-REFERENCE

Unsupported Object COBOL feature.

### 28.1.319 4.1.319 OCCURS

Controls multiple occurrences of data structures.

```
01 main-table.  
  03 main-record occurs 366 times depending on the-day.  
    05 main-field pic x occurs 132 times depending on the-len.
```

### 28.1.320 4.1.320 OF

A data structure reference and name conflict resolution qualifier.

```
MOVE "abc" TO the-field OF the-record OF the-structure
```

Synonym for IN

### 28.1.321 4.1.321 OFF

Turn off a switch. See ON.

```
SPECIAL-NAMES.  
  SWITCH-1 IS mainframe  
    ON STATUS IS bigiron  
    OFF STATUS IS pc
```

...

```
SET mainframe TO OFF
```

### 28.1.322 4.1.322 OMITTED

Allows for placeholders in call frames and testing for said placeholders. Also allows for omitted label records, and void returns. OMITTED is only allowed with BY REFERENCE data.

```

CALL "thing" USING
    BY REFERENCE string-var
    BY VALUE number-var
    BY REFERENCE OMITTED
    GIVING NULL
END-CALL

...

PROGRAM-ID. thing.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 default-float usage float-long.

LINKAGE-SECTION.
77 string-var pic x(80).
77 number-var pic 9(8).
77 float-var usage float-long.

PROCEDURE DIVISION
    USING
        BY REFERENCE OPTIONAL string-var
        BY VALUE number-var
        BY REFERENCE OPTIONAL float-var
    RETURNING OMITTED.

IF float-var IS OMITTED
    SET ADDRESS OF float-var TO default-float
END-IF

```

### 28.1.323 4.1.323 ON

Turn on a switch. See OFF.

```

SPECIAL-NAMES.
    SWITCH-1 IS mainframe
        ON STATUS IS bigiron
        OFF STATUS IS pc

```

...

```
SET mainframe TO ON
```

Starts declaratives.

```

ADD 1 TO wafer-thin-mint
    ON SIZE ERROR
        SET get-a-bucket TO TRUE
END-ADD

```

See SIZE, EXCEPTION.

### 28.1.324 4.1.324 ONLY

Sharing control. SHARING WITH READ ONLY

### 28.1.325 4.1.325 OPEN

Opens a file selector. Modes include INPUT, OUTPUT, I-O, EXTEND. May be OPTIONAL in the FD.

```
OPEN INPUT SHARING WITH ALL OTHER infile
OPEN EXTEND SHARING WITH NO OTHER myfile
```

### 28.1.326 4.1.326 OPTIONAL

Allows for referencing non-existent files. Allows for optionally OMITTED call arguments.

Code below shows optional file open and optional CALL arguments.

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT OPTIONAL nofile ASSIGN TO "file.not"
        ORGANIZATION IS LINE SEQUENTIAL.
```

...

```
DATA DIVISION.
LINKAGE SECTION.
77 arg PIC 99.
```

```
PROCEDURE DIVISION USING OPTIONAL arg
```

```
OPEN INPUT nofile
CLOSE nofile
```

```
IF arg IS OMITTED OR NOT NUMERIC
    MOVE 0 TO RETURN-CODE
ELSE
    MOVE arg TO RETURN-CODE
END-IF
GOBACK.
```

### 28.1.327 4.1.327 OPTIONS

A currently unsupported paragraph of the IDENTIFICATION division.

### 28.1.328 4.1.328 OR

Logical operation. See AND, NOT. OpenCOBOL supports COBOL's logical expression shortcuts. Order of precedence can be controlled with parenthesis, and default to NOT, AND, OR, right to left.

```
IF A NOT EQUAL 1 OR 2 OR 3 OR 5
    DISPLAY "FORE!" END-DISPLAY
END-IF
```

### 28.1.329 4.1.329 ORDER

Sort clause to influence how duplicates are managed.

```
sort sort-work
  ascending key work-rec with duplicates in order
  using sort-in
  giving sort-out.
```

In 1.1pre-rel, WITH DUPLICATES IN ORDER is a default.

### 28.1.330 4.1.330 ORGANIZATION

Defines a file's storage organization. One of INDEXED, RELATIVE, SEQUENTIAL. OpenCOBOL also supports a LINE SEQUENTIAL structure.

### 28.1.331 4.1.331 OTHER

File sharing option, ALL OTHER, NO OTHER.

EVALUATE's else clause.

```
OCOBOL*> Here be dragons <*
  EVALUATE TRUE
    WHEN a IS 1
      PERFORM paragraph-1
    WHEN OTHER
      ALTER paragraph-1 TO paragraph-2
      PERFORM paragraph-3
  END-EVALUATE
```

### 28.1.332 4.1.332 OUTPUT

File OPEN mode. Procedure named in SORT

```
sort sort-work
  on descending key work-rec
  collating sequence is mixed
  input procedure is sort-transform
  output procedure is output-uppercase.
```

### 28.1.333 4.1.333 OVERFLOW

Declarative clause for STRING and UNSTRING that will trigger on space overflow conditions.

### 28.1.334 4.1.334 OVERLINE

A display control for SCREEN section fields.

### 28.1.335 4.1.335 OVERRIDE

Unsupported Object COBOL METHOD-ID clause.

**28.1.336 4.1.336 PACKED-DECIMAL**

Numeric USAGE clause, equivalent to COMPUTATIONAL-3. Holds each digit in a 4-bit field.

From the opencobol-2.0 tarball testsuite

OCOBOL

```

IDENTIFICATION DIVISION.
PROGRAM-ID.      prog.
DATA             DIVISION.
WORKING-STORAGE SECTION.

01 G-1.
   02 X-1          PIC 9(1) VALUE 1
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-2.
   02 X-2          PIC 9(2) VALUE 12
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-3.
   02 X-3          PIC 9(3) VALUE 123
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-4.
   02 X-4          PIC 9(4) VALUE 1234
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-5.
   02 X-5          PIC 9(5) VALUE 12345
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-6.
   02 X-6          PIC 9(6) VALUE 123456
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-7.
   02 X-7          PIC 9(7) VALUE 1234567
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-8.
   02 X-8          PIC 9(8) VALUE 12345678
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-9.
   02 X-9          PIC 9(9) VALUE 123456789
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-10.
   02 X-10         PIC 9(10) VALUE 1234567890
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-11.
   02 X-11         PIC 9(11) VALUE 12345678901
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

01 G-12.
   02 X-12         PIC 9(12) VALUE 123456789012
                   PACKED-DECIMAL.

   02 FILLER      PIC X(18) VALUE SPACE.

```

```

01 G-13.
  02 X-13          PIC 9(13) VALUE 1234567890123
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-14.
  02 X-14          PIC 9(14) VALUE 12345678901234
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-15.
  02 X-15          PIC 9(15) VALUE 123456789012345
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-16.
  02 X-16          PIC 9(16) VALUE 1234567890123456
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-17.
  02 X-17          PIC 9(17) VALUE 12345678901234567
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-18.
  02 X-18          PIC 9(18) VALUE 123456789012345678
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-S1.
  02 X-S1          PIC S9(1) VALUE -1
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-S2.
  02 X-S2          PIC S9(2) VALUE -12
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-S3.
  02 X-S3          PIC S9(3) VALUE -123
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-S4.
  02 X-S4          PIC S9(4) VALUE -1234
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-S5.
  02 X-S5          PIC S9(5) VALUE -12345
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-S6.
  02 X-S6          PIC S9(6) VALUE -123456
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-S7.
  02 X-S7          PIC S9(7) VALUE -1234567
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-S8.
  02 X-S8          PIC S9(8) VALUE -12345678
                  PACKED-DECIMAL.
  02 FILLER       PIC X(18) VALUE SPACE.
01 G-S9.
  02 X-S9          PIC S9(9) VALUE -123456789

```

```
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
01 G-S10.  
02 X-S10      PIC S9(10) VALUE -1234567890  
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
01 G-S11.  
02 X-S11      PIC S9(11) VALUE -12345678901  
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
01 G-S12.  
02 X-S12      PIC S9(12) VALUE -123456789012  
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
01 G-S13.  
02 X-S13      PIC S9(13) VALUE -1234567890123  
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
01 G-S14.  
02 X-S14      PIC S9(14) VALUE -12345678901234  
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
01 G-S15.  
02 X-S15      PIC S9(15) VALUE -123456789012345  
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
01 G-S16.  
02 X-S16      PIC S9(16) VALUE -1234567890123456  
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
01 G-S17.  
02 X-S17      PIC S9(17) VALUE -12345678901234567  
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
01 G-S18.  
02 X-S18      PIC S9(18) VALUE -123456789012345678  
          PACKED-DECIMAL.  
02 FILLER      PIC X(18) VALUE SPACE.  
PROCEDURE      DIVISION.  
*>  Dump all values <*>  
      CALL "dump" USING G-1  
      END-CALL.  
      CALL "dump" USING G-2  
      END-CALL.  
      CALL "dump" USING G-3  
      END-CALL.  
      CALL "dump" USING G-4  
      END-CALL.  
      CALL "dump" USING G-5  
      END-CALL.  
      CALL "dump" USING G-6  
      END-CALL.  
      CALL "dump" USING G-7  
      END-CALL.  
      CALL "dump" USING G-8  
      END-CALL.  
      CALL "dump" USING G-9  
      END-CALL.
```



```
CALL "dump" USING G-10
END-CALL.
CALL "dump" USING G-11
END-CALL.
CALL "dump" USING G-12
END-CALL.
CALL "dump" USING G-13
END-CALL.
CALL "dump" USING G-14
END-CALL.
CALL "dump" USING G-15
END-CALL.
CALL "dump" USING G-16
END-CALL.
CALL "dump" USING G-17
END-CALL.
CALL "dump" USING G-18
END-CALL.
CALL "dump" USING G-S1
END-CALL.
CALL "dump" USING G-S2
END-CALL.
CALL "dump" USING G-S3
END-CALL.
CALL "dump" USING G-S4
END-CALL.
CALL "dump" USING G-S5
END-CALL.
CALL "dump" USING G-S6
END-CALL.
CALL "dump" USING G-S7
END-CALL.
CALL "dump" USING G-S8
END-CALL.
CALL "dump" USING G-S9
END-CALL.
CALL "dump" USING G-S10
END-CALL.
CALL "dump" USING G-S11
END-CALL.
CALL "dump" USING G-S12
END-CALL.
CALL "dump" USING G-S13
END-CALL.
CALL "dump" USING G-S14
END-CALL.
CALL "dump" USING G-S15
END-CALL.
CALL "dump" USING G-S16
END-CALL.
CALL "dump" USING G-S17
END-CALL.
CALL "dump" USING G-S18
END-CALL.
INITIALIZE X-1.
CALL "dump" USING G-1
END-CALL.
INITIALIZE X-2.
```

```
CALL "dump" USING G-2
END-CALL.
INITIALIZE X-3.
CALL "dump" USING G-3
END-CALL.
INITIALIZE X-4.
CALL "dump" USING G-4
END-CALL.
INITIALIZE X-5.
CALL "dump" USING G-5
END-CALL.
INITIALIZE X-6.
CALL "dump" USING G-6
END-CALL.
INITIALIZE X-7.
CALL "dump" USING G-7
END-CALL.
INITIALIZE X-8.
CALL "dump" USING G-8
END-CALL.
INITIALIZE X-9.
CALL "dump" USING G-9
END-CALL.
INITIALIZE X-10.
CALL "dump" USING G-10
END-CALL.
INITIALIZE X-11.
CALL "dump" USING G-11
END-CALL.
INITIALIZE X-12.
CALL "dump" USING G-12
END-CALL.
INITIALIZE X-13.
CALL "dump" USING G-13
END-CALL.
INITIALIZE X-14.
CALL "dump" USING G-14
END-CALL.
INITIALIZE X-15.
CALL "dump" USING G-15
END-CALL.
INITIALIZE X-16.
CALL "dump" USING G-16
END-CALL.
INITIALIZE X-17.
CALL "dump" USING G-17
END-CALL.
INITIALIZE X-18.
CALL "dump" USING G-18
END-CALL.
INITIALIZE X-S1.
CALL "dump" USING G-S1
END-CALL.
INITIALIZE X-S2.
CALL "dump" USING G-S2
END-CALL.
INITIALIZE X-S3.
CALL "dump" USING G-S3
```

```
END-CALL.  
INITIALIZE X-S4.  
CALL "dump" USING G-S4  
END-CALL.  
INITIALIZE X-S5.  
CALL "dump" USING G-S5  
END-CALL.  
INITIALIZE X-S6.  
CALL "dump" USING G-S6  
END-CALL.  
INITIALIZE X-S7.  
CALL "dump" USING G-S7  
END-CALL.  
INITIALIZE X-S8.  
CALL "dump" USING G-S8  
END-CALL.  
INITIALIZE X-S9.  
CALL "dump" USING G-S9  
END-CALL.  
INITIALIZE X-S10.  
CALL "dump" USING G-S10  
END-CALL.  
INITIALIZE X-S11.  
CALL "dump" USING G-S11  
END-CALL.  
INITIALIZE X-S12.  
CALL "dump" USING G-S12  
END-CALL.  
INITIALIZE X-S13.  
CALL "dump" USING G-S13  
END-CALL.  
INITIALIZE X-S14.  
CALL "dump" USING G-S14  
END-CALL.  
INITIALIZE X-S15.  
CALL "dump" USING G-S15  
END-CALL.  
INITIALIZE X-S16.  
CALL "dump" USING G-S16  
END-CALL.  
INITIALIZE X-S17.  
CALL "dump" USING G-S17  
END-CALL.  
INITIALIZE X-S18.  
CALL "dump" USING G-S18  
END-CALL.  
MOVE ZERO TO X-1.  
CALL "dump" USING G-1  
END-CALL.  
MOVE ZERO TO X-2.  
CALL "dump" USING G-2  
END-CALL.  
MOVE ZERO TO X-3.  
CALL "dump" USING G-3  
END-CALL.  
MOVE ZERO TO X-4.  
CALL "dump" USING G-4  
END-CALL.
```

```
MOVE ZERO TO X-5.  
CALL "dump" USING G-5  
END-CALL.  
MOVE ZERO TO X-6.  
CALL "dump" USING G-6  
END-CALL.  
MOVE ZERO TO X-7.  
CALL "dump" USING G-7  
END-CALL.  
MOVE ZERO TO X-8.  
CALL "dump" USING G-8  
END-CALL.  
MOVE ZERO TO X-9.  
CALL "dump" USING G-9  
END-CALL.  
MOVE ZERO TO X-10.  
CALL "dump" USING G-10  
END-CALL.  
MOVE ZERO TO X-11.  
CALL "dump" USING G-11  
END-CALL.  
MOVE ZERO TO X-12.  
CALL "dump" USING G-12  
END-CALL.  
MOVE ZERO TO X-13.  
CALL "dump" USING G-13  
END-CALL.  
MOVE ZERO TO X-14.  
CALL "dump" USING G-14  
END-CALL.  
MOVE ZERO TO X-15.  
CALL "dump" USING G-15  
END-CALL.  
MOVE ZERO TO X-16.  
CALL "dump" USING G-16  
END-CALL.  
MOVE ZERO TO X-17.  
CALL "dump" USING G-17  
END-CALL.  
MOVE ZERO TO X-18.  
CALL "dump" USING G-18  
END-CALL.  
MOVE ZERO TO X-S1.  
CALL "dump" USING G-S1  
END-CALL.  
MOVE ZERO TO X-S2.  
CALL "dump" USING G-S2  
END-CALL.  
MOVE ZERO TO X-S3.  
CALL "dump" USING G-S3  
END-CALL.  
MOVE ZERO TO X-S4.  
CALL "dump" USING G-S4  
END-CALL.  
MOVE ZERO TO X-S5.  
CALL "dump" USING G-S5  
END-CALL.  
MOVE ZERO TO X-S6.
```

```

CALL "dump" USING G-S6
END-CALL.
MOVE ZERO TO X-S7.
CALL "dump" USING G-S7
END-CALL.
MOVE ZERO TO X-S8.
CALL "dump" USING G-S8
END-CALL.
MOVE ZERO TO X-S9.
CALL "dump" USING G-S9
END-CALL.
MOVE ZERO TO X-S10.
CALL "dump" USING G-S10
END-CALL.
MOVE ZERO TO X-S11.
CALL "dump" USING G-S11
END-CALL.
MOVE ZERO TO X-S12.
CALL "dump" USING G-S12
END-CALL.
MOVE ZERO TO X-S13.
CALL "dump" USING G-S13
END-CALL.
MOVE ZERO TO X-S14.
CALL "dump" USING G-S14
END-CALL.
MOVE ZERO TO X-S15.
CALL "dump" USING G-S15
END-CALL.
MOVE ZERO TO X-S16.
CALL "dump" USING G-S16
END-CALL.
MOVE ZERO TO X-S17.
CALL "dump" USING G-S17
END-CALL.
MOVE ZERO TO X-S18.
CALL "dump" USING G-S18
END-CALL.
STOP RUN.

```

With a support file to dump the first 10 bytes of each record

```

#include <stdio.h>
#ifdef __INTEL_COMPILER
#pragma warning ( disable : 1419 )
#endif
int dump (unsigned char *data);
int dump (unsigned char *data)
{
    int i;
    for (i = 0; i < 10; i++)
        printf ("%02x", data[i]);
    puts ("");
    return 0;
}
/**/

```

Which captures:

1f20202020202020202020  
012f202020202020202020  
123f202020202020202020  
01234f2020202020202020  
12345f2020202020202020  
0123456f20202020202020  
1234567f20202020202020  
012345678f202020202020  
123456789f202020202020  
01234567890f2020202020  
12345678901f2020202020  
0123456789012f20202020  
1234567890123f20202020  
012345678901234f202020  
123456789012345f202020  
01234567890123456f2020  
12345678901234567f2020  
0123456789012345678f2020  
1d20202020202020202020  
012d202020202020202020  
123d202020202020202020  
01234d2020202020202020  
12345d2020202020202020  
0123456d20202020202020  
1234567d20202020202020  
012345678d202020202020  
123456789d202020202020  
01234567890d2020202020  
12345678901d2020202020  
0123456789012d20202020  
1234567890123d20202020  
012345678901234d202020  
123456789012345d202020  
01234567890123456d2020  
12345678901234567d2020  
0123456789012345678d2020  
0f20202020202020202020  
000f202020202020202020  
000f202020202020202020  
00000f2020202020202020  
00000f2020202020202020  
0000000f20202020202020  
0000000f20202020202020  
000000000f202020202020  
000000000f202020202020  
0000000000f202020202020  
00000000000f2020202020  
000000000000f2020202020  
0000000000000f2020202020  
00000000000000f2020202020  
000000000000000f2020202020  
0000000000000000f2020202020  
00000000000000000f2020202020  
000000000000000000f2020202020  
0000000000000000000f2020202020  
00000000000000000000f2020202020  
0c20202020202020202020  
000c202020202020202020  
000c202020202020202020  
00000c2020202020202020

```
00000c20202020202020
0000000c202020202020
0000000c202020202020
000000000c2020202020
000000000c2020202020
00000000000c20202020
00000000000c20202020
000000000000c202020
000000000000c202020
0000000000000c2020
0000000000000c2020
0000000000000c20
000000000000000c20
000000000000000c
0f2020202020202020
000f20202020202020
000f20202020202020
00000f202020202020
00000f202020202020
0000000f2020202020
0000000f2020202020
000000000f20202020
000000000f20202020
000000000f20202020
0000000000f20202020
00000000000f202020
000000000000f202020
0000000000000f2020
00000000000000f20
000000000000000f20
0000000000000000f
0c2020202020202020
000c20202020202020
000c20202020202020
00000c202020202020
00000c202020202020
0000000c2020202020
0000000c2020202020
000000000c20202020
000000000c20202020
000000000c20202020
000000000c20202020
00000000000c202020
0000000000000c20
000000000000000c20
0000000000000000c
```

### 28.1.337 4.1.337 PADDING

Defines a character to use for short record padding.

```
ORGANIZATION IS LINE SEQUENTIAL PADDING CHARACTER IS '*'
```

### 28.1.338 4.1.338 PAGE

Write and Report writer clause.

```
WRITE theline AFTER ADVANCING PAGE

PAGE LIMITS ARE 66 LINES 132 COLUMNS
HEADING IS 4 FIRST DETAIL IS 6
LAST CONTROL HEADING IS 58
LAST DETAIL IS 60
FOOTING IS 62
```

### 28.1.339 4.1.339 PAGE-COUNTER

A special register, qualified by Report Name. Report Writer is recognized but not yet supported.

### 28.1.340 4.1.340 PARAGRAPH

An allowable EXIT point.

```
NAMED-PARAGRAPH.
  PERFORM FOREVER
    IF solution
      EXIT PARAGRAPH
    END-IF
  PERFORM solve-the-puzzle.
END-PERFORM.
```

### 28.1.341 4.1.341 PERFORM

A COBOL procedural and inline control flow verb.

```
beginning.
  PERFORM FOREVER
    PERFORM miracles
  END-PERFORM
GOBACK.

miracles.
  DISPLAY wonders END-DISPLAY.
```

### 28.1.342 4.1.342 PF

Report Writer alias for PAGE FOOTING.

### 28.1.343 4.1.343 PH

Report Writer alias for PAGE HEADING.



## 28.1.344 4.1.344 PIC

A commonly used shorthand of PICTURE.

## 28.1.345 4.1.345 PICTURE

The PICTURE clause is easily one of COBOL's greatest strengths. Fully detailed pictorial data definitions. The internal complexity is left to compiler authors, while developers and management are free to describe data at a very high conceptual level.

The two most common picture characters are 9 and X, for numeric and alphanumeric data respectively. For alphabetic data, A can be used.

Aside from data storage pictures, a vast array of *edit* pictures are allowed for control of input and output formatting.

+, -, A, B, N, X, Z, "\*", 'CR', 'DB', E, S, V, ., P, currency symbol

OpenCOBOL offers full standards support of all alpha, alphanumeric and numeric storage specifiers as well as full support for edit and numeric-edit clauses.

An example of some of the PICTURE options

```
*>>source format is free
*> *****
*> Author:    jrjs (John Ellis)
*> Date:      Oct-2008
*> Purpose:   formatted output examples using pic strings.
*> *****
```

identification division.

program-id. picstring.

data division.

working-storage section.

\*><\*

01 header.

```
05 filler          pic xxx value "ln".
05 filler          pic x(11) value "  disp1".
05 filler          pic x(11) value "  disp2".
05 filler          pic x(11) value "  disp3".
05 filler          pic x(11) value "  disp4".
05 filler          pic x(12) value "  disp5".
05 filler          pic x(9) value "  an1".
05 filler          pic x(14) value "  phone".
05 filler          pic x(10) value "  date".
```

\*><\*

01 headerLines pic x(90) value all "-".

\*><\*

01 displayformats.

```
05 linenum          pic 99 value 1.
05 disp1           pic zzz,zz9.99 value zero.
05 filler          pic x value spaces.
05 disp2           pic $zz,zz9.99 value zero.
05 filler          pic x value spaces.
05 disp3           pic ---,--9.99 value zero.
05 filler          pic x value spaces.
05 disp4           pic $-z,zz9.99 value zero.
05 filler          pic x value spaces.
05 disp5           pic -zz,zz9.zz- blank zero value zero.
```

```

05 filler          pic x value spaces.
*><*an1 is actually a string field because of the embedded blanks, thus you put value spaces.
05 an1            pic 99b99b99 value spaces.
05 filler          pic x value spaces.
05 phone          pic bxxxxbxxxxbxxxx value spaces.
05 filler          pic x value spaces.
05 dispdate       pic 99/99/9999 value zero.
*><*

```

procedure division.

0000-start.

\*><\*

```

display headerLines.
display header.
display headerLines.

```

\*><\*\*\*\*\*

```

move 220.22          to disp1,
                    disp2.

move -220.22         to disp3,
                    disp4,
                    disp5.

```

```

inspect disp5 replacing first "-" by "(",
                    first "-" by ")".

```

```

move 10122008        to dispdate.

```

\*><\*\*\*\*\*

```

*><*Please note the results of moving 'abcd' to an1.
*><*an1 will show up as 00 00 00 because alpha data was
*><*moved into instead of numeric data.

```

\*><\*

```

*><*The phone field will display " abc def ghij" because
*><*'b' in the pic string.

```

\*><\*\*\*\*\*

```

move "abcd"          to an1.
move "abcdefghij"    to phone.

```

```

display displayformats.

```

```

add 1                to linenum.
move zero             to disp4,
                    disp5.

```

\*><\*\*\*\*\*

```

*><*Here after moving data to an1 and phone, I use the
*><*inspect statement to replace the blanks.

```

\*><\*\*\*\*\*

```

move "123456"        to an1.
move "5555551234"    to phone.

```

```

inspect an1 replacing all " " by "-".

```

```

inspect phone replacing first " " by "(",
                    first " " by ")",
                    first " " by "-".

```

```

display displayformats.

```

```

inspect phone converting "23456789" to "adgjmtw".

```

```

    display phone.

    perform 0010-endProgram.
*><*
0010-endProgram.
    stop run.
*><*

```

Outputs:

```

-----
ln      disp1      disp2      disp3      disp4      disp5      an1      phone      date
-----
01      220.22      $220.22    -220.22    $-220.22    (220.22)  00 00 00  abc def ghij 10/12/2008
02      220.22      $220.22    -220.22    $ 0.00      12-34-56  (555)555-1234 10/12/2008
(jjj)jjj-ladg

```

## 28.1.346 4.1.346 PLUS

Screen section relative line / column control during layout.

```

01 form-1 AUTO.
   05 LINE 01 COLUMN 01 VALUE "Form!".
   05 LINE PLUS 3 COLUMN 01 VALUE value-4.

```

## 28.1.347 4.1.347 POINTER

Allocates a restricted use variable for holding addresses.

```

01 c-handle          USAGE IS POINTER.

CALL "open-lib" RETURNING c-handle
    ON EXCEPTION
        DISPLAY "Can't link open-lib" END-DISPLAY
        STOP RUN RETURNING 1
END-CALL
IF c-handle EQUAL NULL
    DISPLAY "Can't open-lib" END-DISPLAY
    STOP RUN RETURNING 1
END-IF

CALL "use-lib" USING BY VALUE c-handle BY CONTENT "Hello" & x"00"
CALL "close-lib" USING BY VALUE c-handle

```

```

*> Interfacing with the C ABI is just a teenie-weenie bit of voodoo
*> Pass the REFERENCE or use RETURNING if C sets the value. Use
*> VALUE when you want C to have its pointer, not the
*> REFERENCE address of the COBOL POINTER. So most inits are
*> BY REFERENCE (or RETURNING) and most usage, including
*> rundown of C ABI tools, is USING BY VALUE.
*> <*>

```

## 28.1.348 4.1.348 POSITION

Alias for COLUMN in screen section layouts. Also an obsolete, recognized but not supported:

MULTIPLE FILE TAPE CONTAINS file-1 POSITION 1 file-2 POSITION 80

### 28.1.349 4.1.349 POSITIVE

Class condition.

```
IF amount IS POSITIVE
  DISPLAY "Not broke yet" END-DISPLAY
END-IF
```

### 28.1.350 4.1.350 PRESENT

Report Writer clause used for optional field and group output.

```
05 field PIC X(16) PRESENT WHEN sum > 0.
```

### 28.1.351 4.1.351 PREVIOUS

Previous key READ control for INDEXED files.

```
READ file-1 PREVIOUS RECORD
```

### 28.1.352 4.1.352 PRINTER

Special name.

```
SPECIAL-NAMES.
  PRINTER IS myprint

DISPLAY "test" UPON PRINTER END-DISPLAY
```

### 28.1.353 4.1.353 PRINTING

Report Writer declarative to SUPPRESS report printing.

### 28.1.354 4.1.354 PROCEDURE

The COBOL DIVISION that holds the executable statements. Also used with INPUT and OUTPUT sort procedures.

### 28.1.355 4.1.355 PROCEDURE-POINTER

Alias for PROGRAM-POINTER, capable of holding a callable address.

### 28.1.356 4.1.356 PROCEDURES

Debug module declarative clause.

USE FOR DEBUGGING ON ALL PROCEDURES

### 28.1.357 4.1.357 PROCEED

Used in ALTER.

```
ALTER paragraph-1 TO PROCEED TO paragraph-x
```

### 28.1.358 4.1.358 PROGRAM

An EXIT point.

```
EXIT PROGRAM.
```

### 28.1.359 4.1.359 PROGRAM-ID

The program identifier. Case sensitive, unlike all other OpenCOBOL identifiers. OpenCOBOL produces C Application Binary Interface linkable entities and this identifier must conform to those rules. Dashes in names are replaced by a hex string equivalent.

### 28.1.360 4.1.360 PROGRAM-POINTER

A data USAGE clause defining a field that can hold the executable address of a CALL routine.

```
77 callback USAGE PROGRAM-POINTER.  
...  
SET callback TO ENTRY a-program-id  
CALL callback
```

### 28.1.361 4.1.361 PROMPT

Screen section input control.

```
PROMPT IS ':'
```

### 28.1.362 4.1.362 PROPERTY

Unsupported Object COBOL phrase.

### 28.1.363 4.1.363 PROTOTYPE

Unsupported Object COBOL phrase.

### 28.1.364 4.1.364 PURGE

Unsupported Communication Section clause.

### 28.1.365 4.1.365 QUEUE

Unsupported Communication Section clause.

### 28.1.366 4.1.366 QUOTE

A figurative constant representing ""

```
DISPLAY QUOTE 123 QUOTE END-DISPLAY
```

Outputs:

```
"123"
```

### 28.1.367 4.1.367 QUOTES

A figurative constant representing ""

```
01 var PICTURE X(4).
```

```
MOVE ALL QUOTES TO var  
DISPLAY var END-DISPLAY
```

Outputs:

```
"" ""
```

### 28.1.368 4.1.368 RAISE

Exception handling. There IS support for exceptions in OpenCOBOL but it is currently fairly limited. See [FUNCTION EXCEPTION-LOCATION](#) for a sample. RAISE is not yet recognized.

### 28.1.369 4.1.369 RAISING

Exception handling. There IS support for exceptions in OpenCOBOL but it is currently limited. RAISING is not yet recognized.

### 28.1.370 4.1.370 RANDOM

A file access mode. RANDOM access allows seeks to any point in a file, usually by [KEY](#).

### 28.1.371 4.1.371 RD

Report writer DATA division, REPORT section descriptor. Currently unsupported.

```
DATA DIVISION.  
REPORT SECTION.  
RD report-1  
    PAGE LIMIT IS 66 LINES.
```

### 28.1.372 4.1.372 READ

A staple of COBOL. Read a record.

```

READ infile PREVIOUS RECORD INTO back-record
  AT END
    SET attop TO TRUE
  NOT AT END
    PERFORM cursor-calculator
END-READ

```

### 28.1.373 4.1.373 RECEIVE

An unsupported Communication Section clause.

### 28.1.374 4.1.374 RECORD

Multiple use phrase.

```

FD file
  RECORD IS VARYING IN SIZE FROM 1 TO 80 CHARACTERS
  DEPENDING ON size-field

SELECT file
  ASSIGN TO filename
  ACCESS MODE IS RANDOM
  RECORD KEY IS key-field
  ALTERNATE KEY IS alt-key WITH DUPLICATES.

READ infile NEXT RECORD INTO display-rec END-READ

```

### 28.1.375 4.1.375 RECORDING

An obsolete, recognized, but ignored file descriptor clause.

```

FD file
  RECORD IS VARYING IN SIZE FROM 1 TO 80 CHARACTERS
  DEPENDING ON size-field
  RECORDING MODE IS F.

```

### 28.1.376 4.1.376 RECORDS

Multiple use phrase.

```

UNLOCK file-1s RECORDS

```

### 28.1.377 4.1.377 RECURSIVE

Specifies a PROGRAM-ID as having the recursive attribute. Recursive sub programs can CALL themselves.

This qualifier has implications on how OpenCOBOL allocates storage. Normally storage is stacked, recursion can chew through stack space very quickly. Sub programs marked RECURSIVE are usually allocated using the memory heap.

```
PROGRAM-ID nextbigthing IS RECURSIVE.
```

### 28.1.378 4.1.378 REDEFINES

A very powerful DATA division control allowing for redefinition of memory storage, including incompatible data by type.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.      prog.
DATA             DIVISION.
WORKING-STORAGE SECTION.
01 X             PIC X.
01 G             REDEFINES X.
   02 A          PIC X.
   02 B          REDEFINES A PIC 9.
PROCEDURE       DIVISION.
   STOP RUN.
```

### 28.1.379 4.1.379 REEL

A tape device qualifier

```
CLOSE file REEL FOR REMOVAL
```

### 28.1.380 4.1.380 REFERENCE

The default COBOL CALL argument handler. CALL arguments can be

```
BY REFERENCE
BY CONTENT
BY VALUE
```

where by reference passes a reference pointer, allowing data modification inside sub programs.

### 28.1.381 4.1.381 RELATION

Unsupported.

### 28.1.382 4.1.382 RELATIVE

File organization where the position of a logical record is determined by its relative record number.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20110806
*> Purpose:     RELATIVE file organization
*> Tectonics:   cobb -g -debug -W -x relatives.cob
*> *****
identification division.
program-id. relatives.
```



```

environment division.
configuration section.
repository.
    function all intrinsic.

input-output section.
file-control.
    select optional relatives
        assign to "relatives.dat"
        file status is filestatus
        organization is relative
        access mode is dynamic
        relative key is nickname.

data division.
file section.
fd relatives.
    01 person.
        05 firstname      pic x(48).
        05 lastname       pic x(64).
        05 relationship    pic x(32).

working-storage section.
77 filestatus pic 9(2).
    88 ineof value 1 when set to false is 0.

77 satisfaction pic 9.
    88 satisfied value 1 when set to false is 0.

77 nickname    pic 9(2).

77 title-line pic x(34).
    88 writing-names value "Adding, Overwriting. 00 to finish".
    88 reading-names value "Which record? 00 to quit".
77 problem    pic x(80).

screen section.
01 detail-screen.
    05          line 1 column 1 from title-line erase eos.
    05          line 2 column 1 value "Record: ".
    05 pic 9(2) line 2 column 16 using nickname.
    05          line 3 column 1 value "First name: ".
    05 pic x(48) line 3 column 16 using firstname.
    05          line 4 column 1 value "Last name: ".
    05 pic x(64) line 4 column 16 using lastname.
    05          line 5 column 1 value "Relation: ".
    05 pic x(32) line 5 column 16 using relationship.
    05 pic x(80) line 6 column 1 from problem.

01 show-screen.
    05          line 1 column 1 from title-line erase eos.
    05          line 2 column 1 value "Record: ".
    05 pic 9(2) line 2 column 16 using nickname.
    05          line 3 column 1 value "First name: ".
    05 pic x(48) line 3 column 16 from firstname.
    05          line 4 column 1 value "Last name: ".
    05 pic x(64) line 4 column 16 from lastname.
    05          line 5 column 1 value "Relation: ".

```

```

05 pic x(32) line 5 column 16 from relationship.
05 pic x(80) line 6 column 1 from problem.

*> -*****-*****-*****-*****-*****-*****-*****-*****-*****-*****
procedure division.
beginning.

*> Open the file and find the highest record number
*> which is a sequential read operation after START
    open input relatives

    move 99 to nickname
    start relatives key is less than or equal to nickname
        invalid key
            move concatenate('NO START' space filestatus)
                to problem
            move 00 to nickname
        not invalid key
            read relatives next end-read
    end-start

*> Close and open for i-o
    close relatives
    open i-o relatives

*> Prompt for numbers and names to add until 00
    set writing-names to true
    set satisfied to false
    perform fill-file through fill-file-end
        until satisfied

    close relatives

*> Prompt for numbers to view names of until 00
    open input relatives

    set reading-names to true
    set satisfied to false
    perform record-request through record-request-end
        until satisfied

    perform close-shop
.
ending.
goback.

*> get some user data to add
fill-file.
    display detail-screen end-display.
    accept detail-screen end-accept.
    move spaces to problem
    if nickname equal 0
        set satisfied to true
        go to fill-file-end
    end-if.
.
write-file.
    write person

```

```

        invalid key
        move concatenate("overwriting: " nickname) to problem
        rewrite person
            invalid key
            move concatenate(
                exception-location() space nickname
                space filestatus)
            to problem
            end-rewrite
        end-write.
    display detail-screen end-display

.
fill-file-end.
.

*> get keys to display
record-request.
    display show-screen end-display
    accept show-screen end-accept
    move spaces to problem
    if nickname equals 0
        set satisfied to true
        go to record-request-end
    end-if
.

*> The magic of relative record number reads
read-relation.
    read relatives
        invalid key
            move exception-location() to problem
        not invalid key
            move spaces to problem
    end-read
    display show-screen end-display
.

record-request-end.
.

*> get out <*>
close-shop.
    close relatives.
    goback.
.
end program relatives.

```

with sample screens:

```

Adding, Overwriting. 00 to finish
Record:      04
First name:  Brad_____
Last name:   Tiffin_____
Relation:    brother_____

```

allowing for new record additions or overwrites of existing key numbers, and:

```
Which record?          00 to quit
Record:                03
First name:            Brian
Last name:             Tiffin
Relation:
```

where typing in a *nickname* record number retrieves the relative record.

### 28.1.383 4.1.383 RELEASE

Release a record to a SORT. Used with INPUT PROCEDURE of SORT verb.

```
RELEASE record-1 FROM identifier-1
```

### 28.1.384 4.1.384 REMAINDER

Access to integer remainders during division.

```
DIVIDE
    hex-val BY 16 GIVING left-nibble REMAINDER right-nibble
END-DIVIDE
```

### 28.1.385 4.1.385 REMOVAL

A close clause.

```
CLOSE filename-1 REEL FOR REMOVAL
```

Specifies that the file is stored on multiple removable tapes/disks. Not all systems support such devices.

### 28.1.386 4.1.386 RENAMES

OpenCOBOL supports regrouping of level 02-49 data items with level 66 and RENAMES.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20110606
*> Purpose:     Demonstration of 66-level datanames
*> Tectonics:   cobb
*> *****
identification division.
program-id. sixty-six.

data division.
working-storage section.
01 master.
   05 field-1 pic s9(9).
   05 field-2 pic x(16).
   05 field-3 pic x(4).
   05 field-4 pic s9(9).
66 sixty-six renames field-2.
66 group-66 renames field-2 through field-4.
```

```

*> *****
procedure division.
move -66 to field-1
move "sixtysix" to field-2
move "ABCD" to field-3
multiply field-1 by -1 giving field-4 end-multiply
display "master : " master end-display
display "field-1 : " field-1 end-display
display "sixtysix: " sixtysix end-display
display "group-66: " group-66 end-display

goback.
end program sixtysix.

```

giving:

```

$ ./sixtysix
master : 00000006vsixtysix          ABCD000000066
field-1 : -000000066
sixtysix: sixtysix
group-66: sixtysix                ABCD000000066

```

## 28.1.387 4.1.387 REPLACE

A COBOL text preprocessing operator.

```

REPLACE ==MARKER== BY ==DISPLAY "REPLACE EXAMPLE" END-DISPLAY==.
identification division.
program-id. prog.

```

```

procedure division.
MARKER
goback.
end program prog.

```

And then to see how that REPLACE is working, use *cobc* with the -E argument

```

# 1 "replacing.cob"

identification division.
program-id. prog.

procedure division.
DISPLAY "REPLACE EXAMPLE" END-DISPLAY
goback.
end program prog.

```

## 28.1.388 4.1.388 REPLACING

An INSPECT subclause. A COPY preprocessor clause.

## 28.1.389 4.1.389 REPORT

Unsupported Report Writer section and File descriptor clause.

## 28.1.390 4.1.390 REPORTING

Unsupported declarative for Report Writer.

## 28.1.391 4.1.391 REPORTS

Unsupported Report Writer file descriptor clause associating files with named reports.

## 28.1.392 4.1.392 REPOSITORY

A paragraph of the CONFIGURATION SECTION. OpenCOBOL supports the **FUNCTION ALL INTRINSIC** clause of the REPOSITORY. Allows source code to use intrinsic functions without the **FUNCTION** keyword.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20110213
*> Purpose:     Demonstrate an intrinsic function shortcut
*> Tectonics:   cobc -x functionall.cob
*> *****
  identification division.
  program-id.    functionall.

  environment division.
  configuration section.
  repository.
    function all intrinsic.

*> *****
  procedure division.
  display function pi space function e end-display
  display pi space e end-display

  goback.
  end program functionall.
```

Sample output:

```
$ cobc -x functionall.cob
$ ./functionall
3.1415926535897932384626433832795029 2.7182818284590452353602874713526625
3.1415926535897932384626433832795029 2.7182818284590452353602874713526625
```

Without the **repository** paragraph:

```
$ cobc -x functionall.cob
functionall.cob:19: Error: 'pi' undefined
functionall.cob:19: Error: 'e' undefined
```

## 28.1.393 4.1.393 REQUIRED

Recognized but ignored Screen section field attribute.

### 28.1.394 4.1.394 RESERVE

An unsupported `SELECT` clause.

### 28.1.395 4.1.395 RESET

Unsupported Report Writer data control field clause.

### 28.1.396 4.1.396 RESUME

Unsupported declarative control flow statement.

### 28.1.397 4.1.397 RETRY

Unsupported record locking wait and retry clause.

- `RETRY n TIMES`
- `RETRY FOR n SECONDS`
- `RETRY FOREVER`

### 28.1.398 4.1.398 RETURN

Return records in a `SORT OUTPUT PROCEDURE`.

### 28.1.399 4.1.399 RETURNING

Specify the destination of `CALL` results.

```
01 result PIC S9(8).
```

```
CALL "libfunc" RETURNING result END-CALL
```

Specify the return field for a sub-program.

```
PROCEDURE DIVISION USING thing RETURNING otherthing
```

### 28.1.400 4.1.400 REVERSE-VIDEO

`SCREEN` section field display attribute. Functionality dependent on terminal and operating system support and settings.

### 28.1.401 4.1.401 REWIND

A really cool lyric in the Black Eyed Peas song, “Hey Mama”.

### 28.1.402 4.1.402 REWRITE

Allow overwrite of records where primary key exists.

```
write person
  invalid key
    move concatenate("overwriting: " nickname) to problem
    rewrite person
      invalid key
        move concatenate(
          exception-location() space nickname
          space filestatus)
          to problem
      end-rewrite
  end-write.
```

### 28.1.403 4.1.403 RF

Short form for unsupported REPORT FOOTING.

### 28.1.404 4.1.404 RH

Short form for unsupported REPORT HEADING.

### 28.1.405 4.1.405 RIGHT

Ignored SYNCHRONIZED clause.

### 28.1.406 4.1.406 ROLLBACK

Recognized but not fully supported revert of transactional revert of file writes. See [COMMIT](#).

### 28.1.407 4.1.407 ROUNDED

Well defined rounding clause applied to arithmetic. Defined well enough for bank managers to feel comfortable handing their calculations over to a bunch of nerds.

```
COMPUTE total-value ROUNDED = 1.0 / 6.0 END-COMPUTE
```

### 28.1.408 4.1.408 RUN

A stopping point.

```
STOP RUN RETURNING 1
```

Terminates run regardless of nesting depth, returning control (and result) to operating system. See [GOBACK](#) and [EXIT PROGRAM](#) for other run unit terminations.



## 28.1.409 4.1.409 SAME

I-O-CONTROL clause for SAME RECORD AREA.

## 28.1.410 4.1.410 SCREEN

Screen section.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> ***** < *
*> Author:    Brian Tiffin
*> Date:      20110701
*> Purpose:   Play with 2.0 screen section
*> Tectonics: cobb
*> ***** < *
  identification division.
  program-id. screening.

  data division.
  working-storage section.
  01 some-data pic s9(9).

  screen section.
  01 detail-screen.
     03 line 1 column 1 value "title line".
     03 line 2 column 1 value "field: ".
     03 line 2 column 16 using some-data.

*> ***** < *
  procedure division.
  display detail-screen end-display
  accept detail-screen end-accept
  goback.

  end program screening.

```

being a poor representation of the plethora of field attribute control allowed in OpenCOBOL screen section.

Screen field attributes include:

- JUSTIFIED RIGHT
- BLANK WHEN ZERO
- OCCURS integer-val TIMES
- BELL, BEEP
- AUTO, AUTO-SKIP, AUTOTERMINATE
- UNDERLINE
- OVERLINE
- SECURE
- REQUIRED
- FULL
- PROMPT

- REVERSE-VIDEO
- BLANK LINE
- BLANK SCREEN
- ERASE EOL
- ERASE EOS
- SIGN IS LEADING SEPERATE CHARACTER
- SIGN IS TRAILING SEPERATE CHARACTER
- LINE NUMBER IS [PLUS] integer-val
- COLUMN NUMBER IS [PLUS] integer-val
- FOREGROUND-COLOR IS integer-val HIGHLIGHT, LOWLIGHT
- BACKGROUND-COLOR IS integer-val BLINK
- PICTURE IS picture-clause USING identifier
- PICTURE IS picture-clause FROM identifier, literal
- PICTURE IS picture-clause TO identifier
- VALUE is literal

During ACCEPT, **USING** fields are read/write, **FROM** fields are read and **TO** fields are write.

See [What are the OpenCOBOL SCREEN SECTION colour values?](#) for colour values.

### 28.1.411 4.1.411 SD

**SD** file data descriptor.

```
SD sort-file-1
  RECORD CONTAINS 80 CHARACTERS.
```

### 28.1.412 4.1.412 SEARCH

A powerful table and file search verb. See [Linear SEARCH](#) for an example.

### 28.1.413 4.1.413 SECONDS

Clause of unsupported read/write **RETRY** on lock.

### 28.1.414 4.1.414 SECTION

COBOL source code is organized in **DIVISION**, **SECTION**, paragraphs and sentences. OpenCOBOL supports user named sections and recognizes the following list of pre-defined sections.

- **CONFIGURATION**
- **INPUT-OUTPUT**
- **FILE**
- **WORKING-STORAGE**

- LOCAL-STORAGE
- LINKAGE
- REPORT (recognized but unsupported)
- SCREEN

User defined sections provide for source code organization and use of `PERFORM` with `THROUGH` for tried and true COBOL procedural programming.

### 28.1.415 4.1.415 SECURE

`SCREEN` section field attribute. Displayed as asterisks.

### 28.1.416 4.1.416 SEGMENT

Unsupported Communication section clause.

### 28.1.417 4.1.417 SELECT

`FILE-CONTROL` phrase. Associates files with names, descriptors, and options.

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT fileresource
        ASSIGN TO external-name
        FILE STATUS IS identifier
        COLLATING SEQUENCE IS alphabet-name
        LOCK MODE IS MANUAL WITH LOCK ON MULTIPLE RECORDS
        RECORD DELIMITER IS STANDARD
        RESERVE num AREA
        SHARING WITH NO OTHER
        ORGANIZATION IS INDEX
            ACCESS MODE IS DYNAMIC
            RECORD KEY IS key-field
            ALTERNATE RECORD KEY IS key-field-2 WITH DUPLICATES
            ALTERNATE RECORD KEY IS key-field-3.
```

though, naming a quick file can be as simple as

```
SELECT myfile ASSIGN TO "name.txt".
```

which will be a default `LINE SEQUENTIAL` file.

### 28.1.418 4.1.418 SELF

Unsupported Object COBOL clause.

### 28.1.419 4.1.419 SEND

Unsupported Communication section verb.

### 28.1.420 4.1.420 SENTENCE

An obsolete control flow clause. `CONTINUE` is preferred to `NEXT SENTENCE`.

### 28.1.421 4.1.421 SEPARATE

Fine tuned control over leading and trailing sign indicator.

```
77 field-1 PICTURE S9(8) SIGN IS TRAILING SEPARATE.
```

### 28.1.422 4.1.422 SEQUENCE

Controls `COLLATING` sequence for character compares, by defining a character set.

### 28.1.423 4.1.423 SEQUENTIAL

OpenCOBOL supports both fixed length `SEQUENTIAL` and newline terminated `LINE SEQUENTIAL` file access.

### 28.1.424 4.1.424 SET

- `SET ADDRESS OF ptr-var TO var.`
- `SET ENVIRONMENT "name" TO "value".`
- `SET cond-1 TO TRUE`

That last one is pretty cool. An 88 level conditional set `TRUE` will cause the associated value to change to a value that satisfies the condition as true.

```
01 field-1 pic 99.  
88 cond-1 value 42.
```

```
MOVE 0 TO field-1  
DISPLAY field-1 END-DISPLAY  
SET cond-1 TO TRUE  
DISPLAY field-1 END-DISPLAY
```

00 and 42 are displayed.

### 28.1.425 4.1.425 SHARING

File sharing option.

- `SHARING WITH NO OTHER`
- `SHARING WITH ALL OTHER`
- `SHARING WITH READ ONLY`

Functionality dependent on build options and operating system running OpenCOBOL.

### 28.1.426 4.1.426 SIGN

Fine tuned control over leading and trailing sign indicator.

```
77 field-1 PICTURE S9(8) SIGN IS TRAILING SEPARATE.
```

### 28.1.427 4.1.427 SIGNED

OpenCOBOL supports the full gamut of COBOL numeric data storage. SIGNED and UNSIGNED being part and parcel.

### 28.1.428 4.1.428 SIGNED-INT

A native storage format NUMERIC data USAGE clause. Equivalent to BINARY-LONG, BINARY-LONG SIGNED, and SIGNED-LONG.

### 28.1.429 4.1.429 SIGNED-LONG

A native storage format NUMERIC data USAGE clause. Equivalent to BINARY-LONG, BINARY-LONG SIGNED, and SIGNED-INT.

### 28.1.430 4.1.430 SIGNED-SHORT

A native storage format NUMERIC data USAGE clause. Equivalent to BINARY-SHORT SIGNED.

### 28.1.431 4.1.431 SIZE

Multi purpose.

OpenCOBOL allows SIZE IS control on CALL arguments.

Arithmetic operations allow for declaratives on size errors.

```
ADD 1 TO ocobol
  ON SIZE ERROR
    SET erroneous TO TRUE
  NOT ON SIZE ERROR
    DISPLAY "Whee, ADD 1 TO COBOL" END-DISPLAY
END-ADD
```

STRING has a DELIMITED BY SIZE option to include entire fields.

### 28.1.432 4.1.432 SORT

OpenCOBOL supports USING, GIVING as well as INPUT PROCEDURE and OUTPUT PROCEDURE clauses for the SORT verb.

```
OCOBOL* OpenCOBOL SORT verb example using standard in and standard out
  identification division.
  program-id. sorting.
```

```
environment division.
input-output section.
file-control.
    select sort-in
        assign keyboard
        organization line sequential.
    select sort-out
        assign display
        organization line sequential.
    select sort-work
        assign "sortwork".

data division.
file section.
fd sort-in.
    01 in-rec          pic x(255).
fd sort-out.
    01 out-rec         pic x(255).
sd sort-work.
    01 work-rec        pic x(255).

procedure division.
sort sort-work
    ascending key work-rec
    using sort-in
    giving sort-out.

goback.
exit program.
end program sorting.
```

In the next sample, demonstrating INPUT PROCEDURE and OUTPUT PROCEDURE take note of the `RETURN` and `RELEASE` verbs as they are key to record by record control over sort operations.

Also, just to complicate things, this sample sorts using a mixed-case alphabet (but also places capital A out of order to demonstrate special cases that can codified in an `ALPHABET`).

`OCOBOL >>SOURCE FORMAT IS FIXED`

```
*****
* Author:    Brian Tiffin
* Date:      02-Sep-2008
* Purpose:   An OpenCOBOL SORT verb example
* Tectonics: cobb -x sorting.cob
*           ./sorting <input >output
*           or simply
*           ./sorting
*           for keyboard and screen demos
*****
identification division.
program-id. sorting.

environment division.
configuration section.
* This sets up a sort order lower then upper except for A and a
special-names.
    alphabet mixed is " AabBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTu
-"UvVwWxXyYzZ0123456789".

input-output section.
```

```

file-control.
  select sort-in
    assign keyboard
    organization is line sequential.
  select sort-out
    assign display
    organization is line sequential.
  select sort-work
    assign "sortwork".

data division.
file section.
fd sort-in.
  01 in-rec          pic x(255).
fd sort-out.
  01 out-rec         pic x(255).
sd sort-work.
  01 work-rec        pic x(255).

working-storage section.
01 loop-flag        pic x value low-value.

procedure division.
sort sort-work
  on descending key work-rec
  collating sequence is mixed
  input procedure is sort-transform
  output procedure is output-uppercase.

display sort-return end-display.
goback.

*****
sort-transform.
move low-value to loop-flag
open input sort-in
read sort-in
  at end move high-value to loop-flag
end-read
perform
  until loop-flag = high-value
    move FUNCTION LOWER-CASE(in-rec) to work-rec
    release work-rec
    read sort-in
      at end move high-value to loop-flag
    end-read
end-perform
close sort-in
.

*****
output-uppercase.
move low-value to loop-flag
open output sort-out
return sort-work
  at end move high-value to loop-flag
end-return
perform

```

```
    until loop-flag = high-value
      move FUNCTION UPPER-CASE(work-rec) to out-rec
      write out-rec end-write
      return sort-work
      at end move high-value to loop-flag
    end-return
  end-perform
close sort-out
.

exit program.
end program sorting.
```

Here is a snippet describing TABLE sorts by [jrls\_swla]

table define

```
01  nbr-of-columns  pic 9(4) value zero.
01  tcindex2       usage is index.
01  dbtables.
    03  tables-columns occurs 1 to 1000 times
         depending on nbr-of-columns
         ascending key tcTable, tcColumn
         indexed by tcindex.
    05  tcTable     pic x(64)  value spaces.
    05  tcColumn   pic x(64)  value spaces.
    05  tcAlias    pic x(10)  value spaces.
    05  tcOrder    pic 9(4)   value zero.
    05  tcType     pic x(10)  value spaces.
    05  tcMaxLen   pic 9(4)   value zero.
*><*
01  aliasName.
    05  pic x value "t".
    05  anVal      pic 9(3)   value zero.

01  showdata.
    05  sdTable    pic x(17)  value spaces.
    05  sdColumn  pic x(17)  value spaces.
    05  sdType    pic x(10)  value spaces.
    05  sdOrder   pic zzzzz-.
    05  sdMaxLen  pic zzzzz.
```

table load

```
perform varying rows from 1 by 1
  until rows > dbNumRows
  call "dbNextRow"      using by value dbResult,
                           by reference ColumnBuff,
                           by reference CbuffDesc
                           returning dbResult
  add 1                to nbr-of-columns
  set tcindex          up by 1
  move cbTable         to tcTable(tcindex)
  move cbColumn        to tcColumn(tcindex)
  move cbType          to tcType(tcindex)
  move cbOrder         to tcOrder(tcindex)
  move cbMaxLen        to tcMaxLen(tcindex)
  if nbr-of-columns = 1
```



```

        add 1          to anVal
    else
        set tcindex2   to tcindex
        set tcindex2   down by 1
        if cbTable <> tcTable(tcindex2)
            add 1       to anVal
        end-if
    end-if
    move aliasName     to tcAlias(tcindex)
end-perform.

```

table sort

```

    sort tables-columns ascending key tcTable, tcColumn.

```

display table

```

perform varying tcindex from 1 by 1
until tcindex > nbr-of-columns
move tcTable(tcindex)      to sdTable
move tcColumn(tcindex)     to sdColumn
move tcOrder(tcindex)      to sdOrder
move tcType(tcindex)       to sdType
move tcMaxLen(tcindex)     to sdMaxLen
display showdata
end-perform.

```

Excercise for the audience. Could the above code be simplified by using

```

MOVE CORRESPONDING cbRecord to table-columns(tcindex)
...
MOVE CORRESPONDING table-columns(tcindex) to showdata

```

with a few judicious field name changes?

#### 4.1.432.1 An OCSORT support tool

There is an external sort utility referenced in [What is ocsort?](#)

### 28.1.433 4.1.433 SORT-MERGE

Used in an I-O-CONTROL paragraph with the SAME clause:

```

SAME SORT-MERGE AREA FOR filename-1.

```

The SORT-MERGE keyword and SORT keyword are equivalent in this case.

### 28.1.434 4.1.434 SORT-RETURN

A *SPECIAL-REGISTER* used by the OpenCOBOL SORT routines.

- +000000000 for success
- +000000016 for failure

A programmer may set SORT-RETURN in an INPUT PROCEDURE.

### 28.1.435 4.1.435 SOURCE

Compiler directive controlling source code handling.

```
>>SOURCE FORMAT IS FIXED
>>SOURCE FORMAT IS FREE
```

OpenCOBOL allows use of this directive at programmer whim. **cobc** defaults to FIXED format source handling, so the directive must occur beyond the sequence and indicator columns unless the **-free** compile option is used.

Split keys are a pending feature in OpenCOBOL.

```
SELECT ...
  RECORD KEY IS key-name SOURCE is dname-2 dname-3
```

Also a pending Report Writer data source clause.

### 28.1.436 4.1.436 SOURCE-COMPUTER

A paragraph of the IDENTIFICATION division. Treated as a comment.

### 28.1.437 4.1.437 SOURCES

Currently unsupported SOURCES ARE report writer clause.

### 28.1.438 4.1.438 SPACE

A figurative constant representing a space character.

### 28.1.439 4.1.439 SPACES

A figurative constant representing space characters.

### 28.1.440 4.1.440 SPECIAL-NAMES

OpenCOBOL supports a fair complete set of the SPECIAL-NAMES in common use.

- CONSOLE IS CRT
- SYSIN IS mnemonic-name-1
- SYSOUT IS
- SYSLIST IS
- SYSLST IS
- PRINTER IS
- SYSERR IS
- CONSOLE IS mnemonic-name-7
- SWITCH-1 IS mnemonic-name-n ON STATUS IS condition-name-1 OFF STATUS IS condition-name-2
- SWITCH-2

- ...
- SWITCH-8 IS ...
- C01 IS mnemonic-name-m
- ...
- C12 IS
- ALPHABET alphabet-name IS NATIVE, STANDARD-1, STANDARD-2, EBCDIC literal-1 THRU literal-2 [ALSO literal-3]
- SYMBOLIC CHARACTERS symbol-character IS integer-1 IN alphabet-name
- CLASS class-name IS literal THRU literal-2
- LOCALE locale-name IS identifier-1
- CURRENCY SIGN IS literal
- DECIMAL-POINT IS COMMA
- CURSOR IS identifier-1
- CRT STATUS IS identifier-1
- SCREEN CONTROL IS identifier-1 **PENDING**
- EVENT STATUS IS identifier-1 **PENDING**

#### **28.1.441 4.1.441 STANDARD**

- LABEL RECORDS ARE STANDARD

#### **28.1.442 4.1.442 STANDARD-1**

- ALPHABET IS STANDARD-1
- RECORD DELIMITER IS STANDARD-1

equivalent to [ASCII](#)

#### **28.1.443 4.1.443 STANDARD-2**

- ALPHABET IS STANDARD-1
- RECORD DELIMITER IS STANDARD-1

equivalent to [ASCII](#)

#### **28.1.444 4.1.444 START**

Sets internal file fields that will influence sequential [READ NEXT](#) and [READ PREVIOUS](#) for [INDEXED](#) files. Can also be used to seek to the [FIRST](#) or [LAST](#) record of a file for [SEQUENTIAL](#) access modes.

```
start indexing
  key is less than
    keyfield of indexing-record
  invalid key
    display
      "bad start: " keyfield of indexing-record
    end-display
    set no-more-records to true
  not invalid key
    read indexing previous record
      at end set no-more-records to true
    end-read
end-start
```

The conditionals are quite powerful.

```
KEY IS GREATER THAN
KEY IS >
KEY IS LESS THAN
KEY IS <
KEY IS EQUAL TO
KEY IS =

KEY IS NOT GREATER THAN
KEY IS NOT >
KEY IS NOT LESS THAN
KEY IS NOT <
KEY IS NOT EQUAL TO
KEY IS NOT =

KEY IS <>
KEY IS GREATER THAN OR EQUAL TO
KEY IS >=
KEY IS LESS THAN OR EQUAL TO
KEY IS <=
```

See [Does OpenCOBOL support ISAM?](#) for some example source code.

### 28.1.445 4.1.445 STATEMENT

Unsupported.

### 28.1.446 4.1.446 STATUS

Multi-purpose.

- CRT STATUS IS
- FILE STATUS IS
- EVENT STATUS IS
- SWITCH-1 IS thing ON STATUS IS conditional-1

### 28.1.447 4.1.447 STEP

Unsupported Report Writer OCCURS subclause.

## 28.1.448 4.1.448 STOP

End a run and return control to the operating system.

```
STOP RUN RETURNING 5.
```

Forms include:

- STOP RUN
- STOP RUN RETURNING stat
- STOP RUN GIVING stat
- STOP literal
- STOP RUN WITH ERROR STATUS stat
- STOP RUN WITH NORMAL STATUS stat

## 28.1.449 4.1.449 STRING

String together a set of variables with controlled delimiters.

```
01 var PICTURE X(5).
```

```
STRING
```

```
"abc" DELIMITED BY "b"
```

```
"def" DELIMITED BY SIZE
```

```
"ghi" DELIMITED BY "z"
```

```
INTO var
```

```
ON OVERFLOW
```

```
    DISPLAY "var is full at" SPACE LENGTH OF var END-DISPLAY
```

```
END-STRING
```

```
DISPLAY var END-DISPLAY
```

Outputs:

```
var is full at 5
adefg
```

OpenCOBOL also fully supports the WITH POINTER clause to set the initial and track the position in the output character variable.

## 28.1.450 4.1.450 STRONG

Unsupported.

## 28.1.451 4.1.451 SUB-QUEUE-1

Unsupported Communication section clause.

## 28.1.452 4.1.452 SUB-QUEUE-2

Unsupported Communication section clause.

### 28.1.453 4.1.453 SUB-QUEUE-3

Unsupported Communication section clause.

### 28.1.454 4.1.454 SUBTRACT

Arithmetic operation.

```
SUBTRACT a b c FROM d ROUNDED END-SUBTRACT
```

```
SUBTRACT a FROM b GIVING c
  ON SIZE ERROR
    SET math-error TO TRUE
  NOT ON SIZE ERROR
    SET math-error TO FALSE
END-SUBTRACT
```

```
SUBTRACT CORRESPONDING record-a FROM record-b ROUNDED
  ON SIZE ERROR
    SET something-wrong TO TRUE
END-SUBTRACT
```

### 28.1.455 4.1.455 SUM

A REPORT SECTION control break summation field clause. Unsupported.

### 28.1.456 4.1.456 SUPER

Unsupported Object COBOL clause.

### 28.1.457 4.1.457 SUPPRESS

Unsupported declarative to suppress printing.

### 28.1.458 4.1.458 SYMBOL

Unsupported.

### 28.1.459 4.1.459 SYMBOLIC

SPECIAL-NAMES clause allowing user defined figurative constants.

### 28.1.460 4.1.460 SYNC

Alias for SYNCHRONIZED

### 28.1.461 4.1.461 SYNCHRONIZED

Control padding inside record definitions, in particular to match C structures.

```

01 infile.
   03 slice    occurs 64 times depending on slices.
       05 stext usage pointer synchronized.
       05 val   float-long   synchronized.
       05 ftext usage pointer synchronized.

```

### 28.1.462 4.1.462 SYSTEM-DEFAULT

OBJECT-COMPUTER clause for locale support.

```
CHARACTER CLASSIFICATION IS SYSTEM-DEFAULT
```

### 28.1.463 4.1.463 TABLE

Unsupported keyword, but OpenCOBOL fully supports tables, including SORT.

### 28.1.464 4.1.464 TALLYING

INSPECT clause for counting occurrences of a literal.

```
INSPECT record-1 TALLYING ident-1 FOR LEADING "0"
```

### 28.1.465 4.1.465 TAPE

A device type used in ASSIGN.

### 28.1.466 4.1.466 TERMINAL

Unsupported Communication section clause.

### 28.1.467 4.1.467 TERMINATE

Currently unsupported Report Writer verb to finish up a report. See INITIATE.

### 28.1.468 4.1.468 TEST

Allows control over when loop conditionals are tested. WITH TEST BEFORE is the default. WITH TEST AFTER will always evaluate the body of the loop at least once.

```

perform
  with test after
  varying x from 1 by xstep
  until x >= function e
    if x > function e
      move function e to x-value

```

```
    else
        move x to x-value
    end-if
    compute recip = 1 / x end-compute
    move recip to y-value
    write outrec end-write
end-perform
```

### 28.1.469 4.1.469 TEXT

Unsupported Communication section clause.

### 28.1.470 4.1.470 THAN

Part of the conditional clauses for readability.

```
IF A GREATER THAN 10
    DISPLAY "A > 10" END-DISPLAY
END-IF
```

### 28.1.471 4.1.471 THEN

A somewhat disdained keyword that is part of the IF THEN ELSE control structure.

```
IF A > 10 THEN
    DISPLAY "A GREATER THAN 10" END-DISPLAY
ELSE
    DISPLAY "A LESS THAN OR EQUAL TO 10" END-DISPLAY
END-IF
```

### 28.1.472 4.1.472 THROUGH

Used in definitions for alphabets in [SPECIAL-NAMES](#) and a procedural clause allowing [PERFORM](#) from one label THROUGH (inclusive) to another label and all paragraphs in between. Also used to specify grouping with [RENAMES](#).

```
PERFORM 100-open-files THROUGH 100-files-end
```

### 28.1.473 4.1.473 THRU

Commonly used alias for [THROUGH](#)

### 28.1.474 4.1.474 TIME

An [ACCEPT FROM](#) source. Allows access to current clock.

```
01 current-time.
   05 ct-hours      pic 99.
   05 ct-minutes    pic 99.
   05 ct-seconds    pic 99.
   05 ct-hundredths pic 99.
```



```
ACCEPT current-time FROM TIME
```

## 28.1.475 4.1.475 TIMES

A counted loop.

```
PERFORM 5 TIMES  
    DISPLAY "DERP" END-DISPLAY  
END-PERFORM
```

## 28.1.476 4.1.476 TO

Multi-purpose destination specifier.

```
ADD 1 TO cobol GIVING OpenCOBOL  
    ON SIZE ERROR  
        DISPLAY "Potential exceeds expectations" END-DISPLAY  
END-ADD
```

## 28.1.477 4.1.477 TOP

A LINAGE clause.

## 28.1.478 4.1.478 TRAILING

Multi-purpose. FUNCTION TRIM allows a TRAILING keyword. An INSPECT TALLYING subclause.

## 28.1.479 4.1.479 TRUE

A SET target. Used in EVALUATE to control when the operation succeeds. When used with a conditional 88 level name, will set the corresponding field to a listed VALUE.

```
01 field-1 pic x.  
   88 cond-1 values 'a','b','c'.
```

```
SET cond-1 TO TRUE  
DISPLAY field-1 END-DISPLAY
```

## 28.1.480 4.1.480 TYPE

An unsupported Report Writer report group clause. Also unsupported data description clause.

## 28.1.481 4.1.481 TYPEDEF

Unsupported data description clause that will allow user defined record layouts.

### 28.1.482 4.1.482 UCS-4

Currently unsupported Universal Character Set alphabet. UCS-4 would store international code points in 4 bytes.

### 28.1.483 4.1.483 UNDERLINE

SCREEN section field attribute.

### 28.1.484 4.1.484 UNIT

A close option, alias for REEL.

```
CLOSE file-1 UNIT WITH NO REWIND
```

### 28.1.485 4.1.485 UNIVERSAL

Unsupported Object COBOL exception object clause.

### 28.1.486 4.1.486 UNLOCK

Manual record unlock and buffer write sync.

```
UNLOCK filename-1 RECORDS
```

### 28.1.487 4.1.487 UNSIGNED

Usage clause specifying that a value will not include any sign and therefore can't be negative.

### 28.1.488 4.1.488 UNSIGNED-INT

A native storage format NUMERIC data USAGE clause. Equivalent to BINARY-LONG UNSIGNED and UNSIGNED-LONG.

### 28.1.489 4.1.489 UNSIGNED-LONG

A native storage format NUMERIC data USAGE clause. Equivalent to BINARY-LONG UNSIGNED and UNSIGNED-INT.

### 28.1.490 4.1.490 UNSIGNED-SHORT

A native storage format NUMERIC data USAGE clause. Equivalent to BINARY-SHORT UNSIGNED and UNSIGNED-SHORT.

## 28.1.491 4.1.491 UNSTRING

A powerful string decomposition verb.

```

UNSTRING Input-Address
  DELIMITED BY ", " OR "/"
  INTO
    Street-Address DELIMITER D1 COUNT C1
    Apt-Number DELIMITER D2 COUNT C2
    City DELIMITER D3 COUNT C3
    State DELIMITER D4 COUNT C4
    Zip-Code DELIMITER D5 COUNT C5
  WITH POINTER ptr-1
  ON OVERFLOW
    SET more-fields TO TRUE
END-UNSTRING

```

## 28.1.492 4.1.492 UNTIL

Sets a loop conditional.

```

PERFORM VARYING ident-1 FROM 1 BY 1 UNTIL ident-1 > 10
  CALL "thing" USING BY VALUE ident-1 END-CALL
END-PERFORM

```

## 28.1.493 4.1.493 UP

Index and pointer modification.

```

SET ptr-1 UP BY 4
SET ind-1 UP BY 1

```

## 28.1.494 4.1.494 UPDATE

SCREEN section field attribute.

## 28.1.495 4.1.495 UPON

A DISPLAY destination clause.

## 28.1.496 4.1.496 USAGE

OpenCOBOL uses standard **big-endian** internal storage by default. USAGE clauses influence the data representation. The INTEL architecture uses **little-endian** form and OpenCOBOL programmers developing for this common chipset may need to pay heed to this for performance purposes. As per the standards, OpenCOBOL supports COMPUTATIONAL-5 native usage.

OpenCOBOL enables use of one to eight byte binary representations in both big and little endian forms.

Along with full support of all common COBOL PICTURE clauses both storage and display, OpenCOBOL supports USAGE clauses of:

- BINARY

- COMPUTATIONAL, COMP
- COMP-1
- COMP-2
- COMP-3
- COMP-4
- COMP-5
- COMP-X
- FLOAT-LONG
- FLOAT-SHORT
- DISPLAY
- INDEX
- PACKED-DECIMAL
- POINTER
- PROGRAM-POINTER
- SIGNED-SHORT
- SIGNED-INT
- SIGNED-LONG
- UNSIGNED-SHORT
- UNSIGNED-INT
- UNSIGNED-LONG
- BINARY-CHAR SIGNED
- BINARY-CHAR UNSIGNED
- BINARY-CHAR
- BINARY-SHORT SIGNED
- BINARY-SHORT UNSIGNED
- BINARY-SHORT
- BINARY-LONG SIGNED
- BINARY-LONG UNSIGNED
- BINARY-LONG
- BINARY-DOUBLE SIGNED
- BINARY-DOUBLE UNSIGNED
- BINARY-DOUBLE
- BINARY-C-LONG SIGNED
- BINARY-C-LONG UNSIGNED
- BINARY-C-LONG

### 28.1.497 4.1.497 USE

Sets up `DECLARATIVES` paragraphs.

- `USE BEFORE DEBUGGING`
- `USE AFTER EXECEPTION`

### 28.1.498 4.1.498 USER-DEFAULT

`OBJECT-COMPUTER` clause for locale support.

```
CHARACTER CLASSIFICATION IS USER-DEFAULT
```

### 28.1.499 4.1.499 USING

Specifies arguments to `CALL` and in `PROCEDURE` declarations.

- `BY REFERENCE` (default, pointer to modifiable data is passed)
- `BY CONTENT` (reference to a copy of the data)
- `BY VALUE` (actual dereferenced value is placed into call frame)

### 28.1.500 4.1.500 UTF-16

Unsupported internationalization clause.

### 28.1.501 4.1.501 UTF-8

Unsupported internationalization clause.

### 28.1.502 4.1.502 VAL-STATUS

Alias for the unsupported `VALIDATE-STATUS` clause of the `VALIDATE` statement.

### 28.1.503 4.1.503 VALID

Unsupported.

### 28.1.504 4.1.504 VALIDATE

Unsupported data validation verb.

### 28.1.505 4.1.505 VALIDATE-STATUS

Unsupported clause of the `VALIDATE` statement.

### 28.1.506 4.1.506 VALUE

An `CALL` frame argument modifier. Sets values in data descriptions as well as values used with 88 level conditional names.

### 28.1.507 4.1.507 VALUES

Plural of `VALUE` when more than one entry is used in an 88 conditional name.

### 28.1.508 4.1.508 VARYING

Sets a looping variable.

```
PERFORM VARYING loop-counter FROM 1 BY 1 UNTIL loop-counter > 10
    DISPLAY loop-counter END-DISPLAY
END-PERFORM
```

### 28.1.509 4.1.509 WHEN

A very powerful keyword used in `EVALUATE` phrases for specifying conditional expressions.

```
EVALUATE TRUE
    WHEN A = 10
        DISPLAY "A = 10" END-DISPLAY
    WHEN A = 15
        PERFORM A-IS-15
    WHEN B IS EQUAL 6
        PERFORM B-IS-6
    WHEN C IS GREATER THAN 5
        DISPLAY "C > 5" END-DISPLAY
    WHEN OTHER
        DISPLAY "Default imperative" END-DISPLAY
END-EVALUATE
```

### 28.1.510 4.1.510 WITH

Multi-purpose.

- WITH LOCK
- DISPLAY WITH screen-attribute
- WITH ROLLBACK (pending)

### 28.1.511 4.1.511 WORKING-STORAGE

A `DATA` division section. Unless `BASED`, all fields are allocated and fixed in memory (for the running program within a module).

### 28.1.512 4.1.512 WRITE

Record write. Unlike `READ` that uses filenames syntax, `WRITE` uses record buffer syntax which must be related to the file through the `FD` file descriptor. OpenCOBOL supports `LINAGE` and `WRITE` has support for 'report' paging and line control.

```
WRITE record-buff END-WRITE
```

```
WRITE record-name-1 AFTER ADVANCING PAGE END-WRITE.
```

```
WRITE record-name-1
  AT END-OF-PAGE
  DISPLAY "EOP" END-DISPLAY
END-WRITE
```

### 28.1.513 4.1.513 YYYYDDD

Modifies `ACCEPT` var FROM `DAY` to use full 4 digit year for the Julian date retrieval.

```
ACCEPT date-var FROM DAY YYYYDDD
```

### 28.1.514 4.1.514 YYYYMMDD

Modifies `ACCEPT` var FROM `DATE` to use full 4 digit year.

```
ACCEPT date-var FROM DATE YYYYMMDD
```

### 28.1.515 4.1.515 ZERO

Figurative and numeric constant for the value 0.

### 28.1.516 4.1.516 ZEROES

Plural of `ZERO`.

### 28.1.517 4.1.517 ZEROS

Alternate spelling for `ZEROES`.

## 28.2 4.2 Does OpenCOBOL implement any Intrinsic FUNCTIONS?

Yes, many. As of the July 2008 1.1 pre-release





**Intrinsic FUNCTION**

- 4.2.1 FUNCTION ABS
- 4.2.2 FUNCTION ACOS
- 4.2.3 FUNCTION ANNUITY
- 4.2.4 FUNCTION ASIN
- 4.2.5 FUNCTION ATAN
- 4.2.6 FUNCTION BYTE-LENGTH
- 4.2.7 FUNCTION CHAR
- 4.2.8 FUNCTION COMBINED-DATETIME
- 4.2.9 FUNCTION CONCATENATE
- 4.2.10 FUNCTION COS
- 4.2.11 FUNCTION CURRENT-DATE
- 4.2.12 FUNCTION DATE-OF-INTEGERS
- 4.2.13 FUNCTION DATE-TO-YYYYMMDD
- 4.2.14 FUNCTION DAY-OF-INTEGERS
- 4.2.15 FUNCTION DAY-TO-YYYYDDD
- 4.2.16 FUNCTION E
- 4.2.17 FUNCTION EXCEPTION-FILE
- 4.2.18 FUNCTION EXCEPTION-LOCATION
- 4.2.19 FUNCTION EXCEPTION-STATEMENT
- 4.2.20 FUNCTION EXCEPTION-STATUS
- 4.2.21 FUNCTION EXP
- 4.2.22 FUNCTION EXP10
- 4.2.23 FUNCTION FACTORIAL
- 4.2.24 FUNCTION FRACTION-PART
- 4.2.25 FUNCTION INTEGER
- 4.2.26 FUNCTION INTEGER-OF-DATE
- 4.2.27 FUNCTION INTEGER-OF-DAY
- 4.2.28 FUNCTION INTEGER-PART
- 4.2.29 FUNCTION LENGTH
- 4.2.30 FUNCTION LOCALE-DATE
- 4.2.31 FUNCTION LOCALE-TIME
- 4.2.32 FUNCTION LOCALE-TIME-FROM-SECONDS
- 4.2.33 FUNCTION LOG
- 4.2.34 FUNCTION LOG10
- 4.2.35 FUNCTION LOWER-CASE
- 4.2.36 FUNCTION MAX
- 4.2.37 FUNCTION MEAN
- 4.2.38 FUNCTION MEDIAN
- 4.2.39 FUNCTION MIDRANGE
- 4.2.40 FUNCTION MIN
- 4.2.41 FUNCTION MOD
- 4.2.42 FUNCTION NUMVAL
- 4.2.43 FUNCTION NUMVAL-C
- 4.2.44 FUNCTION ORD
- 4.2.45 FUNCTION ORD-MAX
- 4.2.46 FUNCTION ORD-MIN
- 4.2.47 FUNCTION PI
- 4.2.48 FUNCTION PRESENT-VALUE
- 4.2.49 FUNCTION RANDOM
- 4.2.50 FUNCTION RANGE
- 4.2.51 FUNCTION REM
- 4.2.52 FUNCTION REVERSE
- 4.2.53 FUNCTION SECONDS-FROM-FORMATTED-TIME

- 4.2.54 FUNCTION SECONDS-PAST-MIDNIGHT
- 4.2.55 FUNCTION SIGN
- 4.2.56 FUNCTION SIN
- 4.2.57 FUNCTION SQRT
- 4.2.58 FUNCTION STANDARD-DEVIATION

ABS, ACOS, ANNUITY, ASIN, ATAN, BYTE-LENGTH, CHAR, CONCATENATE, COS, CURRENT-DATE, DATE-OF-INTEGERS, DATE-TO-YYYYMMDD, DAY-OF-INTEGERS, DAY-TO-YYYYDDD, E, EXCEPTION-FILE, EXCEPTION-LOCATION, EXCEPTION-STATEMENT, EXCEPTION-STATUS, EXP, EXP10, FACTORIAL, FRACTION-PART, INTEGER, INTEGER-OF-DATE, INTEGER-OF-DAY, INTEGER-PART, LENGTH, LOCALE-DATE, LOCALE-TIME, LOG, LOG10, LOWER-CASE, MAX, MEAN, MEDIAN, MIDRANGE, MIN, MOD, NUMVAL, NUMVAL-C, ORD, ORD-MAX, ORD-MIN, PI, PRESENT-VALUE, RANDOM, RANGE, REM, REVERSE, SECONDS-FROM-FORMATTED-TIME, SECONDS-PAST-MIDNIGHT, SIGN, SIN, SQRT, STANDARD-DEVIATION, STORED-CHAR-LENGTH, SUBSTITUTE, SUBSTITUTE-CASE, SUM, TAN, TEST-DATE-YYYYMMDD, TEST-DAY-YYYYDDD, TRIM, UPPER-CASE, VARIANCE, WHEN-COMPILED, YEAR-TO-YYYY

## 28.2.1 4.2.1 FUNCTION ABS

Absolute value of numeric argument

```
DISPLAY FUNCTION ABS(DIFFERENCE).
```

## 28.2.2 4.2.2 FUNCTION ACOS

The ACOS function returns a numeric value (in radians) that approximates the arccosine of the argument.

The domain of the arccosine function is -1 to +1. Domain errors return a result of 0. The inverse cosine function returns a range of 0 thru  $\pi$

```
DISPLAY FUNCTION ACOS(-1).
```

## 28.2.3 4.2.3 FUNCTION ANNUITY

Compute the ratio of an annuity paid based on arguments of interest and number of periods.

**WORKING-STORAGE SECTION.**

```
77 INTEREST          PIC S9V9999 VALUE 0.08.
77 MONTHLY           PIC S9V9999 VALUE ZERO.
77 PERIODS           PIC 99          VALUE 36.
77 ANNUITY-VALUE     PIC S9V9999 VALUE ZERO.
```

**PROCEDURE DIVISION.**

```
    COMPUTE MONTHLY ROUNDED = INTEREST / 12
    COMPUTE ANNUITY-VALUE ROUNDED =
        FUNCTION ANNUITY (MONTHLY PERIODS)
    DISPLAY "Monthly rate: " MONTHLY
        " Periods: " PERIODS
        " Annuity ratio: " ANNUITY-VALUE
    END-DISPLAY.
```

Outputs:

```
Monthly rate: +0.0067 Periods: 36 Annuity ratio: +0.0314
```

## 28.2.4 4.2.4 FUNCTION ASIN

The ASIN function returns a numeric value (in radians) that approximates the arcsine of the argument.

The domain of the arcsine function is  $-1$  to  $+1$ . Domain errors return a result of  $0$ . The inverse sine function returns a range of  $-\pi/2$  thru  $\pi/2$

```
DISPLAY FUNCTION ASIN(-1) .
```

## 28.2.5 4.2.5 FUNCTION ATAN

The ATAN function returns a numeric value (in radians) that approximates the arctangent of the argument.

The domain of the arctangent function is all real numbers. The inverse tangent function returns a range of  $-\pi/2$  thru  $\pi/2$

```
DISPLAY FUNCTION ATAN(1) .
```

## 28.2.6 4.2.6 FUNCTION BYTE-LENGTH

The BYTE-LENGTH function returns an integer that is the internal storage length of the given argument.

```
COBOL >>SOURCE FORMAT IS FIXED
*****
* Purpose:  demonstrate intrinsic FUNCTION BYTE-LENGTH
*****
identification division.
program-id. bytelength.

data division.
working-storage section.
01 char-var          usage binary-char.
01 short-var         usage binary-short.
01 long-var          usage binary-long.
01 double-var        usage binary-double.

01 num1-var          pic 9.
01 num4-var          pic 99v99.
01 num9-var          pic s9(9).
01 num18-var         pic s9(18).
01 num18c-var        pic s9(18) usage comp.
01 num18p-var        pic s9(18) usage comp-3.
01 edit-var          pic $zzzz9.99.

01 string-var        pic x(10) value "abc".

01 newline           pic x value x'0a'.

procedure division.
display
"num1-var   len = " function byte-length(num1-var) newline
"num4-var   len = " function byte-length(num4-var) newline
"num9-var   len = " function byte-length(num9-var) newline
"num18-var  len = " function byte-length(num18-var) newline
"num18c-var len = " function byte-length(num18c-var) newline
"num18p-var len = " function byte-length(num18p-var) newline
"edit-var   len = " function byte-length(edit-var) newline

"12         len = " function byte-length(12) newline
"12.12      len = " function byte-length(12.12) newline
```

```
"1234567890.123 = " function
    byte-length(1234567890.123) newline

"string-var len = " function byte-length(string-var) newline
"trim string    = " function
    byte-length(function trim(string-var)) newline

"char-var  len = " function byte-length(char-var) newline
"short-var len = " function byte-length(short-var) newline
"long-var  len = " function byte-length(long-var)  newline
"double-var len = " function byte-length(double-var)

end-display
goback.
exit program.
```

#### Outputs:

```
num1-var  len = 1
num4-var  len = 4
num9-var  len = 9
num18-var len = 18
num18c-var len = 8
num18p-var len = 10
edit-var  len = 9
12        len = 2
12.12     len = 4
1234567890.123 = 13
string-var len = 10
trim string    = 00000003
char-var  len = 1
short-var  len = 2
long-var   len = 4
double-var len = 8
```

## 28.2.7 4.2.7 FUNCTION CHAR

The CHAR function returns a ONE character alphanumeric field whose value is the character in the current collating sequence having the ordinal position equal to the value of the integer argument. The argument must be greater than 0 and less than or equal to the number of positions in the collating sequence. Errors in the argument range return 0 (the LOW-VALUE by default).

See [ASCII](#) or [EBCDIC](#) and details of the [ALPHABET](#) clause.

```
DISPLAY FUNCTION CHAR(66).
```

Would output **A** in the ASCII character set. Note this may be different than what some expect. OpenCOBOL CHAR is 1 thru 128 not 0 thru 127 as a C programmer may be used to.

*And to add a little confusion, most personal computers use an extended character set, usually erroneously called ASCII with a range of 0 to 255. A more appropriate name may be ISO-8859-1 Latin 1. See [ASCII](#) for more accurate details. This author is often guilty of this misnomer of the use of the term ASCII.*

## 28.2.8 4.2.8 FUNCTION COMBINED-DATETIME

Returns a common datetime form from integer date (years and days from 1600 to 10000) and numeric time arguments (seconds in day). Date should be from 1 to 3067671 and time should be from 1 to 86400. The character string returned

is in the form 7.5.

```
DISPLAY FUNCTION COMBINED-DATETIME (1; 1) END-DISPLAY
```

Outputs:

```
0000001.00001
```

## 28.2.9 4.2.9 FUNCTION CONCATENATE

Concatenate the given fields. CONCATENATE is an OpenCOBOL extension.

```
MOVE "COBOL" TO stringvar
MOVE FUNCTION CONCATENATE ("Open"; stringvar) TO goodsystem
DISPLAY goodsystem END-DISPLAY
```

## 28.2.10 4.2.10 FUNCTION COS

The COS function returns a numeric value that approximates the cosine of the argument (in radians).

The domain of the cosine function is all real numbers, with a nominal domain of 0 thru  $\pi$  with a zero returned at  $\pi/2$ . The cosine function returns a range of -1 thru +1.

```
DISPLAY FUNCTION COS (1.5707963267949) .
```

## 28.2.11 4.2.11 FUNCTION CURRENT-DATE

Returns an alphanumeric field of length 21 with the current date, time and timezone information in the form YYYYMMDDhhmmsscc±tznn

```
DISPLAY FUNCTION CURRENT-DATE .
```

Example Output:

```
2008080921243796-0400
```

## 28.2.12 4.2.12 FUNCTION DATE-OF-INTEGGER

Converts an integer date, days on the Gregorian since December 31 1600 to YYYYMMDD form

```
DISPLAY DATE-OF-INTEGGER (1)
DISPLAY DATE-OF-INTEGGER (50000)
```

Outputs:

```
16010101
17371123
```

50,000 days after December 31, 1600 being November 23rd, 1737.

### 28.2.13 4.2.13 FUNCTION DATE-TO-YYYYMMDD

Converts a two digit year date format to four digit year form using a sliding window pivot of the optional second argument. The pivot defaults to 50.

The OpenCOBOL implementation of DATE-TO-YYYYMMDD also accepts an optional third argument, replacing the default century value of 1900 and is treated as the years added to the given year portion of the first argument and modified by the sliding 100 window pivot.

Domain errors occur for year values less than 1600 and greater than 999,999. There is no validation of the input date.

Because of the sliding window, this function is dependent on the date of evaluation

```
DISPLAY FUNCTION DATE-TO-YYYYMMDD (000101)
DISPLAY FUNCTION DATE-TO-YYYYMMDD (500101)
DISPLAY FUNCTION DATE-TO-YYYYMMDD (610101)
DISPLAY FUNCTION DATE-TO-YYYYMMDD (990101)

DISPLAY FUNCTION DATE-TO-YYYYMMDD (990101, 50, 1900)
DISPLAY FUNCTION DATE-TO-YYYYMMDD (990101, -10, 1900)
DISPLAY FUNCTION DATE-TO-YYYYMMDD (990101, 50, 2000)
DISPLAY FUNCTION DATE-TO-YYYYMMDD (990101, 50, 2100)
```

When run in August, 2008 produces:

```
20000101
20500101
19610101
19990101
18990101
17990101
19990101
20990101
```

### 28.2.14 4.2.14 FUNCTION DAY-OF-INTEGERS

Converts a Gregorian integer date form to Julian date form (YYYYDDD) based on days since December 31, 1600. Errors return 0

```
DISPLAY FUNCTION DAY-OF-INTEGERS (97336) .
1867182
```

97,336 days after 16001231 being the 182nd day of the year 1867. Canada's date of Confederation and recognized birthday.

### 28.2.15 4.2.15 FUNCTION DAY-TO-YYYYDDD

Converts a Julian 2 digit year and three digit day integer to a four digit year form. See [FUNCTION DATE-TO-YYYYMMDD](#) for some of the details of the calculations involved.

### 28.2.16 4.2.16 FUNCTION E

Returns Euler's number as an alphanumeric field to 34 digits of accuracy after the decimal. E forms the base of the natural logarithms. It has very unique and important properties such as:

- the derivative of  $e^x$  is  $e^x$

- and the area below the curve of  $y = 1/x$  for  $1 \leq x \leq e$  is exactly 1.
- making it very useful in calculations of Future Value with compound interest.

**DISPLAY FUNCTION E END-DISPLAY**

outputs:

2.7182818284590452353602874713526625

A small graph to show the magic area.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        29-May-2009, Modified 20110505 to add e tic mark
*> Purpose:     Plot Euler's number (using integral of 1 over x)
*> Tectonics:   requires access to gnuplot. http://www.gnuplot.info
*>              cobc -Wall -x ploteuler.cob
*> OVERWRITES  ocgenplot.gp, ocgpdata.txt and images/euler.png
*> *****
identification division.
program-id. ploteuler.

environment division.
input-output section.
file-control.
    select scriptfile
        assign to "ocgenplot.gp"
        organization is line sequential.
    select outfile
        assign to "ocgpdata.txt"
        organization is line sequential.

data division.
file section.
fd scriptfile.
    01 gnuplot-command pic x(82).

fd outfile.
    01 outrec.
        03 x-value    pic -z9.999.
        03 filler     pic x.
        03 y-value    pic -z9.999.

working-storage section.
    01 xstep    pic 9v99999.
    01 x        pic 9v99999.
    01 recip    pic 9v99999.

*> The plot command is xrange 0:3, y 0:2 data col 1 for x 2 for y
01 gpcmds pic x(400) value is
    "set style fill solid 1.0;           " &
    "set grid;                           " &
    "set xtics add ('e' 2.718281);        " &
    "plot [0:3] [0:2] 'ocgpdata.txt' using 1:2 \  " &
    " with filledcurves below x1 title '1/x';    " &
    "set terminal png;                       " &
    "set output 'images/euler.png';         " &
    "replot                                  ".

```

```
01 line-cnt pic 999.
01 gptable.
   05 gpcmd pic x(50) occurs 8 times.

01 gplot pic x(40) value is 'gnuplot -persist ocgenplot.gp'.
01 result pic s9(9).

*> *****
procedure division.
display function e end-display

*><* Create the script to plot the area of Euler's number
open output scriptfile.
move gpcmds to gptable
perform varying line-cnt from 1 by 1 until line-cnt > 8
   move gpcmd(line-cnt) to gnuplot-command
   write gnuplot-command end-write
end-perform
close scriptfile

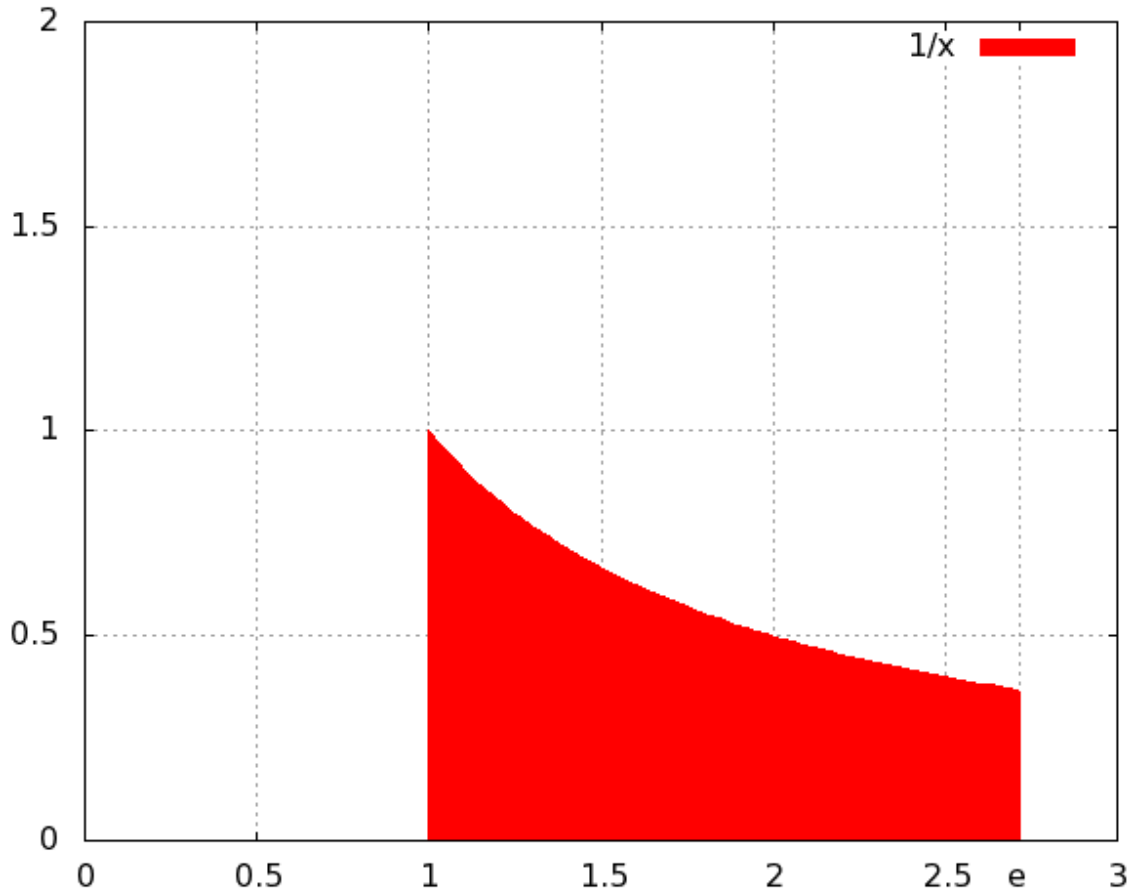
*><* Create the reciprocal data
open output outfile
move spaces to outrec
compute xstep = function e / 100 end-compute
perform
   with test after
   varying x from 1 by xstep
   until x >= function e
   if x > function e
      move function e to x-value
   else
      move x to x-value
   end-if
   compute recip = 1 / x end-compute
   move recip to y-value
   write outrec end-write
end-perform
close outfile

*><* Invoke gnuplot
call "SYSTEM" using gplot returning result end-call
if result not = 0
   display "Problem: " result end-display
   stop run returning result
end-if

goback.
end program ploteuler.
```

The area in red is exactly 1. Well, not on this plot exactly, as it is somewhat sloppy with the *xstep* end case and the precisions.





See [Can OpenCOBOL be used for plotting?](#) for some details on plotting.

### 28.2.17 4.2.17 FUNCTION EXCEPTION-FILE

This special-register holds the error number and name of the source file that caused an input output exception. See [FUNCTION EXCEPTION-STATUS](#) for an example.

### 28.2.18 4.2.18 FUNCTION EXCEPTION-LOCATION

This special-register can be queried for the location of the last exception. See [FUNCTION EXCEPTION-STATUS](#) for example source code. Note: This feature requires compilation with *-fsource-location* compiler switch. This option is also turned on with *-g* and *-debug* debugging info compiles. Information includes PROGRAM-ID, section and source line.

### 28.2.19 4.2.19 FUNCTION EXCEPTION-STATEMENT

This special-register holds the statement that was executing when the latest exception was raised. See [FUNCTION EXCEPTION-STATUS](#) for an example. Note: This feature requires compilation with *-fsource-location* compiler switch. This option is also turned on with *-g* debugging info compiles.

## 28.2.20 4.2.20 FUNCTION EXCEPTION-STATUS

This FUNCTION returns the current exception status. The example below is courtesy of Roger While, from a post he made announcing the *FUNCTION EXCEPTION-* features.

Source format is free, compile with *cobc -x -g -free except.cob*

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  MINIPROG.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  LINUX.
OBJECT-COMPUTER.  LINUX.
SPECIAL-NAMES.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT PRINTFILE ASSIGN TO "XXRXWXX"
FILE STATUS RXWSTAT.

DATA DIVISION.
FILE SECTION.
FD PRINTFILE.
01 PRINTREC PIC X(132).

WORKING-STORAGE SECTION.
01 RXWSTAT PIC XX.

PROCEDURE DIVISION.
A00-MAIN SECTION.
001-MAIN-PROCEDURE.
OPEN INPUT PRINTFILE.
DISPLAY "File Status: " RXWSTAT.
DISPLAY "EXCEPTION-FILE: " FUNCTION EXCEPTION-FILE.
DISPLAY "Return Length: "
    FUNCTION LENGTH (FUNCTION EXCEPTION-FILE).
DISPLAY "EXCEPTION-STATUS: " FUNCTION EXCEPTION-STATUS.
DISPLAY "EXCEPTION-STATEMENT: " FUNCTION EXCEPTION-STATEMENT.
STRING "TOOLONG" DELIMITED SIZE INTO RXWSTAT.
DISPLAY "EXCEPTION-STATUS: " FUNCTION EXCEPTION-STATUS.
DISPLAY "EXCEPTION-STATEMENT: " FUNCTION EXCEPTION-STATEMENT.
DISPLAY "EXCEPTION-LOCATION: " FUNCTION EXCEPTION-LOCATION.

STOP RUN.
```

Example output:

```
File Status: 35
EXCEPTION-FILE: 35PRINTFILE
Return Length: 00000011
EXCEPTION-STATUS: EC-I-O-PERMANENT-ERROR
EXCEPTION-STATEMENT: OPEN
EXCEPTION-STATUS: EC-OVERFLOW-STRING
EXCEPTION-STATEMENT: STRING
EXCEPTION-LOCATION: MINIPROG; 001-MAIN-PROCEDURE OF A00-MAIN; 29
```

---

**Tip:** See the source file `libcob/exception.def` for a list of the plethora of run-time exceptions supported by OpenCOBOL.

### 28.2.21 4.2.21 FUNCTION EXP

Returns an approximation of Euler's number (see [FUNCTION E](#)) raised to the power of the numeric argument.

```
DISPLAY FUNCTION EXP(1) END-DISPLAY
```

outputs:

```
2.718281828459045091
```

**Note:** Be aware that this approximation seems accurate to “only” 15 decimal places. Diligent programmers need to be aware of the foibles of floating point mathematics and take these issues into consideration.

### 28.2.22 4.2.22 FUNCTION EXP10

Returns an approximation of the value 10 raised to the power of the numeric argument.

```
DISPLAY FUNCTION EXP10(1.0) END-DISPLAY
DISPLAY FUNCTION EXP10(1.2) END-DISPLAY
DISPLAY FUNCTION EXP10(10) END-DISPLAY
```

Outputs:

```
10.000000000000000000
15.848931924611132871
10000000000.000000000000000000
```

### 28.2.23 4.2.23 FUNCTION FACTORIAL

Computes the factorial of the integral argument. Valid Range of 0 to 19 with a domain of 1 to 121645100408832000.

```
OCOBOL*> *****
*> Program to find range and domain of FUNCTION FACTORIAL
identification division.
program-id. fact.

data division.
working-storage section.
01 ind pic 999.
01 result pic 9(18).

*> *****
procedure division.
perform varying ind from 0 by 1 until ind > 20
    add zero to function factorial(ind) giving result
    on size error
        display "overflow at " ind end-display
    end-add
    display ind " = " function factorial(ind) end-display
end-perform

goback.
end program fact.
```

## Outputs:

```
000 = 00000000000000000001
001 = 00000000000000000001
002 = 00000000000000000002
003 = 00000000000000000006
004 = 00000000000000000024
005 = 00000000000000000120
006 = 00000000000000000720
007 = 00000000000000005040
008 = 0000000000000040320
009 = 0000000000000362880
010 = 0000000000003628800
011 = 0000000000039916800
012 = 000000000479001600
013 = 000000006227020800
014 = 000000087178291200
015 = 000001307674368000
016 = 000020922789888000
017 = 000355687428096000
018 = 006402373705728000
019 = 121645100408832000
overflow at 020
020 = 432902008176640000
```

Kind of the same thing, with some zero out formatting.

```
OCOBOL*> *****
*> Program to find range and domain of FUNCTION FACTORIAL
identification division.
program-id. fact.

data division.
working-storage section.
01 ind                pic 99.
01 z-ind              pic z9.
01 result             pic 9(18).
01 pretty-result     pic z(17)9.

*> *****
procedure division.
perform varying ind from 0 by 1 until ind > 21
    add zero to function factorial(ind) giving result
    on size error
        display
            "overflow at " ind ", result undefined: "
            function factorial(ind)
        end-display
    not on size error
        move ind to z-ind
        move result to pretty-result
        display
            "factorial(" z-ind ") = " pretty-result
        end-display
    end-add
end-perform

goback.
end program fact.
```

Which outputs:

```
factorial( 0) =          1
factorial( 1) =          1
factorial( 2) =          2
factorial( 3) =          6
factorial( 4) =         24
factorial( 5) =        120
factorial( 6) =        720
factorial( 7) =       5040
factorial( 8) =      40320
factorial( 9) =     362880
factorial(10) =    3628800
factorial(11) =   39916800
factorial(12) =  479001600
factorial(13) = 6227020800
factorial(14) = 87178291200
factorial(15) = 1307674368000
factorial(16) = 20922789888000
factorial(17) = 355687428096000
factorial(18) = 6402373705728000
factorial(19) = 121645100408832000
overflow at 20, result undefined, 432902008176640000
overflow at 21, result undefined, 197454024290336768
```

## 28.2.24 4.2.24 FUNCTION FRACTION-PART

Returns a numeric value that is the fraction part of the argument. Keeping the sign.

```
DISPLAY FUNCTION FRACTION-PART(FUNCTION E) END-DISPLAY
DISPLAY FUNCTION FRACTION-PART(-1.5) END-DISPLAY
DISPLAY FUNCTION FRACTION-PART(-1.0) END-DISPLAY
DISPLAY FUNCTION FRACTION-PART(1) END-DISPLAY
```

Outputs:

```
+ .718281828459045235
- .500000000000000000
+ .000000000000000000
+ .000000000000000000
```

## 28.2.25 4.2.25 FUNCTION INTEGER

Returns the greatest integer less than or equal to the numeric argument.

```
DISPLAY
  FUNCTION INTEGER (-3)      SPACE
  FUNCTION INTEGER (-3.141)
END-DISPLAY
DISPLAY
  FUNCTION INTEGER (3)      SPACE
  FUNCTION INTEGER (3.141)
END-DISPLAY
DISPLAY
  FUNCTION INTEGER (-0.3141) SPACE
  FUNCTION INTEGER (0.3141)  SPACE
```

```

FUNCTION INTEGER (0)
END-DISPLAY

```

Outputs:

```

-00000000000000000003 -00000000000000000004
+00000000000000000003 +00000000000000000003
-00000000000000000001 +00000000000000000000 +00000000000000000000

```

Note the -4, greatest integer **less than or equal to** the argument.

## 28.2.26 4.2.26 FUNCTION INTEGER-OF-DATE

Converts a date in the Gregorian calendar to an integer form. Expects a numeric argument in the form *YYYYMMDD* based on years greater than or equal to 1601 and less than 10000. Month values range from 1 to 12. Days range from 1 to 31 and should be valid for the specified month and year. Invalid input returns unpredictable results and sets the exception EC-ARGUMENT-FUNCTION to exist. See [FUNCTION DATE-OF-INTEG](#) for the converse function.

## 28.2.27 4.2.27 FUNCTION INTEGER-OF-DAY

Converts a Julian date of *YYYYDDD* to integer date form. See [FUNCTION DAY-OF-INTEG](#) for the converse intrinsic function. Invalid arguments return an undefined result and set the exception EC-ARGUMENT-FUNCTION to exist.

## 28.2.28 4.2.28 FUNCTION INTEGER-PART

Returns the integer part of the numeric argument. Similar to [FUNCTION INTEGER](#) but returns different values for negative arguments.

```

DISPLAY
  FUNCTION INTEGER-PART (-3)      SPACE
  FUNCTION INTEGER-PART (-3.141)
END-DISPLAY
DISPLAY
  FUNCTION INTEGER-PART (3)       SPACE
  FUNCTION INTEGER-PART (3.141)
END-DISPLAY
DISPLAY
  FUNCTION INTEGER-PART (-0.3141) SPACE
  FUNCTION INTEGER-PART (0.3141)  SPACE
  FUNCTION INTEGER-PART (0)
END-DISPLAY

```

Outputs:

```

-00000000000000000003 -00000000000000000003
+00000000000000000003 +00000000000000000003
+00000000000000000000 +00000000000000000000 +00000000000000000000

```

## 28.2.29 4.2.29 FUNCTION LENGTH

Returns an integer that is the length in character positions of the given argument.

```

working storage.
01 nat      pic n(10) .
01 cha      pic x(10) .
01 bin      constant as h'ff' .

01 num      pic s9(8)v9(8) .
01 form     pic $-z(7)9.9(8) .

procedure division.
display
    function length(nat) space
    function length(cha) space
    function length(bin)
end-display
display
    function length(num) space
    function length(form)
end-display

```

Outputs:

```

20 10 3
16 19

```

## 28.2.30 4.2.30 FUNCTION LOCALE-DATE

Returns a culturally appropriate date given an alphanumeric of 8 character positions in the form “YYYYMMDD” and an optional locale name that has been associated with a locale in the SPECIAL-NAMES paragraph.

See <http://en.wikipedia.org/wiki/Locale> for a start at the very detail rich computational requirements of LOCALE.

Will set EC-ARGUMENT-FUNCTION to exist for invalid input.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20120116
*> Purpose:     Demonstrate locale functions
*> Tectonics:   cobb -x locales.cob
*> *****

identification division.
program-id.  locales.

environment division.
configuration section.
repository.
    function all intrinsic.

*> -*****-*****-*****-*****-*****-*****-
procedure division.

*> Display cultural norm date and times as set in environment.
*>   Google LC_ALL.
*> 20120622 represents June 22 2012
*> 141516 represents 2pm (14th hour), 15 minutes, 16 seconds
*> 39600 represents 11 hours in seconds

display locale-date(20120622)          end-display

```

```
display locale-time(141516)          end-display
display locale-time-from-seconds(39600) end-display

goback.
end program locales.
```

Which produced:

```
[btiffin@home cobol]$ cobc -x locales.cob
[btiffin@home cobol]$ ./locales
06/22/2012
02:15:16 PM
11:00:00 AM
```

I live in Canada, but usually run Fedora with LANG=en\_US.utf8

and so

```
[btiffin@home cobol]$ export LANG='en_CA.utf8'
[btiffin@home cobol]$ ./locales
22/06/12
02:15:16 PM
11:00:00 AM
```

Boo, day month year form. Sad, 2 digit year? What kinda backwater land do I live in? Time to write strongly worded letters to some committees. :)

I just looked, and it seems Canada is listed as DD/MM/YY; I'm moving to Germany.

```
[btiffin@home cobol]$ export LANG=en_DK.utf8
[btiffin@home cobol]$ ./locales
2012-06-22
14:15:16
11:00:00
```

Joy. year month day. Hmm, what about Hong Kong?

```
[btiffin@home cobol]$ LANG=en_HK.utf8 ./locales
Sunday, June 22, 2012
02:15:16 EST
11:00:00 EST
```

Nice.

If you want to run your system through its locales, try

```
$ locs=( $(locale -a) )
$ for l in ${locs[@]}; do echo $l; LANG=$l ./locales; done
```

and expect some unicode in the output.

Oh, and along with FUNCTION EXCEPTION-STATUS you can detect invalid arguments.

```
000100 >>SOURCE FORMAT IS FIXED
000200*> *****
000300*> Author:      Brian Tiffin
000400*> Date:        20120116
000500*> Purpose:     Demonstrate locale function invalid arguments
000600*> Tectonics:   cobc -x -g -debug locales.cob
000700*> *****
000800 identification division.
000900 program-id.  locales.
```



```

001000
001100 environment division.
001200 configuration section.
001300 repository.
001400     function all intrinsic.
001500
001600*> -*****-*****-*****-*****-*****-*****-***
001700 procedure division.
001800
001900*> Display cultural norm date and times as set in environment.
002000*>     Google LC_ALL.
002100*> 20120622 represents June 22 2012
002200*> 141516 represents 2pm (14th hour), 15 minutes, 16 seconds
002300*> 39600 represents 11 hours in seconds
002400
002500 display locale-date(20120622)           end-display
002600 display locale-time(141516)             end-display
002700 display locale-time-from-seconds(39600) end-display
002800
002900*> invalid arguments are detected through EXCEPTION-STATUS
003000 display locale-date(20120699)           end-display
003100 DISPLAY "EXCEPTION-STATUS: " EXCEPTION-STATUS
003200 DISPLAY "EXCEPTION-STATEMENT: " EXCEPTION-STATEMENT
003300 DISPLAY "EXCEPTION-LOCATION: " EXCEPTION-LOCATION
003400
003500 display locale-time(941516)             end-display
003600 DISPLAY "EXCEPTION-STATUS: " EXCEPTION-STATUS
003700 DISPLAY "EXCEPTION-STATEMENT: " EXCEPTION-STATEMENT
003800 DISPLAY "EXCEPTION-LOCATION: " EXCEPTION-LOCATION
003900
004000 display locale-time-from-seconds(-39600) end-display
004100
004200 goback.
004300 end program locales.

```

giving:

```

$ ./locales
06/22/2012
02:15:16 PM
11:00:00 AM

```

```

EXCEPTION-STATUS: EC-ARGUMENT-FUNCTION
EXCEPTION-STATEMENT: DISPLAY
EXCEPTION-LOCATION: locales; MAIN PARAGRAPH OF MAIN SECTION; 30

```

```

EXCEPTION-STATUS: EC-ARGUMENT-FUNCTION
EXCEPTION-STATEMENT: DISPLAY
EXCEPTION-LOCATION: locales; MAIN PARAGRAPH OF MAIN SECTION; 35
-11:00:00 AM

```

## 28.2.31 4.2.31 FUNCTION LOCALE-TIME

Returns a culturally appropriate date given an alphanumeric of 6 character positions in the form “HHMMSS” and an optional locale name that has been associated with a locale in the SPECIAL-NAMES paragraph. See <http://en.wikipedia.org/wiki/Locale> for a start at the very detail rich computational requirements of LOCALE.

Will set EC-ARGUMENT-FUNCTION to exist for invalid input.

See [FUNCTION LOCALE-DATE](#).

### 28.2.32 4.2.32 FUNCTION LOCALE-TIME-FROM-SECONDS

Returns a culturally appropriate date given an alphanumeric number of seconds and an optional locale name that has been associated with a locale in the SPECIAL-NAMES paragraph.

See <http://en.wikipedia.org/wiki/Locale> for a start at the very detail rich computational requirements of LOCALE.

Will set EC-ARGUMENT-FUNCTION to exist for invalid input.

See [FUNCTION LOCALE-DATE](#).

### 28.2.33 4.2.33 FUNCTION LOG

Returns an approximation of the natural logarithmic value of the given numeric argument. Uses a base of `FUNCTION E`.

```
DISPLAY FUNCTION LOG(100) END-DISPLAY
DISPLAY FUNCTION LOG(FUNCTION E) END-DISPLAY
```

gives:

```
4.60517018598809137
000000001
```

### 28.2.34 4.2.34 FUNCTION LOG10

Returns an approximation of the base-10 logarithmic value of the given numeric argument.

```
DISPLAY FUNCTION LOG10(100) END-DISPLAY
```

gives:

```
000000002
```

### 28.2.35 4.2.35 FUNCTION LOWER-CASE

Convert any uppercase character values (A-Z) in the argument to lowercase (a-z).

### 28.2.36 4.2.36 FUNCTION MAX

Returns the maximum value from the list of arguments.

```
DISPLAY FUNCTION MAX ( "def"; "abc"; ) END-DISPLAY
DISPLAY FUNCTION MAX ( 123.1; 123.11; 123 ) END-DISPLAY
```

Outputs:

```
def
123.11
```

### 28.2.37 4.2.37 FUNCTION MEAN

Returns the arithmetic mean (average) of the list of numeric arguments.

```
DISPLAY FUNCTION MEAN(1; 2; 3; 4; 5; 6; 7; 8; 9) END-DISPLAY
```

Outputs:

```
+5.000000000000000000
```

### 28.2.38 4.2.38 FUNCTION MEDIAN

Returns the middle value of the arguments formed by arranging the list in sorted order.

```
DISPLAY FUNCTION MEDIAN(1; 2; 3; 4; 5; 6; 7; 8; 9) END-DISPLAY
```

Outputs:

```
5
```

### 28.2.39 4.2.39 FUNCTION MIDRANGE

Returns the arithmetic mean (average) of the minimum and maximum argument from the list of numeric arguments.

```
DISPLAY FUNCTION MIDRANGE(1; 2; 3; 4; 5; 6; 7; 8; 9) END-DISPLAY
```

Outputs:

```
5.000000000000000000
```

### 28.2.40 4.2.40 FUNCTION MIN

Returns the minimum value from the list of arguments.

```
DISPLAY FUNCTION MIN ( "def"; "abc"); END-DISPLAY
DISPLAY FUNCTION MIN ( 123.1; 123.11; 123) END-DISPLAY
```

Outputs:

```
abc
123
```

### 28.2.41 4.2.41 FUNCTION MOD

Returns an integer value of that is the first-argument modulo second-argument.

```
DISPLAY FUNCTION MOD(123; 23) END-DISPLAY
```

Outputs:

```
+000000000000000008
```

## 28.2.42 4.2.42 FUNCTION NUMVAL

Returns the numeric value represented by the character string argument.

```
OCOBOL IDENTIFICATION DIVISION.  
PROGRAM-ID. prog.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
01 X PIC X(12) VALUE "-9876.1234".  
01 F PIC X(12) VALUE "B-9876.1234".  
PROCEDURE DIVISION.  
DISPLAY FUNCTION NUMVAL ( X )  
DISPLAY FUNCTION NUMVAL ( F )  
END-DISPLAY.  
STOP RUN.
```

gives:

```
-09876.1234  
000000000
```

The “B” in field F, breaks the numeric conversion. NUMVAL is actually fairly complicated and forgiving of inputs, but will return 0 on invalid numeric conversions.

OpenCOBOL 2 will also provide FUNCTION TEST-NUMVAL.

## 28.2.43 4.2.43 FUNCTION NUMVAL-C

Returns the numeric value represented by the culturally appropriate currency specification argument.

```
OCOBOL IDENTIFICATION DIVISION.  
PROGRAM-ID. prog.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
01 X PIC X(14) VALUE "%-9876.1234".  
PROCEDURE DIVISION.  
DISPLAY FUNCTION NUMVAL-C ( X , "%" )  
END-DISPLAY.  
STOP RUN.
```

gives:

```
-09876.1234
```

in a LOCALE that uses the percent sign as a currency symbol.

OpenCOBOL 2 will also provide FUNCTION TEST-NUMVAL-C.

## 28.2.44 4.2.44 FUNCTION ORD

Returns the integer value that is the ordinal position of the character argument in the program’s collating sequence. COBOL uses 1 as the lowest ordinal for character sequencing.

```
DISPLAY FUNCTION ORD ("J") END-DISPLAY
```

Outputs (on an ASCII system with no ALPHABET clause):

00000075

Note that COBOL uses 1 as the first value for collating. So ASCII 74 is ORD 75 for “J”.

## 28.2.45 4.2.45 FUNCTION ORD-MAX

Returns the integer that is the ordinal position of the maximum value of the given argument list.

```
DISPLAY ORD-MAX(9; 8; 7; 6; 5; 4; 3; 2; 1) END-DISPLAY
DISPLAY ORD-MAX('abc'; 'def'; 'ghi') END-DISPLAY
```

Outputs:

```
00000001
00000003
```

## 28.2.46 4.2.46 FUNCTION ORD-MIN

Returns the integer that is the ordinal position of the minimum value from the argument list.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20090531
*> Purpose:     Demonstration of FUNCTION ORD-MIN and REPOSITORY
*> Tectonics:   cobb -x ordmin.cob
*> *****
  identification division.
  program-id.   ordmin.

  environment division.
  configuration section.
  repository.
  function all intrinsic.

  data division.
  working-storage section.
  01 posmin          pic 9(8).

*> *****
  procedure division.
  move ord-min (9; 8; 7; 6; 5; 4; 3; 2; 1; 2; 3; 4; 5) to posmin
  display posmin end-display
  move ord-min ("abc"; "def"; "000"; "def"; "abc") to posmin
  display posmin end-display
  goback.
  end program ordmin.
```

Outputs:

```
00000009
00000003
```

Notice how ord-min did not require FUNCTION, as the REPOSITORY entry allows this to be skipped in the source codes.

## 28.2.47 4.2.47 FUNCTION PI

Returns an approximation of the ratio of the circumference by the diameter of a circle. It returns an alphanumeric with 34 digits after the decimal. Please be aware of the limitations of using these types of approximated values in computations.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20101030
*> Purpose:     Demonstrate PI
*> Tectonics:   cobc -x pi-demo.cob
*> *****
identification division.
program-id. pi-demo.

data division.
working-storage section.
01 args pic x(80).
01 diameter pic 999 value 1.
01 show-diameter pic zz9.
01 circumference usage float-long.
01 plural pic xx.
01 plural-length pic 9 value 1.
01 newline pic x value x'0a'.

*> *****
procedure division.
accept args from command-line end-accept
if args not equal spaces
    move args to diameter
end-if

if diameter not equal 1
    move "s " to plural
    move 2 to plural-length
else
    move " " to plural
    move 1 to plural-length
end-if
move diameter to show-diameter

display "FUNCTION PI is " function pi newline end-display

compute circumference = function pi * diameter end-compute
display
    "A wheel, " show-diameter " metre" plural(1:plural-length)
    "wide will roll, very close to but only approximately, "
    newline circumference " metres in ONE full rotation."
    newline
end-display

goback.
end program pi-demo.
```

Outputs:

```
$ cobc -x pi-demo.cob && ./pi-demo && ./pi-demo 42
FUNCTION PI is 3.1415926535897932384626433832795029
```

A wheel, 1 metre wide will roll, very close to but only approximately, 3.14159265358979312 metres in ONE full rotation.

FUNCTION PI is 3.1415926535897932384626433832795029

A wheel, 42 metres wide will roll, very close to but only approximately, 131.946891450771318 metres in ONE full rotation.

## 28.2.48 4.2.48 FUNCTION PRESENT-VALUE

Returns an approximation of the present value from a discount rate and list of future period end amounts. It attempts to reflect the future value of \$1.00 given time, inflation and interest.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20101030
*> Purpose:     Demo of PRESENT-VALUE
*> Tectonics:   cobb -x present-value-demo.cob
*> *****
identification division.
program-id. present-value-demo.

data division.
working-storage section.
01 args pic x(80).
01 newline pic x value x'0a'.
01 rate pic s9v9999 value 0.7000.
01 the-value pic s9(6)v99.

*> *****
procedure division.
accept args from command-line end-accept

if args not equal to spaces
  move args to rate
end-if

compute the-value rounded =
  function present-value(rate; 1000, 1010, 1000, 1100)
end-compute
display
  "A discount rate of " rate " gives a PRESENT-VALUE of "
  the-value " given" newline
  "end-amounts of 1000, 1010, 1000 and 1100"
end-display

compute the-value rounded =
  function present-value(rate; 1000, 1000, 1000, 1000)
end-compute
display
  "A discount rate of " rate " gives a PRESENT-VALUE of "
  the-value " given" newline
  "end-amounts of 1000, 1000, 1000 and 1000"
end-display
```

```
goback.  
end program present-value-demo.
```

Outputs:

```
$ ./present-value-demo  
A discount rate of +0.7000 gives a PRESENT-VALUE of +001272.96 given  
end-amounts of 1000, 1010, 1000 and 1100  
A discount rate of +0.7000 gives a PRESENT-VALUE of +001257.53 given  
end-amounts of 1000, 1000, 1000 and 1000  
$ ./present-value-demo 0.333  
A discount rate of +0.3330 gives a PRESENT-VALUE of +002089.18 given  
end-amounts of 1000, 1010, 1000 and 1100  
A discount rate of +0.3330 gives a PRESENT-VALUE of +002051.88 given  
end-amounts of 1000, 1000, 1000 and 1000  
$ ./present-value-demo 0.935  
A discount rate of +0.9350 gives a PRESENT-VALUE of +001003.03 given  
end-amounts of 1000, 1010, 1000 and 1100  
A discount rate of +0.9350 gives a PRESENT-VALUE of +000993.23 given  
end-amounts of 1000, 1000, 1000 and 1000
```

For details, talk to a professional.

*rant* Any COBOL programmer using financial functions for use by others **HAS** to attain some level of *domain expertise* in the mathematics at work, as well as a level of *technical competence* to read through and defend both the COBOL source code and the generated C code that OpenCOBOL emits before compiling. *rant over*

## 28.2.49 4.2.49 FUNCTION RANDOM

Returns a pseudo-random number given a numeric seed value as argument.

```
DISPLAY FUNCTION RANDOM(1) END-DISPLAY  
DISPLAY FUNCTION RANDOM(1) END-DISPLAY  
DISPLAY FUNCTION RANDOM() END-DISPLAY
```

Outputs:

```
+00000000.1804289383  
+00000000.1804289383  
+000000000.846930886
```

## 28.2.50 4.2.50 FUNCTION RANGE

Returns the value of the minimum argument subtracted from the maximum argument from the list of numeric arguments.

```
DISPLAY FUNCTION RANGE(1; 2; 3; 4; 5; 6; 7; 8; 9) END-DISPLAY
```

Outputs:

```
+00000000000000000008
```

## 28.2.51 4.2.51 FUNCTION REM

Returns the numeric remainder of the first argument divided by the second.



```
DISPLAY FUNCTION REM(123; 23) END-DISPLAY
```

Outputs:

```
+00000000000000000008
```

## 28.2.52 4.2.52 FUNCTION REVERSE

Returns the reverse of the given character string.

```
DISPLAY FUNCTION REVERSE("abc") END-DISPLAY
```

Outputs:

```
cba
```

## 28.2.53 4.2.53 FUNCTION SECONDS-FROM-FORMATTED-TIME

This function converts a time that is in a specified format to a numeric value representing the number of seconds after midnight.

```
OCOBOL IDENTIFICATION DIVISION.
PROGRAM-ID. prog.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 X PIC X(6) VALUE "hhmmss".
01 Y PIC 9(8) COMP-5.
01 Z PIC X(6) VALUE "010203".
PROCEDURE DIVISION.
MOVE FUNCTION SECONDS-FROM-FORMATTED-TIME (X, Z) TO Y.
IF Y NOT = 3723
DISPLAY Y
END-DISPLAY
END-IF.
STOP RUN.
```

This test would fail if 01:02:03 was not returned as 3723 seconds past midnight.

Argument 1 takes the form *hhmmss* and expects argument 2 to be a matching length numeric item, or 0 is returned.

## 28.2.54 4.2.54 FUNCTION SECONDS-PAST-MIDNIGHT

Returns the seconds past midnight from the current system time.

## 28.2.55 4.2.55 FUNCTION SIGN

Returns +1 for positive, 0 for zero and -1 for a negative numeric argument.

## 28.2.56 4.2.56 FUNCTION SIN

Returns an approximation for the trigonometric sine of the given numeric angle (expressed in radians) argument. See [Can OpenCOBOL be used for plotting?](#) for a sample graph using gnuplot.

## 28.2.57 4.2.57 FUNCTION SQRT

Returns an approximation of the square root of the given numeric argument.

```
DISPLAY FUNCTION SQRT(-1) END-DISPLAY
CALL "perror" USING NULL END-CALL
DISPLAY FUNCTION SQRT(2) END-DISPLAY
```

Outputs:

```
0.00000000000000000000
Numerical argument out of domain
1.414213562373095145
```

Note: CALL “perror” reveals a bug in OpenCOBOL versions packaged before June 2009 where the stack will eventually underflow due to improper handling of the **void** return specification. Versions supporting RETURNING NULL fix this problem. An actual application that needed to verify the results of square roots or other numerical function would be better off placing a small C wrapper to set and get the global `errno`.

## 28.2.58 4.2.58 FUNCTION STANDARD-DEVIATION

Returns an approximation of the standard deviation from the given list of numeric arguments.

```
DISPLAY
  FUNCTION STANDARD-DEVIATION(1 2 3 4 5 6 7 8 9 10) SPACE
  FUNCTION STANDARD-DEVIATION(1 2 3 4 5 6 7 8 9 100)
END-DISPLAY
```

```
2.872281323269014308 28.605069480775604518
```

## 28.2.59 4.2.59 FUNCTION STORED-CHAR-LENGTH

Returns the numeric value of the internal storage length of the given argument in bytes, not counting spaces.

## 28.2.60 4.2.60 FUNCTION SUBSTITUTE

FUNCTION SUBSTITUTE is an OpenCOBOL extension to the suite of intrinsic functions.

```
DISPLAY
  FUNCTION SUBSTITUTE("this is a test",
    "this", "that",
    "is a", "was",
    "test", "very cool!")
END-DISPLAY
```

Will display:

```
that was very cool!
```

having changed *this* for *that*, *is a* for *was* and *test* with **very cool!**

The new intrinsic accepts:

```
SUBSTITUTE(subject, lit-pat-1, repl-1 [, lit1-pat-2, repl-2, ...])
```

where *lit-pat* just means the scan is for literals, not that you have to use literal constants. WORKING-STORAGE identifiers are fine for any of the subject, the search patterns or the replacements.

As with all intrinsics, you receive a new field and the subject is untouched.

---

**Note:** The resulting field can be shorter, the same length or longer than the subject string.

---

This is literal character **global** find and replace, and there are no wildcards or other pattern expressions. Unlike INSPECT, this function **does not require same length** patterns and replacements. Each pattern replacement pair uses the original subject, not any intermediate in progress result.

As this is an alphanumeric operation, a reference modification is also allowed

```
MOVE FUNCTION SUBSTITUTE(subject, pat, repl) (2:4) TO xvar4
```

to result in 4 characters starting at the second position after the substitution.

### 28.2.61 4.2.61 FUNCTION SUBSTITUTE-CASE

Similar to SUBSTITUTE, but ignores upper and lower case of subject when matching patterns.

### 28.2.62 4.2.62 FUNCTION SUM

Returns the numeric value that is the sum of the given list of numeric arguments.

### 28.2.63 4.2.63 FUNCTION TAN

Returns an approximation for the trigonometric tangent of the given numeric angle (expressed in radians) argument. Returns ZERO if the argument would cause an infinity or other size error.

### 28.2.64 4.2.64 FUNCTION TEST-DATE-YYYYMMDD

Test for valid date in numeric yyyyymmdd form.

### 28.2.65 4.2.65 FUNCTION TEST-DAY-YYYYDDD

Test for valid date in numeric yyyyddd form.

### 28.2.66 4.2.66 FUNCTION TRIM

Returns a character string that is the argument trimmed of spaces. Defaults to trimming both ends, but can be passed LEADING or TRAILING qualifier arguments.

```
DISPLAY ' "' FUNCTION TRIM(" abc ") ' "' END-DISPLAY
DISPLAY ' "' FUNCTION TRIM(" abc " LEADING) ' "' END-DISPLAY
DISPLAY ' "' FUNCTION TRIM(" abc " TRAILING) ' "' END-DISPLAY
```

Outputs:

```
"abc"  
"abc  "  
"  abc"
```

## 28.2.67 4.2.67 FUNCTION UPPER-CASE

Returns a copy of the alphanumeric argument with any lower case letters replaced by upper case letters.

```
DISPLAY FUNCTION UPPER-CASE("# 123 abc DEF #") END-DISPLAY
```

Outputs:

```
# 123 ABC DEF #
```

## 28.2.68 4.2.68 FUNCTION VARIANCE

Returns the variance of a series of numbers. The variance is defined as the square of the FUNCTION STANDARD-DEVIATION

```
DISPLAY FUNCTION VARIANCE(1 2 3 4 5 6 7 8 9 100) END-DISPLAY.
```

```
+818.2500000000000000
```

## 28.2.69 4.2.69 FUNCTION WHEN-COMPILED

Returns a 21 character alphanumeric field of the form YYYYMMDDhhmmsscc±zzzz e.g. 2008070505152000-0400 representing when a module or executable is compiled. The WHEN-COMPILED special register reflects when an object module was compiled

```
program-id. whenpart1. procedure division.  
display "First part :" FUNCTION WHEN-COMPILED end-display.
```

```
program-id. whenpart2. procedure division.  
display "Second part:" FUNCTION WHEN-COMPILED end-display.
```

```
program-id. whenshow. procedure division.  
call "whenpart1" end-call.  
call "whenpart2" end-call.  
display "Main part  :" FUNCTION WHEN-COMPILED end-display.
```

For a test

```
$ cobc -c whenpart1.cob && sleep 15 && cobc -c whenpart2.cob &&  
> sleep 15 && cobc -x whenshow.cob whenpart1.o whenpart2.o  
$ ./whenshow
```

gives:

```
First part :2008082721391500-0400  
Second part:2008082721393000-0400  
Main part  :2008082721394500-0400
```

## 28.2.70 4.2.70 FUNCTION YEAR-TO-YYYY

Converts a two digit year to a sliding window four digit year. The optional second argument (default 50) is added to the date at execution time to determine the ending year of a 100 year interval.

## 28.3 4.3 Can you clarify the use of FUNCTION in OpenCOBOL?

Yes. This information is from [Roger], posted to the `opencobol` forums.

Just to clarify the use of FUNCTION.

(Applies to 0.33)

FUNCTION (generally speaking, there are exceptions) can be used anywhere where a source item is valid.

It always results in a new temporary field.

This will have the desired characteristics dependant on the parameters.

eg. FUNCTION MIN (x, y, z)

with x PIC 99

    y PIC 9(8) COMP

    z PIC 9(6)V99

will result in returning a field that has at least 8 positions before the (implied) decimal point and 2 after.

It does NOT ever change the contents of parameters to the function.

FUNCTION's are nestable.

eg.

```
DISPLAY FUNCTION REVERSE (FUNCTION UPPER-CASE (myfield)).
```

One clarification to the above quote was pointed out by Roger. The line:

be used anywhere where a source item is valid.

should be:

be used anywhere where a sending field is valid.

## 28.4 4.4 What is the difference between the LENGTH verb and FUNCTION LENGTH?

From [Roger]:

The standard only defines FUNCTION LENGTH.

The LENGTH OF phrase is an extension (from MF)

## 28.5 4.5 What STOCK CALL LIBRARY does OpenCOBOL offer?

OpenCOBOL 1.0 ships with quite a few callable features. See `CALL`. Looking through the source code, you'll find the current list of service calls in:

libcob/system.def

With the 1.1 pre-release of July 2008, that list included

```

/* COB_SYSTEM_GEN (external name, number of parameters, internal name) */

COB_SYSTEM_GEN ("SYSTEM", 1, SYSTEM)
COB_SYSTEM_GEN ("CBL_ERROR_PROC", 2, CBL_ERROR_PROC)
COB_SYSTEM_GEN ("CBL_EXIT_PROC", 2, CBL_EXIT_PROC)
COB_SYSTEM_GEN ("CBL_OPEN_FILE", 5, CBL_OPEN_FILE)
COB_SYSTEM_GEN ("CBL_CREATE_FILE", 5, CBL_CREATE_FILE)
COB_SYSTEM_GEN ("CBL_READ_FILE", 5, CBL_READ_FILE)
COB_SYSTEM_GEN ("CBL_WRITE_FILE", 5, CBL_WRITE_FILE)
COB_SYSTEM_GEN ("CBL_CLOSE_FILE", 1, CBL_CLOSE_FILE)
COB_SYSTEM_GEN ("CBL_FLUSH_FILE", 1, CBL_FLUSH_FILE)
COB_SYSTEM_GEN ("CBL_DELETE_FILE", 1, CBL_DELETE_FILE)
COB_SYSTEM_GEN ("CBL_COPY_FILE", 2, CBL_COPY_FILE)
COB_SYSTEM_GEN ("CBL_CHECK_FILE_EXIST", 2, CBL_CHECK_FILE_EXIST)
COB_SYSTEM_GEN ("CBL_RENAME_FILE", 2, CBL_RENAME_FILE)
COB_SYSTEM_GEN ("CBL_GET_CURRENT_DIR", 3, CBL_GET_CURRENT_DIR)
COB_SYSTEM_GEN ("CBL_CHANGE_DIR", 1, CBL_CHANGE_DIR)
COB_SYSTEM_GEN ("CBL_CREATE_DIR", 1, CBL_CREATE_DIR)
COB_SYSTEM_GEN ("CBL_DELETE_DIR", 1, CBL_DELETE_DIR)
COB_SYSTEM_GEN ("CBL_AND", 3, CBL_AND)
COB_SYSTEM_GEN ("CBL_OR", 3, CBL_OR)
COB_SYSTEM_GEN ("CBL_NOR", 3, CBL_NOR)
COB_SYSTEM_GEN ("CBL_XOR", 3, CBL_XOR)
COB_SYSTEM_GEN ("CBL_IMP", 3, CBL_IMP)
COB_SYSTEM_GEN ("CBL_NIMP", 3, CBL_NIMP)
COB_SYSTEM_GEN ("CBL_EQ", 3, CBL_EQ)
COB_SYSTEM_GEN ("CBL_NOT", 2, CBL_NOT)
COB_SYSTEM_GEN ("CBL_TOUPPER", 2, CBL_TOUPPER)
COB_SYSTEM_GEN ("CBL_TOLOWER", 2, CBL_TOLOWER)
COB_SYSTEM_GEN ("\364", 2, CBL_XF4)
COB_SYSTEM_GEN ("\365", 2, CBL_XF5)
COB_SYSTEM_GEN ("\221", 2, CBL_X91)
COB_SYSTEM_GEN ("C$NARG", 1, cob_return_args)
COB_SYSTEM_GEN ("C$PARAMSIZE", 1, cob_parameter_size)
COB_SYSTEM_GEN ("C$MAKEDIR", 1, cob_acuw_mkdir)
COB_SYSTEM_GEN ("C$CHDIR", 2, cob_acuw_chdir)
COB_SYSTEM_GEN ("C$SLEEP", 1, cob_acuw_sleep)
COB_SYSTEM_GEN ("C$COPY", 3, cob_acuw_copyfile)
COB_SYSTEM_GEN ("C$FILEINFO", 2, cob_acuw_file_info)
COB_SYSTEM_GEN ("C$DELETE", 2, cob_acuw_file_delete)
COB_SYSTEM_GEN ("C$TOUPPER", 2, CBL_TOUPPER)
COB_SYSTEM_GEN ("C$TOLOWER", 2, CBL_TOLOWER)
COB_SYSTEM_GEN ("C$JUSTIFY", 1, cob_acuw_justify)
COB_SYSTEM_GEN ("CBL_OC_NANOSLEEP", 1, cob_oc_nanosleep)

/**/

```

Note the “SYSTEM”. This CALL sends a command string to the shell. It acts as a wrapper to the standard C library “system” call. “SYSTEM” removes any trailing spaces from the argument and appends the null terminator required for the C library “system” call. While shell access opens yet another powerful door for the OpenCOBOL programmer, diligent developers will need to pay heed to cross platform issues when calling the operating system.

## 28.5.1 4.5.1 A CBL\_ERROR\_PROC example

```

OCOBOL >>SOURCE FORMAT IS FIXED
*****
* OpenCOBOL demonstration
* Author: Brian Tiffin
* Date: 26-Jun-2008
* History:
* 03-Jul-2008
* Updated to compile warning free according to standards
* Purpose:
* CBL_ERROR_PROC and CBL_EXIT_PROC call example
* CBL_ERROR_PROC installs or removes run-time error procedures
* CBL_EXIT_PROC installs or removes exit handlers
* Also demonstrates the difference between Run time errors
* and raised exceptions. Divide by zero is raises an
* exception, it does not cause a run time error.
* NB:
* Please be advised that this example uses the functional but
* now obsolete ENTRY verb. Compiling with -Wall will display
* a warning. No warning will occur using -std=MF
* Tectonics: ccbc -x errorproc.cob
identification division.
program-id. error_exit_proc.

data division.
working-storage section.
* entry point handlers are procedure addresses
01 install-address usage is procedure-pointer.
01 install-flag pic 9 comp-x value 0.
01 status-code pic s9(9) comp-5.

* exit handler address and priority (prio is IGNORED with OC1.1)
01 install-params.
02 exit-addr usage is procedure-pointer.
02 handler-prio pic 999 comp-x.

* indexing variable for back scanning error message strings
01 ind pic s9(9) comp-5.

* work variable to demonstrate raising exception, not RTE
01 val pic 9.

* mocked up error procedure reentrancy control, global level
01 once pic 9 value 0.
88 been-here value 1.

* mocked up non-reentrant value
01 global-value pic 99 value 99.

* LOCAL-STORAGE SECTION comes into play for ERROR_PROCS that
* may themselves cause run-time errors, handling reentry.
local-storage section.
01 reenter-value pic 99 value 11.

* Linkage section for the error message argument passed to proc
* By definition, error messages are 325 alphanumeric
linkage section.

```

```
01  err-msg          pic x(325).

* example of OpenCOBOL error and exit procedures
procedure division.

* Demonstrate problem installing procedure
* get address of WRONG handler. NOTE: Invalid address
set exit-addr to entry "nogo-proc".

* flag: 0 to install, 1 to remove
call "CBL_EXIT_PROC" using install-flag
                             install-params
                             returning status-code

end-call.
* status-code 0 on success, in this case expect error.
if status-code not = 0
    display
        "Intentional problem installing EXIT PROC"
        ", Status: " status-code
    end-display
end-if.

* Demonstrate install of an exit handler
* get address of exit handler
set exit-addr to entry "exit-proc".

* flag: 0 to install, 1 to remove
call "CBL_EXIT_PROC" using install-flag
                             install-params
                             returning status-code

end-call.
* status-code 0 on success.
if status-code not = 0
    display
        "Problem installing EXIT PROC"
        ", Status: " status-code
    end-display
    stop run
end-if.

* Demonstrate installation of an error procedure
* get the procedure entry address
set install-address to entry "err-proc".

* install error procedure. install-flag 0 installs, 1 removes
call "CBL_ERROR_PROC" using install-flag
                             install-address
                             returning status-code

end-call.
* status-code is 0 on success.
if status-code not = 0
    display "Error installing ERROR PROC" end-display
    stop run
end-if.

* example of error that raises exception, not a run-time error
divide 10 by 0 giving val end-divide.
* val will be a junk value, use at own risk
```



```

divide 10 by 0 giving val
  on size error display "DIVIDE BY ZERO Exception" end-display
end-divide.

* intentional run-time error
call "erroneous" end-call.          *> ** Intentional error **

* won't get here.  RTS error handler will stop run
display
  "procedure division, following run-time error"
end-display.
display
  "global-value: " global-value
  ", reenter-value: " reenter-value
end-display.

exit program.
*****

*****
* Programmer controlled Exit Procedure:
entry "exit-proc".

display
  "**Custom EXIT HANDLER (will pause 3 and 0.5 seconds)**"
end-display.

* sleep for 3 seconds
call "C$SLEEP" using "3" end-call.
* demonstrate nanosleep; argument in billionth's of seconds
* Note: also demonstrates OpenCOBOL's compile time
*       string catenation using ampersand;
*       500 million being one half second
call "CBL_OC_NANOSLEEP" using "500" & "000000" end-call.

exit program.

*****
* Programmer controlled Error Procedure:
entry "err-proc" using err-msg.

display "**ENTER error procedure**" end-display.

* These lines are to demonstrate local and working storage
display
  "global-value: " global-value
  ", reenter-value: " reenter-value
end-display.
* As reenter-value is local-storage
* the 77 will NOT display on reentry, while the global 66 will
move 66 to global-value.
move 77 to reenter-value.

* Process err-msg.
* Determine Length of error message, looking for null terminator
perform varying ind from 1 by 1
  until (err-msg(ind:1) = x"00") or (ind = length of err-msg)
  continue

```

```

end-perform.
display err-msg(1:ind) end-display.

* demonstrate trapping an error caused in error-proc
if not been-here then
    set been-here to true
    display "Cause error while inside error-proc" end-display
    call "very-erroneous" end-call      *> Intentional error
end-if.

* In OpenCOBOL 1.1, the return-code is local and does
* not influence further error handlers
*move 1 to return-code.
move 0 to return-code.

display "**error procedure EXIT**" end-display.

exit program.

```

with tectonics:

```

$ cobc -x errorproc.cob
$ ./errorproc
Intentional problem installing EXIT PROC, Status: -000000001
DIVIDE BY ZERO Exception
**ENTER error procedure**
global-value: 99, reenter-value: 11
Cannot find module 'erroneous'
Cause error while inside error-proc
**ENTER error procedure**
global-value: 66, reenter-value: 11
Cannot find module 'very-erroneous'
**error procedure EXIT**
libcob: Cannot find module 'very-erroneous'
**Custom EXIT HANDLER (will pause 3 and 0.5 seconds)**

```

## 28.5.2 4.5.2 Some stock library explanations

This small gem of a help file was written up by Vincent Coen, included here for our benefit.

---

**Note:** The code below is a work in progress. If you see this attention box; the file is not yet deemed complete.

---

System Calls v1.1.0 for OC v1.1 Author: Vincent B Coen dated 12/01/2009

```

COB_SYSTEM_GEN ("CBL_ERROR_PROC", 2, CBL_ERROR_PROC):      Register error proc in Linux??? needs ch
  call using      install-flag  pic x comp-x      Indicates operation to be performed
                                          (0 = install error procedure)
                                          (1 = un-install error procedure)
                                          install-addr Usage procedure pointer Create by 'set install-addr to entry entry
                                          (the address of error procedure to install or un-in

```

```

COB_SYSTEM_GEN ("CBL_EXIT_PROC", 2, CBL_EXIT_PROC)      Register closedown proc
  call using      install-flag  pic x comp-x      Indicate operation to be performed
                                          (0 = install closedown proc. with default priority
                                          (1 = un=install closedown proc.)
                                          (2 = query priority of installed proc.)

```

```

                                (3 = install shutdown proc. with given priority)
install-param  group item defined as:
install-addr  USAGE PROCEDURE POINTER (addr of shutdown proc to install, uninst
install-prty  pic x comp-x      (when install-flag = 3, priority of proc. being in
returning     status-code      (See section key).
on exit      install-prty      (when install-flag = 2, returns priority of select

COB_SYSTEM_GEN ("CBL_OPEN_FILE", 5, CBL_OPEN_FILE)      Open byte stream file
call using   file-name        pic x(n)      space or null terminated
            access-mode       pic x comp-5   (1 = read only, 2 = write only [deny must = 0]
            3 = read / write)
            deny-mode         pic x comp-5   (0 = deny both, 1 = deny write, 2 = deny read
            3 = deny neither read nor write)
            device            pic x comp-5   (must be zero)
            file-handle       pic x(4)      (Returns a file handle for a successful open)
returning    status-code      (See section key)

COB_SYSTEM_GEN ("CBL_CREATE_FILE", 5, CBL_CREATE_FILE)  Create byte stream file
call using   file-name        pic x(n)      (space or null terminated)
            access-mode       pic x comp-x   (1 = read only)
            2 = write only (deny must be 0)
            3 = read / write)
            deny-mode         pic x comp-x   (0 = deny both read & write exclusive)
            1 = deny write)
            2 = deny read)
            3 = deny neither read nor write)
            device            pic x comp-x   (must be zero) (reserved for future use)
            file-handle       pic x(4)      (Returns a file handle for a successful open)
returning    status-code      (See section key)

COB_SYSTEM_GEN ("CBL_READ_FILE", 5, CBL_READ_FILE)     Read byte stream file
call using   file-handle      pic x(4)      (File handle returned when file opened)
            file-offset       pic x(8) comp-x (offset in the file at which to read) (Max limit X"
            byte-count        pic x(4) comp-x (number of bytes to read. Poss limit x"00FFFF")
            flags              pic x comp-x   (0 = standard read, 128 = current file size returned
            file-offset field)
returning    status-code      (See section key)
on exit:    file-offset       pic x(n)      (Current file size on return if flags = 128 on entry)
            buffer            pic x(n)      (Buffer into which bytes are read. IT IS YOUR RESPONSIB
            TO ENSURE THAT THE BUFFER IS LARGE ENOUGH TO HOLD A
            READ)

Remarks:   See Introduction to Byte Stream Routines as well as example code taken
            from old version of CobXref

COB_SYSTEM_GEN ("CBL_WRITE_FILE", 5, CBL_WRITE_FILE)   Write byte stream file
call using   file-handle      pic x(4)      (File handle returned when file opened)
            file-offset       pic x(8) comp-x (offset in the file at which to write) (Max limit X"
            byte-count        pic x(4) comp-x (number of bytes to write. Poss limit x"00FFFF")
            Putting a value of zero here causes file to be truncated
            to the size specified in file-offset)
            flags              pic x comp-x   (0 = standard write)
            buffer            pic x(n)      (Buffer into which bytes are written from)
returning    status-code      (See section key)

Remarks:   See Introduction to Byte Stream Routines as well as example code taken
            from old version of CobXref

```

```

COB_SYSTEM_GEN ("CBL_CLOSE_FILE", 1, CBL_CLOSE_FILE)          Close byte stream file
  call using file-handle      pic x(4)      on entry the file handle returned when file opened
           returning status-code      (see section key)

COB_SYSTEM_GEN ("CBL_FLUSH_FILE", 1, CBL_FLUSH_FILE)          ??????????????????
  call using      ???????      pic ?????      No Idea

COB_SYSTEM_GEN ("CBL_DELETE_FILE", 1, CBL_DELETE_FILE)        Delete File
  call using file-name      pic x(n)      file to delete terminated by space can contain path.
           returning status-code

COB_SYSTEM_GEN ("CBL_COPY_FILE", 2, CBL_COPY_FILE)            Copy file
  call using file-name1      (pic x(n)      File to copy, can contain path terminated by space
           file-name2      (pic x(n)      File name of new file, can contain path terminated by
           For both, if no path current directory is assumed.
           returning status-code      (see section key)

COB_SYSTEM_GEN ("CBL_CHECK_FILE_EXIST", 2, CBL_CHECK_FILE_EXIST)  Check if file exists & return de
  Call using      file-name
           file-details
           returning status-code

file-name      pic x(n)
file-details      Group item defined as:
  file-size      pic x(8) comp-x
  file-date
  day      pic x comp-x
  month      pic x comp-x
  year      pic xx comp-x

file-time
  hours      pic x comp-x
  minutes      pic x comp-x
  seconds      pic x comp-x
  hundredths      pic x comp-x
status-code      see section key

On entry: file-name      The file to look for. name can cotain path and is terminated by a space
           If no path given current directory is assumed.
On Exit:  file-size      Size if file in bytes
           file-date      Date the file was created
           file-time      Time file created

COB_SYSTEM_GEN ("CBL_RENAME_FILE", 2, CBL_RENAME_FILE)        Rename file
  call using      old-file-name      pic x(n)      (file to rename can contain path terminated by spa
           new-file-name      pic x(n)      (new file name as above path must be same)
           returning status-code      (see section key)

COB_SYSTEM_GEN ("CBL_GET_CURRENT_DIR", 3, CBL_GET_CURRENT_DIR) Get details of current directory
  call using      ???      pic x(n)      ???
           ???      pic x(n)      ???
           returning status-code      (see section key)

COB_SYSTEM_GEN ("CBL_CHANGE_DIR", 1, CBL_CHANGE_DIR)          Change current directory
  Call using      path-name      pic x(n)      (relative or absolute terminated by x"00")
           returning status-code      (see section key)

COB_SYSTEM_GEN ("CBL_CREATE_DIR", 1, CBL_CREATE_DIR)          Create directory

```

Call using path-name pic x(n) (relative or absolute path-name terminate by x"00")  
 returning status-code (see section key)

COB\_SYSTEM\_GEN ("CBL\_DELETE\_DIR", 1, CBL\_DELETE\_DIR) Delete directory  
 Call using path-name pic x(n) (relative or absolute name terminated by space or null [x"00"  
 returning status-code (see section key)

COB\_SYSTEM\_GEN ("CBL\_AND", 3, CBL\_AND) logical AND  
 Call using source (Any data item)  
 target (Any data item)  
 by value length (numeric literal or pic x(4) comp-5  
 returning status-code (see section key)

COB\_SYSTEM\_GEN ("CBL\_OR", 3, CBL\_OR) logical OR  
 call using source (Any data item)  
 target (Any data item)  
 by value length (numeric literal or pic x(4) comp-5  
 returning status-code (see section key)

COB\_SYSTEM\_GEN ("CBL\_NOR", 3, CBL\_NOR) Logical Not OR ?  
 Call using source (Any data item)  
 target (Any data item)  
 by value length (numeric literal or pic x(4) comp-5  
 returning status-code (see section key)

COB\_SYSTEM\_GEN ("CBL\_XOR", 3, CBL\_XOR) logical eXclusive OR  
 Call using source (Any data item)  
 target (Any data item)  
 by value length (numeric literal or pic x(4) comp-5  
 returning status-code (see section key)

COB\_SYSTEM\_GEN ("CBL\_IMP", 3, CBL\_IMP) Logical IMPLies  
 call using source Any data item  
 target Any data Item  
 by value length Nuneric literal or pic x(4) comp-5  
 returning status-code (see section key)

COB\_SYSTEM\_GEN ("CBL\_NIMP", 3, CBL\_NIMP) Logical Not IMPLies  
 call using source Any data item  
 target Any data Item  
 by value length Nuneric literal or pic x(4) comp-5  
 returning status-code (see section key)

COB\_SYSTEM\_GEN ("CBL\_EQ", 3, CBL\_EQ) Logical EQUIVALENCE between bits of bot  
 Call using source (Any data item)  
 target (Any data item)  
 by value length (numeric literal or pic x(4) comp-5  
 returning status-code (see section key)

COB\_SYSTEM\_GEN ("CBL\_NOT", 2, CBL\_NOT) Logical NOT  
 Call using target Any data item  
 by value length numeric lit or pic x(4) comp-5

COB\_SYSTEM\_GEN ("CBL\_TOUPPER", 2, CBL\_TOUPPER) Convert a string to Upper case  
 Call using string pic x(n) (The string to convert)  
 by value length pic x(4) comp-5 (Number of bytes to change)  
 returning status-code (see section key)



```

104700      05 Fht-Byte-Count      pic x(4)          comp-x value 4096.
104800      05 Fht-CopyRefNo2     pic 9(6)          value zero.
104900      05 Fht-Pointer        pic s9(5)         comp value zero.
105000      05 Fht-Copy-Line-End  pic s9(5)         comp value zero.
105100      05 Fht-Copy-Words     pic s9(5)         comp value zero.
105200      05 Fht-sw-Eof        pic 9              value zero.
105300      88 Fht-Eof            value 1.
105400      05 Fht-Current-Rec    pic x(160)        value spaces.
105500      05 Fht-File-Name     pic x(256) .
105600      05 Fht-Buffer        pic x(4097) .
105700      05 filler            pic x              value x"FF".
105800 01  Fht-Table-Size       pic s9(5) comp value zero.
105900*
106000 01  Cbl-File-Fields.
106100      03 Cbl-File-name     pic x(256) .
106200      03 Cbl-Access-Mode   pic x            comp-x value 1.
106300      03 Cbl-Deny-Mode     pic x            comp-x value 3.
106400      03 Cbl-Device        pic x            comp-x value zero.
106500      03 Cbl-Flags         pic x            comp-x value zero.
106600      03 Cbl-File-Handle   pic x(4)          value zero.
106700      03 Cbl-File-OffSet   pic x(8)          comp-x value zero.
106800*
106900 01  Cbl-File-Details.
107000      03 Cbl-File-Size     pic x(8)          comp-x value zero.
107100      03 Cbl-File-Date.
107200          05 Cbl-File-Day   pic x            comp-x value zero.
107300          05 Cbl-File-Mth  pic x            comp-x value zero.
107400          05 Cbl-File-Year  pic x            comp-x value zero.
107500      03 Cbl-File-time.
107600          05 Cbl-File-Hour  pic x            comp-x value zero.
107700          05 Cbl-File-Min  pic x            comp-x value zero.
107800          05 Cbl-File-Sec  pic x            comp-x value zero.
107900          05 Cbl-File-Hund  pic x            comp-x value zero.
...
...
*****
*
* zz300, zz400, zz500 & zz600 all relate to copy files/libraries
* via the COPY verb
* As it is hoped to only use the filename.i via Open-Cobol
* then this lot can be killed off as well as all the other related
* code.
* NOTE that the COPY verb is implemented in a very basic way despite
* the fact that this code allows for 99 levels of COPY, eg, there is
* NO replacing so hopefully I can remove it all after primary testing
* When it is built into cobc
*
356400 zz300-Open-File.
356500*****
356600* Open a Copy file using CBL-OPEN-File
356700* filename is using Cbl-File-name
356800*
356900      move zero to Return-Code.
357000      if Fht-Table-Size > 99
357100          move 24 to Return-Code
357200          display Msg11
357300          go to zz300-Exit.
357400*

```

```

357500* set up New entry in File Table
357600*
357700    add      1 to Fht-Table-Size.
357800    move     Fht-Table-Size to e.
357900    move     zeroes to Fht-File-OffSet (e) Fht-File-Size (e)
358000                Fht-File-Handle (e) Fht-Block-OffSet (e)
358100                Fht-CopyRefNo2 (e) Fht-sw-Eof (e)
358200                Fht-Copy-Line-End (e) Fht-Copy-Words (e).
358300    move     4096 to Fht-Byte-Count (e).
358400    move     spaces to Fht-Current-Rec (e).
358500    move     1      to Fht-pointer (e).
358600*
358700    perform  zz400-Check-File-Exists thru zz400-Exit.
358800    if      Return-Code not = zero
358900        subtract 1 from Fht-Table-Size
359000        go to  zz300-Exit.
359100*
359200    move     Fht-Table-Size to e.
359300    move     Cbl-File-Size to Fht-File-Size (e).
359400    move     Cbl-File-name to Fht-File-Name (e).
359500    move     1      to Cbl-Access-Mode
359600                Cbl-Deny-Mode.
359700    move     zero to Cbl-Device
359800                Cbl-File-Handle.
359900    move     zero to Return-Code.
360000    call    "CBL_OPEN_FILE" using
360100        Cbl-File-name
360200        Cbl-Access-Mode
360300        Cbl-Deny-Mode
360400        Cbl-Device
360500        Cbl-File-Handle.
360600    if      Return-Code not = zero
360700        display Msg12 cbl-File-name
360800        display "      This should not happen here"
360900        subtract 1 from Fht-Table-Size
361000        go to  zz300-exit.
361100*
361200    move     Cbl-File-Handle to Fht-File-Handle (e).
361300    add      1 to Copy-Depth.
361400    move     1 to sw-Copy.
361500    move     zero to Fht-CopyRefNo2 (e)
361600                Return-Code.
362000    zz300-Exit.
362100    exit.
362200/
362300    zz400-Check-File-Exists.
362400*
362500* check for correct filename and extention taken from COPY verb
362600*
362700* input : wsFoundNewWord2
362800* Output : Return-Code = 0 : Cbl-File-Details & Cbl-File-name
362900*         Return-Code = 25 : failed fn in wsFoundNewWord2
363000*
363100    move     zero to e.
363200    inspect  wsFoundNewWord2 tallying e for all ".".
363300    if      e not zero
363400        go to  zz400-Try1.
363500    perform  varying a from 1 by 1 until Return-Code = zero

```



```

363600         move 1 to e
363700         move spaces to Cbl-File-name
363800         string wsFoundNewWord2 delimited by space
363900                                 into Cbl-File-name pointer e
364000         string File-Ext (a) delimited by size
364100                                 into Cbl-File-name pointer e
364200         move zero to Return-Code
364300         call "CBL_CHECK_FILE_EXIST" using
364400             Cbl-File-name
364500             Cbl-File-Details
364600         end-call
364700         if Return-Code not = zero
364800             and a = 7
364900             exit perform
365000         end-if
365100     end-perform
365200     if Return-Code not = zero
365300         display "zz400A Check File exist err=" Return-Code
365400         display Msg13 wsFoundNewWord2
365500         move 25 to Return-Code
365600         go to zz400-Exit.
365700* ok file now found
365900     go to zz400-Exit.
366000*
366100 zz400-Try1.
366200     move wsFoundNewWord2 to Cbl-File-name.
366300     move zero to Return-Code.
366400     call "CBL_CHECK_FILE_EXIST" using
366500         Cbl-File-name
366600         Cbl-File-Details.
366700     if Return-Code not = zero
366800         move function lower-case (wsFoundNewWord2) to
366900             Cbl-File-name
367000         go to zz400-Try2.
367100* ok file now found
367200     go to zz400-exit.
367300*
367400 zz400-Try2.
367500     move zero to Return-Code.
367600     call "CBL_CHECK_FILE_EXIST" using
367700         Cbl-File-name
367800         Cbl-File-Details.
367900     if Return-Code not = zero
368000         display "zz400C Check File exist err=" Return-Code
368100         display Msg13 wsFoundNewWord2 " or " Cbl-File-name
368200         move 25 to Return-Code
368300         go to zz400-Exit.
368400*
368500* ok file now found
368600*
368700 zz400-Exit.
368800     exit.
368900/
369000 zz500-Close-File.
369100     call "CBL_CLOSE_FILE" using
369200         Fht-File-Handle (Fht-Table-Size).
369300     if Return-Code not = zero
369400         display Msg14

```

```
369500          Cbl-File-name.
369800  subtract 1 from Fht-Table-Size.
369900*
370000  if      Fht-Table-Size = zero
370100      move zero to sw-Copy.
370200  subtract 1 from Copy-Depth.
370300  move    zero to Return-Code.
370400  go      to zz500-Exit.
370500*
370600  zz500-Exit.
370700  exit.
370800/
370900  zz600-Read-File.
371000*****
371100* called using file-handle
371200* returning CopySourceRecin1 size 160 chars
371300* If buffer empty read a block
371400* and regardless, move record terminated by x"0a"
371500* to Fht-Current-Rec (Fht-Table-Size)
371600*
371700  if      Fht-Eof (Fht-Table-Size)
371800      perform zz500-Close-File
371900      go to zz600-Exit.
372000*
372100  if      Fht-File-OffSet (Fht-Table-Size) = zero
372200      and Fht-Block-OffSet (Fht-Table-Size) = zero
372300      perform zz600-Read-A-Block
372400      go to zz600-Get-A-Record.
372500*
372600  zz600-Get-A-Record.
372700*****
372800* Now to extract a record from buffer and if needed read a block
372900* then extract
373000*
373100  move    spaces to Fht-Current-Rec (Fht-Table-Size).
373200  add    1 to Fht-Block-OffSet (Fht-Table-Size) giving g.
373300*
373400* note size is buffer size + 2
373500*
373600  unstring Fht-Buffer (Fht-Table-Size) (1:4097)
373700      delimited by x"0A" or x"FF"
373800      into      Fht-Current-Rec (Fht-Table-Size)
373900      delimiter  Word-Delimit3
374000      pointer   g.
374100*
374200* Get next Block of data ?
374300*
374400  if      Word-Delimit3 = x"FF"
374500      and g not < 4097
374600      add Fht-Block-OffSet (Fht-Table-Size)
374700          to Fht-File-OffSet (Fht-Table-Size)
374800      perform zz600-Read-A-Block
374900      go to zz600-Get-A-Record.
375000* EOF?
375100  move    1 to Fht-Pointer (Fht-Table-Size).
375200  if      Word-Delimit3 = x"FF"
375300      move 1 to Fht-sw-Eof (Fht-Table-Size)
375400      go to zz600-Exit.
```

```

375500* Now so tidy up
375600     subtract 1 from g giving Fht-Block-OffSet (Fht-Table-Size).
375700     go      to zz600-exit.
375800*
375900 zz600-Read-A-Block.
      *****
376000     move    all x"FF" to Fht-Buffer (Fht-Table-Size).
376100*     if      Fht-File-Size (Fht-Table-Size) < 4096 and not = zero
376200*         move Fht-File-Size (Fht-Table-Size)
376300*             to Fht-Byte-Count (Fht-Table-Size).
376400     call    "CBL_READ_FILE" using
376500         Fht-File-Handle (Fht-Table-Size)
376600         Fht-File-OffSet (Fht-Table-Size)
376700         Fht-Byte-Count (Fht-Table-Size)
376800         Cbl-Flags
376900         Fht-Buffer (Fht-Table-Size).
377000     if      Return-Code not = zero
377100         display Msg15 Return-Code
377200         go to zz600-Exit.
377300* just in case all ff does not work
377400     move    x"FF" to Fht-Buffer (Fht-Table-Size) (4097:1).
377500     move    zero to Fht-Block-OffSet (Fht-Table-Size).
377600     subtract Fht-Byte-Count (Fht-Table-Size)
377700             from Fht-File-Size (Fht-Table-Size).
377800 zz600-Exit.
377900     exit.

```

## 28.6 4.6 What are the XF4, XF5, and X91 routines?

From [opencobol.org](http://opencobol.org)

The CALL's X"F4", X"F5", X"91" are from MF.  
You can find them in the online MF doc under  
Library Routines.

F4/F5 are for packing/unpacking bits from/to bytes.  
91 is a multi-use call. Implemented are the subfunctions  
get/set cobol switches (11, 12) and get number of call params (16).

Roger

Use

```

CALL X"F4" USING
        BYTE-VAR
        ARRAY-VAR
        RETURNING STATUS-VAR

```

to pack the last bit of each byte in the 8 byte ARRAY-VAR into corresponding bits of the 1 byte BYTE-VAR.  
The X"F5" routine takes the eight bits of byte and moves them to the corresponding occurrence within array.  
X"91" is a multi-function routine.

```

CALL X"91" USING
        RESULT-VAR
        FUNCTION-NUM

```

PARAMETER-VAR  
RETURNING STATUS-VAR

As mentioned by Roger, OpenCOBOL supports FUNCTION-NUM of 11, 12 and 16.

11 and 12 get and set the on off status of the 8 (eight) run-time OpenCOBOL switches definable in the SPECIAL-NAMES paragraph. 16 returns the number of call parameters given to the current module.

## 28.7 4.7 What is CBL\_OC\_NANOSLEEP OpenCOBOL library routine?

CBL\_OC\_NANOSLEEP allows (upto) nanosecond sleep timing. It accepts a 64 bit integer value which may be in character or numeric data forms.

```
CALL "CBL_OC_NANOSLEEP" USING 500000000  
RETURNING STATUS  
END-CALL
```

Would wait one-half second. *It may be easier to grok if the source code uses string catenation; "500" & "000000" for example.*

## 28.8 4.8 How do you use C\$JUSTIFY?

The C\$JUSTIFY sub program can centre, or justify strings left or right.

```
OCOBOL >>SOURCE FORMAT IS FIXED  
*> *****  
*> Author: Brian Tiffin  
*> Date: 01-Jul-2008  
*> Purpose: Demonstrate the usage of OpenCOBOL call library  
*> C$JUSTIFY, C$TOUPPER, C$TOLOWER  
*> Tectonics: Using OC1.1 post 02-Jul-2008, cobc -x -Wall  
*> History: 02-Jul-2008, updated to remove warnings  
*> *****  
identification division.  
program-id. justify.  
  
environment division.  
configuration section.  
source-computer. IBMPC.  
object-computer. IBMPC.  
  
data division.  
WORKING-STORAGE section.  
01 source-str pic x(80)  
value " this is a test of the internal voice communication  
- " system".  
01 just-str pic x(80).  
01 justification pic x.  
01 result pic s9(9) comp-5.  
  
procedure division.  
move source-str to just-str.
```

```

*> Left justification
move "L" to justification.
perform demonstrate-justification.

*> case change to upper, demonstrate LENGTH verb
call "C$TOUPPER" using just-str
                  by value function length( just-str )
                  returning result

end-call.

*> Centre
move "C" to justification.
perform demonstrate-justification.

*> case change to lower
call "C$TOLOWER" using just-str
                  by value 80
                  returning result

end-call.

*> Right, default if no second argument
call "C$JUSTIFY" using just-str
                  returning result

end-call.
move "R" to justification.
perform show-justification.

exit program.
stop run.

*> *****
demonstrate-justification.
call "C$JUSTIFY" using just-str
                  justification
                  returning result

end-call
if result not equal 0 then
    display "Problem: " result end-display
    stop run
end-if
perform show-justification
.

*> *****
show-justification.
evaluate justification
    when "L"    display "Left justify" end-display
    when "C"    display "Centred (in UPPERCASE)" end-display
    when other display "Right justify" end-display
end-evaluate
display "Source:  |" source-str "|" end-display
display "Justified:|" just-str "|" end-display
display space end-display
.

```

Producing:

```
$ ./justify
Left justify
Source:   |   this is a test of the internal voice communication system   |
Justified:|this is a test of the internal voice communication system   |

Centred (in UPPERCASE)
Source:   |   this is a test of the internal voice communication system   |
Justified:|           THIS IS A TEST OF THE INTERNAL VOICE COMMUNICATION SYSTEM   |

Right justify
Source:   |   this is a test of the internal voice communication system   |
Justified:|           this is a test of the internal voice communication system|
```

## 28.9 4.9 What compiler directives are supported by OpenCOBOL?

OpenCOBOL 1.1pre-rel supports a limited number of **directives**.

- >>D
- >>SOURCE

OpenCOBOL 2.0 supports a much wider subset of standard directives and existent extensions. Some are only recognized and will be ignored with a warning until implemented.

- >>D
- >>SOURCE
- >>DEFINE
- >>DISPLAY
- >>IF
- >>ELSE
- >>ELIF
- >>ELSE-IF
- >>END-IF
- >>SET
- >>LEAP-SECOND
- >>TURN

### 28.9.1 4.9.1 >>D

Debug line control. OpenCOBOL only compiles these lines if the **-fdebugging-line** command line is used.

### 28.9.2 4.9.2 >>SOURCE

OpenCOBOL fully supports FREE and FIXED format source. The compiler defaults FIXED form sources, so this directive is usually placed at column 8 or beyond. The command line arguments **-free** and **-fixed** controls the default for the *first line* of source.

See [What source formats are accepted by OpenCOBOL?](#) for more details.

**28.9.3 4.9.3 >>DEFINE**

Define a compile time symbol.

- >>DEFINE identifier AS literal
- >>DEFINE identifier AS literal OVERRIDE
- >>DEFINE identifier OFF
- >>DEFINE identifier PARAMETER
- >>DEFINE identifier CONSTANT
- >>DEFINE identifier working-variable

The -D command line option can be used to define symbols.

**28.9.4 4.9.4 >>IF**

Conditional compile directive. Will include source lines upto >>END-IF, an >>ELSE-IF or >>ELSE clause if condition is true.

- >>IF identifier DEFINED
- >>IF conditional-expression

**28.9.5 4.9.5 >>ELSE-IF**

Allows for multiple conditions in a conditional compile sequence.

**28.9.6 4.9.6 >>ELIF**

Alias for >>ELSE-IF.

**28.9.7 4.9.7 >>ELSE**

Compiles in source lines upto an >>END-IF if the previous >>IF or >>ELSE-IF conditions test false.

**28.9.8 4.9.8 >>END-IF**

Terminates a conditional compile block.

**28.9.9 4.9.9 >>SET**

Allows modification of compiler source text handling behaviour.

- >>SET CONSTANT
- >>SET SOURCEFORMAT
- >>SET FOLDCOPYNAME | FOLD-COPY-NAME
- >>SET NOFOLDCOPYNAME | NOFOLD-COPY-NAME
- >>SET AS

- >>SET literal
- >>SET {SET\_PAREN\_LIT}
- >>SET working-store-var

#### **28.9.10 4.9.10 >>LEAP-SECOND**

Ignored.

#### **28.9.11 4.9.11 >>TURN**

Will allow modification of exception code handling, when implemented.



## **5 FEATURES AND EXTENSIONS**

## OpenCOBOL Features

- 5.1 How do I use OpenCOBOL for CGI?
  - 5.1.1 AJAX
- 5.2 What is ocdoc?
  - 5.2.1 ocdoc generated ocdoc documentation
- 5.3 What is CBL\_OC\_DUMP?
  - 5.3.1 Update to OC\_CBL\_DUMP
- 5.4 Does OpenCOBOL support any SQL databases?
  - 5.4.1 Oracle procob and binary data sizes
  - 5.4.2 PostgreSQL Sample
- 5.5 Does OpenCOBOL support ISAM?
  - 5.5.1 FILE STATUS
- 5.6 Does OpenCOBOL support modules?
- 5.7 What is COB\_PRE\_LOAD?
- 5.8 What is the OpenCOBOL LINKAGE SECTION for?
- 5.9 What does the -fstatic-linkage OpenCOBOL compiler option do?
- 5.10 Does OpenCOBOL support Message Queues?
- 5.11 Can OpenCOBOL interface with Lua?
- 5.12 Can OpenCOBOL use ECMAScript?
- 5.13 Can OpenCOBOL use JavaScript?
- 5.14 Can OpenCOBOL interface with Scheme?
- 5.15 Can OpenCOBOL interface with Tcl/Tk?
- 5.16 Can OpenCOBOL interface with Falcon PL?
- 5.17 Can OpenCOBOL interface with Ada?
- 5.18 Can OpenCOBOL interface with Vala?
  - 5.18.1 Call OpenCOBOL programs from Vala
  - 5.18.2 Call OpenCOBOL from a Vala GTK gui application
  - 5.18.3 Call Genie program from OpenCOBOL
  - 5.18.4 Pass data to and from Genie
- 5.19 Can OpenCOBOL interface with S-Lang?
  - 5.19.1 Setup
  - 5.19.2 Keyboard control
  - 5.19.3 Scripting
- 5.20 Can the GNAT Programming Studio be used with OpenCOBOL?
- 5.21 Does OpenCOBOL support SCREEN SECTION?
  - 5.21.1 Environment variables in source code
- 5.22 What are the OpenCOBOL SCREEN SECTION colour values?
- 5.23 Does OpenCOBOL support CRT STATUS?
- 5.24 What is CobCurses?
- 5.25 What is CobXRef?
- 5.26 Does OpenCOBOL implement Report Writer?
- 5.27 Does OpenCOBOL implement LINAGE?
- 5.28 Can I use ctags with OpenCOBOL?
- 5.29 What about debugging OpenCOBOL programs?
  - 5.29.1 Some debugging tricks
  - 5.29.2 Animator
  - 5.29.3 Unit testing
- 5.30 Is there a C interface to OpenCOBOL?
- 5.31 What are some idioms for dealing with C char \* data from OpenCOBOL?
- 5.32 Does OpenCOBOL support COPY includes?
- 5.33 Does OpenCOBOL support WHEN-COMPILED?
- 5.34 What is PI in OpenCOBOL?
- 5.35 Does OpenCOBOL support the Object features of the 2002 standard?
- 5.36 Does OpenCOBOL implement PICTURE 78?

## OpenCOBOL Features

## 29.1 5.1 How do I use OpenCOBOL for CGI?

OpenCOBOL is more than capable of being a web server backend tool.

- One of the tricks is assigning an input stream to KEYBOARD when you need to get at POST data.
- Another is using the ACCEPT var FROM ENVIRONMENT feature.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*****
* Author:      Brian Tiffin, Francois Hiniger
* Date:        30-Aug-2008
* Purpose:     Display the CGI environment space
* Tectonics:   cobc -x cgienv.cob
* Move cgienv to the cgi-bin directory as cgienv.cgi
* browse http://localhost/cgi-bin/cgienv.cgi or cgienvform.html
*****
identification division.
program-id.  cgienv.

environment division.
input-output section.
file-control.
    select webinput assign to KEYBOARD.

data division.
file section.
fd webinput.
    01 postchunk          pic x(1024).

working-storage section.
78 name-count            value 34.
01 newline               pic x   value x'0a'.
01 name-index            pic 99 usage comp-5.
01 value-string          pic x(256).
01 environment-names.
    02 name-strings.
        03 filler        pic x(20) value 'AUTH_TYPE' .
        03 filler        pic x(20) value 'CONTENT_LENGTH' .
        03 filler        pic x(20) value 'CONTENT_TYPE' .
        03 filler        pic x(20) value 'DOCUMENT_ROOT' .
        03 filler        pic x(20) value 'GATEWAY_INTERFACE' .
        03 filler        pic x(20) value 'HTTP_ACCEPT' .
        03 filler        pic x(20) value 'HTTP_ACCEPT_CHARSET' .
        03 filler        pic x(20) value 'HTTP_ACCEPT_ENCODING' .
        03 filler        pic x(20) value 'HTTP_ACCEPT_LANGUAGE' .
        03 filler        pic x(20) value 'HTTP_COOKIE' .
        03 filler        pic x(20) value 'HTTP_CONNECTION' .
        03 filler        pic x(20) value 'HTTP_HOST' .
        03 filler        pic x(20) value 'HTTP_REFERER' .
        03 filler        pic x(20) value 'HTTP_USER_AGENT' .
        03 filler        pic x(20) value 'LIB_PATH' .
        03 filler        pic x(20) value 'PATH' .
        03 filler        pic x(20) value 'PATH_INFO' .
        03 filler        pic x(20) value 'PATH_TRANSLATED' .
```

```
03 filler      pic x(20) value 'QUERY_STRING'.
03 filler      pic x(20) value 'REMOTE_ADDR'.
03 filler      pic x(20) value 'REMOTE_HOST'.
03 filler      pic x(20) value 'REMOTE_IDENT'.
03 filler      pic x(20) value 'REMOTE_PORT'.
03 filler      pic x(20) value 'REQUEST_METHOD'.
03 filler      pic x(20) value 'REQUEST_URI'.
03 filler      pic x(20) value 'SCRIPT_FILENAME'.
03 filler      pic x(20) value 'SCRIPT_NAME'.
03 filler      pic x(20) value 'SERVER_ADDR'.
03 filler      pic x(20) value 'SERVER_ADMIN'.
03 filler      pic x(20) value 'SERVER_NAME'.
03 filler      pic x(20) value 'SERVER_PORT'.
03 filler      pic x(20) value 'SERVER_PROTOCOL'.
03 filler      pic x(20) value 'SERVER_SIGNATURE'.
03 filler      pic x(20) value 'SERVER_SOFTWARE'.
02 filler redefines name-strings.
03 name-string pic x(20) occurs name-count times.
```

**procedure division.**

*\* Always send out the Content-type before any other IO*

```
display
  "Content-type: text/html"
  newline
end-display.
display
  "<html><body>"
end-display.
display
  "<h3>CGI environment with OpenCOBOL</h3>"
end-display.
display
  '<a href="/cgienvform.html">To cgienvform.html</a>'
  "<p><table>"
end-display.
* Accept and display some of the known CGI environment values
perform varying name-index from 1 by 1
  until name-index > name-count
    accept value-string from environment
      name-string(name-index)
    end-accept
    display
      "<tr><td>"
      name-string(name-index)
      ": </td><td>"
      function trim (value-string trailing)
      "</td></tr>"
    end-display
    if (name-string(name-index) = "REQUEST_METHOD")
      and (value-string = "POST")
        open input webininput
        read webininput
          at end move spaces to postchunk
        end-read
        close webininput
        display
```

```

        ' <tr><td align="right">'
        "First chunk of POST:</td><td>"
        postchunk(1:72)
        "</td></tr>"
    end-display
end-if
end-perform.
display "</table></p></body></html>" end-display.
COOL goback.

```

Once compiled and placed in an appropriate cgi-bin directory of your web server, a simple form can be used to try the example.

#### cgienvform.html

```

<html><head><title>OpenCOBOL sample CGI form</title></head>
<body>
<h3>OpenCOBOL sample CGI form</h3>
<form action="http://localhost/cgi-bin/cgienv.cgi" method="post">
  <p>
    Text: <input type="text" name="text"><br>
    Password: <input type="password" name="password"><br>
    Checkbox: <input type="checkbox" name="checkbox"><br>
    <input type="radio" name="radio" value="ONE"> One<br>
    <input type="radio" name="radio" value="TWO"> Two<br>
    <input type="submit" value="Send"> <input type="reset">
  </p>
</form>
</body>
</html>

```

## 29.1.1 5.1.1 AJAX

From a post on [opencobol.org](http://opencobol.org) by DamonH:

As promised, here is the html for AJAX to use the cgenv.cgi example from the FAQ. You need not change anything with the cobol code.

#### ajax.html

```

<html>
<head>
<title>Simple Ajax Example</title>
<script language="Javascript">
function xmlhttpPost(strURL) {
    var xmlhttpReq = false;
    var self = this;
    // Mozilla/Safari
    if (window.XMLHttpRequest) {
        self.xmlhttpReq = new XMLHttpRequest();
    }
    // IE
    else if (window.ActiveXObject) {
        self.xmlhttpReq = new ActiveXObject("Microsoft.XMLHTTP");
    }
    self.xmlhttpReq.open('POST', strURL, true);
    self.xmlhttpReq.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    self.xmlhttpReq.onreadystatechange = function() {

```

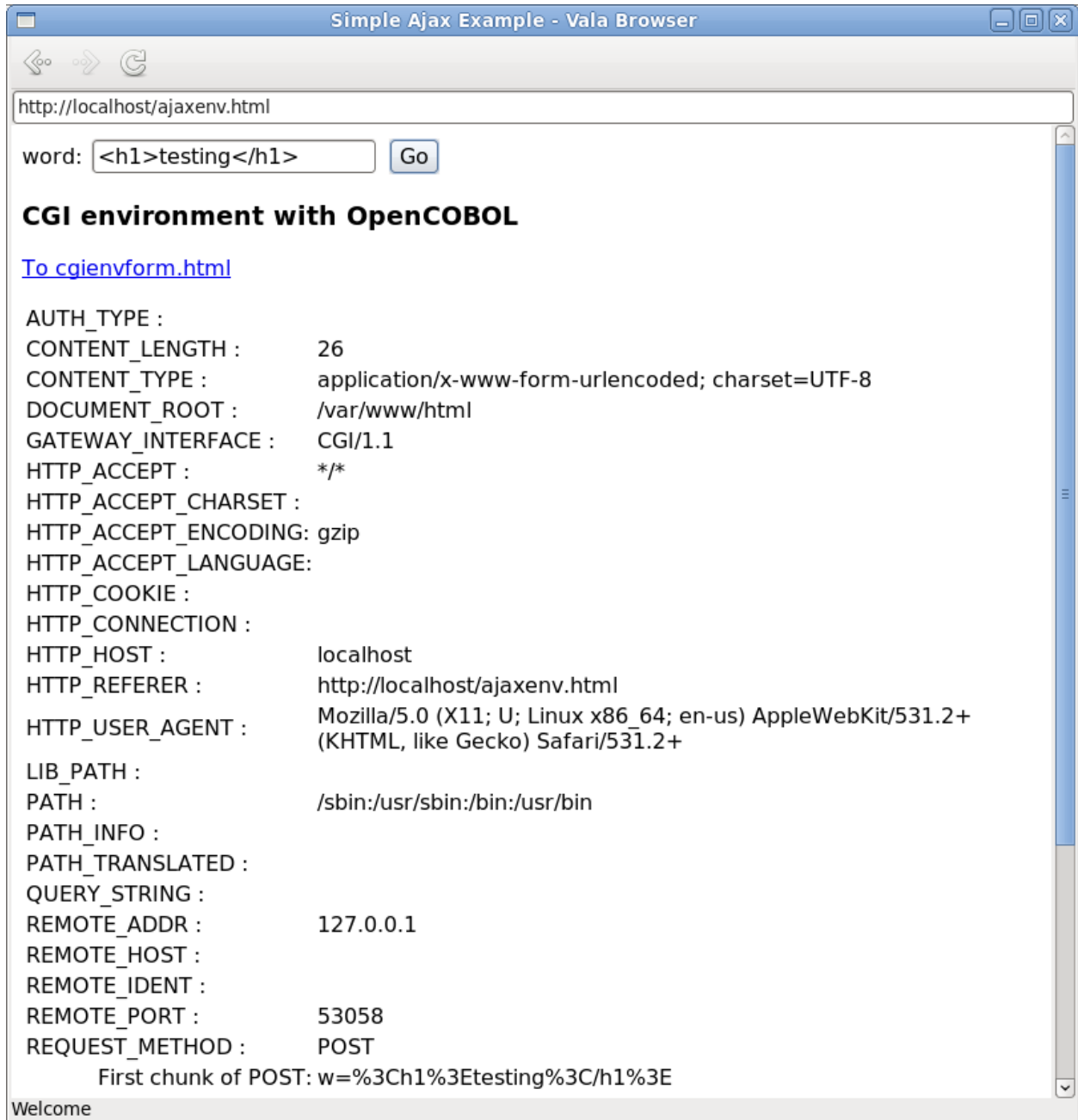
```
        if (self.xmlHttpRequest.readyState == 4) {
            updatepage(self.xmlHttpRequest.responseText);
        }
    }
    self.xmlHttpRequest.send(getquerystring());
}

function getquerystring() {
    var form      = document.forms['f1'];
    var word = form.word.value;
    qstr = 'w=' + escape(word); // NOTE: no '?' before querystring
    return qstr;
}

function updatepage(str) {
    document.getElementById("result").innerHTML = str;
}
</script>
</head>
<body>
<form name="f1">
    <p>word: <input name="word" type="text">
    <input value="Go" type="button" onclick=' javascript:xmlhttpPost ("/cgi-bin/cgienv.cgi") '></p>
    <div id="result"></div>
</form>
</body>
</html>
```

A quick screenshot from the Vala WebKit called from OpenCOBOL sample. *To be clear, this is a screenshot of an OpenCOBOL application that includes an embedded browser, displaying AJAX invoked OpenCOBOL CGI binaries (installed on the host without superuser access).* Take this one step further, and the browser application could utilize libSOUP and be its own webserver.

**Sometimes, just wow.** Ok, feel the need for marketing speak. “Moving beyond COBOL? Why? Move COBOL beyond.”



For those developers looking to serve OpenCOBOL applications on hosted systems and no super user privileges, see [How do I use LD\\_RUN\\_PATH with OpenCOBOL?](#) for some pointers on local library linkage.

## 29.2 5.2 What is ocdoc?

**ocdoc** is a small utility used to annotate sample programs and to support generation of Usage Documentation using COBOL sourced ReStructuredText extract lines.

ocdoc.cob

OCOBOL >>SOURCE FORMAT IS FIXED

```

*> *****
*><* =====
*><* ocdoc.cob usage guide
*><* =====
*><* .. sidebar:: Table of Contents
*><*
*><* .. contents:: :local:
*><*
*><* :Author: Brian Tiffin
*><* :Date: 30-Sep-2008
*><* :Rights: Copyright (c) 2008, Brian Tiffin.
*><* GNU FDL License.
*><* :Purpose: Extract usage document lines from COBOL sources.
*><* Using OpenCOBOL 1.1pr. OpenCOBOL is tasty.
*><* :Tectonics: cobc -x ocdoc.cob
*><* :Docgen: $ ./ocdoc ocdoc.cob ocdoc.rst ocdoc.html skin.css
*> *****
*><*
*><* -----
*><* Command line
*><* -----
*><* *ocdoc* runs in two forms.
*><*
*><* Without arguments, *ocdoc* will act as a pipe filter.
*><* Reading from standard in and writing the extract to standard
*><+ out.
*><*
*><* The *ocdoc* command also takes an input file, an extract
*><+ filename, an optional result file (with optional
*><+ stylesheet) and a verbosity option *-v* or a
*><+ special *-fixed* flag (to force skipping sequence numbers).
*><* If a result file is given, ocdoc will automatically
*><* run an *rst2html* command using the SYSTEM service.
*><*
*><* Due to an overly simplistic argument handler, you can only
*><+ turn on verbosity or -fixed when using all four filenames.
*><*
*><* Examples::
*><*
*><* $ cat ocdoc.cob | ocdoc >ocdoc.rst
*><* $ ./ocdoc ocdoc.cob ocdoc.rst
*><* $ ./ocdoc ocdoc.cob ocdoc.rst
*><+ ocdoc.html skin.css -fixed
*><* ...
*><* Input : ocdoc.cob
*><* Output : ocdoc.rst
*><* Command: rst2html --stylesheet=skin.css
*><+ ocdoc.rst ocdoc.html
*><*
*><* -----
*><* What is extracted
*><* -----
*><* - Lines that begin with \*><* *ignoring spaces*, are
*><+ extracted.
*><*
*><* - Lines that begin with \*><+ are appended to the
*><+ previous output line. As lines are trimmed of trailing

```



```

*><+ spaces, and *ocdoc* removes the space following the
*><+ extract triggers, you may need two spaces after an
*><+ ocdoc append.
*><*
*><* - Lines that begin with \*><[ begin a here document
*><+ with lines that follow extracted as is.
*><*
*><* - Lines that begin with \*><] close a here document.
*><+ Here document start and end lines are excluded from the
*><+ extract.
*><*
*><* -----
*><* Source code
*><* -----
*><* 'Download ocdoc.cob
*><+ <http://opencobol.addltocobol.com/ocdoc.cob> '_'
*><* 'See ocdocseq.cob
*><+ <http://opencobol.addltocobol.com/ocdocseq.html> '_'
*><*
*><! This is not extracted. Reminder of how to include source
*><! .. include:: ocdoc.cob
*><!     :literal:
*><*
*><* -----
*><* identification division
*><* -----
*><*
*><* ::
*><*
*><[
identification division.
program-id. OCDOC.

environment division.
input-output section.
file-control.
    select standard-input assign to KEYBOARD.
    select standard-output assign to DISPLAY.

    select source-input
assign to source-name
organization is line sequential
    .
    select doc-output
assign to doc-name
organization is line sequential
    .
*><]
*><*
*><* -----
*><* data division
*><* -----
*><*
*><* ::
*><*
*><[
data division.
file section.

```

```

fd standard-input.
  01 stdin-record      pic x(256).
fd standard-output.
  01 stdout-record     pic x(256).

fd source-input.
  01 source-record     pic x(256).
fd doc-output.
  01 doc-record        pic x(256).

working-storage section.
01 arguments           pic x(256).
01 source-name         pic x(256).
01 doc-name            pic x(256).
01 result-name         pic x(256).
01 style-name          pic x(256).
01 verbosity           pic x(9).
  88 verbose           values "-v" "--v" "-verbose" "--verbose".
  88 skipseqnum        values "-fix" "-fixed" "--fix" "--fixed".
01 usagehelp           pic x(6).
  88 helping           values "-h" "--h" "-help" "--help".
01 filter-flag         pic x value low-value.
  88 filtering         value high-value.

01 line-count          usage binary-long.
01 line-display        pic z(8)9.

*><]
*><*
*><* Note the conditional test for end of here doc
*><*
*><* ::
*><*
*><[
  01 trimmed           pic x(256).
  88 herestart         value "><[".
  88 hereend           value "><]".

  01 hereflag          pic x value low-value.
  88 heredoc           value high-value.
  88 herenone          value low-value.

*><]
*><*
*><* Note the here-record adds an odoc extract to lines that
*><+ follow.
*><*
*><* ::
*><*
*><[
  01 here-record.
  02 filler            pic x(5) value "><* ".
  02 here-data         pic x(251).

  01 seq-record.
  02 filler            pic x(7) value " ".
  02 seq-data         pic x(249).

```

```

01 doc-buffer          pic x(256).
01 buffer-offset      pic 999 usage comp-5 value 1.
01 buffer-flag        pic x value low-value.
    88 buffer-empty   value low-value.
    88 buffered-output value high-value.

01 counter            pic 999 usage comp-5.
01 len-of-comment     pic 999 usage comp-5.

01 first-part         pic x(8).
    88 special        values "><*" "><+".
    88 autodoc        value "><*".
    88 autoappend     value "><+".

01 rst-command        pic x(256).
01 result             usage binary-long.
*><]
*><*
*><* -----
*><* procedure division
*><* -----
*><*
*><* ::
*><*
*><[
*> *****
procedure division.

*><]
*><*
*><* Accept command line arguments. See if help requested.
*><*
*><* ::
*><*
*><[
accept arguments from command-line end-accept

move arguments to usagehelp
if helping
    display
        "$ ./ocdoc source markover [output [skin [--fixed]]]"
    end-display
    display "$ ./ocdoc" end-display
    display
        " without arguments extracts stdin to stdout"
    end-display
    goback
end-if

*><]
*><*
*><* Either run as filter or open given files. Two filenames
*><+ will generate an extract. Three will run the extract
*><+ through *rst2html* using an optional fourth filename
*><+ as a stylesheet.
*><*
*><* ::
*><*

```

```
*><[
*> Determine if this is running as a filter
  if arguments not equal spaces
    unstring arguments delimited by all spaces
      into source-name doc-name
        result-name style-name
        verbosity
    end-unstring

    open input source-input
    open output doc-output
  else
    set filtering to true

    open input standard-input
    open output standard-output
  end-if

*><]
*><*
*><* Initialize the output buffer, and line count.
*><*
*><* ::
*><*
*><[
  set buffer-empty to true
  move 1 to buffer-offset
  move spaces to doc-record
  move 0 to line-count

*><]
*><*
*><* The read is either from file or stdin. Start with the
*><+ first record.
*><*
*><* ::
*><*
*><[
*> filtering requires different reader loop
  if filtering
    read standard-input
      at end move high-values to stdin-record
    end-read
    move stdin-record to source-record
  else
    read source-input
      at end move high-values to source-record
    end-read
  end-if

*><]
*><*
*><* The main loop starts here, having done a pre-read to start
*><+ things off.
*><*
*><* ::
*><*
*><[
```

```

perform until source-record = high-values
    add 1 to line-count

*><]
*><*
*><* Small wrinkle if processing fixed form with sequence numbers,
*><+ as the heredoc end marker needs to be recognized
*><+ but we still want the sequence numbers in the heredoc.
*><*
*><* So files processed --fixed play some data shuffling games.
*><*
*><* ::
*><*
*><[
    if skipseqnum
        if heredoc
            move source-record(7 : 248) to trimmed
            move source-record to seq-data
            move seq-record to source-record
        else
            move source-record(7 : 248) to source-record
            move source-record to trimmed
        end-if
    else
        move function trim(source-record leading) to trimmed
    end-if

*><]
*><*
*><* First to check for here doc start and end, setting flag
*><+ if trimmed conditional the heredoc start or heredoc end
*><+ strings.
*><*
*><* ::
*><*
*><[
    if herestart
        set heredoc to true
    end-if

    if hereend
        set herenone to true
    end-if

*><]
*><*
*><* Inside the loop, we skip over heredoc entries.
*><+ If it is normal, than check for heredoc and include
*><+ source lines that follow, by prepending the extract tag
*><*
*><* ::
*><*
*><[
    if (not herestart) and (not hereend)
        if heredoc
            move source-record to here-data
            move here-record to trimmed
        end-if

```

```
*><]
*><*
*><* Unstring the line, looking for special tags in the first
*><+ part.
*><*
*><* ::
*><*
*><[

    unstring trimmed delimited by all spaces
        into first-part
        count in counter
    end-unstring

*><]
*><*
*><* If special, we either buffer or append to buffer
*><*
*><* ::
*><*
*><[

    evaluate true when special
        if autoappend and buffer-empty
            move spaces to doc-record
            move 1 to buffer-offset
        end-if

        if autodoc and buffered-output
            if filtering
                move doc-record to stdout-record
                write stdout-record end-write
            else
                write doc-record end-write
            end-if
            if verbose
                display
                    function trim(doc-record trailing)
                end-display
            end-if
            move spaces to doc-record
            set buffer-empty to true
            move 1 to buffer-offset
        end-if

*><]
*><*
*><* Skip over where the tag was found plus an extra space.
*><* Adding 2 skips over the assumed space after a special tag
*><*
*><* ::
*><*
*><[

    add 2 to counter
    compute len-of-comment =
        function length(trimmed) - counter
    end-compute

    if len-of-comment > 0
        move trimmed(counter : len-of-comment)
```

```

                to doc-buffer
            else
                move spaces to doc-buffer
            end-if

*><]
*><*
*><* Buffer the line, either to position 1 or appending to last.
*><*
*><* ::
*><*
*><[
        string
            function trim(doc-buffer trailing)
                delimited by size
            into doc-record
            with pointer buffer-offset
            on overflow
                move line-count to line-display
                display
                    "*** truncation *** reading line "
                    line-display
                end-display
            end-string
            set buffered-output to true
        end-evaluate
    end-if

*><]
*><*
*><* Again, we either read the next record from file or stdin.
*><*
*><* ::
*><*
*><[
        if filtering
            read standard-input
                at end move high-values to stdin-record
            end-read
            move stdin-record to source-record
        else
            read source-input
                at end move high-values to source-record
            end-read
        end-if
    end-perform

*><]
*><*
*><* We may or may not end up with buffered data
*><*
*><* ::
*><*
*><[
        if buffered-output
            set buffer-empty to true
            move 1 to buffer-offset
            if filtering

```

```
        move doc-record to stdout-record
        write stdout-record end-write
    else
        write doc-record end-write
    end-if
    if verbose
        display
            function trim(doc-record trailing)
        end-display
    end-if
    move spaces to doc-record
end-if

*><]
*><*
*><* Close the OpenCOBOL files
*><*
*><* ::
*><*
*><[
    if filtering
        close standard-output
        close standard-input
    else
        close doc-output
        close source-input
    end-if

    if verbose
        display "Input  : " function trim(source-name) end-display
        display "Output : " function trim(doc-name) end-display
    end-if

*><]
*><*
*><* If we have a result file, use the SYSTEM service to
*><+ generate an HTML file, possibly with stylesheet.
*><*
*><* ::
*><*
*><[
*> pass the extract through a markover, in this case ReST
move spaces to rst-command
if result-name not equal spaces
    if style-name equal spaces
        string
            "rst2html " delimited by size
            doc-name delimited by space
            " " delimited by size
            result-name delimited by space
            into rst-command
        end-string
    else
        string
            "rst2html --stylesheet=" delimited by size
            style-name delimited by space
            " " delimited by size
            doc-name delimited by space
```



```

        " " delimited by size
        result-name delimited by space
        into rst-command
    end-string
end-if

if verbose
    display
        "Command: "
        function trim(rst-command trailing)
    end-display
end-if

call "SYSTEM"
    using rst-command
    returning result
end-call

if result not equal zero
    display "HTML generate failed: " result end-display
end-if
end-if

*><]
*><*
*><* And before you know it, we are done.
*><*
*><* ::
*><*
*><[
    goback.

    end program OCDCOC.
*><]
*><*
*><* Don't forget to visit http://opencobol.org
*><*
*><* Cheers
*><*
*><* *Last edit:* 03-Oct-2008

```

### 29.2.1 5.2.1 ocdoc generated ocdoc documentation

See `ocdoc.html` for the output from processing `ocdoc.cob` with `ocdoc` using the tectonics listed in the source. `skin.css` ends up embedded in the html.

```

$ cobc -x ocdoc.cob
$ ./ocdoc ocdoc.cob ocdoc.rst ocdoc.html skin.css

```

## 29.3 5.3 What is CBL\_OC\_DUMP?

CBL\_OC\_DUMP is somewhat of a community challenge application to allow for runtime data dumps. Multiple postings to [opencobol.org](http://opencobol.org) has refined the hex display callable to:

OCOBOL >>SOURCE FORMAT IS FIXED

```

*-----
* Authors:   Brian Tiffin, Asger Kjelstrup, human
* Date:     27-Jan-2010
* Purpose:  Hex Dump display
* Tectonics: cobc -c CBL_OC_DUMP.cob
*          Usage: cobc -x program.cob -o CBL_OC_DUMP
*          export OC_DUMP_EXT=1 for explanatory text on dumps
*          (memory address and dump length)
*          export OC_DUMP_EXT=Y for extended explanatory text
*          (architecture and endian-order)
*-----

identification division.
program-id. CBL_OC_DUMP.
*
ENVIRONMENT      DIVISION.
CONFIGURATION    SECTION.
*
data division.
working-storage section.
77  addr                      usage pointer.
77  addr2addr                  usage pointer.
77  counter                    pic 9999999 usage comp-5.
77  byline                      pic 999 usage comp-5.
77  offset                      pic 9999999.
01  some                        pic 999 usage comp-5.
88  some-is-printable-iso88591
    values 32 thru 126, 160 thru 255.
88  some-is-printable-ebcdic
    values 64, 65, 74 thru 80, 90 thru 97,
           106 thru 111, 121 thru 127, 129 thru 137, 143,
           145 thru 153, 159, 161 thru 169, 176,
           186 thru 188, 192 thru 201, 208 thru 217, 224,
           226 thru 233, 240 thru 249.
77  high-var                    pic 99 usage comp-5.
77  low-var                      pic 99 usage comp-5.
*
01  char-set                      pic x(06).
88  is-ascii                      value 'ASCII'.
88  is-ebdic                      value 'EBCDIC'.
88  is-unknown                    value '?'.
01  architecture                  pic x(06).
88  is-32-bit                      value '32-bit'.
88  is-64-bit                      value '64-bit'.
01  endian-order                  pic x(10).
88  is-big-endian-no              value 'Little-Big'.
88  is-big-endian-yes            value 'Big-Little'.
*
77  hex-line                      pic x(48).
77  hex-line-pointer              pic 9(02) value 1.
*
77  show                          pic x(16).
77  dots                          pic x value '.'.
77  dump-dots                     pic x.
*
77  hex-digit                      pic x(16) value '0123456789abcdef'.
01  extended-infos                pic x.
88  show-extended-infos          values '1', '2', 'Y', 'y'.

```

```

      88 show-very-extended-infos values '2', 'Y', 'y'.
*
77 len                pic 999999 usage comp-5.
77 len-display        pic 999999.
*
linkage section.
01 buffer             pic x          any length.
77 byte               pic x.
*-----
procedure division using buffer.
*
MAIN SECTION.
00.
  perform starting-address
*
  perform varying counter from 0 by 16
    until counter >= len
    move counter to offset
    move spaces to hex-line, show
    move '-' to hex-line (24:01)
    move 1 to hex-line-pointer
    perform varying byline from 1 by 1
      until byline > 16
      if (counter + byline) > len
        if byline < 9
          move space to hex-line (24:01)
        end-if
        inspect show (byline:) replacing all spaces by dots
        exit perform
      else
        move buffer (counter + byline : 1) to byte
        perform calc-hex-value
        if ((some-is-printable-iso88591 and is-ascii) or
            (some-is-printable-ebcdic and is-ebdic) )
          move byte to show (byline:1)
        else
          move dots to show (byline:1)
        end-if
      end-if
    end-perform
    display offset ' ' hex-line ' ' show
    end-display
  end-perform
  display ' '
  end-display
*
  continue.
ex. exit program.
*-----
CALC-HEX-VALUE SECTION.
00.
  subtract 1 from function ord(byte) giving some
  end-subtract
  divide some by 16 giving high-var remainder low-var
  end-divide
  string hex-digit (high-var + 1:1)
         hex-digit (low-var + 1:1)
         space

```

```
        delimited by size
        into hex-line
        with pointer hex-line-pointer
    end-string
*
    continue.
ex. exit.
*-----
STARTING-ADDRESS SECTION.
00.
* Get the length of the transmitted buffer
    CALL 'C$PARAMSIZE' USING 1
        GIVING len
    END-CALL
* If wanted, change the dots to something different than points
    accept dump-dots from environment 'OC_DUMP_DOTS'
        not on exception
        move dump-dots to dots
    end-accept
*
    perform TEST-ASCII
    perform TEST-ENDIAN
    set addr      to address of buffer
    set addr2addr to address of addr
*
    if len > 0
* To show hex-address, reverse if Big-Little Endian
        if is-big-endian-yes
            set addr2addr up   by LENGTH OF addr
            set addr2addr down by 1
        end-if
        move 1 to hex-line-pointer
        perform varying byline from 1 by 1
            until byline > LENGTH OF addr
                set address of byte to addr2addr
                perform calc-hex-value
                if is-big-endian-yes
                    set addr2addr down by 1
                else
                    set addr2addr up   by 1
                end-if
            end-perform
        end-if
*
* Get and display characteristics and headline
    accept extended-infos from environment 'OC_DUMP_EXT'
    end-accept
    if show-extended-infos
        display ' '
        end-display
        if len > 0
            end-display
            display 'Dump of memory beginning at Hex-address: '
                hex-line (1 : 3 * (byline - 1) )
            end-display
        end-if
        move len to len-display
        display 'Length of memory dump is: ' len-display
```

```

end-display
if show-very-extended-infos
  perform TEST-64bit
  display 'Program runs in '
    architecture ' architecture. '
    'Char-set is '
    function trim (char-set) '.'
  end-display
  display 'Byte order is ' endian-order
    ' endian.'
  end-display
end-if
end-if
*
* Do we have anything to dump?
  if len > 0
* Ensure that the passed size is not too big
  if len > 999998
    move 999998 to len, len-display
    display 'Warning, only the first '
      len-display ' Bytes are shown!'
    end-display
  end-if
  display ' '
  end-display
  display 'Offset '
    'HEX-- -- -- -5 -- -- -- -- 10 '
    '-- -- -- -- 15 -- '
    ' '
    'CHARS----1----5-'
  end-display
else
  display ' '
  end-display
  display 'Nothing to dump.'
  end-display
end-if
*
  continue.
ex. exit.
*-----
TEST-ASCII SECTION.
*Function: Discover if running Ascii or Ebcdic
00.
  evaluate space
  when x'20'
    set is-ascii to true
  when x'40'
    set is-ebdic to true
  when other
    set is-unknown to true
  end-evaluate
*
  continue.
ex. exit.
*-----
TEST-64BIT SECTION.
*Function: Discover if running 32/64 bit

```

```

00.
*   Longer pointers in 64-bit architecture
    if function length (addr) <= 4
        set is-32-bit to true
    else
        set is-64-bit to true
    end-if
*
    continue.
ex. exit.
*-----
TEST-ENDIAN SECTION.
00.
*   Number-bytes are shuffled in Big-Little endian
    move 128 to byline
    set address of byte to address of byline
    if function ord(byte) > 0
        set is-big-endian-yes to true
    else
        set is-big-endian-no to true
    end-if
*
    continue.
ex. exit.
*-----
end program CBL_OC_DUMP.

```

Example displays:

Alpha literal Dump

```

Offs  HEX-- -- -- 5- -- -- -- -- 10 -- -- -- -- 15 -- CHARS----1----5-
0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6f 70 71 abcdefghijklmopq
0016  72                                     r.....

```

Integer Dump: +0000000123

```

Offs  HEX-- -- -- 5- -- -- -- -- 10 -- -- -- -- 15 -- CHARS----1----5-
0000  7b 00 00 00                                     {.....

```

Or with OC\_DUMP\_EXT enviroment variable set to Y:

Numeric Literal Dump: 0

Dump of memory beginning at Hex-address: bf 80 fc e4  
Program runs in 32-bit architecture. Char-set is ASCII .  
Byte order is Big-Little endian.

```

Offs  HEX-- -- -- 5- -- -- -- -- 10 -- -- -- -- 15 -- CHARS----1----5-
0000  00                                     .....

```

### 29.3.1 5.3.1 Update to OC\_CBL\_DUMP

human posted a new version that displays the dump upon SYSERR. Goes to show the activity that can spring forth from a keen and engaged community.

*Edit 19-Oct-2010: Put all dump-outputs to syserr. Removed unused paragraphs and minor beauty changes.*

OCOBOL >>SOURCE FORMAT IS FIXED

```

*-----
* Authors:   Brian Tiffin, Asger Kjelstrup, Simon Sobisch
* Date:     19-Oct-2010
* Purpose:  Hex Dump display
* Tectonics: cobb -c CBL_OC_DUMP.cob
* Usage:    export OC_DUMP_EXT=1 for explanatory text on dumps
*           (memory address and dump length)
*           export OC_DUMP_EXT=Y for extended explanatory text
*           (architecture and endian-order)
*-----
IDENTIFICATION DIVISION.
PROGRAM-ID. CBL_OC_DUMP.
*
ENVIRONMENT      DIVISION.
CONFIGURATION    SECTION.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
77  addr                      usage pointer.
77  addr2addr                 usage pointer.
77  counter                   pic 999999 usage comp-5.
77  byline                     pic 999   usage comp-5.
77  offset                     pic 999999.
01  some                       pic 999   usage comp-5.
88  some-is-printable-iso88591
    values 32 thru 126, 160 thru 255.
88  some-is-printable-ebcdic
    values 64, 65, 74 thru 80, 90 thru 97,
           106 thru 111, 121 thru 127, 129 thru 137, 143,
           145 thru 153, 159, 161 thru 169, 176,
           186 thru 188, 192 thru 201, 208 thru 217, 224,
           226 thru 233, 240 thru 249.
77  high-var                   pic 99   usage comp-5.
77  low-var                     pic 99   usage comp-5.
*
01  char-set                    pic x(06).
88  is-ascii                    value 'ASCII'.
88  is-ebdic                    value 'EBCDIC'.
88  is-unknown                  value '?'.
01  architecture                pic x(06).
88  is-32-bit                   value '32-bit'.
88  is-64-bit                   value '64-bit'.
01  endian-order                pic x(10).
88  is-big-endian-no            value 'Little-Big'.
88  is-big-endian-yes          value 'Big-Little'.
*
77  hex-line                    pic x(48).
77  hex-line-pointer            pic 9(02) value 1.
*
77  show                        pic x(16).
77  dots                        pic x value '.'.
77  dump-dots                   pic x.
*
77  hex-digit                   pic x(16) value '0123456789abcdef'.
01  extended-infos              pic x.
88  show-extended-infos         values '1', '2', 'Y', 'y'.
88  show-very-extended-infos   values '2', 'Y', 'y'.

```

```

*
77 len                pic 999999 usage comp-5.
77 len-display        pic 999999.
*
LINKAGE SECTION.
01 buffer             pic x          any length.
77 byte              pic x.
-----
PROCEDURE DIVISION USING buffer.
*
*MAIN SECTION.
*00.
    perform starting-address
*
    perform varying counter from 0 by 16
        until counter >= len
        move counter to offset
        move spaces to hex-line, show
        move '-' to hex-line (24:01)
        move 1 to hex-line-pointer
        perform varying byline from 1 by 1
            until byline > 16
            if (counter + byline) > len
                if byline < 9
                    move space to hex-line (24:01)
                end-if
                inspect show (byline:) replacing all spaces by dots
                exit perform
            else
                move buffer (counter + byline : 1) to byte
                perform calc-hex-value
                if ((some-is-printable-iso88591 and is-ascii) or
                    (some-is-printable-ebcdic and is-ebdic) )
                    move byte to show (byline:1)
                else
                    move dots to show (byline:1)
                end-if
            end-if
        end-perform
        display offset ' ' hex-line ' ' show
            upon SYSERR
        end-display
    end-perform
    display ' '
        upon SYSERR
    end-display
*
    exit program.
-----
CALC-HEX-VALUE SECTION.
*00.
    subtract 1 from function ord(byte) giving some
    end-subtract
    divide some by 16 giving high-var remainder low-var
    end-divide
    string hex-digit (high-var + 1:1)
        hex-digit (low-var + 1:1)
        space

```



```

        delimited by size
        into hex-line
        with pointer hex-line-pointer
    end-string
*
    exit section.
-----
STARTING-ADDRESS SECTION.
*00.
* Get the length of the transmitted buffer
    CALL 'C$PARAMSIZE' USING 1
        GIVING len
    END-CALL
* If wanted, change the dots to something different than points
    accept dump-dots from environment 'OC_DUMP_DOTS'
        not on exception
        move dump-dots to dots
    end-accept
*
    perform TEST-ASCII
    perform TEST-ENDIAN
    set addr      to address of buffer
    set addr2addr to address of addr
*
    if len > 0
* To show hex-address, reverse if Big-Little Endian
        if is-big-endian-yes
            set addr2addr up   by LENGTH OF addr
            set addr2addr down by 1
        end-if
        move 1 to hex-line-pointer
        perform varying byline from 1 by 1
            until byline > LENGTH OF addr
                set address of byte to addr2addr
                perform calc-hex-value
                if is-big-endian-yes
                    set addr2addr down by 1
                else
                    set addr2addr up   by 1
                end-if
            end-perform
        end-if
*
* Get and display characteristics and headline
    accept extended-infos from environment 'OC_DUMP_EXT'
    end-accept
    if show-extended-infos
        display ' '
            upon SYSERR
        end-display
        if len > 0
            display 'Dump of memory beginning at Hex-address: '
                hex-line (1 : 3 * (byline - 1) )
                upon SYSERR
            end-display
        end-if
        move len to len-display
        display 'Length of memory dump is: ' len-display

```

```
        upon SYSERR
end-display
if show-very-extended-infos
    perform TEST-64bit
    display 'Program runs in '
        architecture ' architecture. '
        'Char-set is '
        function trim (char-set) '.'
        upon SYSERR
    end-display
    display 'Byte order is ' endian-order
        ' endian.'
        upon SYSERR
    end-display
end-if
end-if
*
* Do we have anything to dump?
    if len > 0
* Ensure that the passed size is not too big
        if len > 999998
            move 999998 to len, len-display
            display 'Warning, only the first '
                len-display ' Bytes are shown!'
                upon SYSERR
            end-display
        end-if
        display ' '
            upon SYSERR
        end-display
        display 'Offset '
            'HEX-- -- -- -5 -- -- -- -- 10 '
            ' -- -- -- -- 15 -- '
            ' , '
            'CHARS----1----5-'
            upon SYSERR
        end-display
    else
        display ' '
            upon SYSERR
        end-display
        display 'Nothing to dump.'
            upon SYSERR
        end-display
    end-if
*
    exit section.
*-----
TEST-ASCII SECTION.
*Function: Discover if running Ascii or Ebcadic
*00.
    evaluate space
        when x'20'
            set is-ascii to true
        when x'40'
            set is-ebdic to true
        when other
            set is-unknown to true
```

```

    end-evaluate
*
    exit section.
*-----
TEST-64BIT SECTION.
*Function: Discover if running 32/64 bit
*00.
*   Longer pointers in 64-bit architecture
    if function length (addr) <= 4
        set is-32-bit to true
    else
        set is-64-bit to true
    end-if
*
    exit section.
*-----
TEST-ENDIAN SECTION.
*00.
*   Number-bytes are shuffled in Big-Little endian
    move 128 to byline
    set address of byte to address of byline
    if function ord(byte) > 0
        set is-big-endian-yes to true
    else
        set is-big-endian-no to true
    end-if
*
    exit section.
*-----
end program CBL_OC_DUMP.

```

## 29.4 5.4 Does OpenCOBOL support any SQL databases?

Yes. There is no embedded SQL in OpenCOBOL in terms of EXEC but there are *at least* two usable CALL extensions, the EXEC potential of the Firebird **gpre** and the tried and successful use of Oracle's **procob**. There are currently (*February 2013*) quite a few active developments for easing SQL engine access.

- as reported on [opencobol.org](http://opencobol.org) the **procob 10.2 Oracle** preprocessor produces code that compiles and executes just fine with OpenCOBOL 1.1 See note about data sizes and the *binary-size*: configuration below.
- There are workable prototypes for SQLite at [ocshell.c](#)
  - with a sample usage program at [sqlscreen.cob](#)
  - and supporting documentation at [sqlscreen.html](#)
- The SQLite extension comes in two flavours; a shell mode discussed above and a direct API interface housed at [ocsqlite.c](#)
- A libdbi (generic database access) extension is also available. See [cobdbi](#) for full details.
- Efforts toward providing a preprocessor for EXEC are underway.
- Jim Currey's team has kindly posted an ease-of-use MySQL preprocessing layer.
  - <http://svn.wp0.org/add1/libraries/mysql4Windows4OpenCobol/>
- Rumours of a potential Postgres layer have also been heard.
  - Not a rumour anymore. Work on a nicely complete PostgreSQL binding was posted by [gehudyk](#) to

– [http://www.opencobol.org/modules/newbb/viewtopic.php?topic\\_id=868&forum=1&post\\_id=4142](http://www.opencobol.org/modules/newbb/viewtopic.php?topic_id=868&forum=1&post_id=4142)

- **AND** as a *thing to watch for*, one of the good people of the OpenCOBOL community has written a layer that converts READ and WRITE verbage to SQL calls *at run time*. More on this as it progresses.

## 29.4.1 5.4.1 Oracle procob and binary data sizes

Details of the configuration setting for proper Oracle procob processing.

From Angus on [opencobol.org](http://opencobol.org)

Hi

I had some trouble with Oracle procob 10.2 and OpenCobol 1.1 with std=mf. For PIC S9(2) COMP, procob seems to use 2 bytes, and OpenCobol only one. It doesn't work well. It comes from the parameter binary-size in the mf.conf, which seems to tell to opencobol the larger of comp type I modify to binary-size: 2-4-8 and it works (same as the mvs.conf) Our application works with Microfocus / Oracle, and microfocus use 2 bytes, like Oracle. Perhaps because we have the mvs toggle

Except for this thing, opencobol and oracle work like a charm, on a debian 32bit.

Regards,  
Angus

## 29.4.2 5.4.2 PostgreSQL Sample

Nowhere near as complete as the binding that Gerald posted to [opencobol.org](http://opencobol.org), the example below was a starting point.

See [http://www.opencobol.org/modules/newbb/viewtopic.php?topic\\_id=868&forum=1#forumpost4142](http://www.opencobol.org/modules/newbb/viewtopic.php?topic_id=868&forum=1#forumpost4142)

Note that the PostgreSQL runtime library is **libpq**, ending in *q* not *g*.

```
OCOBOL*> *****
*> Author:      Brian Tiffin
*> Date:        20091129
*> Purpose:     PostgreSQL connection test
*> Tectonics:   cobb -x -lpq pgcob.cob
*> *****
identification division.
program-id. pgcob.

data division.
working-storage section.
01 pgconn usage pointer.
01 pgres  usage pointer.
01 resptr usage pointer.
01 resstr pic x(80) based.
01 result usage binary-long.
01 answer pic x(80).

*> *****
procedure division.
display "Before connect:" pgconn end-display

call "PQconnectdb" using
```

```

    by reference "dbname = postgres" & x"00"
    returning pgconn
end-call
display "After connect: " pgconn end-display

call "PQstatus" using by value pgconn returning result end-call
display "Status:      " result end-display

call "PQuser" using by value pgconn returning resptr end-call

set address of resstr to resptr
string resstr delimited by x"00" into answer end-string
display "User:      " function trim(answer) end-display

display "call PQexec" end-display
call "PQexec" using
    by value pgconn
    by reference "select version();" & x"00"
    returning pgres
end-call
display pgres end-display

*> Pull out a result. row 0, field 0 <*>
call "PQgetvalue" using
    by value pgres
    by value 0
    by value 0
    returning resptr
end-call
set address of resstr to resptr
string resstr delimited by x"00" into answer end-string
display "Version:    " answer end-display

call "PQfinish" using by value pgconn returning null end-call
display "After finish: " pgconn end-display

call "PQstatus" using by value pgconn returning result end-call
display "Status:      " result end-display

*> this will now return garbage <*>
call "PQuser" using by value pgconn returning resptr end-call
set address of resstr to resptr
string resstr delimited by x"00" into answer end-string
display "User after:  " function trim(answer) end-display

goback.
end program pgcob.

```

Run from a user account that has default PostgreSQL credentials:

```

$ cobc -x -lpq pgcob.cob
$ ./pgcob
Before connect:0x00000000
After connect: 0x086713e8
Status:      +0000000000
User:        brian
call PQexec
0x08671a28

```

```
Version:      PostgreSQL 8.3.7 on i486-pc-linux-gnu, compiled by GCC gcc-4.3.real (Debian 4.3).
After finish: 0x086713e8
Status:      +0000000001
User after:   PostgreSQL 8.3.7 on i486-pc-linux-gnu, compiled by GCC gcc-4.3.real (Debian 4.3).
```

Note that *User after* is not the valid answer, shown on purpose. The connection had been closed and the status was correctly reported as non-zero, being an error, but this example continued through as a demonstration.

## 29.5 5.5 Does OpenCOBOL support ISAM?

Yes. The official release used Berkeley DB, but there are also experimental configurations of the compiler that use VBISAM, CISAM, DISAM or other external handlers. See [What are the configure options available for building OpenCOBOL?](#) for more details about these options. The rest of this entry assumes the default Berkeley database.

ISAM is an acronym for Indexed Sequential Access Method.

OpenCOBOL has fairly full support of all standard specified ISAM compile and runtime semantics.

For example

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*><* =====
*><* indexing example
*><* =====
*><* :Author:      Brian Tiffin
*><* :Date:        17-Feb-2009
*><* :Purpose:     Fun with Indexed IO routines
*><* :Tectonics:   cobc -x indexing.cob
*> *****
identification division.
program-id. indexing.

environment division.
configuration section.

input-output section.
file-control.
  select optional indexing
  assign to "indexing.dat"
  organization is indexed
  access mode is dynamic
  record key is keyfield of indexing-record
  alternate record key is splitkey of indexing-record
  with duplicates
  .

*> ** OpenCOBOL does not yet support split keys **
*> alternate record key is newkey
*> source is first-part of indexing-record
*> last-part of indexing-record
*> with duplicates

data division.
file section.
fd indexing.
  01 indexing-record.
```

```

03 keyfield          pic x(8) .
03 splitkey.
   05 first-part     pic 99.
   05 middle-part    pic x.
   05 last-part      pic 99.
03 data-part        pic x(54) .

working-storage section.
01 display-record.
   03 filler          pic x(4) value spaces.
   03 keyfield        pic x(8) .
   03 filler          pic xx value spaces.
   03 splitkey.
     05 first-part    pic z9.
     05 filler        pic x value space.
     05 middle-part   pic x.
     05 filler        pic xx value all "+".
     05 last-part     pic z9.
   03 filler          pic x(4) value all "-".
   03 data-part      pic x(54) .

*> control break
01 oldkey           pic 99x99.

*> In a real app this should well be two separate flags
01 control-flag     pic x.
   88 no-more-duplicates value high-value
     when set to false is low-value.
   88 no-more-records value high-value
     when set to false is low-value.

*> *****
procedure division.

*> Open optional index file for read write
open i-o indexing

*> populate a sample database
move "1234567800a01some 12345678 data here" to indexing-record
perform write-indexing-record
move "8765432100a01some 87654321 data here" to indexing-record
perform write-indexing-record
move "1234876500a01some 12348765 data here" to indexing-record
perform write-indexing-record
move "8765123400a01some 87651234 data here" to indexing-record
perform write-indexing-record

move "1234567900b02some 12345679 data here" to indexing-record
perform write-indexing-record
move "9765432100b02some 97654321 data here" to indexing-record
perform write-indexing-record
move "1234976500b02some 12349765 data here" to indexing-record
perform write-indexing-record
move "9765123400b02some 97651234 data here" to indexing-record
perform write-indexing-record

move "1234568900c13some 12345689 data here" to indexing-record
perform write-indexing-record

```

```
move "9865432100c13some 98654321 data here" to indexing-record
perform write-indexing-record
move "1234986500c13some 12349865 data here" to indexing-record
perform write-indexing-record
move "9865123400c13some 98651234 data here" to indexing-record
perform write-indexing-record

*> close it ... not necessary, but for the example
close indexing

*> clear the record space for this example
move spaces to indexing-record

*> open the data file again
open i-o indexing

*> read all the duplicate 00b02 keys
move 00 to first-part of indexing-record
move "b" to middle-part of indexing-record
move 02 to last-part of indexing-record

*> using read key and then next key / last key compare
set no-more-duplicates to false
perform read-indexing-record
perform read-next-record
    until no-more-duplicates

*> read by key of reference ... the cool stuff
move 00 to first-part of indexing-record
move "a" to middle-part of indexing-record
move 02 to last-part of indexing-record

*> using start and read next
set no-more-records to false
perform start-at-key
perform read-next-by-key
    until no-more-records

*> read by primary key of reference
move "87654321" to keyfield of indexing-record

*>
set no-more-records to false
perform start-previous-key
perform read-previous-by-key
    until no-more-records

*> and with that we are done with indexing sample
close indexing

goback.
*> *****

*><* Write paragraph
write-indexing-record.
    write indexing-record
        invalid key
        display
```



```

        "rewrite key: " keyfield of indexing-record
    end-display
    rewrite indexing-record
        invalid key
        display
            "really bad key: "
            keyfield of indexing-record
        end-display
    end-rewrite
end-write
.

*><* read by alternate key paragraph
read-indexing-record.
    display "Reading: " splitkey of indexing-record end-display
    read indexing key is splitkey of indexing-record
invalid key
    display
        "bad read key: " splitkey of indexing-record
    end-display
    set no-more-duplicates to true
end-read
.

*><* read next sequential paragraph
read-next-record.
    move corresponding indexing-record to display-record
    display display-record end-display
    move splitkey of indexing-record to oldkey

    read indexing next record
        at end set no-more-duplicates to true
        not at end
            if oldkey not equal splitkey of indexing-record
                set no-more-duplicates to true
            end-if
    end-read
.

*><* start primary key of reference paragraph
start-prime-key.
    display "Prime < " keyfield of indexing-record end-display
    start indexing
        key is less than
            keyfield of indexing-record
    invalid key
        display
            "bad start: " keyfield of indexing-record
        end-display
        set no-more-records to true
    not invalid key
        read indexing previous record
            at end set no-more-records to true
        end-read
    end-start
.

*><* read previous by key of reference paragraph

```

```
read-previous-by-key.  
  move corresponding indexing-record to display-record  
  display display-record end-display  
  
  read indexing previous record  
  at end set no-more-records to true  
  end-read  
.  
*><* start alternate key of reference paragraph  
start-at-key.  
  display "Seeking >= " splitkey of indexing-record end-display  
  start indexing  
  key is greater than or equal to  
  splitkey of indexing-record  
  invalid key  
  display  
  "bad start: " splitkey of indexing-record  
  end-display  
  set no-more-records to true  
  not invalid key  
  read indexing next record  
  at end set no-more-records to true  
  end-read  
end-start  
.  
*><* read next by key of reference paragraph  
read-next-by-key.  
  move corresponding indexing-record to display-record  
  display display-record end-display  
  
  read indexing next record  
  at end set no-more-records to true  
  end-read  
.  
end program indexing.  
*><*<br>*><* Last Update: 20090220
```

which outputs:

```
Reading: 00b02  
12345679 0 b++ 2----some 12345679 data here  
97654321 0 b++ 2----some 97654321 data here  
12349765 0 b++ 2----some 12349765 data here  
97651234 0 b++ 2----some 97651234 data here  
12345679 0 b++ 2----some 12345679 data here  
97654321 0 b++ 2----some 97654321 data here  
12349765 0 b++ 2----some 12349765 data here  
97651234 0 b++ 2----some 97651234 data here  
12345679 0 b++ 2----some 12345679 data here  
97654321 0 b++ 2----some 97654321 data here  
12349765 0 b++ 2----some 12349765 data here  
97651234 0 b++ 2----some 97651234 data here  
Seeking >= 00a02  
12345679 0 b++ 2----some 12345679 data here  
97654321 0 b++ 2----some 97654321 data here  
12349765 0 b++ 2----some 12349765 data here
```

```

97651234 0 b++ 2----some 97651234 data here
12345679 0 b++ 2----some 12345679 data here
97654321 0 b++ 2----some 97654321 data here
12349765 0 b++ 2----some 12349765 data here
97651234 0 b++ 2----some 97651234 data here
12345679 0 b++ 2----some 12345679 data here
97654321 0 b++ 2----some 97654321 data here
12349765 0 b++ 2----some 12349765 data here
97651234 0 b++ 2----some 97651234 data here
12345689 0 c++13----some 12345689 data here
98654321 0 c++13----some 98654321 data here
12349865 0 c++13----some 12349865 data here
98651234 0 c++13----some 98651234 data here
12345689 0 c++13----some 12345689 data here
98654321 0 c++13----some 98654321 data here
12349865 0 c++13----some 12349865 data here
98651234 0 c++13----some 98651234 data here
12345689 0 c++13----some 12345689 data here
98654321 0 c++13----some 98654321 data here
12349865 0 c++13----some 12349865 data here
98651234 0 c++13----some 98651234 data here
Prime < 87654321
87651234 0 a++ 1----some 87651234 data here
12349865 0 c++13----some 12349865 data here
12349765 0 b++ 2----some 12349765 data here
12348765 0 a++ 1----some 12348765 data here
12345689 0 c++13----some 12345689 data here
12345679 0 b++ 2----some 12345679 data here
12345678 0 a++ 1----some 12345678 data here

```

on any first runs, where **indexing.dat** does not exist. Subsequent runs have the same output with:

```

rewrite key: 12345678
rewrite key: 87654321
rewrite key: 12348765
rewrite key: 87651234
rewrite key: 12345679
rewrite key: 97654321
rewrite key: 12349765
rewrite key: 97651234
rewrite key: 12345689
rewrite key: 98654321
rewrite key: 12349865
rewrite key: 98651234

```

prepended, as the WRITE INVALID KEY clause triggers a REWRITE to allow overwriting key and data.

## 29.5.1 5.5.1 FILE STATUS

Historically, the condition of a COBOL I/O operation is set in an identifier specified in a *FILE STATUS IS* clause.

John Ellis did us the favour of codifying the OpenCOBOL FILE STATUS codes See [ISAM](#) for the details.

## 29.6 5.6 Does OpenCOBOL support modules?

Yes. Quite nicely in fact. Dynamically! COBOL modules, and object files of many other languages are linkable. As OpenCOBOL uses intermediate C, linkage to other languages is well supported across many platforms. The OpenCOBOL `CALL` instruction maps COBOL `USAGE` to many common C stack frame data representations.

Multipart, complex system development is well integrated in the OpenCOBOL model.

```
$ cobc -b hello.cob goodbye.cob
```

Combines both source files into a single dynamically loadable module. Example produces `hello.so`.

Using the `-l` link library option, OpenCOBOL has access to most shared libraries supported on its platforms.

```
$ cobc -x -lcurl showcurl.cob
```

Will link the `/usr/lib/libcurl.so` (from the *cURL project*) to `showcurl`. The OpenCOBOL `CALL` verb will use this linked library to resolve calls at runtime.

Large scale systems are at the heart of COBOL development and OpenCOBOL is no exception.

For more information, see [What is COB\\_PRE\\_LOAD?](#).

## 29.7 5.7 What is COB\_PRE\_LOAD?

`COB_PRE_LOAD` is an environment variable that controls what dynamic link modules are included in a run.

For example:

```
$ cobc occurl.c
$ cobc occgi.c
$ cobc -x myprog.cob
$ export COB_PRE_LOAD=occurl:occgi
$ ./myprog
```

That will allow the OpenCOBOL runtime link resolver to find the entry point for `CALL "CBL_OC_CURL_INIT"` in the `occurl.so` module. *Note:* the modules listed in the `COB_PRE_LOAD` environment variable DO NOT have extensions. OpenCOBOL will do the right thing on the various platforms.

If the `DSO` files are not in the current working directory along with the executable, the `COB_LIBRARY_PATH` can be set to find them.

See [What is COB\\_LIBRARY\\_PATH?](#) for information on setting the module search path.

## 29.8 5.8 What is the OpenCOBOL LINKAGE SECTION for?

Argument passing in COBOL is normally accomplished through the `LINKAGE SECTION`. This section does not allocate or initialize memory as would definitions in the `WORKING-STORAGE SECTION`.

Care must be taken to inform COBOL of the actual source address of these variables before use. Influences `CHAINING` and `USING` phrases. See `CALL` for more details.

## 29.9 5.9 What does the `-fstatic-linkage` OpenCOBOL compiler option do?

Under normal conditions, the *LINKAGE SECTION* is unallocated and uninitialized. When a *LINKAGE SECTION* variable, that is not part of the *USING* phrase (not a named calling argument), any memory that has been addressed becomes unaddressable across calls. *-fstatic-linkage* creates static addressing to the *LINKAGE SECTION*.

From [Roger]:

```
This relates to LINKAGE items that are NOT referred
to in the USING phrase of the PROCEDURE DIVISION.
It also only has relevance when the program is CALL'ed
from another prog.
This means that the addressability of these items must
be programmed (usually with SET ADDRESS) before reference.
```

```
Per default, the item loses its addressability on exit
from the program. This option causes the module to retain
the item's address between CALL invocations of the program.
```

With some rumours that this may become the default in future releases of OpenCOBOL, and the *-fstatic-linkage* option may be deprecated.

## 29.10 5.10 Does OpenCOBOL support Message Queues?

Yes, but not out of the box. A linkable POSIX message queue layer is available.

```
/* OpenCOBOL access to POSIX Message Queues */
/* Author: Brian Tiffin */
/* Date: August, 2008 */
/* Build: gcc -c ocmq.c */
/* Usage: cobc -x -lrt program.cob ocmq.o */

#include <fcntl.h> /* For O_* constants */
#include <sys/stat.h> /* For mode constants */
#include <errno.h> /* Access to error values */
#include <mqueue.h> /* The message queues */
#include <signal.h> /* for notification */
#include <time.h> /* for the timed versions */
#include <stdio.h>
#include <string.h> /* For strerror */

#include <libcob.h> /* for cob_resolve */

/* Forward declarations */
static void ocmq_handler(int, siginfo_t *, void *);
static void (*MQHANDLER)(int *mqid);

/* Return C runtime global errno */
int ERRORNUMBER() {
    return errno;
}

/* Load a COBOL field with an error string */
int ERRORSTRING(char *errbuff, int buflen) {
    void *temperr;
```

```
temperr = strerror(errno);
memcpy((void *)errbuff, temperr, buflen);
return strlen(temperr);
}

/*
/* Open Message Queue */
int MQOPEN(char *mqname, int oflags) {
    mqd_t mqres;

    errno = 0;
    mqres = mq_open(mqname, oflags);
    return (int)mqres;
}

/* Creating a queue requires two extra arguments, permissions and attributes */
int MQCREATE(char *mqname, int oflags, int perms, char *mqattr) {
    mqd_t mqres;

    errno = 0;
    mqres = mq_open(mqname, oflags, (mode_t)perms, (struct mq_attr *)mqattr);
    return (int)mqres;
}

/* Get current queue attributes */
int MQGETATTR(int mqid, char *mqattr) {
    mqd_t mqres;

    errno = 0;
    mqres = mq_getattr((mqd_t)mqid, (struct mq_attr *)mqattr);
    return (int)mqres;
}

/* Set current queue attributes */
/* only accepts mqflags of 0 or MQO_NONBLOCK once created */
int MQSETATTR(int mqid, char *mqattr, char *oldattr) {
    mqd_t mqres;

    errno = 0;
    mqres = mq_setattr((mqd_t)mqid, (struct mq_attr *)mqattr, (struct mq_attr *)oldattr);
    return (int)mqres;
}

/* Send a message to the queue */
int MQSEND(int mqid, char *message, int length, unsigned int mqprio) {
    mqd_t mqres;

    errno = 0;
    mqres = mq_send((mqd_t)mqid, message, (size_t)length, mqprio);
    return (int)mqres;
}

/* Read the highest priority message */
int MQRECEIVE(int mqid, char *msgbuf, int buflen, int *retprio) {
    ssize_t retlen;

    errno = 0;
```

```

    retlen = mq_receive((mqd_t)mqid, msgbuf, buflen, retprio);
    return (int)retlen;
}

/* Timeout send */
int MQTIMEDSEND(int mqid, char *message, int length, unsigned int mqprio, int secs, long nanos) {

    mqd_t mqres;
    struct timespec mqtimer;
    struct timeval curtime;

    /* Expect seconds and nanos to wait, not absolute. Add the OpenCOBOL values */
    gettimeofday(&curtime, NULL);
    mqtimer.tv_sec = curtime.tv_sec + (time_t)secs;
    mqtimer.tv_nsec = nanos;

    errno = 0;
    mqres = mq_timedsend((mqd_t)mqid, message, (size_t)length, mqprio, &mqtimer);
    return (int)mqres;
}

/* Read the highest priority message */
int MQTIMEDRECEIVE(int mqid, char *msgbuf, int buflen, int *retprio, int secs, long nanos) {
    ssize_t retlen;
    struct timespec mqtimer;

    struct timeval curtime;

    /* Expect seconds and nanos to wait, not absolute. Add the OpenCOBOL values */
    gettimeofday(&curtime, NULL);
    mqtimer.tv_sec = curtime.tv_sec + (time_t)secs;
    mqtimer.tv_nsec = nanos;

    errno = 0;
    retlen = mq_timedreceive((mqd_t)mqid, msgbuf, buflen, retprio, &mqtimer);
    return (int)retlen;
}

/* Notify of new message written to queue */
int MQNOTIFY(int mqid, char *procedure) {
    struct sigevent ocsigevent;
    struct sigaction ocsigaction;

    /* Install signal handler for the notify signal - fill in a
     * sigaction structure and pass it to sigaction(). Because the
     * handler needs the siginfo structure as an argument, the
     * SA_SIGINFO flag is set in sa_flags.
     */
    ocsigaction.sa_sigaction = ocmq_handler;
    ocsigaction.sa_flags = SA_SIGINFO;
    sigemptyset(&ocsigaction.sa_mask);

    if (sigaction(SIGUSR1, &ocsigaction, NULL) == -1) {
        fprintf(stderr, "%s\n", "Error posting sigaction");
        return -1;
    }
}

```

```
/* Set up notification: fill in a sigevent structure and pass it
 * to mq_notify(). The queue ID is passed as an argument to the
 * signal handler.
 */
ocsigevent.sigev_signo      = SIGUSR1;
ocsigevent.sigev_notify     = SIGEV_SIGNAL;
ocsigevent.sigev_value.sival_int = (int)mqid;

if (mq_notify((mqd_t)mqid, &ocsigevent) == -1) {
    fprintf(stderr, "%s\n", "Error posting notify");
    return -1;
}
return 0;
}

/* Close a queue */
int MQCLOSE(int mqid) {
    mqd_t mqres;

    errno = 0;
    mqres = mq_close((mqd_t)mqid);
    return (int)mqres;
}

/* Unlink a queue */
int MQUNLINK(char *mqname) {
    mqd_t mqres;

    errno = 0;
    mqres = mq_unlink(mqname);
    return (int)mqres;
}

/* The signal handling section */
/* signal number */
/* signal information */
/* context unused (required by posix) */
static void ocmq_handler(int sig, siginfo_t *pInfo, void *pSigContext) {
    struct sigevent ocnotify;
    mqd_t mqid;

    /* Get the ID of the message queue out of the siginfo structure. */
    mqid = (mqd_t) pInfo->si_value.sival_int;

    /* The MQPROCESSOR is a hardcoded OpenCOBOL resolvable module name */
    /* It must accept an mqd_t pointer */
    cob_init(0, NULL);
    MQHANDLER = cob_resolve("MQPROCESSOR");
    if (MQHANDLER == NULL) {
        /* What to do here? */
        fprintf(stderr, "%s\n", "Error resolving MQPROCESSOR");
        return;
    }

    /* Request notification again; it resets each time a notification
     * signal goes out.
     */
    ocnotify.sigev_signo = pInfo->si_signo;
}
```



```

ocnotify.sigev_value = pInfo->si_value;
ocnotify.sigev_notify = SIGEV_SIGNAL;

if (mq_notify(mqid, &ocnotify) == -1) {
    /* What to do here? */
    fprintf(stderr, "%s\n", "Error posting notify");
    return;
}

/* Call the cobol module with the message queue id */
MQHANDLER(&mqid);
return;
}
/**/

```

With a sample of usage. Note the linkage of the rt.so realtime library.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*****
* Author:      Brian Tiffin
* Date:        August 2008
* Purpose:     Demonstration of OpenCOBOL message queues
* Tectonics:   gcc -c ocmq.c
*              cobc -Wall -x -lrt mqsample.cob ocmq.o
*****
identification division.
program-id. mqsample.

data division.
working-storage section.
* Constants for the Open Flags
01 MQO-RDONLY          constant as 0.
01 MQO-WRONLY         constant as 1.
01 MQO-RDWR           constant as 2.
01 MQO-CREAT          constant as 64.
01 MQO-EXCL           constant as 128.
01 MQO-NONBLOCK       constant as 2048.
* Constants for the protection/permission bits
01 MQS-IREAD          constant as 256.
01 MQS-IWRITE         constant as 128.

* Need a better way of displaying newlines
01 newline            pic x value x'0a'.

* Message Queues return an ID, maps to int
01 mqid               usage binary-long.
01 mqres              usage binary-long.
* Queue names end up in an mqueue virtual filesystem on GNU/Linux
01 mqname.
   02 name-display    pic x(5) value "/ocmq".
   02 filler          pic x value x'00'.
01 mqopenflags        usage binary-long.
01 mqpermissions      usage binary-long.

01 default-message    pic x(20) value 'OpenCOBOL is awesome'.
01 user-message       pic x(80).
01 send-length        usage binary-long.

01 urgent-message     pic x(20) value 'Urgent OpenCOBOL msg'.

```

```
* Data members for access to C global errno and error strings
01 errnumber          usage binary-long.
01 errstr             pic x(256).
* legend to use with the error reporting
01 operation          pic x(7).

01 loopy              pic 9.

* Debian GNU/Linux defaults to Message Queue entry limit of 8K
01 msgbuf             pic x(8192).
01 msglen             usage binary-long value 8192.
* Priorities range from 0 to 31 on many systems, can be more
01 msgprio            usage binary-long.
* MQ attributes. See /usr/include/bits/mqueue.h
01 mqattr.
  03 mqflags          usage binary-long.
  03 mqmaxmsg         usage binary-long.
  03 mqmsgsize        usage binary-long.
  03 mqcurmsqs        usage binary-long.
  03 filler           usage binary-long occurs 4 times.
01 oldattr.
  03 mqflags          usage binary-long.
  03 mqmaxmsg         usage binary-long.
  03 mqmsgsize        usage binary-long.
  03 mqcurmsqs        usage binary-long.
  03 filler           usage binary-long occurs 4 times.
```

#### procedure division.

```
* The ocmq API support MQCREATE and MQOPEN.
* This example uses non blocking, non exclusive create
* read/write by owner and default attributes
compute
  mqopenflags = MQO-RDWR + MQO-CREAT + MQO-NONBLOCK
end-compute.
compute
  mqpermissions = MQS-IREAD + MQS-IWRITE
end-compute.

* Sample shows the two types of open, but only evaluates create
if zero = zero
call "MQCREATE" using mqname
                    by value mqopenflags
                    by value mqpermissions
                    by value 0
                    returning mqid
end-call
else
call "MQOPEN" using mqname
                    by value mqopenflags
                    returning mqid
end-call
end-if.
move "create" to operation.
perform show-error.

* Show the attributes after initial create
perform show-attributes.
```

```

* Register notification
call "MQNOTIFY" using by value mqid
                    mqname
                    returning mqres
end-call.
move "notify" to operation.
perform show-error.

* Create a temporary queue, will be removed on close
* call "MQUNLINK" using mqname
*                               returning mqres
* end-call.
* move "unlink" to operation.
* perform show-error.

* Use the command line arguments or a default message
accept user-message from command-line end-accept.
if user-message equal spaces
    move default-message to user-message
end-if.
move function length
    (function trim(user-message trailing))
    to send-length.

* Queue up an urgent message (priority 31)
call "MQSEND" using by value mqid
                    by reference urgent-message
                    by value 20
                    by value 31

end-call.
move "send-31" to operation.
perform show-error.

* Queue up a low priority message (1)
call "MQSEND" using by value mqid
                    by reference user-message
                    by value send-length
                    by value 1
                    returning mqres

end-call.
move "send-1" to operation.
perform show-error.

* Queue up a middle priority message (16)
inspect urgent-message
    replacing leading "Urgent" by "Middle".
call "MQSEND" using by value mqid
                    by reference urgent-message
                    by value 20
                    by value 16
                    returning mqres

end-call.
move "send-16" to operation.
perform show-error.

* Redisplay the queue attributes
perform show-attributes.

```

```
* Pull highest priority message off queue
call "MQRECEIVE" using by value mqid
                    by reference msgbuf
                    by value msglen
                    by reference msgprio
                    returning mqres

end-call.
display
    newline "receive len: " mqres " prio: " msgprio
end-display.
if mqres > 0
    display
        "priority 31 message: " msgbuf(1:mqres)
    end-display
end-if.
move "receive" to operation.
perform show-error.

* Pull the middling priority message off queue
call "MQRECEIVE" using by value mqid
                    by reference msgbuf
                    by value msglen
                    by reference msgprio
                    returning mqres

end-call.
display
    newline "receive len: " mqres " prio: " msgprio
end-display.
if mqres > 0
    display
        "priority 16 message: " msgbuf(1:mqres)
    end-display
end-if.
move "receive" to operation.
perform show-error.

* ** INTENTIONAL ERROR msglen param too small **
* Pull message off queue
call "MQRECEIVE" using by value mqid
                    by reference msgbuf
                    by value 1024
                    by reference msgprio
                    returning mqres

end-call.
display
    newline "receive len: " mqres " prio: " msgprio
end-display.
if mqres > 0
    display
        "no message: " msgbuf(1:mqres)
    end-display
end-if.
move "receive" to operation.
perform show-error.

* Pull the low priority message off queue, in blocking mode
move MQO-NONBLOCK to mqflags of mqattr.
call "MQSETATTR" using by value mqid
```

```

        by reference mqattr
        by reference oldattr
        returning mqres
end-call
move "setattr" to operation.
perform show-error.
perform show-attributes.

call "MQRECEIVE" using by value mqid
                    by reference msgbuf
                    by value msglen
                    by reference msgprio
                    returning mqres

end-call.
display
    newline "receive len: " mqres " prio: " msgprio
end-display.
if mqres > 0
    display
        "priority 1 message: " msgbuf(1:mqres)
    end-display
end-if.
move "receive" to operation.
perform show-error.

perform varying loopy from 1 by 1
    until loopy > 5
        display "Sleeper call " loopy end-display
        call "CBL_OC_NANOSLEEP" using 50000000000
        returning mqres

    end-call
end-perform.

* Close the queue. As it is set unlinked, it will be removed
call "MQCLOSE" using by value mqid
        returning mqres

end-call.
move "close" to operation.
perform show-error.

* Create a temporary queue, will be removed on close
call "MQUNLINK" using mqname
        returning mqres

end-call.
move "unlink" to operation.
perform show-error.

goback.

*****
* Information display of the Message Queue attributes.
show-attributes.
call "MQGETATTR" using by value mqid
                    by reference mqattr
                    returning mqres

end-call
move "getattr" to operation.
perform show-error.

```

```
* Display the message queue attributes
display
  name-display " attributes:"          newline
  "flags:      " mqflags    of mqattr  newline
  "max msg:    " mqmaxmsg   of mqattr  newline
  "mqs size:   " mqmsgsize  of mqattr  newline
  "cur msgs:   " mqcurmsqs  of mqattr
end-display
.

* The C global errno error display paragraph
show-error.
  call "ERRORNUMBER" returning mqres end-call
  if mqres > 0
    display
      operation " errno: " mqres
    end-display
    call "ERRORSTRING" using errstr
      by value length errstr
      returning mqres end-call

    if mqres > 0
      display
        "      strerror: " errstr(1:mqres)
      end-display
    end-if
  end-if
.
end program mqsample.

*****
* Author:      Brian Tiffin
* Date:        August 2008
* Purpose:     Demonstration of OpenCOBOL message queue notification
* Tectonics:   gcc -c ocmq.c
*              cobc -Wall -x -lrt mqsample.cob ocmq.o
*****
identification division.
program-id. MQSIGNAL.

data division.
working-storage section.
01 msgbuf pic x(8192).
01 msglen usage binary-long value 8192.
01 msgprio usage binary-long.
01 mqres  usage binary-long.

linkage section.
01 mqid usage binary-long.

procedure division using mqid.

display "in MQSIGNAL".
display "In the COBOL procedure with " mqid end-display.
perform
  with test after
  until mqres <= 0

  call "MQRECEIVE" using by value mqid
```

```

        by reference msgbuf
        by value msglen
        by reference msgprio
    returning mqres

end-call
display
    "receive len: " mqres " prio: " msgprio
end-display
if mqres > 0
    display
        "priority 31 message: " msgbuf(1:mqres)
    end-display
end-if
end-perform.

goback.
end program MQSIGNAL.

```

## 29.11 5.11 Can OpenCOBOL interface with Lua?

Yes. Lua can be embedded in OpenCOBOL applications.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*<*<=====  

*<*< OpenCOBOL Lua Interface  

*<*<=====  

*<*<  

*<*< .. sidebar:: Contents  

*<*<  

*<*< .. contents::  

*<*<     :local:  

*<*<     :depth: 2  

*<*<     :backlinks: entry  

*<*<  

*<*< :Author:    Brian Tiffin  

*<*< :Date:      28-Oct-2008  

*<*< :Purpose:   interface to Lua scripting  

*<*< :Rights:    | Copyright 2008 Brian Tiffin  

*<*<             | Licensed under the GNU General Public License  

*<*<             | No warranty expressed or implied  

*<*< :Tectonics: | cobc -c -I/usr/include/lua5.1/ oclua.c  

*<*<             | cobc -x -llua5.1 luacaller.cob oclua.o  

*<*<             | ./ocdoc luacaller.cob oclua.rst oclua.html ocfaq.css  

*<*< :Requires:  lua5.1, liblua5.1, liblua5.1-dev  

*<*< :Link:      http://www.lua.org  

*<*< :Thanks to: The Lua team, Pontifical Catholic University  

*<*<             of Rio de Janeiro in Brazil.  

*<*<             http://www.lua.org/authors.html  

*<*< :Sources:   | http://opencobol.addltocobol.com/luacaller.cob  

*<*<             | http://opencobol.addltocobol.com/oclua.c  

*<*<             | http://opencobol.addltocobol.com/oclua.lua  

*<*<             | http://opencobol.addltocobol.com/oclua.rst  

*<*<             | http://opencobol.addltocobol.com/ocfaq.rss  

*<*<  

*<*> *****  

    identification division.

```

```
program-id. luacaller.

data division.
working-storage section.
01 luastate          usage pointer.
01 luascript        pic x(10) value 'oclua.lua' & x"00".
01 luacommmand      pic x(64).
01 luaresult        pic x(32).
01 lualength        usage binary-long.

01 items            pic 9 usage computational-5.
01 luastack.
   03 luaitem       pic x(32) occurs 5 times.
01 depth            usage binary-long.

*> *****
procedure division.

call "OCLUA_OPEN" returning luastate end-call

move 'return "OpenCOBOL " .. 1.0 + 0.1' & x"00" to luacommmand
call "OCLUA_DOSTRING"
  using
    by value luastate
    by reference luacommmand
    by reference luaresult
    by value function length(luaresult)
  returning depth
end-call
display
  "OpenCOBOL displays: " depth " |" luaresult "|"
end-display

call "OCLUA_DOFIELD"
  using
    by value luastate
    by reference luascript
    by reference luaresult
    by value 32
  returning depth
end-call
display
  "OpenCOBOL displays: " depth " |" luaresult "|"
end-display

call "OCLUA_DOFIELD"
  using
    by value luastate
    by reference luascript
    by reference luaresult
    by value 32
  returning depth
end-call
display
  "OpenCOBOL displays: " depth " |" luaresult "|"
end-display

call "OCLUA_DEPTH"
```



```

        using
            by value luastate
        returning depth
    end-call
    display "Lua depth: " depth end-display

perform varying items from 1 by 1
    until items > depth
        call "OCLUA_GET"
            using
                by value luastate
                by value items
                by reference luaresult
                by value 32
            returning lualength
        end-call
        move luaresult to luaitem(items)
    end-perform

perform varying items from 1 by 1
    until items > depth
        display
            "Item " items ": " luaitem(items)
        end-display
    end-perform

call "OCLUA_POP"
    using
        by value luastate
        by value depth
    returning depth
end-call

call "OCLUA_DEPTH"
    using
        by value luastate
    returning depth
end-call

display "Lua depth: " depth end-display

call "OCLUA_CLOSE" using by value luastate end-call

goback.
end program luacaller.
*> *****
*><* ++++++
*><* Overview
*><* ++++++
*><* The OpenCOBOL Lua interface is defined at a very high level.
*><*
*><* The objective is to provide easy access to Lua through
*><* script files or strings to be evaluated.
*><*
*><* Command strings and script file names passed to Lua MUST be
*><* terminated with a null byte, as per C Language conventions.
*><*
*><* A Lua engine is started with a call to OCLUA_OPEN, which

```

```
*><* returns an OpenCOBOL POINTER that is used to reference
*><* the Lua state for all further calls.
*><*
*><* A Lua engine is run down with a call to OCLUA_CLOSE.
*><*
*><* .. Attention::
*><*   Calls to Lua without a valid state will cause
*><*   undefined behaviour and crash the application.
*><*
*><* Lua uses a stack and results of the Lua RETURN reserved
*><* word are placed on this stack. Multiple values can be
*><* returned from Lua.
*><*
*><* The developer is responsible for stack overflow conditions
*><* and the size of the stack (default 20 elements) is
*><* controlled with OCLUA_STACK using an integer that
*><* determines the numbers of slots to reserve.
*><*
*><* Requires package installs of:
*><*
*><* * lua5.1
*><* * liblua5.1
*><* * liblua5.1-dev
*><*
*><* ++++++
*><* OpenCOBOL Lua API
*><* ++++++
*><* -----
*><* OCLUA_OPEN
*><* -----
*><* Initialize the Lua engine.
*><*
*><* ::
*><*
*><*   01 luastate USAGE POINTER.
*><*
*><*   CALL "OCLUA_OPEN" RETURNING luastate END-CALL
*><*
*><* -----
*><* OCLUA_STACK
*><* -----
*><* Check and possibly resize the Lua data stack. Returns 0 if
*><* Lua cannot expand the stack to the requested size.
*><*
*><* ::
*><*
*><*   01 elements USAGE BINARY-LONG VALUE 32.
*><*   01 result   USAGE BINARY-LONG.
*><*
*><*   CALL "OCLUA_STACK"
*><*     USING
*><*       BY VALUE luastate
*><*       BY VALUE elements
*><*     RETURNING result
*><*   END-CALL
*><*
*><* -----
*><* OCLUA_DOSTRING
```

```

*><* -----
*><* Evaluate a null terminated alphanumeric field as a Lua program
*><* producing any top of stack entry and returning the depth of
*><* stack after evaluation.
*><*
*><* Takes a luastate, a null terminated command string,
*><* a result field and length and returns an integer depth.
*><*
*><* .. Attention::
*><*   The Lua stack is NOT popped while returning the top of stack entry.
*><*
*><* ::
*><*
*><*   01 luaccommand pic x(64).
*><*   01 luaresult  pic x(32).
*><*   01 depth      usage binary-long.
*><*
*><*   move 'return "OpenCOBOL " .. 1.0 + 0.1' & x"00" to luaccommand
*><*   call "OCLUA_DOSTRING"
*><*       using
*><*           by value luastate
*><*           by reference luaccommand
*><*           by reference luaresult
*><*           by value function length(luaresult)
*><*       returning depth
*><*   end-call
*><*   display
*><*       "OpenCOBOL displays: " depth " |" luaresult "|"
*><*   end-display
*><*
*><* Outputs::
*><*
*><*   OpenCOBOL displays: +0000000001 |OpenCOBOL 1.1           ||
*><*
*><* -----
*><* OCLUA_DOFIELD
*><* -----
*><* Evaluate a script using a null terminated alphanumeric field
*><* naming a Lua program source file, retrieving any top of
*><* stack entry and returning the depth of stack after evaluation.
*><*
*><* Takes a luastate, a null terminated filename,
*><* a result field and length and returns an integer depth.
*><*
*><* .. Attention::
*><*   The Lua stack is NOT popped while returning the top of
*><* stack entry.
*><*
*><* ::
*><*
*><*   01 luascript  pic x(10) value 'oclua.lua' & x"00".
*><*   01 luaresult  pic x(32).
*><*
*><*   call "OCLUA_DOFIELD"
*><*       using
*><*           by value luastate
*><*           by reference luascript
*><*           by reference luaresult

```

```

*<*<          by value function length(luaresult)
*<*<          returning depth
*<*< end-call
*<*< display
*<*<      "OpenCOBOL displays: " depth " |" luaresult "|"
*<*< end-display
*<*<
*<*< Given oclua.lua::
*<*<
*<*<      -- Start
*<*<      -- Script: oclua.lua
*<*<      print("Lua prints hello")
*<*<
*<*<      hello = "Hello OpenCOBOL from Lua"
*<*<      return math.pi, hello
*<*<      -- End
*<*<
*<*< Outputs::
*<*<
*<*<      Lua prints hello
*<*<      OpenCOBOL displays: +0000000002 |Hello OpenCOBOL from Lua      ||
*<*<
*<*< and on return from Lua, there is *math.pi* and the
*<*< Hello string remaining on the Lua state stack.
*<*<
*<*< -----
*<*< OCLUA_DEPTH
*<*< -----
*<*< Returns the current number of elements on the Lua stack.
*<*<
*<*< ::
*<*<
*<*<      call "OCLUA_DEPTH"
*<*<          using
*<*<              by value luastate
*<*<              returning depth
*<*<      end-call
*<*<      display "Lua depth: " depth end-display
*<*<
*<*< -----
*<*< OCLUA_GET
*<*< -----
*<*< Retrieves values from the Lua stack, returning the length
*<*< of the retrieved item.
*<*<
*<*< An example that populates and displays an OpenCOBOL table::
*<*<
*<*<      01 items                pic 9 usage computational-5.
*<*<      01 luastack.
*<*<      03 luaitem              pic x(32) occurs 5 times.
*<*<
*<*<      perform varying items from 1 by 1
*<*<          until items > depth
*<*<              call "OCLUA_GET"
*<*<                  using
*<*<                      by value luastate
*<*<                      by value items
*<*<                      by reference luaresult

```

```

*><*          by value function length(luaresult)
*><*          returning lualength
*><*          end-call
*><*          move luaresult to luaitem(items)
*><* end-perform
*><*
*><* perform varying items from 1 by 1
*><*          until items > depth
*><*          display
*><*          "Item " items ": " luaitem(items)
*><*          end-display
*><* end-perform
*><*
*><* Lua numbers the indexes of stacked items from 1, first
*><* item to n, last item (current top of stack). Negative
*><* indexes may also be used as documented by Lua, -1 being
*><* top of stack.
*><*
*><* Sample output::
*><*
*><* Item 1: OpenCOBOL 1.1
*><* Item 2: 3.1415926535898
*><* Item 3: Hello OpenCOBOL from Lua
*><* Item 4: 3.1415926535898
*><* Item 5: Hello OpenCOBOL from Lua
*><*
*><* -----
*><* OCLUA_POP
*><* -----
*><* Pops the given number of elements off of the Lua stack
*><* returning the depth of the stack after the pop.
*><*
*><* Example that empties the Lua stack::
*><*
*><* call "OCLUA_POP"
*><* using
*><*     by value luastate
*><*     by value depth
*><*     returning depth
*><* end-call
*><*
*><* -----
*><* OCLUA_CLOSE
*><* -----
*><* Close and free the Lua engine.
*><*
*><* .. Danger::
*><* Further calls to Lua are unpredictable and may well
*><* lead to a SIGSEGV crash.
*><*
*><* ::
*><*
*><* call "OCLUA_CLOSE" using by value luastate end-call
*><*

```

With usage document at [oclua.html](http://oclua.html)

The above code uses a wrapper layer of C code

```
/* OpenCOBOL Lua interface */
/* tectonics: cobc -c -I/usr/include/lua5.1 oclua.c */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

/* Include the Lua API header files. */
#include <lua.h>
#include <lualib.h>
#include <lua5.1/lua.h>
#include <lua5.1/lualib.h>

/* Open the Lua engine and load all the default libraries */
lua_State *OCLUA_OPEN() {
    lua_State *oclua_state;
    oclua_state = lua_open();
    luaL_openlibs(oclua_state);
    return oclua_state;
}

int OCLUA_DO(lua_State *L, int which, const char *string, unsigned char *cobol, int coblen) {
    int result;
    int stacked;
    const char *retstr;
    int retlen;

    memset(cobol, ' ', coblen);
    result = ((which == 0) ? luaL_dostring(L, string) : luaL_dofile(L, string));
    if (result == 1) {
        /* error condition */
        return -1;
    } else {
        stacked = lua_gettop(L);
        if (stacked > 0) {
            /* populate cobol field with top of stack */
            retstr = lua_tolstring(L, stacked, &retlen);
            memcpy(cobol, retstr, (coblen > retlen) ? retlen : coblen);
        }
        /* return number of items on the stack */
        return stacked;
    }
}

/* by filename */
int OCLUA_DOFIELD(lua_State *L, const char *filename, unsigned char *cobol, int coblen) {
    return OCLUA_DO(L, 1, filename, cobol, coblen);
}

/* by string */
int OCLUA_DOSTRING(lua_State *L, const char *string, unsigned char *cobol, int coblen) {
    return OCLUA_DO(L, 0, string, cobol, coblen);
}

/* retrieve stack item as string */
int OCLUA_GET(lua_State *L, int element, unsigned char *cobol, int coblen) {
    const char *retstr;
    int retlen;
}
```

```

    /* populate cobol field with top of stack */
    memset(cobol, ' ', coblen);
    retstr = lua_tolstring(L, element, &retlen);
    if (retstr == NULL) {
        return -1;
    } else {
        memcpy(cobol, retstr, (coblen > retlen) ? retlen : coblen);
        return retlen;
    }
}

/* check the stack, resize if needed, returns false if stack can't grow */
int OCLUA_STACK(lua_State *L, int extra) {
    return lua_checkstack(L, extra);
}

/* depth of Lua stack */
int OCLUA_DEPTH(lua_State *L) {
    return lua_gettop(L);
}

/* pop elements off stack */
int OCLUA_POP(lua_State *L, int elements) {
    lua_pop(L, elements);
    return lua_gettop(L);
}

/* close the engine */
void OCLUA_CLOSE(lua_State *L) {
    lua_close(L);
}

/**/

```

and this sample Lua script `oclua.lua`

```

-- Start
-- Script: oclua.lua
print("Lua prints hello")

hello = "Hello OpenCOBOL from Lua"
return math.pi, hello
-- End

```

## 29.12 5.12 Can OpenCOBOL use ECMAScript?

Yes. Using the SpiderMonkey engine. See [Can OpenCOBOL use JavaScript?](#)

## 29.13 5.13 Can OpenCOBOL use JavaScript?

Yes. A wrapper for the SpiderMonkey engine allows OpenCOBOL access to core JavaScript.

```

/* OpenCOBOL with embedded spidermonkey javascript */
/* cobc -c -I/usr/include/smjs ocjs.c

```

```
*   cobc -x -lsmjs jscaller.cob
*   some people found mozjs before smjs
*/
#include <stdio.h>
#include <string.h>

/* javascript api requires an environment type */
#define XP_UNIX

#if (defined(XP_WIN) || defined(XP_UNIX) || defined(XP_BEOS) || defined(XP_OS2))
#include "jsapi.h"
#else
#error "Must define one of XP_BEOS, XP_OS2, XP_WIN or XP_UNIX"
#endif

/* Error codes */
#define OCJS_ERROR_RUNTIME    -1
#define OCJS_ERROR_CONTEXT   -2
#define OCJS_ERROR_GLOBAL    -3
#define OCJS_ERROR_STANDARD -4
#define OCJS_ERROR_EVALUATE -5

/* OpenCOBOL main CALL interface */
/* javascript layer requires
 * a runtime per process,
 * a context per thread,
 * a global object per context
 * and will initialize
 * standard classes.
 */
static JSRuntime *rt;
static JSContext *cx;
static JSObject *global;
static JSClass global_class = {
    "global",0,
    JS_PropertyStub,JS_PropertyStub,JS_PropertyStub,JS_PropertyStub,
    JS_EnumerateStub,JS_ResolveStub,JS_ConvertStub,JS_FinalizeStub
};

/* Initialize the engine resources */
int ocjsInitialize(int rtsize, int cxsize) {
    JSBool ok;

    /* on zero sizes, pick reasonable values */
    if (rtsize == 0) { rtsize = 0x100000; }
    if (cxsize == 0) { cxsize = 0x1000; }

    /* Initialize a runtime space */
    rt = JS_NewRuntime(rtsize);
    if (rt == NULL) { return OCJS_ERROR_RUNTIME; }
    /* Attach a context */
    cx = JS_NewContext(rt, cxsize);
    if (cx == NULL) { return OCJS_ERROR_CONTEXT; }
    /* And a default global */
    global = JS_NewObject(cx, &global_class, NULL, NULL);
    if (global == NULL) { return OCJS_ERROR_GLOBAL; }
    /* Load standard classes */
    ok = JS_InitStandardClasses(cx, global);
}
```



```

    /* Return success or standard class load error */
    return (ok == JS_TRUE) ? 0 : OCJS_ERROR_STANDARD;
}

/* Evaluate script */
int ocjsEvaluate(char *script, char *result, int length) {
    jsval rval;
    JSString *str;
    int reslen = OCJS_ERROR_EVALUATE;

    JSBool ok;

    /* filename and line number, not reported */
    char *filename = NULL;
    int lineno = 0;

    /* clear the result field */
    memset(result, ' ', length);

    /* Evaluate javascript */
    ok = JS_EvaluateScript(cx, global, script, strlen(script), filename, lineno, &rval);

    /* Convert js result to JSString form */
    if (ok == JS_TRUE) {
        str = JS_ValueToString(cx, rval);
        reslen = strlen(JS_GetStringBytes(str));
        if (length < reslen) { reslen = length; }
        /* convert down to char and move to OpenCOBOL result field */
        memcpy(result, JS_GetStringBytes(str), reslen);
    }
    return reslen;
}

/* Evaluate script from file */
int ocjsFromFile(char *filename, char *result, int length) {
    FILE *fin;
    int bufsize = 10240;
    char inbuf[bufsize];
    int reslen;

    fin = fopen(filename, "r");
    if (fin == NULL) { return OCJS_ERROR_EVALUATE; }
    //while (fread(inbuf, sizeof(char), bufsize, fin) > 0) {
    if (fread(inbuf, 1, bufsize, fin) > 0) {
        reslen = ocjsEvaluate(inbuf, result, length);
    }
    return reslen;
}

/* release js engine */
int ocjsRunDown() {
    if (cx != NULL) { JS_DestroyContext(cx); }
    if (rt != NULL) { JS_DestroyRuntime(rt); }
    JS_ShutDown();
    return 0;
}

/* Quick call; start engine, evaluate, release engine */

```

```
int ocjsString(char *script, char *result, int length) {
    int reslen;

    reslen = ocjsInitialize(0, 0);
    if (reslen < 0) { return reslen; }
    reslen = ocjsEvaluate(script, result, length);
    ocjsRunDown();
    return reslen;
}
/**/
```

A sample OpenCOBOL application:

```
OCOBOL >>SOURCE FORMAT IS FIXED
*>*****
*>Author:    Brian Tiffin
*>Date:      11-Sep-2008
*>Purpose:   Embed some javascript
*>Tectonics: cobc -c -I/usr/include/smjs ocjs.c
*>          cobc -x -l/smjs jscaller.cob ocjs.o
*>*****
  identification division.
  program-id. jscaller.

  data division.

  working-storage section.
  78 ocjs-error-runtime value -1.
  78 ocjs-error-context value -2.
  78 ocjs-error-global value -3.
  78 ocjs-error-standard value -4.
  78 ocjs-error-evaluate value -5.

  78 newline           value x"0a".
  01 source-data       pic x(40)
    value "----+----1----+-$56.78 90----3----+----4".
  01 result            pic s9(9).
  01 result-field      pic x(81).

  01 javascript        pic x(1024).
  01 safety-null       pic x value x"00".

*>*****
*><* Evaluate spidermonkey code, return the length of js result
  procedure division.

  display "js> " with no advancing end-display
  accept javascript end-accept
  call "ocjsString"
    using javascript
    result-field
    by value function length(result-field)
    returning result
  end-call
  display "OpenCOBOL result-field: " result-field end-display
  display "OpenCOBOL received      : " result newline end-display

*><* Initialize the javascript engine
  call "ocjsInitialize"
```

```

        using by value 65536
            by value 1024
        returning result
    end-call
    if result less 0
        stop run returning result
    end-if

*><* find (zero offset) dollar amount, space, number
move spaces to javascript
string
    "pat = /\$\\d+\\.\\d+\\s\\d+/"
    'a = ' delimited by size
    source-data delimited by size
    ';' delimited by size
    "a.search(pat); " delimited by size
    x"00" delimited by size
    into javascript
end-string

display
    "Script: " function trim(javascript, trailing)
end-display

call "ocjsEvaluate"
    using javascript
        result-field
        by value function length(result-field)
    returning result
end-call
display "OpenCOBOL result-field: " result-field end-display
display "OpenCOBOL received      : " result newline end-display

*><* values held in js engine across calls
move spaces to javascript
string
    'a;' delimited by size
    x"00" delimited by size
    into javascript
end-string

display
    "Script: " function trim(javascript, trailing)
end-display

call "ocjsEvaluate"
    using javascript
        result-field
        by value function length(result-field)
    returning result
end-call
display "OpenCOBOL result-field: " result-field end-display
display "OpenCOBOL received      : " result newline end-display

*><* erroneous script
move spaces to javascript
string
    'an error of some kind;' delimited by size

```

```
        x"00" delimited by size
        into javascript
    end-string

    display
        "Script: " function trim(javascript, trailing)
    end-display

    call "ocjsEvaluate"
        using javascript
        result-field
        by value function length(result-field)
        returning result
    end-call
    if result equal ocjs-error-evaluate
        display " *** script problem ***" end-display
    end-if
    display "OpenCOBOL result-field: " result-field end-display
    display "OpenCOBOL received      : " result newline end-display

*><* script from file
    move spaces to javascript
    string
        'ocjsscript.js' delimited by size
        x"00" delimited by size
        into javascript
    end-string

    display
        "Script: " function trim(javascript, trailing)
    end-display

    call "ocjsFromFile"
        using javascript
        result-field
        by value function length(result-field)
        returning result
    end-call
    if result equal ocjs-error-evaluate
        display " *** script problem ***" end-display
    end-if
    display "OpenCOBOL result-field: " result-field end-display
    display "OpenCOBOL received      : " result newline end-display

*><* Rundown the js engine
    call "ocjsRunDown" returning result

*><* take first name last name, return last ", " first
    move spaces to javascript
    string
        "re = /(\\w+)\\s(\\w+)/; " delimited by size
        'str = "John Smith"; ' delimited by size
        'newstr = str.replace(re, "$2, $1"); ' delimited by size
        "newstr;" delimited by size
        x"00" delimited by size
        into javascript
    end-string
```

```

display
  "Script: " function trim(javascript, trailing)
end-display

call "ocjsString"
  using javascript
  result-field
  by value function length(result-field)
  returning result
end-call
display "OpenCOBOL result-field: " result-field end-display
display "OpenCOBOL received      : " result newline end-display

*><* split a string using numbers return array (as js string form)
move spaces to javascript
string
  'myString = "Hello 1 word. Sentence number 2."; '
  delimited by size
  'splits = myString.split(/(\d)/); ' delimited by size
  'splits;' delimited by size
  x"00" delimited by size
  into javascript
end-string

display
  "Script: " function trim(javascript, trailing)
end-display

call "ocjsString"
  using javascript
  result-field
  by value function length(result-field)
  returning result
end-call
display "OpenCOBOL result-field: " result-field end-display
display "OpenCOBOL received      : " result newline end-display

*><* Get javascript date
move "new Date()" & x"00" to javascript

display
  "Script: " function trim(javascript, trailing)
end-display

call "ocjsString"
  using javascript
  result-field
  by value function length(result-field)
  returning result
end-call
display "OpenCOBOL result-field: " result-field end-display
display "OpenCOBOL received      : " result end-display

goback.
end program jscaller.

```

And with a sample script:

ocjsscript.js

```
var x = 2
var y = 39
var z = "42"
// boths line evaluate to 42
eval("x + y + 1")
eval(z)
```

Sample output:

```
js> 123 * 456 + 789
OpenCOBOL result-field: 56877
OpenCOBOL received      : +000000005
```

```
Script: pat = /\$\d+\.\d+\s\d+;/; a = "----+----1----+-$56.78 90----3----+----4"; a.search(pat);
OpenCOBOL result-field: 16
OpenCOBOL received      : +000000002
```

```
Script: a;
OpenCOBOL result-field: ----+----1----+-$56.78 90----3----+----4
OpenCOBOL received      : +000000040
```

```
Script: an error of some kind;
*** script problem ***
OpenCOBOL result-field:
OpenCOBOL received      : -000000005
```

```
Script: re = /(\w+)\s(\w+)/; str = "John Smith"; newstr = str.replace(re, "$2, $1"); newstr;
OpenCOBOL result-field: Smith, John
OpenCOBOL received      : +000000011
```

```
Script: myString = "Hello 1 word. Sentence number 2."; splits = myString.split(/(\d)/); splits;
OpenCOBOL result-field: Hello ,1, word. Sentence number ,2,.
OpenCOBOL received      : +000000036
```

```
Script: new Date()
OpenCOBOL result-field: Mon Sep 15 2008 04:16:06 GMT-0400 (EDT)
OpenCOBOL received      : +000000039
```

## 29.14 5.14 Can OpenCOBOL interface with Scheme?

Yes, directly embedded with [Guile](#) and [libguile](#).

callguile.cob

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20090215
*> Purpose:     Demonstrate libguile Scheme interactions
*> Tectonics:   cobic -x -lguile callguile.cob
*> *****
identification division.
program-id. callguile.

data division.
working-storage section.
01 tax-scm          usage pointer.
```

```

01 shipping-scm          usage pointer.
01 scm-string            usage pointer.
01 radix-scm            usage pointer.

01 subtotal              pic 999v99 value 80.00.
01 subtotal-display      pic z(8)9.99.
01 weight                pic 99v99 value 10.00.
01 weight-display        pic Z9.99.
01 breadth               pic 99v99 value 20.00.
01 breadth-display       pic Z9.99.

01 answer                pic x(80).
01 len                   usage binary-long.

01 tax                   pic 9(9)v9(2).
01 tax-display           pic z(8)9.9(2).
01 shipping              pic 9(9)v9(2).
01 shipping-display      pic z(8)9.9(2).
01 invoice-total         pic 9(9)v9(2).
01 invoice-display       pic $(8)9.9(2).

*> *****
procedure division.

display "OC: initialize libguile" end-display
call "scm_init_guile" end-call

display "OC: load scheme code" end-display
call "scm_c_primitive_load" using "script.scm" & x"00" end-call
display "OC:" end-display

display "OC: evaluate one of the defined functions" end-display
call "scm_c_eval_string" using "(do-hello)" & x"00" end-call
display "OC:" end-display

display "OC: perform tax calculation" end-display
move subtotal to subtotal-display
move weight to weight-display
move breadth to breadth-display
call "scm_c_eval_string"
  using
    function concatenate(
      "(compute-tax "; subtotal-display; ")"; x"00"
    )
  returning tax-scm
end-call

display "OC: perform shipping calculation" end-display
display "OC: " function concatenate(
  "(compute-shipping "; weight-display; " ";
  breadth-display; ")"; x"00"
)
end-display
call "scm_c_eval_string"
  using
    function concatenate(
      "(compute-shipping "; weight-display; " ";
      breadth-display; ")"; x"00"
    )

```

```
    )
    returning shipping-scm
end-call

display "OC: have guile build a scheme integer 10" end-display
call "scm_from_int32"
    using by value size is 4 10 returning radix-scm
end-call

display "OC: have guile convert number, base 10" end-display
call "scm_number_to_string"
    using
        by value tax-scm by value radix-scm
    returning scm-string
end-call

display "OC: get numeric string to COBOL" end-display
call "scm_to_locale_stringbuf"
    using
        by value scm-string
        by reference answer
        by value 80
    returning len
end-call
display "OC: tax as string: " answer end-display
move answer to tax

call "scm_number_to_string"
    using
        by value shipping-scm by value radix-scm
    returning scm-string
end-call

call "scm_to_locale_stringbuf"
    using
        by value scm-string
        by reference answer
        by value 80
    returning len
end-call
display "OC: shipping as string: " answer end-display
move answer to shipping

compute invoice-total = subtotal + tax + shipping end-compute

move subtotal to subtotal-display
move tax to tax-display
move shipping to shipping-display
move invoice-total to invoice-display
display "OC:" end-display
display "OC: subtotal    " subtotal-display end-display
display "OC: tax        " tax-display end-display
display "OC: shipping   " shipping-display end-display
display "OC: total:     " invoice-display end-display
goback.
end program callguile.
```

script.scm



```

(define (do-hello)
  (begin
    (display "Welcome to Guile")
    (newline)))

(define (compute-tax subtotal)
  (* subtotal 0.0875))

(define (compute-shipping weight length)

  ;; For small, light packages, charge the minimum
  (if (and (< weight 20) (< length 5))
      0.95

      ;; Otherwise for long packages, charge a lot
      (if (> length 100)
          (+ 0.95 (* weight 0.1))

          ;; Otherwise, charge the usual
          (+ 0.95 (* weight 0.05)))))

(display "Loaded script.scm") (newline)

```

**Outputs:**

```

OC: initialize libguile
OC: load scheme code
Loaded script.scm
OC:
OC: evaluate one of the defined functions
Welcome to Guile
OC:
OC: perform tax calculation
OC: perform shipping calculation
OC: (compute-shipping 10.00 20.00)
OC: have guile build a scheme integer 10
OC: have guile convert number, base 10
OC: get numeric string to COBOL
OC: tax as string: 7.0
OC: shipping as string: 1.45
OC:
OC: subtotal          80.00
OC: tax              7.00
OC: shipping         1.45
OC: total:           $88.45

```

Of course using Scheme for financial calculations in an OpenCOBOL application would not be a smart usage. This is just a working sample.

## 29.15 5.15 Can OpenCOBOL interface with Tcl/Tk?

Yes. OpenCOBOL supports the Tcl/Tk embedding engine developed by Rildo Pragana as part of the TinyCOBOL project. We have been given permission by Rildo to embed his engine in OpenCOBOL.

See <http://ww1.pragana.net/cobol.html> for sources.

A working sample

```

OCOBOL IDENTIFICATION DIVISION.
PROGRAM-ID.    tclgui.
AUTHOR. Rildo Pragana.
*> REMARKS.
*>    Example tcl/tk GUI program for Cobol.
*>
ENVIRONMENT DIVISION.
DATA DIVISION.
*>
WORKING-STORAGE SECTION.
01 DATA-BLOCK.
   05 NAME          PIC X(40) .
   05 W-ADDRESS     PIC X(50) .
   05 PHONE         PIC X(15) .
   05 END-PGM      PIC X .
           05 QUICK-RET      PIC X .
01 SITE-INFO.
   05 TITLE        PIC X(20) .
   05 URL          PIC X(50) .
77 GUI-01         PIC X(64) VALUE "formA.tcl".
77 GUI-02         PIC X(64) VALUE "formB.tcl".
77 END-OF-STRING  pic X value LOW-VALUES.
77 T-SCRIPT       PIC X(128) .
77 T-RESULT       PIC X(80) .
01 dummy         pic X value X"00".

PROCEDURE DIVISION.

CALL "initTcl"

*> test for stcleval function
string "expr 12 * 34" END-OF-STRING into T-SCRIPT
call "stcleval" using T-SCRIPT T-RESULT
display "eval by tcl: |" T-SCRIPT "|" returned " T-RESULT

MOVE "Your name here" to NAME
MOVE "Your address" TO W-ADDRESS
MOVE "Phone number" to PHONE
*> this variable tells Cobol that the user required an exit
MOVE "0" to END-PGM
MOVE "1" to QUICK-RET
MOVE "Afonso Pena" to NAME
*> now we may have the script name as a variable, terminated by a space
CALL "tcleval" USING DATA-BLOCK "./formA.tcl "
MOVE "Deodoro da Fonseca" to NAME
CALL "tcleval" USING DATA-BLOCK GUI-01
MOVE "Rui Barbosa" to NAME
CALL "tcleval" USING DATA-BLOCK GUI-01
MOVE "Frei Caneca" to NAME
CALL "tcleval" USING DATA-BLOCK GUI-01

MOVE "0" to QUICK-RET
MOVE "Your name here" to NAME.
100-restart.
*> call C wrapper, passing data block and size of data
CALL "tcleval" USING DATA-BLOCK GUI-01

```

```

DISPLAY "Returned data:"
DISPLAY "NAME      [" NAME  "]"
DISPLAY "ADDRESS  [" W-ADDRESS  "]"
DISPLAY "PHONE    [" PHONE  "]"
*> if not end of program required, loop
if END-PGM = 0
    go to 100-restart.
*> to start a new GUI (graphical interface), call this first
call "newGui"
MOVE "Title of the site" to TITLE
MOVE "URL (http://..., ftp://..., etc)" to URL
*> now we may draw other main window...
CALL "tclevel" USING SITE-INFO GUI-02
DISPLAY "Returned data:"
DISPLAY "TITLE    [" TITLE  "]"
DISPLAY "URL      [" URL   "]"

STOP RUN.

```

Which uses two Tcl/Tk scripts

```

#!/bin/sh
# the next line restarts using wish\
exec wish "$0" "$@"

if {[info exists vTcl(sourcing)]} {

    package require Tk
    switch $tcl_platform(platform) {
        windows {
            option add *Button.pady 0
        }
        default {
            option add *Scrollbar.width 10
            option add *Scrollbar.highlightThickness 0
            option add *Scrollbar.elementBorderWidth 2
            option add *Scrollbar.borderWidth 2
        }
    }
}

#####
# Visual Tcl v1.60 Project
#

#####
# VTCL LIBRARY PROCEDURES
#

if {[info exists vTcl(sourcing)]} {
#####
## Library Procedure: Window

proc ::Window {args} {
    ## This procedure may be used free of restrictions.
    ## Exception added by Christian Gavin on 08/08/02.
    ## Other packages and widget toolkits have different licensing requirements.

```

```

## Please read their license agreements for details.

global vTcl
foreach {cmd name newname} [lrange $args 0 2] {}
set rest [lrange $args 3 end]
if {$name == "" || $cmd == ""} { return }
if {$newname == ""} { set newname $name }
if {$name == "."} { wm withdraw $name; return }
set exists [winfo exists $newname]
switch $cmd {
    show {
        if {$exists} {
            wm deiconify $newname
        } elseif {[info procs vTclWindow$name] != ""} {
            eval "vTclWindow$name $newname $rest"
        }
        if {[winfo exists $newname] && [wm state $newname] == "normal"} {
            vTcl:FireEvent $newname <<Show>>
        }
    }
    hide {
        if {$exists} {
            wm withdraw $newname
            vTcl:FireEvent $newname <<Hide>>
            return
        }
        iconify { if $exists {wm iconify $newname; return} }
        destroy { if $exists {destroy $newname; return} }
    }
}
}
#####
## Library Procedure: vTcl:DefineAlias

proc ::vTcl:DefineAlias {target alias widgetProc top_or_alias cmdalias} {
    ## This procedure may be used free of restrictions.
    ## Exception added by Christian Gavin on 08/08/02.
    ## Other packages and widget toolkits have different licensing requirements.
    ## Please read their license agreements for details.

    global widget
    set widget($alias) $target
    set widget(rev,$target) $alias
    if {$cmdalias} {
        interp alias {} $alias {} $widgetProc $target
    }
    if {$top_or_alias != ""} {
        set widget($top_or_alias,$alias) $target
        if {$cmdalias} {
            interp alias {} $top_or_alias.$alias {} $widgetProc $target
        }
    }
}
#####
## Library Procedure: vTcl:DoCmdOption

proc ::vTcl:DoCmdOption {target cmd} {
    ## This procedure may be used free of restrictions.
    ## Exception added by Christian Gavin on 08/08/02.

```

```

## Other packages and widget toolkits have different licensing requirements.
##   Please read their license agreements for details.

## menus are considered toplevel windows
set parent $target
while {[wininfo class $parent] == "Menu"} {
    set parent [wininfo parent $parent]
}

regsub -all {\%widget} $cmd $target cmd
regsub -all {\%top} $cmd [wininfo toplevel $parent] cmd

uplevel #0 [list eval $cmd]
}
#####
## Library Procedure:  vTcl:FireEvent

proc ::vTcl:FireEvent {target event {params {}}} {
    ## This procedure may be used free of restrictions.
    ##   Exception added by Christian Gavin on 08/08/02.
    ## Other packages and widget toolkits have different licensing requirements.
    ##   Please read their license agreements for details.

    ## The window may have disappeared
    if {![wininfo exists $target]} return
    ## Process each binding tag, looking for the event
    foreach bindtag [bindtags $target] {
        set tag_events [bind $bindtag]
        set stop_processing 0
        foreach tag_event $tag_events {
            if {$tag_event == $event} {
                set bind_code [bind $bindtag $tag_event]
                foreach rep "\{%W $target\} $params" {
                    regsub -all [lindex $rep 0] $bind_code [lindex $rep 1] bind_code
                }
                set result [catch {uplevel #0 $bind_code} errortext]
                if {$result == 3} {
                    ## break exception, stop processing
                    set stop_processing 1
                } elseif {$result != 0} {
                    bgerror $errortext
                }
                break
            }
        }
        if {$stop_processing} {break}
    }
}
#####
## Library Procedure:  vTcl:Toplevel:WidgetProc

proc ::vTcl:Toplevel:WidgetProc {w args} {
    ## This procedure may be used free of restrictions.
    ##   Exception added by Christian Gavin on 08/08/02.
    ## Other packages and widget toolkits have different licensing requirements.
    ##   Please read their license agreements for details.

    if {[llength $args] == 0} {

```

```

    ## If no arguments, returns the path the alias points to
    return $w
}
set command [lindex $args 0]
set args [lrange $args 1 end]
switch -- [string tolower $command] {
    "setvar" {
        foreach {varname value} $args {}
        if {$value == ""} {
            return [set ::${w}::${varname}]
        } else {
            return [set ::${w}::${varname} $value]
        }
    }
    "hide" - "show" {
        Window [string tolower $command] $w
    }
    "showmodal" {
        ## modal dialog ends when window is destroyed
        Window show $w; raise $w
        grab $w; tkwait window $w; grab release $w
    }
    "startmodal" {
        ## ends when endmodal called
        Window show $w; raise $w
        set ::${w}::_modal 1
        grab $w; tkwait variable ::${w}::_modal; grab release $w
    }
    "endmodal" {
        ## ends modal dialog started with startmodal, argument is var name
        set ::${w}::_modal 0
        Window hide $w
    }
    default {
        uplevel $w $command $args
    }
}
}
#####
## Library Procedure:  vTcl:WidgetProc

proc ::vTcl:WidgetProc {w args} {
    ## This procedure may be used free of restrictions.
    ## Exception added by Christian Gavin on 08/08/02.
    ## Other packages and widget toolkits have different licensing requirements.
    ## Please read their license agreements for details.

    if {[llength $args] == 0} {
        ## If no arguments, returns the path the alias points to
        return $w
    }

    set command [lindex $args 0]
    set args [lrange $args 1 end]
    uplevel $w $command $args
}
#####
## Library Procedure:  vTcl:toplevel

```

```

proc ::vTcl:toplevel {args} {
    ## This procedure may be used free of restrictions.
    ## Exception added by Christian Gavin on 08/08/02.
    ## Other packages and widget toolkits have different licensing requirements.
    ## Please read their license agreements for details.

    uplevel #0 eval toplevel $args
    set target [lindex $args 0]
    namespace eval ::$target {set _modal 0}
}
}

```

```
if {[info exists vTcl(sourcing)]} {
```

```

proc vTcl:project:info {} {
    set base .top43
    namespace eval ::widgets::$base {
        set set,origin 1
        set set,size 1
        set runvisible 1
    }
    namespace eval ::widgets::$base.lab44 {
        array set save {-disabledforeground 1 -font 1 -text 1}
    }
    namespace eval ::widgets::$base.cpd45 {
        array set save {-disabledforeground 1 -font 1 -text 1}
    }
    namespace eval ::widgets::$base.cpd46 {
        array set save {-disabledforeground 1 -font 1 -text 1}
    }
    namespace eval ::widgets::$base.che47 {
        array set save {-disabledforeground 1 -font 1 -text 1 -variable 1}
    }
    namespace eval ::widgets::$base.but48 {
        array set save {-command 1 -disabledforeground 1 -font 1 -text 1}
    }
    namespace eval ::widgets::$base.ent49 {
        array set save {-background 1 -insertbackground 1 -textvariable 1}
    }
    namespace eval ::widgets::$base.cpd50 {
        array set save {-background 1 -insertbackground 1 -textvariable 1}
    }
    namespace eval ::widgets::$base.cpd51 {
        array set save {-background 1 -insertbackground 1 -textvariable 1}
    }
    namespace eval ::widgets::$base.lis43 {
        array set save {-background 1 -listvariable 1}
    }
    namespace eval ::widgets::$base.lab45 {
        array set save {-disabledforeground 1 -font 1 -text 1}
    }
    namespace eval ::widgets::$base.but47 {
        array set save {-command 1 -disabledforeground 1 -text 1}
    }
    namespace eval ::widgets::$base.but51 {
        array set save {-command 1 -disabledforeground 1 -text 1}
    }
}

```

```

set base .top47
namespace eval ::widgets::$base {
    set set,origin 1
    set set,size 1
    set runvisible 1
}
namespace eval ::widgets::$base.ent48 {
    array set save {-background 1 -disabledforeground 1 -insertbackground 1 -textvariable 1}
}
namespace eval ::widgets::$base.but49 {
    array set save {-command 1 -disabledforeground 1 -text 1}
}
namespace eval ::widgets::$base.but50 {
    array set save {-command 1 -disabledforeground 1 -text 1}
}
namespace eval ::widgets_bindings {
    set taglist _TopLevel
}
namespace eval ::vTcl::modules::main {
    set procs {
        init
        main
        cobol_update
    }
    set compounds {
    }
    set projectType single
}
}
}

#####
# USER DEFINED PROCEDURES
#
#####
## Procedure:  main

proc ::main {argc argv} {
global cobol_fields widget

set cobol_fields {
    name      40
    address   50
    phone     15
    endpgm    1
    quickret  1
}

global nomes_anteriores
if {[info exists nomes_anteriores]} {
    set nomes_anteriores {}
}

#bind all <Return> do_exit
}

proc ::cobol_preprocess {args} {
    global quickret

```



```

        if {$quickret} {
            do_exit
        }
    }
#####
## Procedure:  cobol_update

proc ::cobol_update {} {
    global widget
    global nomes_antiores name

    #puts "tcl-TC LOG: lappend nomes_antiores $name"
    lappend nomes_antiores $name
    focus $widget(nome_entry)
}

#####
## Initialization Procedure:  init

proc ::init {argc argv} {

}

init $argc $argv

#####
# VTCL GENERATED GUI PROCEDURES
#

proc vTclWindow. {base} {
    if {$base == ""} {
        set base .
    }
    #####
    # CREATING WIDGETS
    #####
    wm focusmodel $top passive
    wm geometry $top lxl+0+0; update
    wm maxsize $top 1265 994
    wm minsize $top 1 1
    wm overrideredirect $top 0
    wm resizable $top 1 1
    wm withdraw $top
    wm title $top "vtcl.tcl"
    bindtags $top "$top Vtcl.tcl all"
    vTcl:FireEvent $top <<Create>>
    wm protocol $top WM_DELETE_WINDOW "vTcl:FireEvent $top <<DeleteWindow>>"

    #####
    # SETTING GEOMETRY
    #####

    vTcl:FireEvent $base <<Ready>>
}

proc vTclWindow.top43 {base} {
    if {$base == ""} {

```

```

        set base .top43
    }
    if {[wininfo exists $base]} {
        wm deiconify $base; return
    }
    set top $base
    #####
    # CREATING WIDGETS
    #####
    vTcl:toplevel $top -class Toplevel \
        -highlightcolor black
    wm focusmodel $top passive
    wm geometry $top 570x523+318+169; update
    wm maxsize $top 1265 994
    wm minsize $top 1 1
    wm overrideredirect $top 0
    wm resizable $top 1 1
    wm deiconify $top
    wm title $top "New Toplevel 1"
    vTcl:DefineAlias "$top" "Toplevel1" vTcl:Toplevel:WidgetProc "" 1
    bindtags $top "$top Toplevel all _TopLevel"
    vTcl:FireEvent $top <<Create>>
    wm protocol $top WM_DELETE_WINDOW "vTcl:FireEvent $top <<DeleteWindow>>"

    label $top.lab44 \
        -disabledforeground #a1a4a1 -font {helvetica 18 bold} -text Nome:
    vTcl:DefineAlias "$top.lab44" "Label1" vTcl:WidgetProc "Toplevel1" 1
    label $top.cpd45 \
        -disabledforeground #a1a4a1 -font {helvetica 18 bold} -text Endereço:
    vTcl:DefineAlias "$top.cpd45" "Label2" vTcl:WidgetProc "Toplevel1" 1
    label $top.cpd46 \
        -disabledforeground #a1a4a1 -font {helvetica 18 bold} -text Telefone:
    vTcl:DefineAlias "$top.cpd46" "Label3" vTcl:WidgetProc "Toplevel1" 1
    checkbox $top.che47 \
        -disabledforeground #a1a4a1 -font {helvetica 10} -text concluido \
        -variable endpgm
    vTcl:DefineAlias "$top.che47" "Checkbox1" vTcl:WidgetProc "Toplevel1" 1
    button $top.but48 \
        -command do_exit -disabledforeground #a1a4a1 \
        -font {helvetica 10 bold} -text entra
    vTcl:DefineAlias "$top.but48" "Button1" vTcl:WidgetProc "Toplevel1" 1
    entry $top.ent49 \
        -background white -insertbackground black -textvariable name
    vTcl:DefineAlias "$top.ent49" "nome_entry" vTcl:WidgetProc "Toplevel1" 1
    entry $top.cpd50 \
        -background white -insertbackground black -textvariable address
    vTcl:DefineAlias "$top.cpd50" "Entry2" vTcl:WidgetProc "Toplevel1" 1
    entry $top.cpd51 \
        -background white -insertbackground black -textvariable phone
    vTcl:DefineAlias "$top.cpd51" "Entry3" vTcl:WidgetProc "Toplevel1" 1
    listbox $top.lis43 \
        -background white -listvariable nomes_anteriores
    vTcl:DefineAlias "$top.lis43" "Listbox1" vTcl:WidgetProc "Toplevel1" 1
    label $top.lab45 \
        -disabledforeground #a1a4a1 -font {verdana -11} \
        -text {nomes
anteriores}
    vTcl:DefineAlias "$top.lab45" "Label4" vTcl:WidgetProc "Toplevel1" 1

```

```

button $top.but47 \
    -command {source /usr/bin/tkcon} -disabledforeground #a1a4a1 \
    -text tkcon
vTcl:DefineAlias "$top.but47" "Button2" vTcl:WidgetProc "Toplevel1" 1
button $top.but51 \
    -command {MinhaJanela show} -disabledforeground #a1a4a1 \
    -text {nome (aux)}
vTcl:DefineAlias "$top.but51" "Button3" vTcl:WidgetProc "Toplevel1" 1
#####
# SETTING GEOMETRY
#####
place $top.lab44 \
    -x 25 -y 35 -anchor nw -bordermode ignore
place $top.cpd45 \
    -x 25 -y 100 -anchor nw
place $top.cpd46 \
    -x 25 -y 170 -anchor nw
place $top.che47 \
    -x 30 -y 440 -anchor nw -bordermode ignore
place $top.but48 \
    -x 205 -y 430 -anchor nw -bordermode ignore
place $top.ent49 \
    -x 140 -y 40 -width 403 -height 27 -anchor nw -bordermode ignore
place $top.cpd50 \
    -x 175 -y 100 -width 368 -height 27 -anchor nw
place $top.cpd51 \
    -x 175 -y 175 -width 273 -height 27 -anchor nw
place $top.lis43 \
    -x 155 -y 245 -width 383 -height 156 -anchor nw -bordermode ignore
place $top.lab45 \
    -x 35 -y 250 -anchor nw -bordermode ignore
place $top.but47 \
    -x 470 -y 430 -anchor nw -bordermode ignore
place $top.but51 \
    -x 320 -y 430 -anchor nw -bordermode ignore

vTcl:FireEvent $base <<Ready>>
}

proc vTclWindow.top47 {base} {
    if {$base == ""} {
        set base .top47
    }
    if {[winfo exists $base]} {
        wm deiconify $base; return
    }
    set top $base
    #####
    # CREATING WIDGETS
    #####
    vTcl:toplevel $top -class Toplevel \
        -highlightcolor black
    wm withdraw $top
    wm focusmodel $top passive
    wm geometry $top 433x150+169+728; update
    wm maxsize $top 1265 994
    wm minsize $top 1 1
    wm overrideredirect $top 0

```

```
wm resizable $top 1 1
wm title $top "New Toplevel 2"
vTcl:DefineAlias "$top" "MinhaJanela" vTcl:Toplevel:WidgetProc " " 1
bindtags $top "$top Toplevel all _TopLevel"
vTcl:FireEvent $top <<Create>>
wm protocol $top WM_DELETE_WINDOW "vTcl:FireEvent $top <<DeleteWindow>>"

entry $top.ent48 \
    -background white -disabledforeground #a1a4a1 -insertbackground black \
    -textvariable name1
vTcl:DefineAlias "$top.ent48" "Entry1" vTcl:WidgetProc "MinhaJanela" 1
button $top.but49 \
    -command {global name name1
set name $name1
MinhaJanela hide} \
    -disabledforeground #a1a4a1 -text ok
vTcl:DefineAlias "$top.but49" "Button1" vTcl:WidgetProc "MinhaJanela" 1
button $top.but50 \
    -command {MinhaJanela hide} -disabledforeground #a1a4a1 -text fechar
vTcl:DefineAlias "$top.but50" "Button2" vTcl:WidgetProc "MinhaJanela" 1
#####
# SETTING GEOMETRY
#####
place $top.ent48 \
    -x 50 -y 30 -width 353 -height 27 -anchor nw -bordermode ignore
place $top.but49 \
    -x 145 -y 90 -anchor nw -bordermode ignore
place $top.but50 \
    -x 240 -y 90 -anchor nw -bordermode ignore

vTcl:FireEvent $base <<Ready>>
}

#####
## Binding tag: _TopLevel

bind "_TopLevel" <<Create>> {
    if ![info exists _topcount] {set _topcount 0}; incr _topcount
}
bind "_TopLevel" <<DeleteWindow>> {
    if {[set ::%W::_modal]} {
        vTcl:Toplevel:WidgetProc %W endmodal
    } else {
        destroy %W; if {$_topcount == 0} {exit}
    }
}
bind "_TopLevel" <Destroy> {
    if {[winfo toplevel %W] == "%W"} {incr _topcount -1}
}

Window show .
Window show .top43
Window show .top47

main $argc $argv
## ** ##

and
```

```

#!/bin/sh
# the next line restarts using wish\
exec wish "$0" "$@"
# this script receives "data_block" with the (group) value
# of the cobol variable and returns "result"

## visual tcl leaves the main window iconified, so let's show it
wm deiconify .

##### put in this list varname, size pairs

set cobol_fields {
    title          20
    url            50
}

grid [label .msg -text \
      "Use <Tab> to navigate, <Return> (or click button) \n\
to return to main program."] -columnspan 2

grid \
  [label .lab1 -text "Title:"] \
  [entry .e1 -width 20 -textvariable title] -padx 5 -pady 5 -sticky nsw
grid \
  [label .lab2 -text "URL:"] \
  [entry .e2 -width 50 -textvariable url] -padx 5 -pady 5 -sticky nsw

grid [button .ready -text Enter -command do_exit] \
      -columnspan 2 -pady 20 -sticky ns

bind all <Return> do_exit
focus .e1

#trace add variable ::ready write show_variables

proc show_variables {args} {
    uplevel #0 {
        set exclude {^::(env|auto_index|tcl_*|widget|tk_*|auto_*)$}
        puts "variables: -----"
        foreach v [info vars ::*] {
            if {[regexp $exclude $v]} {
                continue
            }
            if {[array exists $v]} {
                puts "$v: [array get $v]"
            } else {
                puts "$v: [set $v]"
            }
        }
    }
}

```

## 29.16 5.16 Can OpenCOBOL interface with Falcon PL?

Not yet, but work with Giancarlo to allow embedding of Falcon scripts is in progress. **Update on December 31st, 2010**

Yes, yes it can.

This is from the linked post ... but the Falcon programming language embeds in OpenCOBOL just fine.

### **falconscript.fal**

```
> "Falcon list comprehension called from OpenCOBOL"
sums = [].mfcomp( {x,y=> x+y}, .[1 2 3], .[4 5 6] )
return sums.describe()
```

it goes

```
$ ./callfalcon
argv[1]: falconscript.fal
Falcon list comprehension called from OpenCOBOL
VM Output: [ 5, 6, 7, 6, 7, 8, 7, 8, 9]
Intermediate: [ 5, 6, 7, 6, 7, 8, 7, 8, 9]
Falcon says: [ 5, 6, 7, 6, 7, 8, 7, 8, 9]
```

A Falcon list comprehension with **mfcomp** applies the reduction  $x+y$  on 1 and 4, 1 and 5, 1 and 6, then 2 and 4, 2 and 5 etc.

See [http://www.opencobol.org/modules/newbb/viewtopic.php?topic\\_id=1221&forum=1&post\\_id=6113#forumpost6113](http://www.opencobol.org/modules/newbb/viewtopic.php?topic_id=1221&forum=1&post_id=6113#forumpost6113) for details.

FalconPL has some nice features.

```
saying = List("Have", "a", "nice", "day")
```

```
for elem in saying
  >> elem
  formiddle: >> " "
  forlast: > "!"
end
```

giving:

```
Have a nice day!
```

Source files can be found in <http://fossile.plpwebs.com/ocsamples.cgi/dir?ci=tip>

## 29.17 5.17 Can OpenCOBOL interface with Ada?

Yes. The freely available **gnat** system can be used and will create object files that can be included in an OpenCOBOL project.

This example compiles an gnat package that includes *hello* and *ingress* PROCEDURE and a *echo* FUNCTION. These will be called from an OpenCOBOL **adacaller.cob** program.

The gnat specification file

```
with Interfaces.C;
use Interfaces.C;
package HelloAda is
```

```

procedure hello;
procedure ingress(value : in INTEGER);
function echo(message : in char_array) return integer;
pragma export(C, hello);
pragma export(C, ingress);
pragma export(C, echo);

```

end HelloAda;

The gnat implementation body

```

with Ada.Text_IO, Ada.Integer_Text_IO, Interfaces.C;
use Ada.Text_IO, Ada.Integer_Text_IO, Interfaces.C;
package body HelloAda is

```

```

procedure hello is
begin

```

```

    Put_Line("Hello from Ada and OpenCOBOL");
    New_Line;

```

```

end hello;

```

```

procedure ingress(value : in integer) is
begin

```

```

    Put_Line("Passing integer to Ada from OpenCOBOL");
    Put("OpenCOBOL passed: ");
    Put(value);
    New_Line;
    New_Line;

```

```

end ingress;

```

```

function echo(message : in char_array) return integer is
begin

```

```

    Put (To_Ada (message, true));
    return To_Ada (message, true)'length;

```

```

end echo;

```

end HelloAda;

The adacaller.cob source file

```

OCOBOL***** adacaller.cob *****
>>SOURCE FORMAT IS FIXED
*****
* Author:      Brian Tiffin
* Date:        08-Sep-2008
* Purpose:     Demonstrate using Ada sub-programs
* Tectonics:   gnatgcc -c helloada.adb
*              gnatbind -n helloada
*              gnatgcc -c b~helloada.abd
*              cobc -x -lgnat caller.cob helloada.o b~helloada.o
*****
identification division.
program-id. caller.

data division.
working-storage section.
01 ada-message      pic x(10) value "Ada echo" & x'0a' & x'00'.
01 result           pic s9(9) value high-value.

```

```
*****  
procedure division.  
begin.  
call "adainit" end-call  
  
call "hello" end-call  
  
call "ingress" using by value 42 end-call  
  
call "echo" using  
    by reference ada-message  
    returning result  
end-call  
display "Ada return: " result end-display  
  
call "adafinal" end-call  
  
goback  
.  
end program caller.
```

And the tectonics; Debian GNU/Linux *build.sh*

```
gnatgcc -c helloada.adb  
gnatbind -n helloada  
gnatgcc -c b~helloada.adb  
cobc -x -lgnat adacaller.cob helloada.o b~helloada.o
```

An important step is the creation of the object file from the *gnatbind* output *with -n* that is used in the final OpenCOBOL executable.

Sample run using *./adacaller*:

```
Hello from Ada and OpenCOBOL  
  
Passing integer to Ada from OpenCOBOL  
OpenCOBOL passed:          42  
  
Ada echo  
Ada return: +000000009
```

See [Can the GNAT Programming Studio be used with OpenCOBOL?](#) for more.

## 29.18 5.18 Can OpenCOBOL interface with Vala?

Yes. Very easily. The Vala design philosophy of producing C application binary interface code means that Vala is directly usable with OpenCOBOL's *CALL* statement.

See <http://live.gnome.org/Vala> for some details on this emerging programming environment.

This interface will be seeing more and more use as it really does open the door to some very powerful extensions.

- WebKit embedding
- PDF Viewers
- GTK
- Media streaming



- much more

### 29.18.1 5.18.1 Call OpenCOBOL programs from Vala

Using a few simple tricks, Vala can easily call OpenCOBOL programs. Vala uses a predictable link module naming convention. Inside a class, `from.vala`, the linker will try and find `from_vala_name`, in this case `from_vala_ochello`.

```
/* Call OpenCOBOL from Vala */

public class from.vala
{

    public static int main(string[] args)
    {
        stdout.printf("Result: %d\n", ochello());
        return 0;
    }
    [import()]
    public extern static int ochello();
}
/**/
```

So the PROGRAM-ID here is `from_vala_ochello`.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin
*> Date:     20101017
*> Purpose:   Call ochello from Vala in a from.vala Class
*> Tectonics:
*>  cobc -fimplicit-init -C ochello.cob
*>  valac callcobol.vala ochello.c -X -lcob
*> *****
identification division.
program-id. from_vala_ochello.

*> *****
procedure division.
display "Hello OpenCOBOL's World!" end-display
move 42 to return-code
goback.
end program from_vala_ochello.
```

The tectonics might seem a little bit mysterious. `cobc` is used to produce C source code, including calls for initialization of the OpenCOBOL runtime.

`valac` is then used to compile and link the Vala source, the generated `ochello.c` and then the `gcc` compiler is passed the `-lcob` to link in `libcob.so`.

### 29.18.2 5.18.2 Call OpenCOBOL from a Vala GTK gui application

And another experiment, with a gui button and repeated timer calls.

`callhellogui.vala`

```
// Call OpenCOBOL program from Vala and show the return code on a button
using Gtk;

public class from.vala {
    public static int cobolcode;
    public static char[] valarray = new char[80];

    public static int main (string[] args) {

        Gtk.init (ref args);
        var time = new TimeoutSource(50);

        var window = new Window (WindowType.TOPLEVEL);
        window.title = "Invoke OpenCOBOL program";
        window.set_default_size (300, 50);
        window.position = WindowPosition.CENTER;
        window.destroy.connect (Gtk.main_quit);

        cobolcode = ochello();

        var button = new Button.with_label (cobolcode.to_string());
        button.clicked.connect (() => {
            button.label = "Thanks for all the fish!";
            stdout.printf("%d\n", fishy());
        });

        time.set_callback(() => {
            var t = Time.local(time_t());
            string fromvala = "From vala string type + time to_string: " + t.to_string();
            string fromcobol = "xxxx/xx/xxbxx/xx/xxxxxxx/xx";

            stdout.printf("Vala fromcobol string was : %s\n", fromcobol);

            datey(fromvala, fromcobol);

            stdout.printf("Vala fromcobol string set to: %s\n", fromcobol);
            return true;
        });

        time.attach(null);

        window.add (button);
        window.show_all ();

        Gtk.main ();
        return 0;
    }

    [import()]
    public extern static int ochello();
    public extern static int fishy();
    public extern static int datey(string arg1, string arg2);
}
```

*ochellogui.cob*

And here we define *from\_vala\_ochello*, *from\_vala\_fishy*, *from\_vala\_datey*.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20101017
*> Purpose:     Call ochello from Vala in a from.vala Class
*> Tectonics:
*>  cobc -fimplicit-init -C ochellogui.cob
*>  valac --pkg gtk+-2.0 callcobolgui.vala ochellogui.c -X -lcob
*> *****
  identification division.
  program-id. from_vala_ochello.
  procedure division.
  display "Hello OpenCOBOL's Wonderful World!" end-display
  move 42 to return-code
  goback.
  end program from_vala_ochello.

*> *****
*> *****

  program-id. from_vala_fishy.
  procedure division.
  display "We really do mean, thanks for all the fish!" end-display
  goback.
  end program from_vala_fishy.

*> *****
*> *****

  program-id. from_vala_datey.
  data division.
  working-storage section.
  01 edited-date pic xxxx/xx/xxbxx/xx/xxxxxxxx/xx.

  linkage section.
  01 datafromvala pic x(60).
  01 datafromcobol pic x(27).

  procedure division using datafromvala datafromcobol.
  move function current-date to edited-date
  inspect edited-date replacing all "/" by ":" after initial space
  display edited-date end-display

  display datafromvala end-display

  move edited-date to datafromcobol
  goback.
  end program from_vala_datey.

```

Tectonics similar to the first sample. With this one, a timer fires every 50 milliseconds passing data back and forth between Vala and OpenCOBOL **unsafely, mind you**. If you push button “42”, a message is printed to standard out.



Along with the GUI button, produces:

```
$ ./callcobolgui
...
Vala fromcobol string was : xxxx/xx/xxbxx/xx/xxxxxxxx/xx
2010/10/17 18:19:5598-04:00
From vala string type + time to_string: 2010-10-17 18:19:55
Vala fromcobol string set to: 2010/10/17 18:19:5598-04:00
Vala fromcobol string was : xxxx/xx/xxbxx/xx/xxxxxxxx/xx
2010/10/17 18:19:5603-04:00
From vala string type + time to_string: 2010-10-17 18:19:56
Vala fromcobol string set to: 2010/10/17 18:19:5603-04:00
...
```

### 29.18.3 5.18.3 Call Genie program from OpenCOBOL

Here is a sample that calls a small Genie program.

**pipng.gs**, a small program that spawns out some shell commands. One fails on purpose, *ech* is not a valid executable. The next *echo* call has the output captured in *ret\_stdout*. 42 is then passed as the return code to OpenCOBOL.

```
// Tectonics: valac -c pipng.gs
[indent=4]
class wrapper : Object
  def static hellogenie() : int
    ret_stdout : string
    ret_stderr : string
    ret_status : int

    try
      Process.spawn_command_line_sync("ech 'ech?'",out ret_stdout,out ret_stderr,out ret_status)
    except ex : Error
      print("in catch")
      print(ex.message)

    print("stdout: %s", ret_stdout)
    print("stderr: %s", ret_stderr)
    print("status: %d", ret_status)

    try
      Process.spawn_command_line_sync("echo -n 'hey it works!'",out ret_stdout,out ret_stderr,out ret_status)
    except ex : Error
      print("in catch")
      print(ex.message)

    print("stdout: %s", ret_stdout)
    print("stderr: %s", ret_stderr)
    print("status: %d", ret_status)

  return 42
```

**callgenie.cob**

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*><* =====
*><* callgenie
*><* =====
*><* :Author:    Brian Tiffin
*><* :Date:      29-Sep-2010
*><* :Purpose:   Demonstrate getting at Genie code
*><* :Tectonics:
*><*   valac -c piping.gs
*><*   cobc -x callgenie.cob piping.vala.o
*><*   -lglib-2.0 -lobject-2.0
*> *****
identification division.
program-id. callgenie.

data division.
working-storage section.
01 result usage binary-long.

*> *****
procedure division.
call "wrapper_hellogenie" returning result end-call
display "Result from Genie: " result end-display
.
goback.
end program callgenie.
*><*
*><* Last Update: 29-Sep-2010

```

The Vala/Genie link naming is predictable. Inside a *class*, **wrapper**, the Genie generated link name is *wrapper\_hellogenie*.

With a sample run producing:

```

[btiffin@home vala]$ ./callgenie
in catch
Failed to execute child process "ech" (No such file or directory)
stdout: (null)
stderr: (null)
status: 0
stdout: hey it works!
stderr:
status: 0
Result from Genie: +0000000042

```

**29.18.4 5.18.4 Pass data to and from Genie**

The Genie

```

// Tectonics: valac -c genierex.gs
[indent=4]
class cbl.oc.genie : Object
  def static regexing(pattern : string, subject : string, out value : string, out leng : int) : int
    print " "
    print "Pattern: %s", pattern

```

```

print "Subject: %s", subject
try
  var r = new Regex(pattern)
  var s = subject
  s = r.replace(s, s.length, 0, "COBOL")
  value = s
  leng = (int)s.length
except ex : Error
  print ex.message
  value = subject
  leng = (int)subject.length
  return 1
return 0

```

## The COBOL

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*><* =====
*><* Call Genie Regex
*><* =====
*><* :Author:      Brian Tiffin
*><* :Date:        20101101
*><* :Purpose:     Getting at Genie Regex code
*><* :Tectonics:   vala -c genieregex.gs
*><*               cobc -x callgenieregex.cob genieregex.vala.o
*><*               -lglib-2.0 -lgobject-2.0
*> *****
identification division.
program-id. callgenieregex.

data division.
working-storage section.
01 pattern pic x(80) value "Fortran|APL|Python" & x"00".
01 subject pic x(80) value
   "OpenCOBOL, Fortran, Vala, Genie, Python, C, APL" & x"00".
01 out-pointer usage pointer.
01 out-length usage binary-long.
01 middleman pic x(80) based.
01 replacement pic x(80).
01 result usage binary-long.

*> *****
procedure division.
call "cbl_oc_genie_regexing"
  using
    by reference pattern
    by reference subject
    by reference out-pointer
    by reference out-length
  returning result
end-call
display "Result from Genie: " result end-display

set address of middleman to out-pointer
move middleman(1:out-length) to replacement
display "replacement now: " replacement end-display

move "(red)" & x'00' to pattern

```

```

move "The red car was going too fast" & x'00' to subject
move 0 to out-length
set out-pointer to null
free middleman

call "cbl_oc_genie_regexing"
  using
    by reference pattern
    by reference subject
    by reference out-pointer
    by reference out-length
  returning result
end-call
display "Result from Genie: " result end-display

set address of middleman to out-pointer
move middleman(1:out-length) to replacement
display "replacement now: " replacement end-display

move "[:digit:]" & x'00' to pattern
move "The Regex fails" & x'00' to subject
move 0 to out-length
set out-pointer to null
free middleman

call "cbl_oc_genie_regexing"
  using
    by reference pattern
    by reference subject
    by reference out-pointer
    by reference out-length
  returning result
end-call
display "Result from Genie: " result end-display

set address of middleman to out-pointer
move middleman(1:out-length) to replacement
display "replacement now: " replacement end-display
goback.
end program callgenierex.

```

## The Output

```

$ valac -g -v -c genierex.gs
cc -g -c '/home/btiffin/lang/cobol/genierex.vala.c' -pthread -I/usr/include/glib-2.0 -I/usr/lib64/

$ cobc -g -debug -v -x callgenierex.cob genierex.vala.o -lgobject-2.0 -lglib-2.0
Preprocessing: callgenierex.cob to callgenierex.i
Return status: 0
Parsing:      callgenierex.i
Return status: 0
Translating:  callgenierex.i to callgenierex.c
Executing:    gcc -c -I/usr/local/include -pipe -g -Wno-unused -fsigned-char
              -Wno-pointer-sign -o "/tmp/cob3411_0.o" "callgenierex.c"
Return status: 0
Executing:    gcc -Wl,--export-dynamic -o "callgenierex"
              "/tmp/cob3411_0.o" "genierex.vala.o" -L/usr/local/lib -lcob
              -lm -lgmp -lnurses -ldb -ldl -l"gobject-2.0" -l"glib-2.0"

```

Return status: 0

```
$ ./callgenieregex
```

```
Pattern: Fortran|APL|Python
Subject: OpenCOBOL, Fortran, Vala, Genie, Python, C, APL
Result from Genie: +0000000000
replacement now: OpenCOBOL, COBOL, Vala, Genie, COBOL, C, COBOL
```

```
Pattern: (red)
Subject: The red car was going too fast
Result from Genie: +0000000000
replacement now: The COBOL car was going too fast
```

```
Pattern: [:digit:]
Subject: The Regex fails
Error while compiling regular expression [:digit:] at char 0: POSIX named classes are supported only
Result from Genie: +0000000001
replacement now: The Regex fails
```

## 29.19 5.19 Can OpenCOBOL interface with S-Lang?

Yes. The S-Lang engine can be used with OpenCOBOL for two purposes. Supporting a very nice terminal and keyboard programmer interface S-Lang can be used to scan the keyboard for non-waiting ACCEPT key routines. As a bonus, S-Lang has a very nice scripting engine that allows easy and direct linkage of script variables with OpenCOBOL defined storage members.

### 29.19.1 5.19.1 Setup

You will need the S-Lang library for this interface. Under Debian that is simply

```
$ apt-get install libslang2
```

See <http://www.s-lang.org/> for details of this very capable library.

### 29.19.2 5.19.2 Keyboard control

This sample only show S-Lang terminal input. A very sophisticated terminal output control interface is also available.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin
*> Date:     20090503
*> Purpose:   Experimental S-Lang interface
*> Tectonics: cobb -x slangkey.cob -lslang
*> *****
identification division.
program-id. slangkey.

data division.
working-storage section.
01 thekey          usage binary-long unsigned.
01 thekm           usage binary-long.
```



```

01 result                usage binary-long.

*> exit handler address and priority (prio is IGNORED with OC1.1)
01 install-flag          pic 9 comp-x value 0.
01  install-params.
    02 exit-addr         usage is procedure-pointer.
    02 handler-prio     pic 999 comp-x.

*> *****
procedure division.

*> Initialize low and high level S-Lang terminal routines
call "SLtt_get_terminfo" end-call
call "SLkp_init" returning result end-call
if result equal -1
    display "problem intializing S-Lang tty" end-display
    stop run giving 1
end-if

call "SLang_init_tty" using
    by value -1    *> abort char
    by value -1    *> flow ctrl
    by value 0     *> output processing
    returning result
end-call
if result equal -1
    display "problem intializing S-Lang tty" end-display
    stop run giving 1
else
    display "Keyboard in special mode" x"0d" end-display
end-if

*> install an exit handler to put terminal back
set exit-addr to entry "tty-reset"
call "CBL_EXIT_PROC" using
    install-flag
    install-params
    returning result
end-call
if result not equal zero
    display "error installing exit procedure" end-display
end-if

*> Not sure? Have SLang handle ^C or let OpenCOBOL take over?
call "SLang_set_abort_signal" using by value 0 end-call

*> The demo. Fetch a key, then fetch a keycode. 4 times.
*> SLang terminals display newline as newline. Need explicit
*> CR to get a carriage return. Hence the x"0d".
*> Plus, output is buffered until line terminators.
display
    "Tap a normal key, then tap a 'special' key, ie F1, 4 times"
    x"0d"
end-display
perform 4 times
    call "SLang_getkey" returning thekey end-call
    display thekey space with no advancing end-display
    call "SLkp_getkey" returning thekm end-call

```

```
    display thekm x"0d" end-display
end-perform

*> Exit handler will take care of resetting terminal
goback.

*> *****
*> Exit procedure to ensure terminal properly reset
*> *****
entry "tty-reset".
call "Slang_reset_tty" end-call
display "exit proc reset the tty" end-display
goback.

end program slangkey.
```

#### Outputs:

```
Keyboard in special mode
Tap a normal key, then tap a 'special' key, ie F1, 4 times
0000000097 +0000000513
0000000001 +0000000002
0000000099 +0000065535
0000000003 +0000000003
exit proc reset the tty
```

having tapped, A, F1, Ctrl-A, Ctrl-B, C, EscEsc and Ctrl-C. The S-Lang abort handler pretty much takes over the Ctrl-C handling in this sample so it looks at though Ctrl-C was tapped twice, but it wasn't.

### 29.19.3 5.19.3 Scripting

S-Lang also provides a very comprehensive scripting language, which is very easy to embed.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin
*> Date:     20090505
*> Purpose:  Experimental S-Lang interface
*> Tectonics: cobb -x callslang.cob -lslang
*> *****
identification division.
program-id. callslang.

data division.
working-storage section.
01 result          usage binary-long.
01 cobol-integer   usage binary-long value 42.
01 cobol-float     usage float-long value 0.0.
01 sl-int-type     constant as 20.
01 sl-double-type  constant as 27.
01 read-write     constant as 0.

*> *****
procedure division.

*> Initialize S-Lang
```

```

call "SLang_init_all" returning result
if result equal -1
    display "Sorry, problem initializing SLang" end-display
end-if

*> Register "slint" variable
call "SLadd_intrinsic_variable" using
    by reference "slint" & x"00"
    by reference cobol-integer
    by value sl-int-type
    by value read-write
    returning result
end-call
if result equal -1
    display "Could not register cobol-integer" end-display
end-if

*> Register "sldbl" variable
call "SLadd_intrinsic_variable" using
    by reference "sldbl" & x"00"
    by reference cobol-float
    by value sl-double-type
    by value read-write
    returning result
end-call
if result equal -1
    display "Could not register cobol-float" end-display
end-if

call "SLang_load_string" using
    "sldbl = sum([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]);" & x"00"
    returning result
end-call
if result equal -1
    display "Could not interpret sum intrinsic" end-display
end-if
display "S-Lang set cobol-float to " cobol-float end-display

display "Next lines of output are S-Lang printf" end-display
call "SLang_load_string" using
    ' () = printf("slint (cobol-integer) = %d\n", slint);' & x"00"
    returning result
end-call
if result equal -1
    display "Could not interpret printf" end-display
end-if

add 1 to cobol-integer

call "SLang_load_string" using
    ' () = printf("slint after COBOL add = %d\n", slint);' & x"00"
    returning result
end-call
if result equal -1
    display "error with printf after cobol add" end-display
end-if

*> Let's get out of here and do the Dilbert Nerd Dance...Woohoo!

```

```
goback.  
end program callslang.
```

Which produces:

```
S-Lang set cobol-float to 45.000000000000000000  
Next lines of output are S-Lang printf  
slint (cobol-integer) = 42  
slint after COBOL add = 43
```

## 29.20 5.20 Can the GNAT Programming Studio be used with OpenCOBOL?

Yes. Extensions to smooth the integration of OpenCOBOL development in gnat-gps is posted at <http://svn.wp0.org/ocdocs/brian/opencobol.xml>

```
<?xml version="1.0"?>  
<Custom>  
  <Language>  
    <Name>OpenCOBOL</Name>  
    <Spec_Suffix>.cob</Spec_Suffix>  
    <Extension>.cbl</Extension>  
    <Extension>.cpy</Extension>  
  
    <Keywords>^(identification|id|environment|data|procedure|division|</Keywords>  
    <Keywords>program-id|author|</Keywords>  
    <Keywords>configuration|source-computer|object-computer|</Keywords>  
    <Keywords>special-names|repository|</Keywords>  
    <Keywords>input-output|file-control|io-control|</Keywords>  
    <Keywords>file|working-storage|local-storage|linkage|</Keywords>  
    <Keywords>communication|report|screen|</Keywords>  
    <Keywords>section|declaratives|</Keywords>  
    <Keywords>end|</Keywords>  
    <Keywords>perform|end-perform|until|times|varying|</Keywords>  
    <Keywords>add|subtract|multiply|divide|compute|</Keywords>  
    <Keywords>end-add|end-subtract|end-multiply|end-divide|end-compute|</Keywords>  
    <Keywords>accept|display|read|write|rewrite|sort|</Keywords>  
    <Keywords>end-accept|end-display|end-read|end-write|end-rewrite|</Keywords>  
    <Keywords>move|evaluate|end-evaluate|if|end-if|when|</Keywords>  
    <Keywords>(un)?string|end-(un)?string|call|end-call|</Keywords>  
    <Keywords>goback|stop[\s]+run|</Keywords>  
    <Keywords>filler|low-value[s]?|high-value[s]?|space[s]?|zero[es]?[s]?|b</Keywords>  
  
  <Context>  
    <New_Line_Comment_Start>\*&gt;|[ ]{6}\*</New_Line_Comment_Start>  
    <String_Delimiter>&quot;</String_Delimiter>  
    <Constant_Character>&apos;</Constant_Character>  
    <Can_Indent>True</Can_Indent>  
    <Syntax_Highlighting>True</Syntax_Highlighting>  
    <Case_Sensitive>False</Case_Sensitive>  
  </Context>  
  
  <Categories>  
    <Category>  
      <Name>procedure</Name>  
      <Pattern>^[0-9a-z]+\.</Pattern>
```

```

    <Index>1</Index>
    <Icon>subprogram_xpm</Icon>
  </Category>
</Categories>
</Language>

<alias name="program">
  <param name="pid">prog</param>
  <text>* &gt; OC &lt; *
    * &gt; &gt; SOURCE FORMAT IS FIXED
    * &gt; *****
    * &gt; Author:      Brian Tiffin
    * &gt; Date:        %D
    * &gt; Purpose:    %_
    * &gt; Tectonics:  make
    * &gt; *****
    identification division.
    program-id %(pid).

    environment division.
    configuration section.
    repository.
    special-names.
    input-output section.

    data division.
    file section.
    working-storage section.
    local-storage section.
    linkage section.
    screen section.

    procedure division.
    declaratives.
    end declaratives.

    00-main.

    .
    00-finish.
    goback.
    * &gt; *****

    end program %(pid).
  </text>
</alias>

<Language>
  <Name>Vala</Name>
  <Spec_Suffix>.vala</Spec_Suffix>

  <Keywords>^(bool|char|constpointer|double|float|size_t|ssize_t|string|unichar|void|</Keywords>
  <Keywords>int|int8|int16|int32|int64|long|short|</Keywords>
  <Keywords>uint|uint8|uint16|uint32|uint64|ulong|ushort|</Keywords>
  <Keywords>class|delegate|enum|errordomain|interface|namespace|struct|</Keywords>
  <Keywords>break|continue|do|for|foreach|return|while|</Keywords>
  <Keywords>else|if|switch|</Keywords>
  <Keywords>case|default|</Keywords>

```

```

<Keywords>abstract|const|dynamic|ensures|extern|inline|internal|override|</Keywords>
<Keywords>private|protected|public|requires|signal|static|virtual|volatile|weak|</Keywords>
<Keywords>>false|null|true|</Keywords>
<Keywords>try|catch|finally|throw|</Keywords>
<Keywords>as|base|construct|delete|get|in|is|lock|new|out|params|ref|</Keywords>
<Keywords>sizeof|set|this|throws|typeof|using|value|var|yield|yields)\b</Keywords>

<Context>
  <New_Line_Comment_Start>//</New_Line_Comment_Start>
  <Comment_Start>/*</Comment_Start>
  <Comment_End>*</Comment_End>
  <String_Delimiter>&quot;</String_Delimiter>
  <Constant_Character>&apos;</Constant_Character>
  <Can_Indent>True</Can_Indent>
  <Syntax_Highlighting>True</Syntax_Highlighting>
  <Case_Sensitive>True</Case_Sensitive>
</Context>

<Categories>
  <Category>
    <Name>procedure</Name>
    <Pattern>^[0-9a-z]+\.</Pattern>
    <Index>1</Index>
    <Icon>subprogram_xpm</Icon>
  </Category>
</Categories>
</Language>

<tool name="cobc" package="OpenCOBOL" index="opencobol">
  <language>OpenCOBOL</language>
  <initial-cmd-line>-m</initial-cmd-line>
  <switches lines="3" columns="2">
    <title line="1" column="1" >Code generation</title>
    <title line="1" column="2" >Run-time options</title>
    <title line="2" column="1" line-span="2" >Source forms and Warnings</title>
    <title line="3" column="1" line-span="0" />
    <title line="2" column="2" >Debugging</title>
    <title line="3" column="2" >Syntax</title>

    <radio>
      <radio-entry label="Build dynamic module (default)" switch="-m" />
      <radio-entry label="Build executable" switch="-x" />
      <radio-entry label="Build object file" switch="-c" />
      <radio-entry label="Preprocess only" switch="-E" />
      <radio-entry label="Translation only, COBOL to C" switch="-C" />
      <radio-entry label="Compile only, output assembly file" switch="-S" />
    </radio>
    <check label="Syntax checking only" switch="-fsyntax-only"
      tip="Syntax error checking only; no output emitted" />

    <combo label="Optimization" switch="-O" nodigit="1" noswitch="0"
      tip="Controls the optimization level">
      <combo-entry label="No optimization" value="0" />
      <combo-entry label="Simple optimization" value="1" />
      <combo-entry label="Some more optimization" value="s" />
      <combo-entry label="Full optimization" value="2" />
    </combo>
  </switches>
</tool>

```

```

<field label="Generate Listing to " switch="-t" separator=" " as-file="true"
      tip="Generate a listing file to given filename" />
<field label="Save Generated files to " switch="-save-temps" separator="=" as-directory="tr
      tip="Save temporary files to given directory" />

<radio line="2" column="1">
  <radio-entry label="Format FIXED" switch="-fixed"
    tip="Standards mandate default is fixed format source code" />
  <radio-entry label="Format FREE (FIXED is default)" switch="-free"
    tip="Assume free format source code" />
</radio>
<check label="MF comment (may lead to ambiguous source)" switch="-fmfcomment" line="2" colu
      tip="Allow * or / in column 1 as FIXED format line comment" />
<check label="FUNCTION implied" switch="-ffunctions-all" line="2" column="1"
      tip="Allow use of intrinsic functions without FUNCTION keyword" />
<check label="Fold Copy LOWER" switch="-ffold-copy-lower" line="2" column="1"
      tip="Fold COPY subject to lower case" />
<check label="Fold Copy UPPER" switch="-ffold-copy-upper" line="2" column="1"
      tip="Fold COPY subject to upper case" />
<check label="Full Warnings" switch="-W" line="2" column="1"
      tip="ALL possible warnings" />
<popup label="Warnings" line="2" column="1">
  <check label="All (exceptions listed below)" switch="-Wall" />
  <check label="Obsolete" switch="-Wobsolete"
    tip="Warn if obsolete features used" />
  <check label="Archaic" switch="-Warchaic"
    tip="Warn if archaic features used" />
  <check label="Redefinition" switch="-Wredefinition"
    tip="Warn of incompatible redefinition of data items" />
  <check label="Constant" switch="-Wconstant"
    tip="Warn of inconsistent constant" />
  <check label="Parentheses" switch="-Wparentheses"
    tip="Warn of lack of parentheses around AND within OR" />
  <check label="Strict typing" switch="-Wstrict-typing"
    tip="Warn of type mismatch, strictly" />
  <check label="Implicit define" switch="-Wimplicit-define"
    tip="Warn of implicitly defined data items" />
  <check label="Call params (Not set for All)" switch="-Wcall-params"
    tip="Warn of non 01/77 items for CALL" />
  <check label="Column overflow (Not set for All)" switch="-Wcolumn-overflow"
    tip="Warn for FIXED format text past column 72" />
  <check label="Terminator (Not set for All)" switch="-Wterminator"
    tip="Warn when missing scope terminator (END-xxx)" />
  <check label="Truncate (Not set for All)" switch="-Wtruncate"
    tip="Warn of possible field truncation" />
  <check label="Linkage (Not set for All)" switch="-Wlinkage"
    tip="Warn of dangling LINKAGE items" />
  <check label="Unreachable (Not set for All)" switch="-Wunreachable"
    tip="Warn of unreachable statements" />
</popup>

<check label="Internal run-time error checks" switch="-debug" column="2"
      tip="generate extra internal tests" />
<check label="Implicit initialize" switch="-fimplicit-init" column="2"
      tip="Do automatic initialization of the Cobol runtime system" />
<check label="No truncation" switch="-fnotrunc" column="2"
      tip="Do not truncate binary fields according to PICTURE" />
<check label="Sign ASCII" switch="-fsign-ascii" column="2"

```

```

        tip="Numeric display sign ASCII (Default on ASCII machines)" />
<check label="Sign EBCDIC" switch="-fsign-ebcdic" column="2"
        tip="Numeric display sign EBCDIC (Default on EBCDIC machines)" />
<check label="Stack checking for PERFORM" switch="-fstack-check" column="2"
        tip="Generate code to verify that you do not go beyond the boundary of the stack" />
<check label="Pass extra NULL" switch="-fnull-param" column="2"
        tip="Pass extra NULL terminating pointers on CALL statements" />

<check label="Enable Debugging lines" switch="-fdebugging-line" line="2" column="2"
        tip="Enable column 7 D (FIXED FORMAT) debug lines and &gt;&gt;D inline compiler dire
<check label="Object Debug Information" switch="-g" line="2" column="2"
        tip="Link level debug information" />
<check label="Trace (SECTION/PARAGRAPH)" switch="-ftrace" line="2" column="2"
        tip="Enable output of trace statements for SECTION and PARAGRAPH" />
<check label="Trace all (SECTION/PARAGRAPH/STATEMENT)" switch="-ftraceall" line="2" column="2"
        tip="Enable output of trace statements for SECTION, PARAGRAPH and STATEMENTS" />
<check label="Source locations" switch="-fsource-location" line="2" column="2"
        tip="Generate source location code (Turned on by -debug or -g)" />

<check label="COBOL2002" switch="-std=cobol2002" line="3" column="2"
        tip="Override the compiler's default, and configure for COBOL 2002" />
<check label="COBOL 85" switch="-std=cobol85" line="3" column="2"
        tip="Override the compiler's default, and configure for COBOL 85" />
<check label="Micro Focus" switch="-std=mf" line="3" column="2"
        tip="Override the compiler's default, and Micro Focus compatibility" />
</switches>
</tool>

<action name="make">
  <external>make</external>
</action>

<action name="cobc">
  <external>cobc -x %f</external>
</action>

<action name="cobcrun">
  <external>cobcrun %p</external>
</action>

<action name="valac">
  <external>valac --pkg gtk+-2.0 %f</external>
</action>

<action name="gdb">
  <external>konsole --vt_sz 132x24 -e gdb ./%p</external>
</action>

<action name="cgdb">
  <external>konsole --vt_sz 132x24 -e cgdb ./%p</external>
</action>

<action name="cgdb...">
  <shell>MDI.input_dialog "Enter command arguments" "Args"</shell>
  <external>konsole --vt_sz 132x24 -e cgdb --args ./%p %1</external>
</action>

<action name="gdbtui">

```



```

    <external>konsole --vt_sz 132x24 -e gdbtui --args ./%p %1</external>
</action>

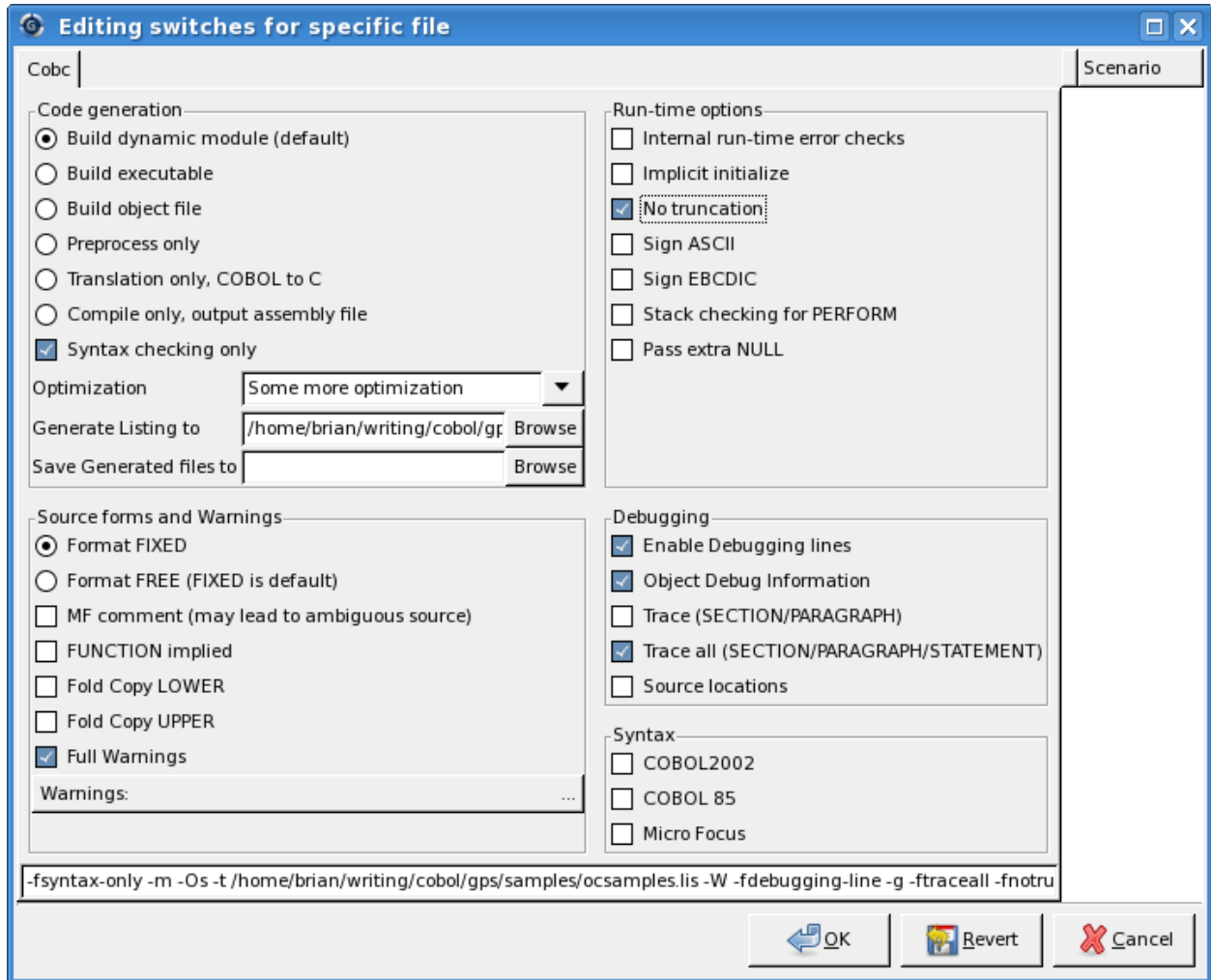
<action name="gdbtui...">
  <shell>MDI.input_dialog "Enter command arguments" "Args"</shell>
  <external>konsole --vt_sz 132x24 -e gdbtui --args ./%p %1</external>
</action>

<action name="DDD">
  <external>ddd ./%p</external>
</action>

<submenu after="Build">
  <title>OpenCOBOL</title>
  <menu action="make">
    <title>make</title>
  </menu>
  <menu action="cobc">
    <title>cobc</title>
  </menu>
  <menu action="cobcrun">
    <title>cobcrun</title>
  </menu>
  <menu action="valac">
    <title>valac</title>
  </menu>
  <menu><title /></menu>
  <menu action="gdb">
    <title>gdb</title>
  </menu>
  <menu action="cgdb">
    <title>cgdb</title>
  </menu>
  <menu action="cgdb...">
    <title>cgdb...</title>
  </menu>
  <menu action="gdbtui">
    <title>gdbtui</title>
  </menu>
  <menu action="gdbtui...">
    <title>gdbtui...</title>
  </menu>
  <menu action="DDD">
    <title>ddd</title>
  </menu>
</submenu>
</Custom>

```

which allows for development screens like



or *to be honest* would do, if the final touches were added to the XML to integrate more with the GPS suite. There is more work required to make a proud developer's interface. *Anyone?*

## 29.21 5.21 Does OpenCOBOL support SCREEN SECTION?

Yes. The OpenCOBOL 1.1 pre-release now includes support for SCREEN SECTION. Experimental release for this support occurred in early July, 2008.

The compiler recognizes most (if not all) of the *Screen description entry* of the COBOL 20xx Draft standard.

External variables that influence screen handling include

**COB\_SCREEN\_EXCEPTIONS=Y** To enable exceptions during ACCEPT.

**COB\_SCREEN\_ESC=Y** To enable handling of the escape key.

See [Does OpenCOBOL support CRT STATUS?](#) for more information on key codes and exception handling.

According to the standard a SCREEN SECTION ACCEPT does not need to be preceded by a DISPLAY. The extra DISPLAY won't hurt, but is not necessary.

### 29.21.1 5.21.1 Environment variables in source code

Thanks to Gary Cutler and [opencobol.org](http://opencobol.org).

In order to detect the PgUp, PgDn or PrtSc (screen print) keys, you must first set the environment variable COB\_SCREEN\_EXCEPTIONS to a non-blank value.

If you want to detect the Esc key, you must set COB\_SCREEN\_EXCEPTIONS as described above AND you must also set COB\_SCREEN\_ESC to a non-blank value. Fortunately, both of these can be done within your OpenCOBOL program, as long as they're done before the ACCEPT.

```
SET ENVIRONMENT 'COB_SCREEN_EXCEPTIONS' TO 'Y'
SET ENVIRONMENT 'COB_SCREEN_ESC' TO 'Y'
```

### 29.22 5.22 What are the OpenCOBOL SCREEN SECTION colour values?

The FOREGROUND-COLOR and BACKGROUND-COLOR clauses will accept

```
78 black value 0.
78 blue value 1.
78 green value 2.
78 cyan value 3.
78 red value 4.
78 magenta value 5.
78 brown value 6.
78 white value 7.
```

The display of these colours are also influenced by HIGHLIGHT, LOWLIGHT and REVERSE-VIDEO options. For instance, brown will display as yellow when HIGHLIGHT is used.

### 29.23 5.23 Does OpenCOBOL support CRT STATUS?

Yes.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CRT STATUS IS screen-status.

DATA DIVISION.
WORKING-STORAGE SECTION.
COPY screenio.
01 screen-status pic 9(4).

PROCEDURE DIVISION.
ACCEPT screen-sample.
IF screen-status = COB-SCR-F1
    ...
```

There is also a special OpenCOBOL variable, **COB-CRT-STATUS** which can be used instead of the CRT STATUS special name.

There is also a COPY text that ships with OpenCOBOL, copy/screenio.cpy that can be included in the DATA DIVISION and provides 78 level constants for supported key status codes. Some values include:

- COB-SCR-F1 thru
- COB-SCR-F64
- COB-SCR-ESC

examine the file to see the other values.

## 29.24 5.24 What is CobCurses?

CobCurses is an optional package designed to work with OpenCOBOL 1.0, before OpenCOBOL 1.1 SCREEN SECTION support was initiated. It has many features beyond simple SCREEN SECTION handling.

See <http://sourceforge.net/projects/cobcurses> for full details. This is a major piece of work by Warren Gay, ve3wwg.

From an [opencobol.org](http://opencobol.org) posting by Warren announcing release 0.95:

CobCurses is a package designed to allow Open-Cobol programmers to create screens on open system platforms, or those (like Windows) that can use PDCurses. Since handcrafting screens is tedious work, this package includes a "Screen Designer" utility.

All User Guides and Programmer Guide documentation can be found on the source forge (see link at bottom).

==== RELEASE NOTES ====

A large number of internal changes were implemented in this release, but first let's cover the user visible improvements:

1. MENUS! Popup menus are now supported, and are available in sdesign with every Action field. In fact, any sdesign field that is marked with a diamond graphic, has the ability to popup a menu with F1 (or ^O).
2. To support menus, FUNCTION keys are now available in Action mode (though CONTROL-O is an alternate way of opening a menu). This included a new event callback NC-FKEY-EVENT.
3. GRAPHIC characters in the screen background. It is now possible using sdesign to draw alternate-charset graphics in your screen background. See the notes in the opening help screen for the "Paint" function.
4. TRACE facilities. CobCurses now includes an environment variable that can enable capturing of trace information to a file for debugging. A routine named NC\_TRACE\_MSG can also be used to add custom messages to the trace file.

INTERNAL CHANGES:

The main two major internal changes were:

1. The terminal support has been virtualized, so that

the CobCurses routines deal with a "terminal" object (not curses routines). This will eventually lead to other possible windowing interfaces like perhaps graphic X Window or native Windows support.

The other motivation for this was to allow CobCurses to have one consistent set of constants for colours, attributes and character sets. Previously, these values were different depending upon the platform and implementation of curses used.

2. Menu support has been provided independently of curses. This is important for portability since PDCurses and some platforms do not provide a curses menu library. This also guarantees that CobCurses menus will behave consistently on all platforms (and overcome menu paging bugs in ncurses).

PLANNED FOR THE NEXT RELEASE:

Please avoid writing much code that works with colour pairs. In the next release, it is planned to hide the colour pair value altogether by using a TDC (Terminal Drawing Context). This TDC will tie together attributes and colours, and perhaps other "drawing contexts" so that you won't have to manage colour pairs (this will be transparent). This will also pave the way for graphical interfaces where a selected font and line styles etc. may also be supported.

NOTES:

HPUX users will need to link with ncurses, instead of the native HPUX curses libraries. I didn't have time to fully investigate this, but the native include files define things like MENU and ITEM types that conflict with the CobCurses defined ones.

====

The release is available for download here:

<http://sourceforge.net/projects/cobcurses>

## 29.25 5.25 What is CobXRef?

CobXRef is a COBOL cross-referencing utility written by Vincent Coen and ported to OpenCOBOL 1.1.

Current source code is available at <http://svn.wp0.org/add1/tools/cobxref> or <http://sourceforge.net/projects/cobxref/> and is currently (*February 2013*) in active development.

The system ships with full documentation and information for building from source is included in the *readme* file.

Fetching the utility

```
$ svn checkout http://svn.wp0.org/add1/tools/cobxref
```

Visit the project space at <https://sourceforge.net/projects/cobxref/> for the latest information.

Example using the cobxref.cbl OpenCOBOL program for source code:

```
$ cobc -E cobxref.cbl
$ cobxref cobxref.i
$ cat cobxref.lst | head -300
```

ACS Cobol Cross Reference Xref v0.95.20 (09/02/2013@00:33) Dictionary File for COBXREF 09/02/13

Symbols of Module: COBXREF (COBXREF)

Data Section (FILE)	Defn	Locations
FS-REPLY	000067F	000195 002904
P-CONDITIONS	000098F	001981
P-VARIABLES	000099F	001980
PL-PROG-NAME	000095F	002309 002312
PRINT-FILENAME	000062F	000197 002859 002924
PRINTLINE	000083F	001397 001398 001399 001400 001937 001939 001942 002063 002133 002135 002138 002205 002215 002225 002227 002264 002271 002284 002288 002290 002301 002315 002701 002702 002703 002704 002705 002716 002717 002720 002721 002725 002726 002727 002728 002731 002732 002734 002735 002736 002737 002740 002741 002742 002743 002746 002747 002748 002751 002752 002753 002756 002757 002758 002759 002923
PRINTLINE2	000097F	001979 001982
SDSORTKEY	000119F	001861
SKADATANAME	000108F	001120 001122 001911 001918 001928 001943 002024 002033 002035 002042 002049 002054 002056 002064 002106 002124 002126 002139 002179 002192 002201 002206 002248 002257 002266 002272 002349 002351
SKAREFNO	000111F	001124 001944 001952 002065 002079 002140 002148 002180 002207 002273 002350 002352
SKAWSORPD	000109F	001116 001921 001923 001929 001945 002034 002036 002051 002057 002070 002115 002141 002196 002208 002267 002275 002336 002668
SKAWSORPD2	000110F	001117 001930 002037 002058 002069 002274 002337 002669
SL-GEN-REFNO1	000080F	002332
SORTFILE	000069F	000117 001860
SORTRECORD	000107F	001113 001126 002354 002923
SOURCE-LIST	000079F	002330 002333
SOURCE-LISTING	000062F	000078 001094 001145 001386 001449 001472 001594
SOURCEFILENAME	000065F	000196 002833 002838 002924
SOURCEINPUT	000065F	000101 001146 001383 001449 001472 001594 002371 002903
SOURCEOUTPUT	000081F	002331
SOURCERECIN	000103F	002331 002370 002375
SUPPLEMENTAL-PART1-OUT	000059F	000073 000105 001106 001145 001381 001383 001449 001472 001594 001862
SUPPLEMENTAL-PART2-IN	000056F	000074 000113 001863 001891 001892 001904 001907 002007 002008 002018 002020 002091 002092 002099 002101 002159 002160 002170 002172 002230 002231 002241 002244
XRDATANAME	000084F	001933 001943 002064 002129 002139 002206 002226 002272 002289 002307 002311
XRDEFN	000085F	001944 002065 002140 002207 002273 002303
XRREFERENCE	000089F	001952 002079 002148
XRTYPE	000086F	001945 002070 002072 002141 002208 002275 002277 002314

ACS Cobol Cross Reference Xref v0.95.20 (09/02/2013@00:33) Dictionary File for COBXREF 09/02/13

Symbols of Module: COBXREF (COBXREF)

Data Section (WORKING-STORAGE)	Defn	Locations
A	000138W	001084 001085 001086 001088 001089 001090 001114 001115 001116 001117 001119 001122 001124 001125 001185 001190 001200 001202 001241 001242 001244 001267 001269 001286 001316 001318 001343 001351 001492 001542 001668 001717 001719 001719 001732 001735 001742 001747 001972 001977 001980 001981 002302 002303 002304 002306 002308 002311 002312 002585 002641 002645
ADDITIONAL-RESERVED-WORDS	000490W	001018
ALL-FUN-IDX	000478W	002667 002670
ALL-FUNCTIONS	000477W	001077 002665
ARG-NUMBER	000135W	002823 002824 002826
ARG-VALS	000361W	001101 002864
ARG-VALUE	000362W	002832 002856 002868 002869 002872 002873 002879 002880 002885 002886 002892 002893 002899 002900 002912
B	000139W	001732 001736 001747 001752 002314 002585
BUILD-NUMBER	000137W	001456 001459 001465 001466 001469 001550 001553 001562 001601 001608 001625 002764 002766 002769 002956
C	000140W	001732 001737 001760 002585
COBOLSEQNO	000210W	002374
CON-TAB-BLOCKS	001026W	001967 001990
CON-TAB-COUNT	001032W	001233 001234 001235 001554 001563 001564 001567 001569 001571 001573 001964 001966 001977 001989 002932
CON-TAB-SIZE	001031W	001554 001555 001556 001557 001563
CONDITION-TABLE	001025W	002928
CONDITIONS	001029W	001235 001569 001573 001980 001990
D	000141W	001732 001738 001767 001772 001775 002400 002577 002578 002582 002592 002595 002596 002598 002598
DUMP-RESERVED-WORDS	000156W	001083
E	000142W	002485 002487 002490 002492 002500
END-PROG	000172W	001143 001380 001385 001389 001395 001655 002364 002441
ERROR-MESSAGES	000335W	
F-POINTER	000134W	001139 002664 002670 002934
FS-REPLY	000195W	002904
FULL-SECTION-NAME	000374W	001244 002723
FUNCTION-TABLE	000387W	000476
FUNCTION-TABLE-SIZE	000481W	001089
GEN-REFNO1	000136W	001439 001587 002326 002332 002350 002352
GIT-BUILD-NO	001043W	002956 002962
GIT-ELEMENTS	001035W	001865 002963
GIT-HOLDWSORPD	001041W	001116 001125 002304 002958
GIT-HOLDWSORPD2	001042W	001117 002959
GIT-PROG-NAME	001039W	002308 002312 002957
GIT-REFNO	001040W	001124 002303 002955
GIT-TABLE-COUNT	001045W	001112 001115 001864 001878 002302 002946 002947 002954 002955 002956 002957 002958 002959 002961 002962 002963
GIT-TABLE-SIZE	001044W	002947 002948 002949 002950 002951
GIT-WORD	001038W	001119 001122 001865 002306 002311 002954
GLOBAL-ACTIVE	000178W	001603 001607 001624
GLOBAL-CURRENT-LEVEL	000201W	001425 001428 001460 001463 001586 002944
GLOBAL-CURRENT-REFNO	000200W	001439 001587 002955
GLOBAL-CURRENT-WORD	000199W	001438 001588 002923 002954
GLOBAL-ITEM-TABLE	001034W	001071

GOTASECTION	000185W	001370	001414	001417	001418	002776	002781	002785	002789
		002793	002797	002801	002805	002809	002814		
GOTENDPROGRAM	000186W	002930							
H1-DD	000241W	002696							
H1-HH	000247W	002699							
H1-MIN	000249W	002700							
H1-MM	000243W	002697							
H1-YY	000245W	002698							
H1PROG-NAME	000239W	001134							
H1PROGRAMID	000252W	002694							
HAD-END-PROG	000175W	002361							
HAVE-NESTED	000181W	001164	001171						
HD-D	000310W	002684							
HD-DATE-TIME	000312W	002693							
HD-HH	000303W	002687							
HD-M	000309W	002683							
HD-MM	000304W	002688							
HD-SS	000305W	002689							
HD-UU	000306W	002690							
HD-Y	000308W	002682							
HD2-D	000313W	002684							
HD2-HH	000319W	002687							
HD2-M	000315W	002683							
HD2-MM	000321W	002688							
HD2-SS	000323W	002689							
HD2-UU	000325W	002690							
HD2-Y	000317W	002682							
HDDATE	000307W	002680	002681						
HDR1	000236W	002703							
HDR10	000286W	002751							
HDR11	000290W	002756							
HDR11A-SORTED	000292W	001968	001992						
HDR11B-SORTED	000296W	001969	001991						
HDR12-HYPHENS	000299W	002757							
HDR3	000254W	002726	002735	002741					
HDR5-PROG-NAME	000261W	002707	002712	002713					
HDR5-SYMBOLS	000259W	002716							
HDR6-HYPHENS	000268W	002713	002714						
HDR6-SYMBOLS	000263W	002717							
HDR7-VARIABLE	000272W	002722	002724	002733					
HDR7-WS	000270W	002725	002734						
HDR8-HD	000277W	002119							
HDR8-WS	000276W	002740							
HDR9	000282W	002746							
HDTIME	000302W	002685	002686						
HOLDFOUNDWORD	000189W	001716	001719						
HOLDID	000203W	001096	001098	001100	001167	001170	002691	002708	002957
HOLDID-MODULE	000204W	001100	001174	001177	002710				
HOLDWSORPD	000187W	001291	001328	001354	001372	001376	001415	001461	002336
		002339	002353	002409	002419	002777	002780	002784	002788
		002792	002796	002800	002804	002808	002932	002958	
HOLDWSORPD2	000188W	001292	001329	001355	001436	001707	001710	001718	001720
		002337	002810	002932	002959				
LINE-COUNT	000129W	001872	002090	002102	002118	002120			
LINE-END	000131W	002396	002596						
LIST-SOURCE	000154W	001135	002329						
LSECT	000356W	001945	002070	002141	002208	002275	002314		
MSG1	000336W	001893	002009	002093	002161	002232			



MSG10	000348W	002952								
MSG16	000350W	001298								
MSG2	000337W	001144								
MSG4	000338W	001447								
MSG5	000340W	001469								
MSG6	000342W	001558								
MSG7	000344W	001783								
MSG8	000345W	002367								
MSG9	000346W	002905								
P-FUNCTION	000480W	001077	001090	002667						
P-OC-IMPLEMENTED	000479W									
PRINT-FILENAME	000197W	002859	002924							
PROG-BASENAME	000198W	001096	001098	002856	002858					
PROG-NAME	000122W	001134	002840							
Q	000148W	001139	001732	001758	001761	001787	001788	001790	001796	
		001798	001800	001802	001808	001811	001812	001842	001844	
		001897	001934	001935	001936	001949	001951	001952	002013	
		002076	002078	002079	002095	002130	002131	002132	002145	
		002147	002148	002165	002209	002213	002236	002261	002278	
		002282								
REPORTS-IN-LOWER	000162W	001095	001118	001165	001172	001565	002305	002341		
RESERVED-NAMES	001019W	001076	002642							
RESVD-IDX	001020W	002644	002645							
RESVD-IMPLEMENTED	001021W									
RESVD-TABLE-SIZE	001023W	001085								
RESVD-WORD	001022W	001076	001086	002644						
S	000143W	001732	001758	001761	001790	001810	001811	001846	001848	
		002459	002479	002493	002504	002526	002543			
S-POINTER	000125W	001139	002158	002214	002224	002229	002263	002283	002287	
		002934								
S-POINTER2	000126W	001373	002427	002443	002444	002447	002450	002451	002452	
		002459	002463	002465	002468	002478	002479	002483	002485	
		002490	002493	002497	002499	002504	002511	002526	002532	
		002543	002552	002934						
S-POINTER3	000127W	002934								
S-POINTER4	000128W	002935								
SAVED-VARIABLE	000191W	001220	001234	001457	001535	001543	001551	001566	001571	
		001602	001626							
SAVESKADATANAME	000190W	001890	001918	001928	002005	002033	002035	002042	002049	
		002054	002056	002088	002124	002126	002157	002192	002201	
		002228	002257	002266	002924					
SAVESKAWSORPD	000192W	001929	002006	002036	002043	002050	002057	002089	002158	
		002229	002262	002267	002282					
SAVESKAWSORPD2	000193W	001930	002006	002037	002058	002089	002158			
SECTION-NAME	000375W	002779	002783	002787	002791	002795	002799	002803	002807	
SECTION-NAMES-TABLE	000364W	000373								
SECTION-USED-TABLE	000358W	001873	002931							
SECTABLE	000353W	000355								
SOURCE-EOF	000169W	001142	001297	001657	002366	002440	002554			
SOURCE-LINE-END	000132W	002396	002443	002451						
SOURCE-WORDS	000133W	001709	002428	002560						
SOURCEFILENAME	000196W	002833	002838	002924						
SOURCEIN8-160	000212W	001149	001183	001204	001207	001209	001244	001271	001273	
		001275	001719	002376	002380	002383	002391	002392	002393	
		002394	002404	002410	002411	002420	002444	002447	002450	
		002460	002468	002478	002480	002487	002492	002494	002500	
		002529	002578	002591	002592	002593	002598			
SOURCEINWS	000209W	001471	002370							

STRING-POINTER	000123W	002825	002831	002834	002839	002855	002857	002870	002874
		002911	002912						
STRING-POINTER2	000124W	002855	002860						
SV1WHAT	000213W	002779	002783	002787	002791	002795	002799	002803	002807
SW-2	000153W	002881							
SW-4	000155W	002887							
SW-5	000158W	002895							
SW-6	000161W	002901							
SW-END-PROG	000171W	001401	002412						
SW-GIT	000177W	001437	001462	001521	001621	002815			
SW-HAD-END-PROG	000174W	002362	002412						
SW-NESTED	000180W	002412							
SW-SOURCE-EOF	000168W	002372	002930						
T	000144W	001754	001756	001797	001803	001815	001829	001833	001838
		001839	001840						
USECT	000359W	001125	001886	002353					
VARIABLES	001030W	001234	001567	001571	001967	001981			
WE-ARE-TESTING	000159W	001388	002413	002418	002561	002597	002960		
WORD-DELIMIT	000183W	001214	001224	001236	001280	001288	001311	001316	001331
		001339	001357	001497	001502	001506	001509	001517	001600
		001605	001659	001708	002445	002462	002469	002482	002486
		002488	002491	002496	002501	002531	002537	002562	
WORD-DELIMIT2	000184W	002506	002525	002529	002537	002544			
WORD-LENGTH	000130W	001322	001446	001447	001714	001717	001721	001722	001733
		001744	001745	001748	001749	001751	001753	001755	001756
		001764	001771	001773	001777	001798	001808	001815	001827
		001839	001841	002446	002470	002521	002545	002563	002765
		002768							
WS-ANAL1	000194W	001874	001886	001887	001905	001921	001923	002723	
WS-WC-DD	000330W	002696							
WS-WC-HH	000331W	002699							
WS-WC-MIN	000332W	002700							
WS-WC-MM	000329W	002697							
WS-WC-YY	000328W	002698							
WS-WHEN-COMPILED	000327W	002695							
WSF1-1	000219W	001311	001313	001337	001660	001670	001672	001675	001678
		001680	001683	001686	002467	002473	002475	002476	002477
		002502	002509	002513	002517	002525			
WSF1-2	000218W	001699							
WSF1-3	000217W	001693	001701						
WSF3-1	000226W	002766	002770						
WSF3-2	000227W	002771							
WSFOUNDNEWWORD	000230W	001776	001778	001779	001828	001831	001842	001844	002926
WSFOUNDNEWWORD2	000231W	001828	001836	001846	001848	002926			
WSFOUNDNEWWORD3	000232W	001731	001790	001811	001819	002925			
WSFOUNDNEWWORD4	000233W	002338	002343	002345	002349	002351	002667	002925	
WSFOUNDNEWWORD5	000234W	002586	002592	002593	002594				
WSFOUNDWORD	000216W	000225							
WSFOUNDWORD2	000225W	001157	001166	001169	001173	001176	001216	001220	001226
		001227	001235	001306	001345	001424	001427	001433	001438
		001446	001447	001470	001485	001488	001489	001493	001503
		001507	001520	001523	001534	001551	001568	001572	001588
		001615	001616	001620	001637	001673	001676	001681	001684
		001714	001721	001722	001735	001736	001737	001738	001739
		001740	001745	001748	001753	001754	001755	001768	001771
		001778	001779	001788	001790	001800	001811	001819	001830
		001835	001843	001844	001847	001848	002338	002342	002345
		002448	002458	002461	002471	002481	002495	002503	002505

```

                                002510 002519 002528 002530 002539 002544 002563 002644
Y                                000145W 001732 001739 001741 001757
Y2                               000146W 001732 001740 001741
Z                                000147W 001733 001735 001736 001737 001738 001739 001740 001764
                                001765 001766 001768 001778 001827 001830 001835 001841
                                002518 002519 002521 002538 002539 002543 002544 002545

```

ACS Cobol Cross Reference Xref v0.95.20 (09/02/2013@00:33) Dictionary File for COBXREF 09/02/13

Variable Tested [S]                      Symbol (88-Conditions)

-----  
 SN-TEST-1                                SNT1-ON

Listing above limited to 300 lines

## 29.26 5.26 Does OpenCOBOL implement Report Writer?

Not at this time. *July, 2008*

But it does support LINAGE. See [Does OpenCOBOL implement LINAGE?](#)

## 29.27 5.27 Does OpenCOBOL implement LINAGE?

Yes. LINAGE sets up logical pages inside file descriptors enhancing the WRITE operations and enabling the END-OF-PAGE clause.

```

FILE SECTION.
FD  A-REPORT
   LINAGE IS 13 LINES
   TOP 2
   FOOTING 2
   BOTTOM 3.

```

LINAGE clauses can set:

```

TOP
LINES
FOOTING
BOTTOM

```

The `LINAGE-COUNTER` noun is maintained during writes to LINAGE output files.

See [LINAGE](#) for a sample program.

## 29.28 5.28 Can I use ctags with OpenCOBOL?

Yes. Use the Exuberant version of ctags. Exuberant ctags recognizes COBOL, producing a TAGS or tags file suitable for **emacs**, **vi**, **nedit** and other editors that support the ctags format. *ctags, by default, only supports the competition, C and Fortran.*

After running ctags program.cob

```
$ vi -t WORKING-STORAGE
```

will open program.cob and start at the line defining the working-storage section. Note: tags are case-sensitive and for larger projects, the above vi command would start an edit of the *first* file with an occurrence of WORKING-STORAGE found in the tags.

## 29.29 5.29 What about debugging OpenCOBOL programs?

OpenCOBOL internal runtime checks are enabled with **-debug**.

Support for tracing is enabled with **-ftrace** and **-ftraceall**.

Source line location is enabled with **-fsource-location**, and implied with the **-g** and **-debug** options..

Activation of FIXED format **D** indicator debug lines is enabled with **-fdebugging-line**. In FREE format, **>>D** can be used anywhere on a line. See [Does OpenCOBOL support D indicator debug lines?](#).

**-fstack-check** will perform stack checking when **-debug** or **-g** is used.

**-fsyntax-only** will ask the compiler to only check for syntax errors, and not emit any output.

To view the intermediate files that are generated, using **-C** will produce the .c source files and any .c.l.h and c.h header files. **-save-temps[=dir]** will leave all intermediate files in the current directory or the optional directory specified, including .i files that are the COBOL sources after COPY processing.

Support for **gdb** is enabled with **-g**.

```
$ gdb hello
GNU gdb 6.7.1-debian
Copyright (C) 2007 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...
Using host libthread_db library "/lib/i686/cmov/libthread_db.so.1".
(gdb) break 106
Breakpoint 1 at 0x0BFUSCA: file hello.c, line 106.
(gdb) break 109
Breakpoint 2 at 0xTETHESY: file hello.c, line 109.
(gdb) run
Starting program: /home/brian/writing/cobol/hello
[Thread debugging using libthread_db enabled]
[New Thread 0xSTEMADDR (LWP 5782)]
[Switching to Thread 0xESSES6b0 (LWP 5782)]

Breakpoint 1, hello_ (entry=0) at hello.c:106
106      cob_new_display (0, 1, 1, &c_1);
(gdb) cont
Continuing.
Hello World!

Breakpoint 2, hello_ (entry=0) at hello.c:109
109      cob_set_location ("hello", "hello.cob", 6, "MAIN SECTION", "MAIN PARAGRAPH", "STOP");
(gdb) cont
Continuing.
```

Program exited normally.  
(gdb)

Setting a break at line 106 and 109 was found by a quick look through the C code from `$ cobc -C hello.cob` and seeing where the DISPLAY call and STOP RUN was located. *Note: just because; the gdb displayed addresses were obfuscated from this listing.*

### 29.29.1 5.29.1 Some debugging tricks

From [human] on opencobol.org:

If you want to have different outputs in debug / normal mode use a fake if 1 = 1 like

```
OCOBOL
D   IF 1 = 1
D       DISPLAY "Debug Line"  END-DISPLAY
D   ELSE
D       DISPLAY "Normal Line" END-DISPLAY
D   END-IF
```

For using the environment Just define

```
OCOBOL
01 debugmode pic x.
88 debugmode-on values 'O', 'Y', 'J', 'o', 'y', 'j', '1'.
```

put an

```
OCOBOL
accept debugmode from Environment "DEBUGMODE"
end-accept
```

at the beginning of each program (or define debugmode as external) and use it in your programs like

```
OCOBOL
IF debugmode-on
    DISPLAY "Debug Line"  END-DISPLAY
ELSE
    DISPLAY "Normal Line" END-DISPLAY
END-IF
```

For having no debug code in runtime you can combine these two

```
OCOBOL
D 01 debugmode pic x.
D 88 debugmode-on values 'O', 'Y', 'J', 'o', 'y', 'j', '1'.
...
D accept debugmode from Environment "DEBUGMODE"
D end-accept
...
D IF debugmode-on
D DISPLAY "Debug Line"  END-DISPLAY
D ELSE
D DISPLAY "Normal Line" END-DISPLAY
D END-IF
```

In this way you have fast code at runtime (if not compiled with `-fdebugging-line`) and can switch the output during development.

The advantages over a compiler switch to disable the displays are:

- You can always use display in your program, not only for debug information.
- You see in the code what you do.
- If compiled with lines that have 'D' indicator you can switch at runtime.
- If compiled without lines that have 'D' indicator you can have faster and smaller modules.

### 29.29.2 5.29.2 Animator

Federico Priolo posted this beauty of a present on [opencobol.org](http://opencobol.org)

TP-COBOL-DEBUGGER

<http://sourceforge.net/projects/tp-cobol-debugg/> and on his company site at <http://www.tp-srl.it/>

A system to preprocess OpenCOBOL inserting animator source code that at runtime provides a pretty slick stepper with WORKING-STORAGE display.

This open source bundle is OpenCOBOL. Compile the animator, run it over your own programs and it generates a new source file that when compiled and evaluated, runs in a nice SCREEN SECTION showing original source and a view pane into WORKING-STORAGE.

### 29.29.3 5.29.3 Unit testing

See [What is COBOLUnit?](#) for links to a well define full on Unit testing framework for COBOL, written in OpenCOBOL.

## 29.30 5.30 Is there a C interface to OpenCOBOL?

Most definitely. See [http://www.opencobol.org/modules/bwiki/index.php?cmd=read&page=UserManual%2F2\\_3#content\\_1\\_0](http://www.opencobol.org/modules/bwiki/index.php?cmd=read&page=UserManual%2F2_3#content_1_0) for details.

As a short example, showing off a little of cobc's ease of use when it comes to C source code.

**hello.c**

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    printf("Hello C compiled with cobc\n");
}

int hello(int argc, char *argv[]) {
    printf("Hello C compiled with cobc, run from hello.so with cobcrun\n");
}
```

With a sample run of

```
$ cobc hello.c
$ cobcrun hello
Hello C compiled with cobc, run from hello.so with cobcrun
$ cobc -x hello.c
$ ./hello
```

Hello C compiled with cobc

```
[btiffin@home cobol]$ cobc -v -x hello.c
Executing:      gcc -c -I/usr/local/include -pipe -Wno-unused -fsigned-char
                -Wno-pointer-sign -o "/tmp/cob2785_0.o" "hello.c"
Return status: 0
Executing:      gcc -Wl,--export-dynamic -o "hello" "/tmp/cob2785_0.o"
                -L/usr/local/lib -lcob -lm -lgmp -lncurses -ldb -ldl
Return status: 0
```

## 29.31 5.31 What are some idioms for dealing with C char \* data from OpenCOBOL?

*Thanks to Frank Swarbrick for pointing these idioms out*

To add or remove a null terminator, use the STRING verb. For example

```
OCOBOL
* Add a null for calling C
STRING current-url
  DELIMITED BY SPACE
  X"00" DELIMITED BY SIZE
  INTO display-url
MOVE display-url TO current-url

* Remove a null for display
STRING current-url
  DELIMITED BY LOW-VALUE
  INTO display-url.
```

Or to make changes in place

```
OCOBOL
* Change nulls to spaces
INSPECT current-url
  REPLACING ALL X"00" WITH SPACE.
```

Or there is also modified references in OpenCOBOL

```
OCOBOL
* Assume IND is the first trailing space (or picture limit).
* Note: OpenCOBOL auto initializes working-storage to SPACES or ZEROES
*       depending on numeric or non-numeric pictures.
* Remove null
MOVE SPACE TO current-url(IND:1).

* Add a zero terminator
MOVE X"00" TO current-url(IND:1).
```

And the OpenCOBOL CONCATENATE intrinsic

```
OCOBOL
MOVE FUNCTION CONCATENATE(filename; X"00") TO c-field.
```

[Roger] While points out: X"00" is almost always interchangeable with LOW-VALUE.

In all of the above snippets, the source code X"00" can be replaced by the COBOL noun **LOW-VALUE** or *LOW-VALUES*. Except when a program collating sequence is active and where the first character is not X"00".

With the CALL verb, use ADDRESS OF and/or BY REFERENCE

```
CALL "CFUNCTION" USING BY REFERENCE ADDRESS OF current-url.
```

The above being equivalent to char\*\* in C.

COBOL, by its nature, passes all arguments by reference. That can be overridden with the **BY VALUE** clause and the **BY CONTENT** clause.

## 29.32 5.32 Does OpenCOBOL support COPY includes?

Yes. COPY is fully supported, all variations from the standards up to and including the proposed 20xx standards.

Inline REPLACE text substitutions are also supported.

The **-I** compiler option influences the copybook search path and **-E** can be used to examine the *after* COPY preprocessor output.

There is also **-ffold-copy-upper** and **-ffold-copy-lower** compiler controls.

## 29.33 5.33 Does OpenCOBOL support WHEN-COMPILED?

Both as a noun and as an intrinsic function.

```
DISPLAY WHEN-COMPILED.  
DISPLAY FUNCTION WHEN-COMPILED.
```

```
07/05/0805.15.20  
2008070505152000-0400
```

Note: The noun WHEN-COMPILED is non-standard and was deemed obsolete in the pre 85 standard.

## 29.34 5.34 What is PI in OpenCOBOL?

With OpenCOBOL 1.1

```
DISPLAY FUNCTION PI.  
3.1415926535897932384626433832795029
```

```
DISPLAY FUNCTION E.  
2.7182818284590452353602874713526625
```

That's 34 digits after the decimal. Developers that need to know the tolerances for use in calculations are directed to poke around the freely available source code, and to read up on [GMP](#).

## 29.35 5.35 Does OpenCOBOL support the Object features of the 2002 standard?

Not yet. *July 2008*



## 29.36 5.36 Does OpenCOBOL implement PICTURE 78?

Yes. PICTURE 78 clauses can be used for constants, translated at compile time. This common non-standard extension is supported in OpenCOBOL.

## 29.37 5.37 Does OpenCOBOL implement CONSTANT?

Current OC 1.1 has preliminary support for a subset of the standard conforming “CONSTANT” phrase. eg

```
01 MYCONST CONSTANT AS 1.
```

Note: there is a syntax difference between 78 and CONSTANT.

## 29.38 5.38 What source formats are accepted by OpenCOBOL?

Both FIXED and FREE COBOL source formats are supported. FIXED format follows the 1-6, 7, 8-72 special columns of the COBOL standards. The compiler directives:

```
Column
12345678901234567890
    >>SOURCE FORMAT IS FREE
    >>SOURCE FORMAT IS FIXED
```

can be used. The directive must occur at column 8 or beyond if the ACTIVE scan format is FIXED. As per the 2002 standard this directive can be used to switch formats multiple times within a compilation unit.

Continuation indicators in column 7 are not applicable to FREE format and are not supported in this mode of translation. String catenation can always be used; the **&** operator.

The special *\*> till end of line* comment is supported in both FREE and FIXED forms, but by necessity will need to be placed at column 7 or greater in FIXED format sources.

The **-free** and **-fixed** options to **cobc** also influence the expected source formats, with the default being mandated by the standards as FIXED.

## 29.39 5.39 Does OpenCOBOL support continuation lines?

Yes. A dash - in column 7 can be used for continuation lines. But, by necessity continuation lines only apply in FIXED format source code. FREE format COBOL does not support continuation as there is no real meaning to *column 7* in FREE form source.

Note that in this example there is no terminating quote on the string continuations, but there is an extra starting quote following the dash

```
123456789012345678901234567890123456789012345678901234567890123456789012
  identification division.
  program-id. longcont.

  data division.
  working-storage section.
  01  longstr      pic X(80)
      value "This will all be one string in FIXED forma
-t source code".
```

```
01 otherstr      pic X(148) value "this
-"string will have spaces between the words THIS and STRING, as
-"continuation lines always fill to column 72.".
procedure division.
display longstr.
display length longstr.
display function length(function trim(longstr trailing)).
display otherstr(1:72).
display otherstr(73:75).
display length otherstr.
display function length(function trim(otherstr trailing)).
goback.
```

Compiled with:

```
$ cobc longcont.cob
$ cobcrun longcont
```

produces:

```
This will all be one string in FIXED format source code
80
00000055
this                string will have spaces between the words
THIS and STRING, as  continuation lines always fill to column 72.
148
00000139
```

*Note: The DISPLAY of **otherstr** was split to avoid any wide browser scrolling, not for any COBOL reasons.*

*Also note that the rules for continuation lines are quite difficult to describe simply and concerned OpenCOBOL programmers are urged to read through the standards documents for full details.*

## 29.40 5.40 Does OpenCOBOL support string concatenation?

Absolutely. Sources that need long strings, or those wishing to enhance source code readability, can use the **&** operator

```
identification division.
program-id. longstr.

data division.
working-storage section.
01 longstr      pic X(80)
               value "This " & "will " & "all " & "be " &
                   "one " &
                   "string " & "in both FIXED and FREE" &
                   " format source code".

procedure division.
display longstr.
goback.
```

Run this with

```
$ cobc longstr.cob
$ cobcrun longstr
This will all be one string in both FIXED and FREE format source code
$ cobc -free longstr.cob
```

```
$ cobcrun longstr
```

This will all be one string in both FIXED and FREE format `source` code

And for an Intrinsic FUNCTION unique to OpenCOBOL, see [FUNCTION CONCATENATE](#).

## 29.41 5.41 Does OpenCOBOL support D indicator debug lines?

Yes, in two forms. As for continuation lines, column 7 has no meaning for SOURCE FORMAT IS FREE source code so the standard **D** in column 7 can not be used. FORMAT FREE source code can use the `>>D` compiler directive instead. Use **D** lines as a conditional include of a source code line. These debug lines will only be compiled if the `-fdebugging-line` compiler switch is used.

From human on [opencobol.org](http://opencobol.org)

If you put a D in column 7 OC handles this as a comment. These lines are only compiled if you run cobc with `-fdebugging-line`.

By using this you can put some test messages etc. into your program that are only used if necessary (and therefore build with `-fdebugging-line`).

OpenCOBOL also supports a `>>D` debug compile time directive and a handy trick for those that like to write code that be compiled in both FIXED and FREE forms, is to place the directive in column 5, 6 and 7.

```
Column
12345678901234567890
      DISPLAY "Normal Line" END-DISPLAY
      >>DDISPLAY "Debug Line" END-DISPLAY
```

This allows use of the directive form in FORMAT FREE and also, with the **D** in column 7, will compile properly in FORMAT FIXED. In FORMAT FIXED the `>>` in columns 5 and 6 will be ignored as part of the *sequence number* field.

For more information on debugging support see [What about debugging OpenCOBOL programs?](#)

## 29.42 5.42 Does OpenCOBOL support mixed case source code?

Absolutely, kind of. Mixed case and mixed format, [ASCII](#) and [EBCDIC](#). Most COBOL compilers have not required uppercase only source code for quite a few years now. Still, most COBOL compilers including OpenCOBOL folds parts of the source to uppercase *with certain rules* before translating.

The compiler is case insensitive to names

```
000100 identification division.
000200 program-id. mixcase.
000300 data division.
000400 working-storage section.
000500 01 SOMEUPPER pic x(9).
000600 01 SomeUpper pic x(9).
000700 01 someupper pic x(9).
000800
000900 procedure division.
001000 move "SOMEUPPER" to SOMEUPPER.
001100 move "SomeUpper" to SomeUpper.
001200 move "someupper" to someupper.
001300 display "SOMEUPPER: " SOMEUPPER end-display.
```

```
001400 display "SomeUpper: " SomeUpper end-display.
001500 display "someupper: " someupper end-display.
001600 stop run.
```

Attempted compile with:

```
$ cobc -x mixcase.cob
```

produces:

```
mixcase.cob:10: Error: 'SOMEUPPER' ambiguous; need qualification
mixcase.cob:5: Error: 'SOMEUPPER' defined here
mixcase.cob:6: Error: 'SOMEUPPER' defined here
mixcase.cob:7: Error: 'SOMEUPPER' defined here
```

Note; that although the folded declarations conflict, the DISPLAY quoted strings will NOT be folded, and would display as expected.

*Case sensitivity is also at the mercy of operating system conventions.* Under GNU/Linux, OpenCOBOL's dynamic link loader is case sensitive.

```
CALL "C$JUSTIFY" USING center-string "C" END-CALL.
```

is not the same as

```
CALL "c$justify" USING center-string "C" END-CALL.
```

In support of case folding and COPY libraries, OpenCOBOL supports *-ffold-copy-lower* and *-ffold-copy-upper*. For mixing and matching legacy sources.

**Trivia** The expressions *uppercase* and *lowercase* date back to early moveable type. Typographers would keep two cases of metal casted letters, Capitalized and normal. Usually set on stacked shelves over the workbench. The small letters, being used more frequently, ended up on the lower shelf; the lower case letters.

## 29.43 5.43 What is the shortest OpenCOBOL program?

All that is needed is a program-id. Doesn't do much.

```
program-id. a.
```

Update: It turns out that an empty file is the shortest OpenCOBOL that will do nothing. From [Roger](#)

```
$ ls -l empty.cob
-rw-r--r-- 1 root root 0 Jun 21 12:35 empty.cob

$ cobc -x -frelax-syntax empty.cob
empty.cob: 1: Warning: PROGRAM-ID header missing - assumed

$ ./empty
$
```

(Alternate to *-frelax-syntax* is *-std=mf*)

## 29.44 5.44 What is the shortest Hello World program in OpenCOBOL?

A short version of OpenCOBOL hello world, compiled -free

```
program-id.hello.procedure division.display "Hello World!".
```

Thanks to human and the [opencobol.org](http://opencobol.org) forums.

*Please note:* This is **not good** COBOL form, and is only shown as an example of the possibilities.

Update: From [Roger](#) the shortest hello world program can be

```
$ cat hello.cob
display"Hello World!".
```

```
$ cobc -x -frelax-syntax -free hello.cob
hello.cob: 1: Warning: PROGRAM-ID header missing - assumed
hello.cob: 1: Warning: PROCEDURE DIVISION header missing - assumed
```

```
$ ./hello
Hello World!
$
```

So, that means, **display"Hello World!".** is all you need, if you compile with *relax-syntax*.

## 29.45 5.45 How do I get those nifty sequential sequence numbers in a source file?

FIXED format COBOL uses the first 6 positions of each line as a programmer defined **sequence** field. This field is stripped as part of the preprocessing and is not validated. Historically, the sequence numbers were used to verify that card punch cards were read into a card reader in the proper order. Many legacy COBOL programs have sequentially numbered sequence values. Here is a little **vi** trick to renumber the sequence field by 100s.

Given

```
000005* HELLO.COB OpenCOBOL FAQ example
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. hello.
000030 PROCEDURE DIVISION.
000040 DISPLAY "Hello World!".
000100 STOP RUN.
```

Running the following **ex** filter

```
:%!perl -ne 'printf("%06d%s\n", $, * 100, substr($_, 6, -1));'
```

produces a nicely resequenced source file.

```
000100* HELLO.COB OpenCOBOL FAQ example
000200 IDENTIFICATION DIVISION.
000300 PROGRAM-ID. hello.
000400 PROCEDURE DIVISION.
000500 DISPLAY "Hello World!".
000600 STOP RUN.
```

- Note: Only use this on already FIXED form source. If used on any FREE format COBOL, the first 6 columns will be damaged.

This has no effect on the compilation process, it only effects the appearance of the sources.

---

**Note:** Be careful not to confuse SEQUENCE NUMBERS with source code LINE NUMBERS. They are not the same.

---

- Vim: For users of the Vim editor, the command

```
:set number
```

will display the number of each source line. Many editors support the display of line numbers. Even

```
$ less -N
```

can be used to display line numbers of its input.

## 29.46 5.46 Is there a way to count trailing spaces in data fields using OpenCOBOL?

Yes. Quite a few. But instead of resorting to a PERFORM VARYING sequence try

```
01 B-COUNT                PIC 999 VALUE 0.
01 TEST-CASE              PIC X(80)
   VALUE "This is my string."
```

ONE-WAY.

```
INSPECT FUNCTION REVERSE (TEST-CASE)
  TALLYING B-COUNT
  FOR LEADING ' '.
DISPLAY B-COUNT.
```

TWO-WAY.

```
INSPECT TEST-CASE
  TALLYING B-COUNT
  FOR TRAILING SPACE.
DISPLAY B-COUNT.
```

THREE-WAY.

```
IF TEST-CASE EQUAL SPACES
  COMPUTE B-COUNT = LENGTH OF TEST-CASE
ELSE
  COMPUTE
    B-COUNT = LENGTH TEST-CASE -
      FUNCTION LENGTH (FUNCTION TRIM (TEST-CASE TRAILING))
  END-COMPUTE
END-IF
DISPLAY B-COUNT.
```

produces:

```
062
124
062
```

The second value is 124 as TWO-WAY accumulates another 62 after ONE-WAY. The INSPECT verb does not initialize a TALLYING variable.

Information modified from [opencobol.org](http://opencobol.org) forum post.

## 29.47 5.47 Is there a way to left justify an edited numeric field?

Yes, a couple of ways.

Assuming a working storage of

```
01 mynumber PIC 9(8) VALUE 123.
01 myedit PIC Z(7)9.
01 mychars PIC X(8).

01 spcount PIC 99 USAGE COMPUTATIONAL.

MOVE mynumber TO myedit
MOVE myedit TO mychars
DISPLAY mynumber END-DISPLAY
DISPLAY myedit END-DISPLAY
```

```
00000123
      123
```

With OpenCOBOL, the intrinsic

```
FUNCTION TRIM(myedit LEADING)
```

will trim leading whitespace. The LEADING is not really necessary as TRIM removes both leading and trailing whitespace.

OpenCOBOL also ships with a library function for justification of strings

```
CALL "C$JUSTIFY" USING mychars "L" END-CALL
```

to left justify an alphanumeric field. “R” for right, or “C” for centre.

But a generic idiom that should work across all capable COBOL systems

```
MOVE 0 TO spcount
INSPECT myedit TALLYING spcount FOR LEADING SPACE
MOVE myedit(spcount + 1:) TO mychars

DISPLAY myedit END-DISPLAY
DISPLAY mychars END-DISPLAY
```

```
      123
123
```

```
MOVE 0 TO spcount
INSPECT mynumber TALLYING spcount FOR LEADING ZERO
DISPLAY mynumber
DISPLAY mynumber(spcount + 1:)
```

Uses the INSPECT verb to count leading spaces, then reference modification to move the characters one past the spaces till the end of the edit field to an alpha field.

## 29.48 5.48 Is there a way to determine when OpenCOBOL is running ASCII or EBCDIC?

OpenCOBOL supports both ASCII and EBCDIC character encodings. A simple test such as

```
01 MYSAPCE    PIC X VALUE X"20".
   88 MYISASCII VALUE SPACE.

IF MYISASCII
  DISPLAY "I'm ASCII" END-DISPLAY
END-IF
```

can be used to determine the character set at run-time.

## 29.49 5.49 Is there a way to determine when OpenCOBOL is running on 32 or 64 bits?

OpenCOBOL builds and supports both 32 and 64 bit architectures. A simple test such as

```
01 MYPOINTER  USAGE POINTER.

IF FUNCTION LENGTH(MYPOINTER) EQUALS 8
  DISPLAY "This is a 64 bit machine" END-DISPLAY
END-IF
```

can be used to determine the native bit size at run-time.

## 29.50 5.50 Does OpenCOBOL support recursion?

Yes. Not completely to standard currently (*February 2013*), as there are no restrictions on calling programs in a recursive manner, but yes.

A made up example using a factorial called program

```
OCOBOL*> *****
*> Author:    Brian Tiffin
*> Date:     29-Dec-2008
*> Purpose:   Horsing around with recursion
*> Tectonics: cobc -x recurse.cob
*> *****
identification division.
program-id. recurse.

data division.
working-storage section.
78 n          value 4.
01 fact usage binary-long.

*> *****
procedure division.

call "factorial" using by value n returning fact end-call
display n "!" = " fact end-display

goback.
end program recurse.
*> *****
*> *****
```



```

*> *****
identification division.
program-id. factorial is recursive.

data division.
local-storage section.
01 result usage is binary-long.

linkage section.
01 num usage is binary-long.

*> *****
procedure division using by value num.

display "num: " num end-display
if num equal zero
    move 1 to return-code
    display "ret: " return-code end-display
    goback
end-if

subtract 1 from num end-subtract
call "factorial" using by value num returning result end-call
compute return-code = (num + 1) * result end-compute
display "ret: " return-code end-display
goback.

end program factorial.

```

Produces:

```

num: +0000000004
num: +0000000003
num: +0000000002
num: +0000000001
num: +0000000000
ret: +0000000001
ret: +0000000001
ret: +0000000002
ret: +0000000006
ret: +0000000024
4! = +0000000024

```

Of course the *Intrinsic FUNCTION FACTORIAL* might be a more efficient and much easier way at getting factorials.

## 29.51 5.51 Does OpenCOBOL capture arithmetic overflow?

Yes. Here is one sample using *ADD* with *ON SIZE ERROR*.

And please note that *OVERFLOW* is a conditional for *STRING*. In COBOL, what this author terms 'overflow' is less technically correct than 'size error' when using COBOL arithmetic terminology.

```

OCOBOL*> *****
*> Author:    Brian Tiffin
*> Date:      04-Feb-2009
*> Purpose:   Factorial and overflow
*> Tectonics: ccbc -x overflowing.cob

```

```

*> *****
identification division.
program-id. overflowing.

data division.
working-storage section.
01 fact usage binary-long.
01 answer usage binary-double.

*> *****
procedure division.
00-main.

perform
  varying fact from 1 by 1
  until fact > 21
    add function factorial(fact) to zero giving answer
    on size error
      display
        "overflow at: " fact " is " answer
        " without test " function factorial(fact)
      end-display
    not on size error
      display fact ": " answer end-display
    end-add
  end-perform
.

00-leave.
goback.

end program overflowing.
*> *****

```

which outputs:

```

+0000000001: +00000000000000000001
+0000000002: +00000000000000000002
+0000000003: +00000000000000000006
+0000000004: +00000000000000000024
+0000000005: +00000000000000000120
+0000000006: +00000000000000000720
+0000000007: +00000000000000005040
+0000000008: +0000000000000040320
+0000000009: +0000000000000362880
+0000000010: +0000000000000362880
+0000000011: +00000000000039916800
+0000000012: +00000000000479001600
+0000000013: +00000000006227020800
+0000000014: +00000000087178291200
+0000000015: +00000001307674368000
+0000000016: +00000020922789888000
+0000000017: +00000355687428096000
+0000000018: +00006402373705728000
+0000000019: +00121645100408832000
overflow at: +0000000020 is +00121645100408832000 without test 432902008176640000
overflow at: +0000000021 is +00121645100408832000 without test 197454024290336768

```

## 29.52 5.52 Can OpenCOBOL be used for plotting?

Yes? One way is with an external call to *gnuplot*.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*****
* Author:      Brian Tiffin
* Date:        29-July-2008
* Purpose:     Plot trig and a random income/expense/worth report
* Tectonics:   requires access to gnuplot. http://www.gnuplot.info
*              cobc -Wall -x plotworth.cob
* OVERWRITES  ocgenplot.gp  ocgpdata.txt  sincos.png  ploworth.png
*****
identification division.
program-id. plotworth.

environment division.
input-output section.
file-control.
    select scriptfile
        assign to "ocgenplot.gp"
        organization is line sequential.
    select outfile
        assign to "ocgpdata.txt"
        organization is line sequential.
    select moneyfile
        assign to "ocgpdata.txt"
        organization is line sequential.

data division.
file section.
fd scriptfile.
    01 gnuplot-command pic x(82).
fd outfile.
    01 outrec.
        03 x-value      pic -zzzzzz9.99.
        03 filler       pic x.
        03 sin-value    pic -zzzz9.9999.
        03 filler       pic x.
        03 cos-value    pic -zzzz9.9999.
fd moneyfile.
    01 moneyrec.
        03 timefield    pic 9(8).
        03 filler       pic x.
        03 income       pic -zzzzzz9.99.
        03 filler       pic x.
        03 expense      pic -zzzzzz9.99.
        03 filler       pic x.
        03 networth     pic -zzzzzz9.99.

working-storage section.
01 angle      pic s9(7)v99.

01 dates      pic 9(8).
01 days       pic s9(9).
01 worth      pic s9(9).
01 amount     pic s9(9).

```

```
01 gplot  pic x(80) value is 'gnuplot -persist ocgenplot.gp'.
01 result pic s9(9).

procedure division.

* Create the script to plot sin and cos
open output scriptfile.
move "plot 'ocgpdata.txt' using 1:2 with lines title 'sin(x)'"
- to gnuplot-command.
write gnuplot-command.
move "replot 'ocgpdata.txt' using 1:3 with lines title 'cos(x)'"
- to gnuplot-command.
write gnuplot-command.
move "set terminal png; set output 'sincos.png'; replot"
- to gnuplot-command.
write gnuplot-command.
close scriptfile.

* Create the sinoidal data
open output outfile.
move spaces to outrec.
perform varying angle from -10 by 0.01
  until angle > 10
    move angle to x-value
    move function sin(angle) to sin-value
    move function cos(angle) to cos-value
    write outrec
end-perform.
close outfile.

* Invoke gnuplot
call "SYSTEM" using gplot
  returning result.
if result not = 0
  display "Problem: " result
  stop run returning result
end-if.

* Generate script to plot the random networkh
open output scriptfile.
move "set xdata time" to gnuplot-command.
write gnuplot-command.
move 'set timefmt "%Y%m%d"' to gnuplot-command.
write gnuplot-command.
move 'set format x "%m"' to gnuplot-command.
write gnuplot-command.
move 'set title "Income and expenses"' to gnuplot-command.
write gnuplot-command.
move 'set xlabel "2008 / 2009"' to gnuplot-command.
write gnuplot-command.
move 'plot "ocgpdata.txt" using 1:2 with boxes title "Income"
-' linecolor rgb "green"' to gnuplot-command.
write gnuplot-command.
move 'replot "ocgpdata.txt" using 1:3 with boxes title "Expense"
-' linecolor rgb "red"' to gnuplot-command.
write gnuplot-command.
move 'replot "ocgpdata.txt" using 1:4 with lines title "Worth"
- to gnuplot-command.
```

```

write gnuplot-command.
move 'set terminal png; set output "plotworth.png"; replot'
-   to gnuplot-command.
write gnuplot-command.
close scriptfile.

* Generate a bi-weekly dataset with date, income, expense, worth
open output moneyfile.
move spaces to moneyrec.
move function integer-of-date(20080601) to dates.
move function random(0) to amount.

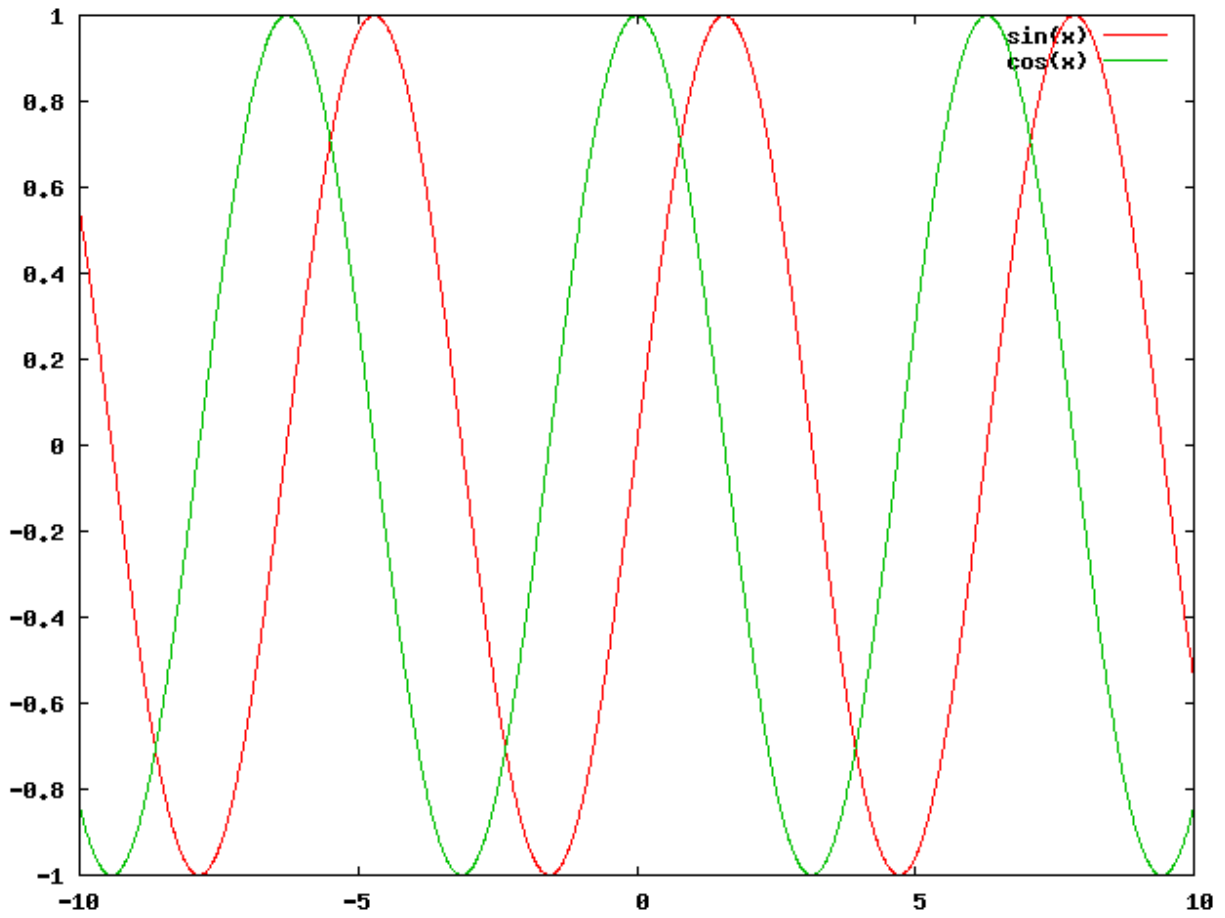
perform varying days from dates by 14
  until days > dates + 365
    move function date-of-integer(days) to timefield
    compute amount = function random() * 2000
    compute worth = worth + amount
    move amount to income
    compute amount = function random() * 1800
    compute worth = worth - amount
    move amount to expense
    move worth to networth
    write moneyrec
  end-perform.
close moneyfile.

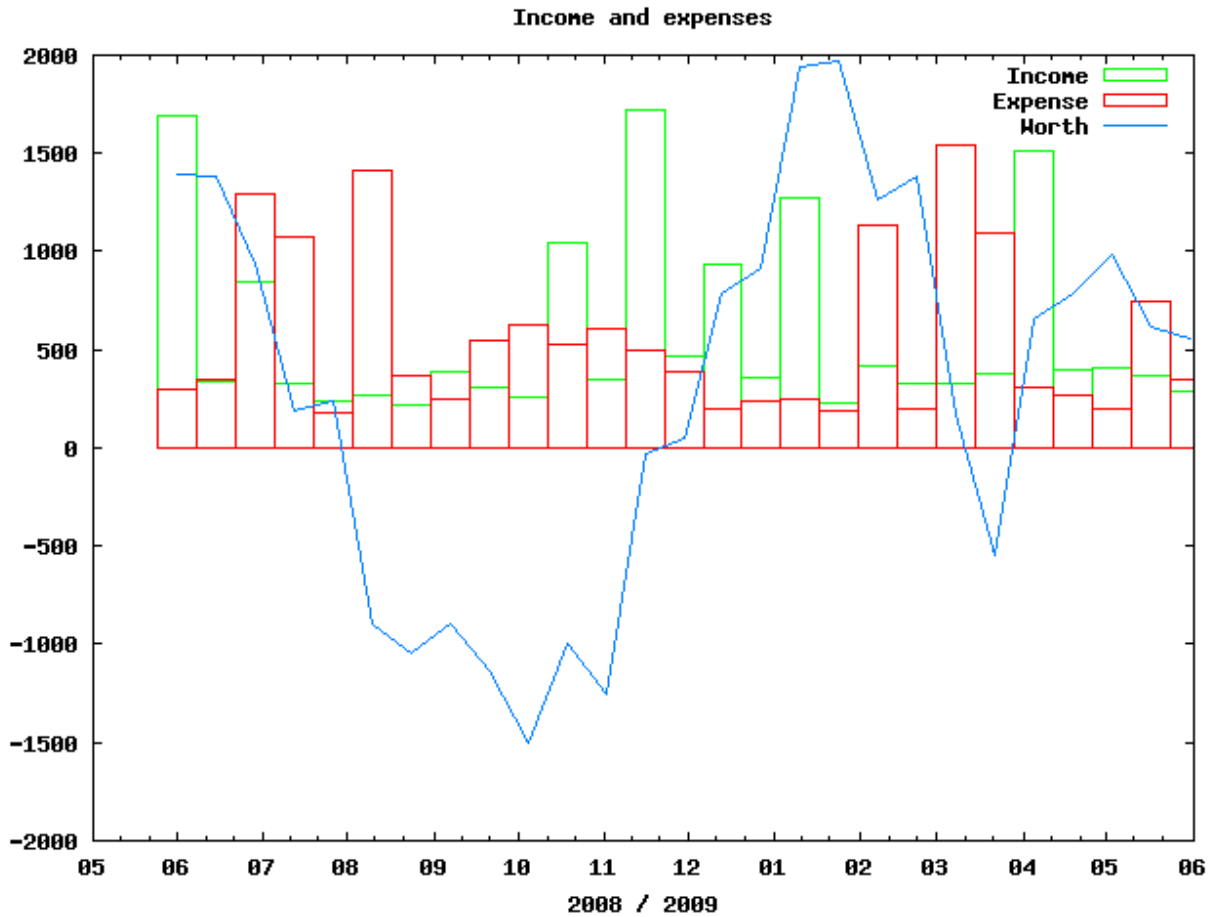
* Invoke gnuplot again. Will open new window.
call "SYSTEM" using gplot
  returning result.
if result not = 0
  display "Problem: " result
  stop run returning result
end-if.

goback.

```

Which displays and saves:





## 29.53 5.53 Does OpenCOBOL support the GIMP ToolKit, GTK+?

Yes. A binding for GTK+ is in the works. Early samples have proven workable and screenshots of OpenCOBOL GUI screens are shown here.

### What does GIMP stand for?

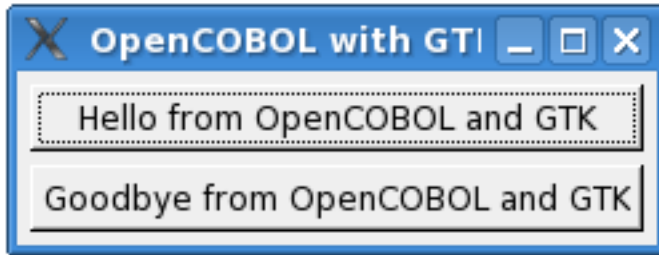
**GIMP** is an acronym for the *GNU Image Manipulation Program*, a very complete and robust graphic design tool. See the [GIMP](#) site for more information.

**GTK+** is the GIMP ToolKit. See the [GTK](#) site for more information.

Simple buttons

```
Hello from GTK in OpenCOBOL at 2008120111495787-0500
Hello from GTK in OpenCOBOL at 2008120111500044-0500
```

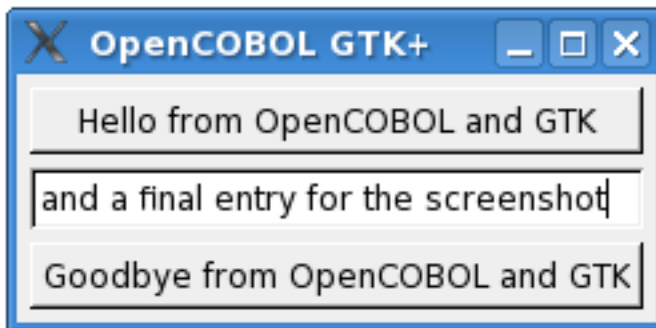
□



Text entry widget

```
Hello from GTK in OpenCOBOL at 2008120312472750-0500
text: first entry , +0000000011
text: first entry - edited , +0000000021
text: then a clear , +0000000012
text: , +0000000000
text: and a final entry for the screen, +0000000032
```

□



Sample OpenCOBOL that generated the above

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author: Brian Tiffin
*> Date: 03-Dec-2008
*> Purpose: Hello from GTK+
*> Requires: libgtk2.0, libgtk2.0-dev, gtk2.0, pkg-config
*> Tectonics:
*> cobc -c 'pkg-config --cflags gtk+-2.0' ocgtk.c
*> cobc -x 'pkg-config --libs gtk+-2.0' gtkhello.cob ocgtk.o
*> *****
identification division.
program-id. gtkhello.

data division.

working-storage section.
01 result usage binary-long.
01 gtk-window usage pointer.
01 gtk-box usage pointer.
01 gtk-hello usage pointer.
01 gtk-textentry usage pointer.
```



```

01 gtk-goodbye          usage pointer.

01 callback             usage procedure-pointer.
01 params              usage pointer.

*> *****
procedure division.

*> Initialize GTK
CALL "CBL_OC_GTK_INIT_CHECK" returning result END-CALL
>>D display "init: " result end-display

*> Create a toplevel window
CALL "CBL_OC_GTK_WINDOW_NEW" returning gtk-window END-CALL
>>D display "win: " gtk-window end-display

*> Set the titlebar - using cob_field now **HERE**
CALL "CBL_OC_GTK_WINDOW_SET_TITLE"
using by value gtk-window
by reference "OpenCOBOL GTK+"
END-CALL
>>D display "title: " gtk-window end-display

*> Set the border width
CALL "CBL_OC_GTK_CONTAINER_SET_BORDER_WIDTH"
using by value gtk-window
by value 5
END-CALL
>>D display "border: " gtk-window end-display

*> connect a window destroy, quit main loop handler
set callback to entry "CBL_OC_destroy"
CALL "CBL_OC_G_SIGNAL_CONNECT"
using by value gtk-window
by reference "delete_event" & x"00"
by value callback
by value params
END-CALL

*> Create a vertically packed box
CALL "CBL_OC_GTK_VBOX_NEW"
using by value 0
by value 5
returning gtk-box
END-CALL
>>D display "box: " gtk-box end-display

*> Add the box to the window
CALL "CBL_OC_GTK_CONTAINER_ADD"
using by value gtk-window
by value gtk-box
END-CALL

*> Create the hello button
CALL "CBL_OC_GTK_BUTTON_NEW_WITH_LABEL"
using by reference "Hello from OpenCOBOL and GTK" & x"00"
returning gtk-hello
END-CALL

```

```
>>D   display "button: " gtk-hello end-display

*> Connect the hello button to the hello code
set callback to entry "CBL_OC_hello"
CALL "CBL_OC_G_SIGNAL_CONNECT"
    using by value gtk-hello
        by reference "clicked" & x"00"
        by value callback
        by value params
END-CALL

*> Pack the button into the box, top to bottom
CALL "CBL_OC_GTK_BOX_PACK_START"
    using by value gtk-box
        by value gtk-hello
        by value 1
        by value 1
        by value 0
END-CALL

*> button is ready to show
CALL "CBL_OC_GTK_WIDGET_SHOW"
    using by value gtk-hello
END-CALL

*> Add a text entry field
CALL "CBL_OC_GTK_ENTRY_NEW"
    returning gtk-textentry
END-CALL

*> Connect code to the text entry, passing the entry widget
set callback to entry "CBL_OC_activate"
CALL "CBL_OC_G_SIGNAL_CONNECT"
    using by value gtk-textentry
        by reference "activate" & x"00"
        by value callback
        by value gtk-textentry
END-CALL

*> Pack the text field into the box, top to bottom
CALL "CBL_OC_GTK_BOX_PACK_START"
    using by value gtk-box
        by value gtk-textentry
        by value 1
        by value 1
        by value 0
END-CALL

*> text field is ready to show
CALL "CBL_OC_GTK_WIDGET_SHOW"
    using by value gtk-textentry
END-CALL

*> Create the bye button
CALL "CBL_OC_GTK_BUTTON_NEW_WITH_LABEL"
    using by reference "Goodbye from OpenCOBOL and GTK" & x"00"
    returning gtk-goodbye
END-CALL
```

```

>>D   display "button: " gtk-goodbye end-display

*> Connect the bye button to the bye code
set callback to entry "CBL_OC_destroy"
CALL "CBL_OC_G_SIGNAL_CONNECT"
    using by value gtk-goodbye
        by reference "clicked" & x"00"
        by value callback
        by value params
END-CALL

*> Pack the button into the box, under hello
CALL "CBL_OC_GTK_BOX_PACK_START"
    using by value gtk-box
        by value gtk-goodbye
        by value 1
        by value 1
        by value 0
END-CALL
>>D   display "pack: " gtk-box end-display

*> button is ready to show
CALL "CBL_OC_GTK_WIDGET_SHOW"
    using by value gtk-goodbye
END-CALL

*> box is ready to show
CALL "CBL_OC_GTK_WIDGET_SHOW"
    using by value gtk-box
END-CALL

*> window is ready to show
CALL "CBL_OC_GTK_WIDGET_SHOW"
    using by value gtk-window
END-CALL

*> Start up the event loop, control returned when GTK main exits
CALL "CBL_OC_GTK_MAIN" END-CALL

*> Something terminated the GTK main loop, sys-close or bye or
display "ending..." end-display

goback.
end program gtkhello.
*> *****

*> **** window shutdown callback *****
identification division.
program-id. CBL_OC_destroy.
data division.
linkage section.
01 gtk-window          usage pointer.
01 gtk-data            usage pointer.

procedure division using by value gtk-window by value gtk-data.

CALL "CBL_OC_GTK_MAIN_QUIT" END-CALL

```

```
goback.
end program CBL_OC_destroy.
*> *****

*> **** hello button click callback *****
identification division.
program-id. CBL_OC_hello.
data division.
linkage section.
01 gtk-window          usage pointer.
01 gtk-data            usage pointer.

procedure division using by value gtk-window by value gtk-data.
display
    "Hello from GTK in OpenCOBOL at "
    function current-date
end-display

goback.
end program CBL_OC_hello.

*> **** text entry activation callback *****
*> This procedure called from GTK on enter key pressed in entry
identification division.
program-id. CBL_OC_activate.
data division.
working-storage section.
01 textfield          pic x(32).
01 textlen            usage binary-long.

linkage section.
01 gtk-window          usage pointer.
01 gtk-data            usage pointer.

procedure division using by value gtk-window by value gtk-data.

CALL "CBL_OC_GTK_ENTRY_GET_TEXT"
    using by value gtk-data
    textfield
    returning textlen
END-CALL
display "text: " textfield ", " textlen end-display

goback.
end program CBL_OC_activate.
```

Using this very early thin wrapper to GTK+

```
/* OpenCOBOL GTK+ 2.0 wrapper */
/* Tectonics: ccbc -c 'pkg-config --cflags gtk+-2.0' ocgtk.c */

#include <memory.h>
#include <stdlib.h>
#include <libcob.h>

#include <gtk/gtk.h>
#include <glib.h>
```

```

#include "ocgtk.h"

/* Initialize the toolkit, abends if not possible */
int
CBL_OC_GTK_INIT(int argc, char *argv[])
{
    gtk_init(&argc, &argv);
    return 0;
}

/* Initialize the toolkit, return false if not possible */
/* Need pointers to argc and argv here */
int
CBL_OC_GTK_INIT_CHECK()
{
    gboolean gres = gtk_init_check(0, NULL);
    return (gres == TRUE) ? 0 : -1;
}

/* Create new window */
GtkWidget*
CBL_OC_GTK_WINDOW_NEW()
{
    return gtk_window_new(GTK_WINDOW_TOPLEVEL);
}

/* set the title */
int
CBL_OC_GTK_WINDOW_SET_TITLE(void *window, char *title)
{
    struct cob_module *module;
    cob_field *title_field;
    char *cstr;

    /* Error conditions simply return, doing nothing */
    if (cob_get_global_ptr()->cob_call_params < 2) { return 1; }

    module = cob_get_global_ptr()->cob_current_module;
    if (module == NULL) {
        //cob_runtime_error("No module!");
        //cob_stop_run(1);
        return 1;
    }

    title_field = module->cob_procedure_parameters[1];
    if (!title_field) { return 1; }

    cstr = (char *)malloc(title_field->size + 1);
    if (!cstr) { return 1; }

    memcpy(cstr, title_field->data, title_field->size);
    cstr[title_field->size] = '\0';

    gtk_window_set_title(GTK_WINDOW(window), cstr);

    free(cstr);
    return 0;
}

```

```
/* Widget sizing */
int
CBL_OC_GTK_WIDGET_SET_SIZE_REQUEST(void *widget, int x, int y)
{
    gtk_widget_set_size_request(GTK_WIDGET(widget), x, y);
    return 0;
}

/* Set border width */
int
CBL_OC_GTK_CONTAINER_SET_BORDER_WIDTH(void *window, int pixels)
{
    gtk_container_set_border_width(GTK_CONTAINER(window), pixels);
    return 0;
}

/* New vertical box */
GtkWidget*
CBL_OC_GTK_VBOX_NEW(int homogeneous, int spacing)
{
    return gtk_vbox_new((gboolean)homogeneous, (gint)spacing);
}

/* New horizontal box */
GtkWidget*
CBL_OC_GTK_HBOX_NEW(int homogeneous, int spacing)
{
    return gtk_hbox_new((gboolean)homogeneous, (gint)spacing);
}

/* packing boxes */
int
CBL_OC_GTK_BOX_PACK_START(void *gcont, void *gobj, int expand, int fill, int padding)
{
    gtk_box_pack_start(GTK_BOX(gcont), gobj, (gboolean)expand, (gboolean)fill, (gint)padding);
    return 0;
}

/* menus */
GtkWidget*
CBL_OC_GTK_MENU_BAR_NEW()
{
    return gtk_menu_bar_new();
}

GtkWidget*
CBL_OC_GTK_MENU_NEW()
{
    return gtk_menu_new();
}

GtkWidget*
CBL_OC_GTK_MENU_ITEM_NEW_WITH_LABEL(char *label)
{
    struct cob_module *module;
    cob_field *title_field;
    char *cstr;
    GtkWidget *item;
```

```

/* Error conditions simply return, doing nothing */
if (cob_get_global_ptr()->cob_call_params < 1) { return NULL; }

module = cob_get_global_ptr()->cob_current_module;
if (module == NULL) {
    //cob_runtime_error("No module!");
    cob_stop_run(1);
}

title_field = module->cob_procedure_parameters[0];
if (!title_field) { return NULL; }

cstr = (char *)malloc(title_field->size + 1);
if (!cstr) { return NULL; }

memcpy(cstr, title_field->data, title_field->size);
cstr[title_field->size] = '\0';

item = gtk_menu_item_new_with_label(cstr);
gtk_widget_set_tooltip_text(item, (gchar *)cstr);

free(cstr);

return item;
}

int
CBL_OC_GTK_MENU_ITEM_SET_SUBMENU(void *item, void *menu)
{
    gtk_menu_item_set_submenu(GTK_MENU_ITEM(item), menu);
    return 0;
}

int
CBL_OC_GTK_MENU_SHELL_APPEND(void *menu, void *item)
{
    gtk_menu_shell_append(GTK_MENU_SHELL(menu), item);
    return 0;
}

/* New button */
GtkWidget*
CBL_OC_GTK_BUTTON_NEW_WITH_LABEL(char *label)
{
    GtkWidget *button;
    button = gtk_button_new_with_label(label);
    if (button) {
        gtk_widget_set_tooltip_text(button, (gchar *)label);
    }
    return button;
}

/* New text entry */
GtkWidget*
CBL_OC_GTK_ENTRY_NEW() {
    return gtk_entry_new();
}

```

```
/* Set text in entry */
int
CBL_OC_GTK_ENTRY_SET_TEXT(void *entry, char *text)
{
    gtk_entry_set_text(GTK_ENTRY(entry), text);
    return 0;
}

/* Get the text in an entry */
int
CBL_OC_GTK_ENTRY_GET_TEXT(void *entry, char *text)
{
    struct cob_module *module;
    cob_field *text_field;
    size_t    text_length;

    module = cob_get_global_ptr()->cob_current_module;
    text_field = module->cob_procedure_parameters[1];

    const gchar *entry_text;
    entry_text = gtk_entry_get_text(GTK_ENTRY(entry));
    text_length = entry_text ? strlen(entry_text) : 0;
    text_length = (text_length > text_field->size) ? text_field->size : text_length;

    memset(text_field->data, ' ', text_field->size);
    memcpy(text_field->data, entry_text, text_length);
    return (int)text_length;
}

/* connect event to callback */
int
CBL_OC_G_SIGNAL_CONNECT(int *gobj, char *sgn, void (cb)(void *, void *), void *parm)
{
    g_signal_connect(G_OBJECT(gobj), sgn, G_CALLBACK(cb), parm);
    return 0;
}

/* add object to container */
int
CBL_OC_GTK_CONTAINER_ADD(void *window, void *gobj)
{
    gtk_container_add(GTK_CONTAINER(window), gobj);
    return 0;
}

/* tell gtk that object is now ready */
int
CBL_OC_GTK_WIDGET_SHOW(void *gobj)
{
    gtk_widget_show(gobj);
    return 0;
}

/* tell gtk to ready all the widgets */
int
CBL_OC_GTK_WIDGET_SHOW_ALL(void *window)
{

```



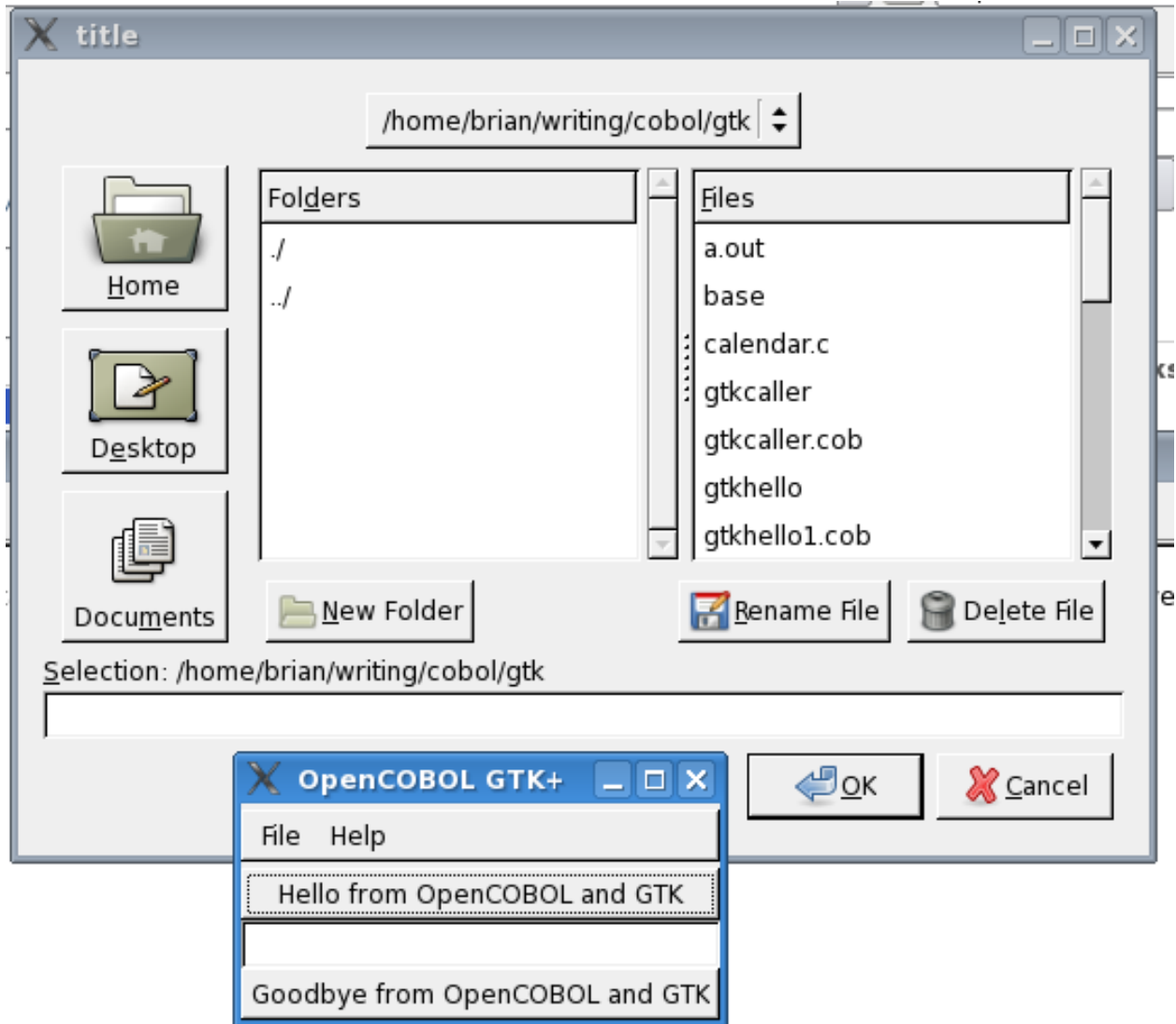
```
    gtk_widget_show_all(window);
    return 0;
}

/* Some dialogs */
GtkWidget*
CBL_OC_GTK_FILE_SELECTION_NEW(char *title)
{
    return gtk_file_selection_new(title);
}

/* the event loop */
int
CBL_OC_GTK_MAIN()
{
    gtk_main();
    return 0;
}

/* stop the gui */
int
CBL_OC_GTK_MAIN_QUIT()
{
    gtk_main_quit();
    return 0;
}
/**/
```

A screenshot with added menu and file dialog after hitting File -> Open



### 29.53.1 5.53.1 A web browsing widget embedded in OpenCOBOL?

Yep.

A short sample, made for OpenCOBOL 1.0's first birthday, Dec 27th, 2008.

```
int
CBL_OC_GTKHTML (char *html_string)
{
    GtkWidget *app;
    GtkWidget *html;
    GtkWidget *scrolled_window;

    char *fakeargv[2] = {"happybday", ""};

    /* prepare our environment, we need gnome and gconf */
    gnome_init ("Example_1", "1.0", 1, fakeargv);
    gconf_init (1, fakeargv);
}
```

```

/* create GtkHTML widget */
html = gtk_html_new ();
gtk_signal_connect (GTK_OBJECT (html), "url_requested",
                   GTK_SIGNAL_FUNC (url_requested), NULL);
gtk_signal_connect (GTK_OBJECT (html), "object_requested",
                   GTK_SIGNAL_FUNC (object_requested), NULL);

gtk_html_load_from_string (GTK_HTML (html), html_string, -1);

/* create GNOME app and put GtkHTML in scrolled window in it */
app = gnome_app_new ("Example_1", "Happy Birthday OpenCOBOL");

scrolled_window = gtk_scrolled_window_new (NULL, NULL);
gtk_scrolled_window_set_policy (GTK_SCROLLED_WINDOW (scrolled_window),
                               GTK_POLICY_AUTOMATIC, GTK_POLICY_AUTOMATIC);
gtk_container_add (GTK_CONTAINER (scrolled_window), html);

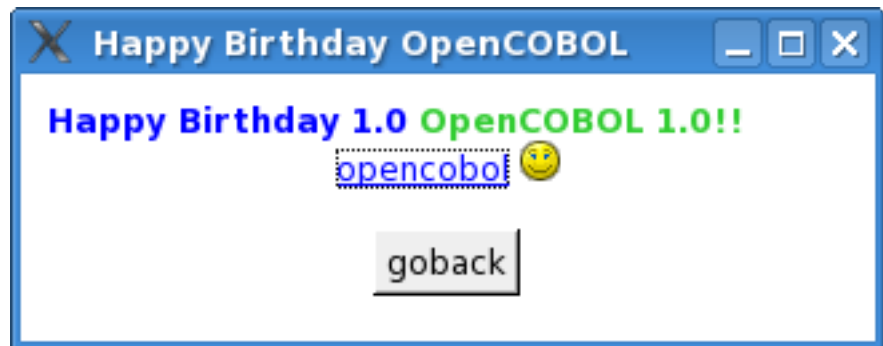
gnome_app_set_contents (GNOME_APP (app), scrolled_window);
gtk_window_set_default_size (GTK_WINDOW (app), 320, 100);
gtk_widget_show_all (app);

/* run the main loop */
gtk_main ();

return 0;
}
/**/

```

displays



when called with this COBOL.

```

*> *****
*> Author:    Brian Tiffin
*> Date:      27-Dec-2008
*> Purpose:   Happy Birthday OpenCOBOL
*> Tectonics:
*> gcc -c `pkg-config --cflags --libs libgnome-2.0 libgnomeui-2.0
*>         gtk+-2.0 libgtkhtml-3.14` hellogtk.c
*> cobc -lgtkhtml-3.14 -lgnomeui-2 -lSM -lICE -lglade-2.0
*>         -lbonoboui-2 -lgnomevfs-2 -lgnomecanvas-2 -lgnome-2 -lpopt
*>         -lbonobo-2 -lbonobo-activation -lORBit-2 -lart_lgpl_2
*>         -lgconf-2 -lgthread-2.0 -lrt -lgtk-x11-2.0 -lxml2
*>         -lgdk-x11-2.0 -latk-1.0 -lgdk_pixbuf-2.0 -lm
*>         -lpangocairo-1.0 -lpango-1.0 -lcairo -lgobject-2.0
*>         -lgmodule-2.0 -ldl -lglib-2.0 -x ocgtkhtml.cob hellogtk.o

```

```
*> *****
identification division.
program-id.  ocgtkhtml.

data division.
working-storage section.
01 result          usage binary-long.
01 html-string     pic x(512) value
    "<B><FONT COLOR=Blue>Happy Birthday 1.0</FONT> " &
    "<FONT COLOR=LimeGreen>OpenCOBOL 1.0!!</FONT></B><br />" &
    "<div align='center'><a href='http://opencobol.org'>" &
    "opencobol</a> <img src='file:smiley.png' />" &
    "<br /><br /><OBJECT CLASSID=close_button>Closebutton" &
    "</OBJECT></div>" & x"00".

*> *****
procedure division.

call "CBL_OC_GTKHTML" using
    by reference html-string
    returning result
end-call

goback.
end program ocgtkhtml.
```

## 29.54 5.54 What is ocsort?

**Proof of concept release** *as of February 2010*

A powerful external sort utility using OpenCOBOL for the sort engine.

A preliminary version can be referenced through [http://www.opencobol.org/modules/newbb/viewtopic.php?topic\\_id=915&forum=1&po](http://www.opencobol.org/modules/newbb/viewtopic.php?topic_id=915&forum=1&po) or directly from [http://oldsite.add1tocobol.com/tiki-download\\_file.php?fileId=74](http://oldsite.add1tocobol.com/tiki-download_file.php?fileId=74)

ocsort supports a variety of sorting options, for example:

```
ocsort sort fields"(1,5,CH,A,11,4,CH,A)" use inputfile record f,391 org sq give outputfile org sq
```

Users of MFSORT may recognize the syntax. Explaining the above example, Angus posted:

This will sort the file "inputfile", a fixed length file (391 byte each record, organization sequential), and create a file "outputfile" sorted (which is of the same type). The sort fields are :

(start, length, type, direction)

=> start=1

=> length=5

=> type = character (you can sort on comp3 fields, but ocsort don't handle it)

=> direction = ascending (or descending)

It's like an order by.

The omit/include condition allow to remove record from the file (ex if character number 5 of this record is 'F', omit the record). You can use and, or, greater than...)

The sources include the parser for the ocsort command language.

## 29.55 5.55 When is Easter?

A short program to display the day of Easter for a given year. *This calculation is known as the Computus*

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        17-Nov-2008
*> Purpose:     Display Easter Day for any given year, 1580 - 2050
*> Tectonics:   cobc -x easter.cob
*>              ./easter [year]
*> *****
identification-division.
program-id. easter.

data-division.
working-storage section.
01 a    picture 9(8) usage comp-x.
01 b    picture 9(8).
01 c    picture 9(8).
01 d    picture 9(8).
01 z    picture 9(8).  *> Why z? COBOL has pi for pi and e for e
01 f    picture 9(8).
01 g    picture 9(8).
01 h    picture 9(8).
01 i    picture 9(8).
01 j    picture 9(8).
01 year picture 9(4).
01 mo   picture 9(2).
01 da   picture 9(2).
01 args picture x(80).

*> *****
procedure-division.

accept args from command-line end-accept
if args not equal spaces
    move args to year
else
    display "Year: " with no advancing end-display
    accept year end-accept
end-if

compute a = function mod(year 19)           end-compute
divide year by 100 giving b remainder c     end-divide
divide b    by    4 giving d remainder z     end-divide
compute f = (b + 8) / 25                    end-compute
compute g = (b - f + 1) / 3                 end-compute
compute h = (19 * a) + b - d - g + 15       end-compute
compute h = function mod(h 30)              end-compute
divide c    by    4 giving i remainder j     end-divide
compute c = (z + i) * 2 + 32 - h - j        end-compute
compute c = function mod(c 7)               end-compute
compute b = (a + (11 * h) + (22 * c)) / 451 end-compute
compute a = h + c - (7 * b) + 114           end-compute
compute da = function mod(a 31) + 1         end-compute
divide a    by 31 giving mo                 end-divide

```

```
display "yyyy/mm/dd: " year "/" mo "/" da end-display
goback.
end program easter.

*> *****
*> Snagged from a REBOL script, easter-day.r by Didier Cadieu
*> http://www.rebol.org/view-script.r?script=easter-day.r
*>
*> easter-day: func [
*>     {Compute the easter date for the wanted year.}
*>     year [integer!] {Year for which you want the easter date}
*>     /local a b c d z f g h i k
*> ] [
*>     a: year // 19
*>     b: to integer! year / 100
*>     c: year // 100
*>     d: to integer! b / 4
*>     z: b // 4
*>     f: to integer! b + 8 / 25
*>     g: to integer! b - f + 1 / 3
*>     h: 19 * a + b - d - g + 15 // 30
*>     i: to integer! c / 4
*>     k: c // 4
*>     c: z + i * 2 + 32 - h - k // 7
*>     b: to integer! a + (11 * h) + (22 * c) / 451
*>     a: h + c - (7 * b) + 114
*>     to date! reduce [
*>         a // 31 + 1
*>         to integer! a / 31
*>         year
*>     ]
*> ]
```

Sample, with and without command line argument.

```
$ cobc -x easter.cob
$ ./easter 2011
yyyy/mm/dd: 2011/04/24
$ ./easter
Year: 2010
yyyy/mm/dd: 2010/04/04
```

### 29.55.1 5.55.1 Easter program critique

What follows is a warning to those people learning COBOL with the help of this document. The variable names used to implement the algorithm to find Easter day are near to useless as to intent and or reason. It's not good COBOL style and I got called on it. Take the critique for what you will, I took it as 'hey, come on, write better code if you're going to show it off'. Keep in mind that if you ever are fortunate enough to work with core business COBOL, what I got as a critique, could well be an embarrassing drumming from a boss and threats of firings. And as a side-note, be willing to take drummings and learn from them before the threats of firings occur. Programmers should never be defensive over code, but open and willing to better.

I posted a link to the easter.cob source code above as a Christmas post on a LinkedIn COBOL group and got this feedback from Huib Klink; I respect his posts and opinions.

It would have been slightly more appropriate to share a COBOL source that tells when its Christmas. Let my give it a try (Proc. div. only):

```

accept args from command-line end-accept
if args not equal spaces
move args to year
else
display "Year: " with no advancing end-display
accept year end-accept
end-if

move 12 to mo
move 25 to da
display "yyyy/mm/dd: " year "/" mo "/" da end-display
goback.
end program xmas.

```

Hmmmmmm. Lot less variables needed so it seems ... should clean up working storage, but since I copy/pasted this and don't want (are forbidden) to fix what ain't broke I will not change that piece of the program. For sure NOBODY will ever need to fix this program anymore so NOBODY will be sitting for hours wondering what a is for. Or b. or c. Or ... whatever, I am a programmer and thus I am lazy by definition, and I want to turn around that logic so doing no clean-up proves my professionalism and eases my job. After all if all programmers are lazy, I must be a very good one and

```

...
(5 minutes contemplating on fuzzy lazy logic)
...

```

Happy Xmas

So, I looked into it and learned something I find very cool. See <http://en.wikipedia.org/wiki/Computus> The calculation has a name and its name is **Computus**. That's awesome. Sadly, the Anonymous Gregorian algorithm detailed on Wikipedia uses the same useless variable names and the sample remains obfuscated, as I think the original sent into a newspaper in 1876 was intended.

## 29.55.2 5.55.2 A real COBOL Computus

From Paul Chandler during a discussion on LinkedIn in COBOL Professionals.

This one is nice folks. Defensible.

```

000100 IDENTIFICATION DIVISION.                                00010025
000200 PROGRAM-ID. RCEASTER.                                    00020025
000300 AUTHOR. PAUL CHANDLER, MARCH 2013.                      00030025
000400*****                                                    00040025
000500*** THIS PROGRAM CALCULATES THE DATE OF EASTER FOR ***   00050025
000600*** YEARS IN THE GREGORIAN CALENDAR. IT'S A PORT OF ***  00060025
000700*** THE DONALD KNUTH ALGORITHM PUBLISHED IN VOLUME 1 ***  00070025
000800*** OF "THE ART OF COMPUTER PROGRAMMING". ***           00080025
000900*** ***                                                 00090025
001000*****                                                    00100025
001100 ENVIRONMENT DIVISION.                                    00110025
001200 DATA DIVISION.                                         00120025
001300 FILE SECTION.                                           00130025
001400 WORKING-STORAGE SECTION.                                 00140025
001500 77 ACCEPT-YEAR PIC 9(08).                                00150025
001600 01 WORKING-FIELDS COMP.                                  00160025
001700 05 TGT-YEAR PIC S9(08).                                 00170025

```

```

001800      05 GOLDEN-NUMBER      PIC S9(08) .           00180025
001900      05 TGT-CENTURY       PIC S9(08) .           00190025
002000      05 LEAP-YEAR-CRCTN   PIC S9(08) .           00200025
002100      05 MOON-SYNC-CRCTN   PIC S9(08) .           00210025
002200      05 FIRST-SUNDAY      PIC S9(08) .           00220025
002300      05 EPACT             PIC S9(08) .           00230025
002400      05 FULL-MOON        PIC S9(08) .           00240025
002500      05 EASTER-SUNDAY     PIC S9(08) .           00250025
002600 01  DISPLAY-FIELDS.      00260025
002700      05 TGT-YEAR-DSP      PIC Z(08)-.           00270025
002800      05 EASTER-MONTH      PIC X(06) .           00280025
002900      05 EASTER-SUNDAY-DSP PIC Z(08)-.           00290025
003000  PROCEDURE DIVISION.    00300025
003100      ACCEPT ACCEPT-YEAR.   00310025
003200      MOVE ACCEPT-YEAR TO TGT-YEAR TGT-YEAR-DSP 00320025
003300      IF TGT-YEAR < 1583   00330025
003400          DISPLAY "YEAR MUST BE 1583 OR GREATER" 00340025
003500          STOP RUN         00350025
003600      ELSE                 00360025
003700          DISPLAY "EASTER DATE FOR:" TGT-YEAR-DSP 00370025
003800      END-IF                00380025
003900      COMPUTE GOLDEN-NUMBER = FUNCTION MOD(TGT-YEAR, 19) + 1 00390025
004000      COMPUTE TGT-CENTURY = (TGT-YEAR / 100) + 1          00400025
004100      COMPUTE LEAP-YEAR-CRCTN = (3 * TGT-CENTURY / 4) - 12 00410025
004200      COMPUTE MOON-SYNC-CRCTN = ((8 * TGT-CENTURY + 5) / 25) - 5 00420025
004300      COMPUTE FIRST-SUNDAY =                               00430025
004400          (5 * TGT-YEAR / 4) - LEAP-YEAR-CRCTN - 10       00440025
004500 *****                                00450025
004600*      TO MAKE THE EPACT CALCULATION MORE READABLE, *      00460025
004700*      THE COMPUTATION WILL BE DONE IN STAGES. *          00470025
004800 *****                                00480025
004900*                                00490025
005000* STAGE #1: GET THE RAW NUMBER..... *                    00500025
005100*                                00510025
005200      COMPUTE EPACT =                                       00520025
005300          (11 * GOLDEN-NUMBER)                               00530025
005400          + 20                                              00540025
005500          + MOON-SYNC-CRCTN                                00550025
005600          - LEAP-YEAR-CRCTN                                00560025
005700*                                00570025
005800* STAGE #2: GET THE MOD 30 VALUE... *                    00580025
005900*                                00590025
006000      COMPUTE EPACT = FUNCTION MOD(EPACT, 30)             00600025
006100*                                00610025
006200* STAGE #3: TO ENSURE THAT EPACT IS A POSITIVE NBR, *      00620025
006300*      ADD 30 AND MOD 30 AGAIN. *                            00630025
006400*                                00640025
006500      ADD 30 TO EPACT                                       00650025
006600      COMPUTE EPACT = FUNCTION MOD(EPACT, 30)             00660025
006700*                                00670025
006800* ADJUST FOR YEARS WHEN ORTHODOX DIFFERS *                00680025
006900*                                00690025
007000      IF (EPACT = 25 AND GOLDEN-NUMBER > 11)             00700025
007100      OR (EPACT = 24)                                       00710025
007200          ADD 1 TO EPACT                                    00720025
007300      END-IF                                               00730025
007400*                                00740025
007500* NEXT 2 STATEMENTS FIND FIRST FULL MOON AFTER MAR.21* 00750025

```



```

007600*                                00760025
007700  SUBTRACT EPACT FROM 44 GIVING FULL-MOON 00770025
007800  IF EPACT > 23                          00780025
007900      ADD 30 TO FULL-MOON                 00790025
008000  END-IF                                  00800025
008100*                                00810025
008200* ADVANCE SUNDAY TO THE FIRST SUNDAY AFTER FULL MOON * 00820025
008300*                                00830025
008400  COMPUTE EASTER-SUNDAY =                 00840025
008500      FULL-MOON                           00850025
008600      + 7                                 00860025
008700      - (FUNCTION MOD((FIRST-SUNDAY + FULL-MOON), 7)) 00870025
008800*                                00880025
008900* IF EASTER-SUNDAY > 31, EASTER IS IN APRIL - MOVE THE 00890025
009000* MONTH TO APRIL AND SUBTRACT 31 FROM THE MONTH.      00900025
009100* OTHERWISE EASTER IS IN MARCH, USE THE DAY AS IS.    00910025
009200*                                00920025
009300  IF EASTER-SUNDAY > 31                  00930025
009400      MOVE 'APRIL' TO EASTER-MONTH        00940025
009500      SUBTRACT 31 FROM EASTER-SUNDAY      00950025
009600  ELSE                                    00960025
009700      MOVE 'MARCH' TO EASTER-MONTH        00970025
009800  END-IF                                  00980025
009900  MOVE EASTER-SUNDAY TO EASTER-SUNDAY-DSP 00990025
010000  DISPLAY EASTER-MONTH EASTER-SUNDAY-DSP 01000025
010100  STOP RUN.                              01010025

```

Tectonics are a simple `cobc -x rceaster.cob`. ACCEPTs the year.

```

$ ./rceaster
2013
EASTER DATE FOR:    2013
MARCH              31

```

Thanks Paul.

## 29.56 5.56 Does Vim support OpenCOBOL?

Very well. See `cobol.vim` for a syntax highlighter tuned for OpenCOBOL.

Vim's Visual Block mode can be very handy at reforming COBOL source code.

Author's choice. `ocfaq.rst` is edited using Vim, Bram Moolenaar's `vi` enhancement. See below for some settings that can make OpenCOBOL more productive.

### 29.56.1 5.56.1 vim code completion

For code completion (Ctrl-P while in insert mode) start by creating a reserved word list using your `cobc` command

```
$ cobc --list-reserved | tail -n+3 | cut -f1 > ~/.vim/ocreserved.lis
```

followed by this change in `~/.vimrc`

```

:set ignorecase
:set infercase
:set complete=k~/.vim/ocreserved.lis

```

## 29.56.2 5.56.2 freedom

To free the cursor (allowing the cursor to travel past line endings) use:

```
:set virtualedit=all
```

## 29.56.3 5.56.3 autoload a skeleton

For a quick template when starting a new file (in `.vimrc`, change the filename `~/lang/cobol/headfix.cob` to where you keep your favourite COBOL starter skeleton).

```
" Auto load COBOL template
autocmd BufNewFile *.cob      0r ~/lang/cobol/headfix.cob
```

## 29.57 5.57 What is w3m?

**w3m** is a text based web browser. OpenCOBOL can leverage some of the power of this application by directly calling it with `SYSTEM`.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin
*> Date:     30-Dec-2008
*> Purpose:   Textualize a webpage
*> Tectonics: cobc -x w3mcaller.cob
*>           ./w3mcaller opencobol.org
*> *****
identification division.
program-id. w3mcaller.

data division.
working-storage section.
01 args          pic x(256) .
01 command       pic x(256) .
01 result        usage binary-long.

*> *****
procedure division.
accept args from command-line.

string
  "w3m -dump " delimited by size
  function trim(args) delimited by size
  into command
end-string
call "SYSTEM" using command returning result end-call

goback.
end program w3mcaller.
```

Sample run on 28-Feb-2010:

```
$ ./w3mcaller opencobol.org
```

```
[logo]
```



```
Commons          Conversion
                  story from          2010/2/
*                OpenCOBOL MicroFocus to 10    768 20 12:23
                  OC, on SUSE          simrw
                  11.2
                  Visit Forums
```

Copyright (C) 2005 The OpenCOBOL Project. All rights reserved.  
Powered by Xoops2 | PHP | MySQL | Apache  
ocean-net

## 29.58 5.58 What is COB\_LIBRARY\_PATH?

If the DSO files are not in the current working directory along with the executable, the COB\_LIBRARY\_PATH can be set to find them.

On GNU/Linux and bash it could be

```
export COB_LIBRARY_PATH=/home/developer/ocnewstuff:/home/developer/ocstuff
```

to search for DSO files in directories **ocnewstuff** then **ocstuff**, giving your testing versions priority during development.

## 29.59 5.59 Can OpenCOBOL interface with Rexx?

Yes, ooRexx linkage is commented on at [http://www.opencobol.org/modules/newbb/viewtopic.php?topic\\_id=456&forum=1#forumpost2](http://www.opencobol.org/modules/newbb/viewtopic.php?topic_id=456&forum=1#forumpost2)

A Regina Rexx layer can be as simple as

### ocrexx.c

```
/* OpenCOBOL interface to Regina Rexx Interpreter */
/* Requires regina3 and regina3-dev */
/* cobc -I/usr/include/regina -c ocrexx.c */

#include <stdio.h>
#include <string.h>
#include <rexxsaa.h>

int ocrexx(char *script, char *args, char *resfield, int reslen, short *result) {
    APIRET rexxapiRET;
    RXSTRING retstr;
    RXSTRING arglist[1];
    short rexxret = 0;

    int ignore = 0;

    /* Initialize the engine, run the script */
    retstr.strptr = NULL;
    retstr.strlength = 0;
    arglist[0].strptr = args;
    arglist[0].strlength = strlen(args);

    rexxapiRET = RexxStart(1, (PRXSTRING)&arglist, script, NULL, NULL,
        RXCOMMAND || RXRESTRICTED, NULL, &rexxret, &retstr);
```

```

    /* set result back to OpenCOBOL */
    memset(resfield, ' ', reslen);
    if (rexxapiRET == 0) {
        memcpy(resfield, retstr.strptr, (retstr.strlength > reslen) ? reslen : retstr.strlength);
        *result = rexxret;
    }

    /* Let REXX do all the memory allocation */
    if (retstr.strptr != NULL) { ignore = REXXFreeMemory(retstr.strptr); }

    return (int)rexxapiRET;
}

int ocrexxcmd(char *cmds, char *args, char *resfield, int reslen, short *result) {
    APIRET rexxapiRET;
    RXSTRING retstr;
    RXSTRING arglist[1];
    RXSTRING instore[2];
    short rexxret = 0;

    int ignore = 0;

    /* For syntax check, no evaluate, taken from 8.4 of the Regina3.4 pdf */
    arglist[0].strptr = "//T";
    arglist[0].strlength = 3;

    arglist[0].strptr = args;
    arglist[0].strlength = strlen(args);

    /* Move the command(s) to the instore array */
    instore[0].strptr = cmds;
    instore[0].strlength = strlen(cmds);
    instore[1].strptr = NULL;
    instore[1].strlength = 0;

    /* Call REXX. Use argcount 1 and &arglist to call syntax check */
    retstr.strptr = NULL;
    retstr.strlength = 0;
    rexxapiRET = REXXStart(1, (PRXSTRING)&arglist, "FILLER", (PRXSTRING)&instore, "COMMAND" /* NULL
        RXCOMMAND, NULL, &rexxret, &retstr);

    /* set result back to OpenCOBOL */
    memset(resfield, ' ', reslen);
    if (rexxapiRET == 0) {
        memcpy(resfield, retstr.strptr, (retstr.strlength > reslen) ? reslen : retstr.strlength);
        *result = rexxret;
    }

    /* Let REXX do all the memory allocation */
    if (instore[1].strptr != NULL) { ignore = REXXFreeMemory(instore[1].strptr); }
    if (retstr.strptr != NULL) { ignore = REXXFreeMemory(retstr.strptr); }

    return (int)rexxapiRET;
}
/**/

```

with a usage example of

## rexccaller.cob

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*><* *****
*><* Rextx in OpenCOBOL
*><* *****
*><*
*><* :Author:      Brian Tiffin
*><* :Date:        13-Nov-2008
*><* :Purpose:     Very High Level Regina Rextx engine
*><* :Requires:    regina-rexx, regina3, regina3-dev, OC 1.1 pre-rel
*><* :Tectonics:
*><*      | cobc -I/usr/include/regina -c ocrexx.c
*><*      | cobc -x -lregina rexxcaller.cob ocrexx.o
*><*      | ocdoc rexxcaller.cob rexxcaller.rst rexxcaller.html
*> *****
identification division.
program-id. rexxcaller.

data division.
*><*
*><* =====
*><* Working Store
*><* =====
*><*
*><* ::
*><*
*><[
working-storage section.
01 newline constant as x"0a".
01 apicode          usage binary-long.
01 resultcode      usage binary-short.
01 scriptname      pic x(12) value 'verrexx.cmd' & x'00'.
01 argument        pic x(256) value 'OC1.1 args' & x"00".
01 cmds            pic x(1024).
01 rexxstring      pic x(1048576).
*><]

*> *****
procedure division.

*><*
*><* ===
*><* API
*><* ===
*><*
*><* -----
*><* ocrexx
*><* -----
*><* Pass a null-term scriptname, a null-term argument string
*><* the return value field and length, the return code and
*><* returning the Rextx api result code.
*><*
*><* Usage::
*><*
compute
    apicode = function length(function trim(scriptname))
end-compute

```

```

display
    "CALL Rexx with |" scriptname(1:apicode - 1) "|"
end-display
*><[
call "ocrexx"
    using
        by reference scriptname
        by reference argument
        by reference rexxstring
        by value function length(rixxstring)
        by reference resultcode
    returning apicode
end-call
display "|" apicode "|" resultcode with no advancing end-display
display "|" function trim(rixxstring trailing) "|" end-display
*><]

*><*
*><* -----
*><* ocrexxcmd
*><* -----
*><* Usage::
*><*
*><[
move "say 'Hello World!'; return 'From Rexx';" & x'00' to cmds.
compute
    apicode = function length(function trim(cmds))
end-compute
display newline
    "CALL Rexx command with |" cmds(1:apicode - 1) "|"
end-display
call "ocrexxcmd"
    using
        by reference cmds
        by reference argument
        by reference rexxstring
        by value function length(rixxstring)
        by reference resultcode
    returning apicode
end-call
display "|" apicode "|" resultcode with no advancing end-display
display "|" function trim(rixxstring trailing) "|" end-display
*><]

*><*
*><* or perhaps::
*><*
*><[
move
    "parse arg argument; say '##' || argument || '##';" & x"0a" &
    "capture = '";" & x"0a" &
    "address system 'cat tectonic && cat verrex.cmd && ls -l" &
    " && w3m rexxcaller.html'" &
    " with output fifo '";" & x"0a" &
    "DO i=1 WHILE queued() \= 0;" & x"0a" &
    "    parse pull line;" & x"0a" &
    "    capture = capture || line || '0a'x;" & x"0a" &
    "END;" & x'0a' &
    "return capture;" & x'00' to cmds

```

```
compute
  apicode = function length(function trim(cmds))
end-compute
display newline
  "CALL Rexx command with |" cmds(1:apicode - 1) "|"
end-display
call "ocrexxcmd"
  using
    by reference cmds
    by reference argument
    by reference rexxstring
    by value function length(reestring)
    by reference resultcode
  returning apicode
end-call
*><]
display "|" apicode "|" resultcode with no advancing end-display
display "|" function trim(reestring trailing) "|" end-display
goback.
end program rexxcaller.
*><*
```

And as a sample Rexx script

#### verrexx.cmd

```
Parse Version ver;
Say ver;
return ver;
```

With a sample run producing:

```
$ ./tectonic
CALL Rexx with |verrexx.cmd|
REXX-Regina_3.3(MT) 5.00 25 Apr 2004
ocrexx.c ocrexx.o rexxcaller rexxcaller.cob rexxcaller.html rexxcaller.rst rexx.output tectonic
|+0000000000|+00000|REXX-Regina_3.3(MT) 5.00 25 Apr 2004|

CALL Rexx command with |say 'Hello World!'; return 'From Rexx';|
Hello World!
|+0000000000|+00000|From Rexx|

CALL Rexx command with |parse arg argument; say '##' || argument || '##';
capture = '';
address system 'cat tectonic && cat verrexx.cmd && ls -l && w3m rexxcaller.html' with output fifo '';
DO i=1 WHILE queued() \= 0;
  parse pull line;
  capture = capture || line || '0a'x;
END;
return capture;|
##OC1.1 args##
|+0000000000|+00000|cobc -I/usr/include/regina/ -c ocrexx.c
cobc -x -lregina rexxcaller.cob ocrexx.o
../ocdoc rexxcaller.cob rexxcaller.rst rexxcaller.html ../ocfaq.css
./rexxcaller
/* script for OpenCOBOL Regina Rexx */
Parse Version ver;
Say ver;
address system;
```



```
'ls';
return ver;
total 68
-rw-r--r-- 1 btiffin btiffin 2469 2008-11-16 11:09 ocrexx.c
-rw-r--r-- 1 btiffin btiffin 2568 2010-05-06 22:51 ocrexx.o
-rwxr-xr-x 1 btiffin btiffin 18128 2010-05-06 22:51 rexxcaller
-rw-r--r-- 1 btiffin btiffin 4477 2008-11-16 11:28 rexxcaller.cob
-rw-r--r-- 1 btiffin btiffin 9312 2010-05-06 22:51 rexxcaller.html
-rw-r--r-- 1 btiffin btiffin 3187 2010-05-06 22:51 rexxcaller.rst
-rw-r--r-- 1 btiffin btiffin 4131 2008-11-16 11:30 rexx.output
-rwxr-xr-x 1 btiffin btiffin 162 2008-11-16 11:21 tectonic
-rw-r--r-- 1 btiffin btiffin 101 2008-11-15 23:24 verrexx.cmd
Rexx in OpenCOBOL
```

```
Author: Brian Tiffin
Date: 13-Nov-2008
Purpose: Very High Level Regina Rexx engine
Requires: regina-rexx, regina3, regina3-dev, OC 1.1 pre-rel
cobc -I/usr/include/regina -c ocrexx.c
Tectonics: cobc -x -lregina rexxcaller.cob ocrexx.o
ocdoc rexxcaller.cob rexxcaller.rst rexxcaller.html
```

#### Working Store

```
working-storage section.
01 newline constant as x"0a".
01 apicode          usage binary-long.
01 resultcode       usage binary-short.
01 scriptname       pic x(12) value 'verrexx.cmd' & x'00'.
01 argument         pic x(256) value 'OC1.1 args' & x"00".
01 cmds             pic x(1024).
01 rexxstring       pic x(1048576).
```

#### API

##### ocrexx

Pass a null-term scriptname, a null-term argument string the return value field and length, the return code and returning the Rexx api result code.

##### Usage:

```
call "ocrexx"
  using
    by reference scriptname
    by reference argument
    by reference rexxstring
    by value function length(rixxstring)
    by reference resultcode
  returning apicode
end-call
display "|" apicode "|" resultcode with no advancing end-display
display "|" function trim(rixxstring trailing) "|" end-display
```

##### ocrexxcmd

##### Usage:

```
move "say 'Hello World!'; return 'From Rexx';" & x'00' to cmds.
compute
    apicode = function length(function trim(cmds))
end-compute
display newline
    "CALL Rexx command with |" cmds(1:apicode - 1) "|"
end-display
call "ocrexxcmd"
    using
        by reference cmds
        by reference argument
        by reference rexxstring
        by value function length(renxstring)
        by reference resultcode
    returning apicode
end-call
display "|" apicode "|" resultcode with no advancing end-display
display "|" function trim(renxstring trailing) "|" end-display
```

or perhaps:

```
move
    "parse arg argument; say '##' || argument || '##';" & x"0a" &
    "capture = ' '; " & x"0a" &
    "address system 'cat tectonic && cat verrex.cmd && ls -l" &
    " && w3m rexxcaller.html'" &
    " with output fifo ' '; " & x"0a" &
    "DO i=1 WHILE queued() \= 0;" & x"0a" &
    "    parse pull line;" & x"0a" &
    "    capture = capture || line || '0a'x;" & x"0a" &
    "END;" & x'0a' &
    "return capture;" & x'00' to cmds
compute
    apicode = function length(function trim(cmds))
end-compute
display newline
    "CALL Rexx command with |" cmds(1:apicode - 1) "|"
end-display
call "ocrexxcmd"
    using
        by reference cmds
        by reference argument
        by reference rexxstring
        by value function length(renxstring)
        by reference resultcode
    returning apicode
end-call
```

|

and the ocdoc output at [rexxcaller.html](#)

## **29.60 5.60 Does OpenCOBOL support table SEARCH and SORT?**

Yep.

This is a two part example. A small tax table search, and a dictionary sort and lookup.

## 29.60.1 5.60.1 Linear SEARCH

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin, with some suggestions from human
*> Date:      30-Nov-2008, 02-Dec-2008
*> Purpose:   Demonstration of the SEARCH verb
*> Tectonics: cobb -x searchlinear.cob
*> *****
identification division.
program-id. searchlinear.

data division.

working-storage section.
01 taxinfo.
   05 tax-table occurs 4 times indexed by tt-index.
      10 province          pic x(2).
      10 taxrate           pic 999v9999.
      10 federal           pic 999v9999.
01 prov                   pic x(2).
01 percent                pic 999v9999.
01 percentage             pic zz9.99.

*> *****
procedure division.
begin.

*> *****
*> Sample for linear SEARCH, requires INDEXED BY table
*> populate the provincial tax table; not really, only a couple
*> populate Ontario and then PEI using different field loaders
move 'AB' to province(1)
move 'ON' to province(2)
move 0.08 to taxrate(2)
move 0.05 to federal(2)
move 'PE00014000000000' to tax-table(3)
move 'YT' to province(4)

*> Find Ontario tax rate
move "ON" to prov
perform search-for-taxrate

*> Setup for Prince Edward Island
move 'PE' to prov
perform search-for-taxrate

*> Setup for failure
move 'ZZ' to prov
perform search-for-taxrate

goback.
*> *****

search-for-taxrate.
  set tt-index to 1
  search tax-table
  at end display "no province: " prov end-display

```

```
        when province(tt-index) = prov
            perform display-taxrate
        end-search
.

display-taxrate.
    compute percent = taxrate(tt-index) * 100
    move percent to percentage
    display
        "found: " prov " at " taxrate(tt-index)
        ", " percentage "%, federal rate of " federal(tt-index)
    end-display
.

end program searchlinear.
```

A sample run producing:

```
$ cobc -x searchlinear.cob && ./searchlinear
found: ON at 000.0800, 8.00%, federal rate of 000.0500
found: PE at 000.1400, 14.00%, federal rate of 000.0000
no province: ZZ
```

## 29.60.2 5.60.2 SORT and binary SEARCH ALL

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin, with some suggestions from human
*> Date:      30-Nov-2008, 02-Dec-2008
*> Purpose:   Demonstration of the SEARCH ALL verb and table SORT
*> Tectonics: cobc -x -fdebugging-line searchbinary.cob
*> *****
identification division.
program-id. searchbinary.

environment division.
input-output section.
file-control.
    select optional wordfile
    assign to infile
    organization is line sequential.

data division.
file section.
fd wordfile.
    01 wordrec                pic x(20).

working-storage section.
01 infile                    pic x(256) value spaces.
    88 defaultfile          value '/usr/share/dict/words'.
01 arguments                 pic x(256).

*> Note the based clause, this memory is initially unallocated
78 maxwords                  value 100000.
01 wordlist                  based.
    05 word-table occurs maxwords times
        depending on wordcount
```

```

        descending key is wordstr
        indexed by wl-index.
    10 wordstr          pic x(20) .
    10 wordline        usage binary-long.
01 wordcount          usage binary-long.

01 file-eof          pic 9 value low-value.
    88 at-eof        value high-values.

01 word              pic x(20) .

*> *****
procedure division.
begin.

*> Get the word file filename
accept arguments from command-line end-accept
if arguments not equal spaces
    move arguments to infile
else
    set defaultfile to true
end-if

*> *****
*> Try playing with the words file and binary SEARCH ALL
*> requires KEY IS and INDEXED BY table description

*> Point wordlist to valid memory
allocate wordlist initialized

open input wordfile

move low-value to file-eof
read wordfile
    at end set at-eof to true
end-read

perform
    with test before
    until at-eof or (wordcount >= maxwords)
        add 1 to wordcount
        move wordrec to wordstr(wordcount)
        move wordcount to wordline(wordcount)
        read wordfile
            at end set at-eof to true
        end-read
    end-perform

close wordfile

*> ensure a non-zero length table when allowing optional file
evaluate true          also file-eof
    when wordcount = 0    also any
        move 1 to wordcount
        display "No words loaded" end-display
    when wordcount >= maxwords also low-value
        display "Word list truncated to " maxwords end-display
end-evaluate

```

```
>>D display "Count: " wordcount ": " wordstr(wordcount) end-display

*> Sort the words from z to a
  sort word-table on descending key wordstr

*> fetch a word to search for
  display "word to find: " with no advancing end-display
  accept word end-accept

*> binary search the words for word typed in and display
*> the original line number if/when a match is found
  set wl-index to 1
  search all word-table
    at end
      display
        word " not a word of " function trim(infile)
      end-display
    when wordstr(wl-index) = word
      display
        word " sorted to " wl-index ", originally "
        wordline(wl-index) " of " function trim(infile)
      end-display
    end-search

*> Release memory ownership
  free address of wordlist

goback.
end program searchbinary.
```

with some sample words and a Debian 5.0.4 system:

```
$ cobc -x searchbinary.cob
$ ./searchbinary
word to find: zygote
zygote          sorted to +000000018, originally +0000098552 of /usr/share/dict/words
$ ./searchbinary
word to find: abacus
abacus         sorted to +000080466, originally +0000018104 of /usr/share/dict/words
```

See SORT for other examples.

## 29.61 5.61 Can OpenCOBOL handle named pipes?

Yes. Here is a sample, using a tongue-in-cheek **corncob** filename.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin
*> Date:     10-Apr-2010
*> Purpose:   playing with the corncob pipe
*> Tectonics: mkfifo corncob
*>           cobc -x popcorn.cob
*>           ls >corncob & ./popcorn
*> *****
  identification division.
```

```

program-id. popcorn.

environment division.
configuration section.

input-output section.
file-control.
    select corncob
    assign to 'corncob'
    organization is line sequential
    .

data division.
file section.
fd corncob.
    01 tobacco pic x(32768).

working-storage section.
01 filestat pic x value low-value.
    88 done          value high-value.
01 liner          pic 99999.
01 looper        pic 99999.
01 atmost       constant as 32768.
01 bowl.
    02 popcorn occurs atmost times depending on liner
        ascending key kernel.
    03 kernel pic x(132).

*> *****
procedure division.

*> Read from the pipe into a table
open input corncob
move zero to liner
perform until done or (liner greater than or equal to atmost)
    read corncob
        at end
            set done to true
        not at end
            add 1 to liner end-add
            move tobacco to kernel(liner)
    end-read
end-perform
close corncob

*> Sort it descending and display
sort popcorn on descending key kernel

perform varying looper from 1 by 1 until looper > liner
    display
        "OpenCOBOL: " function trim(kernel(looper) trailing)
    end-display
end-perform

goback.
end program popcorn.

```

With a sample run producing:

```
$ rm corncob
$ mkfifo corncob
$ ls -d n* >corncob & ./popcorn
[1] 5033
OpenCOBOL: nums.cob
OpenCOBOL: nums
OpenCOBOL: network
[1]+ Done                ls -d n* > corncob
$ ls -d n*
network  nums  nums.cob
$ date >corncob & ./popcorn
[1] 5037
OpenCOBOL: Sun Apr 11 08:04:48 EDT 2010
[1]+ Done                date > corncob
```

## 29.62 5.62 Can OpenCOBOL interface with ROOT/CINT?

Yes. The February 2009 pre-release generates C code that can be loaded by the ROOT/CINT framework. ROOT is a high energy physics data analysis framework released by CERN. ROOT/CINT embeds the CINT C/C++ interactive interpreter.

See <http://root.cern.ch/drupal/content/cint> for details.

OpenCOBOL programmers can use ROOT/CINT for interactive testing of COBOL subprograms.

Given

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin
*> Date:      20101119
*> Purpose:   Pass arguments to ROOT/CINT invoked subprograms
*> Tectonics: cobb -fimplicit-init -C cobparams.cob
*> *****
identification division.
program-id. cobparams.

data division.
linkage section.
01 a-number usage binary-long.

*> *****
procedure division using by reference a-number.
display a-number end-display
move a-number to return-code
goback.
end program cobparams.
```

and the command line

```
$ cobb -fimplicit-init -C cobparams.cob
```

gives a set of C source code output for cobparams.

ROOT/CINT can then be used to play with the program.

```
$ cobb -fimplicit-init -C cobparams.cob
$ root -l
```



```

root [0] gSystem->Load("/usr/local/lib/libcob.so");
root [1] .L cobparams.c+
root [2] int a = 0;
root [3] int d = 42;
root [4] a = cobparams((unsigned char*)&d);
+0000000042
root [5] printf("%d\n", a);
42
root [6]

```

There is some magic in the above snippet. ROOT preloads the runtime libcob.so. Then its .L command is used with the plus + option to interpret and link load the cobc generated cobparams.c file.

The ROOT/CINT console now has access to the cobparams “function”, defined by OpenCOBOL to have an unsigned char pointer as its BY REFERENCE access; A cast of the integer d’s address allows CINT to call up the COBOL subprogram, passing the 42 for DISPLAY and then returning the same value as the result. The interactively defined integer a, gets this 42 from OpenCOBOL’s RETURN-CODE.

## 29.62.1 5.62.1 Graphing sample

ROOT/CINT is built for analysis. So, plotting and graphing are built-in.

Given

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin
*> Date:      20101119
*> Purpose:   Pass arguments to ROOT/CINT invoked subprograms
*> Tectonics: cobc -fimplicit-init -C cobparams.cob
*> *****

REPLACE ==ARRAYSIZE== BY ==450==.

identification division.
program-id. cobfloats.

data division.
working-storage section.
01 cnt pic 999.
01 val usage float-long.
01 xes.
   02 an-x usage float-long occurs ARRAYSIZE times.
01 yes.
   02 an-y usage float-long occurs ARRAYSIZE times.

linkage section.
01 vxes.
   02 an-x usage float-long occurs ARRAYSIZE times.
01 vyes.
   02 an-y usage float-long occurs ARRAYSIZE times.

*> *****
procedure division using by reference vxes, vyes.
perform varying cnt from 1 by 1 until cnt >= ARRAYSIZE
   compute val = cnt * function random() end-compute
   move cnt to an-x in xes(cnt)
   move val to an-y in yes(cnt)

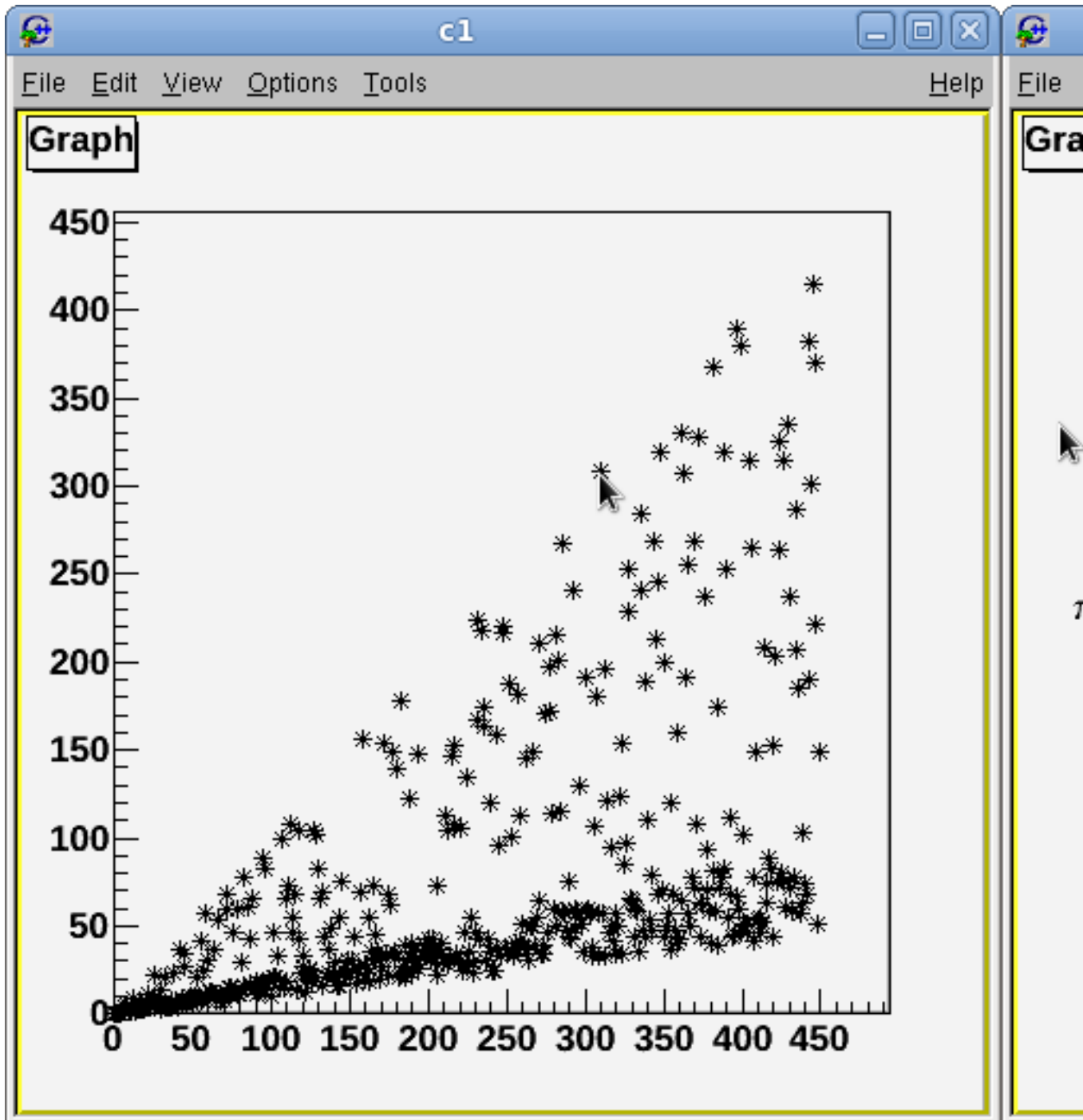
```

```
end-perform
move xes to vxes
move yes to vyes
move cnt to return-code
goback.
end program cobfloats.
```

And then a console session of:

```
$ cobc -fimplicit-init -C cobparams.cob
$ vi cobparams.c
... add a single line
... #pragma K&R
... to lighten up CINT's type safety for ease of use at the console
$ root -l
root [0] gSystem->Load("/usr/local/lib/libcob.so");
root [1] .L cobparams.c+
root [2] int a = 0; double x[450]; double y[450];
root [3] a = cobfloats(&x, &y);
root [4] a
(int)450
root [5] printf("%f %f\n", x[42], y[42]);
43.000000 8.232543
root [6] TGraph *graph1 = new TGraph(450, x, y);
root [7] graph1->Draw("A*");
root [8] TGraphPolar *polar1 = new TGraphPolar(450, x, y);
root [9] polar1->SetLineColor(2);
root [10] polar1->Draw("AOL");
```

produces the following graphs; some constrained random numbers, and a circular view of those random numbers. *Nerd*



heaven.

A trials and tribulations thread can be read at [http://www.opencobol.org/modules/newbb/viewtopic.php?topic\\_id=1165&forum=1](http://www.opencobol.org/modules/newbb/viewtopic.php?topic_id=1165&forum=1)

### 29.63 5.63 Can OpenCOBOL be used to serve HTTP?

Not directly, COBOL preceding the World Wide Web by some 35 years, but yes.

### 29.63.1 5.63.1 libsoup HTTP server

Vala and libsoup is one way.

Given `soupserver.vala`

```
// vala .10 specific. .11 changes string to uint8 array
// valac -c --pkg libsoup-2.4 --thread soupserver.vala

// Give the server a default
void default_handler (Soup.Server server, Soup.Message msg, string path,
                     GLib.HashTable? query, Soup.ClientContext client)
{
    string response_text = ""
    <html>
    <body>
    <p>Current location: %s</p>
    <p><a href="/xml">Test XML</a></p>
    <p><a href="/cobol">Test COBOL</a></p>
    <p><a href="/exit">Tell server to exit</a></p>
    </body>
    </html>"".printf (path);

    msg.set_response ("text/html", Soup.MemoryUse.COPY,
                     response_text, response_text.size ());
    msg.set_status (Soup.KnownStatusCode.OK);
}

void xml_handler (Soup.Server server, Soup.Message msg, string path,
                 GLib.HashTable? query, Soup.ClientContext client)
{
    string response_text = "<node><subnode>test</subnode></node>";
    msg.set_response ("text/xml", Soup.MemoryUse.COPY,
                     response_text, response_text.size ());
}

void cobol_handler (Soup.Server server, Soup.Message msg, string path,
                   GLib.HashTable? query, Soup.ClientContext client)
{
    string response_text = ""
    <html>
    <body>
    <p>Current location: %s</p>
    <p><a href="/xml">Test XML</a></p>
    <p><a href="/">Home</a></p>
    <p><a href="/exit">Tell server to exit</a></p>
    </body>
    </html>"".printf (path);

    msg.set_response ("text/html", Soup.MemoryUse.COPY,
                     response_text, response_text.size ());
    msg.set_status (Soup.KnownStatusCode.OK);
}

void exit_handler (Soup.Server server, Soup.Message msg, string path,
                  GLib.HashTable? query, Soup.ClientContext client)
{
    server.quit ();
}
```

```

}

int CBL_OC_SOUPSERVER(ref Soup.Server* ss, int port) {
    var server = new Soup.Server(Soup.SERVER_PORT, port);
    server.add_handler("/", default_handler);
    server.add_handler("/xml", xml_handler);
    server.add_handler("/cobol", cobol_handler);
    server.add_handler("/exit", exit_handler);
    ss = (owned)server;
    stdout.printf("ss: %X\n", (uint)ss);
    return 0;
}

int CBL_OC_SOUPRUN(Soup.Server ss) {
    ss.run();
    return 0;
}

```

and `ocsoup.cob`

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20101205
*> Purpose:     An HTTP server with libsoup
*> Tectonics:   valac -c --pkg libsoup-2.4 --thread soupserver.vala
*>              cobc -x ocsoup.cob soupserver.vala.o -lglib-2.0
*>              -lsoup-2.4 -lgobject-2.0
*> *****
identification division.
program-id. ocsoup.

data division.
working-storage section.
01 soup-server usage pointer.
01 port usage binary-long value 8088.
01 result usage binary-long.

*> *****
procedure division.
call "g_type_init" end-call
display "Initialize soup HTTP server on port " port end-display
call "CBL_OC_SOUPSERVER" using
    by reference soup-server
    by value port
    returning result
end-call
display "Result: " result " Server at: " soup-server end-display

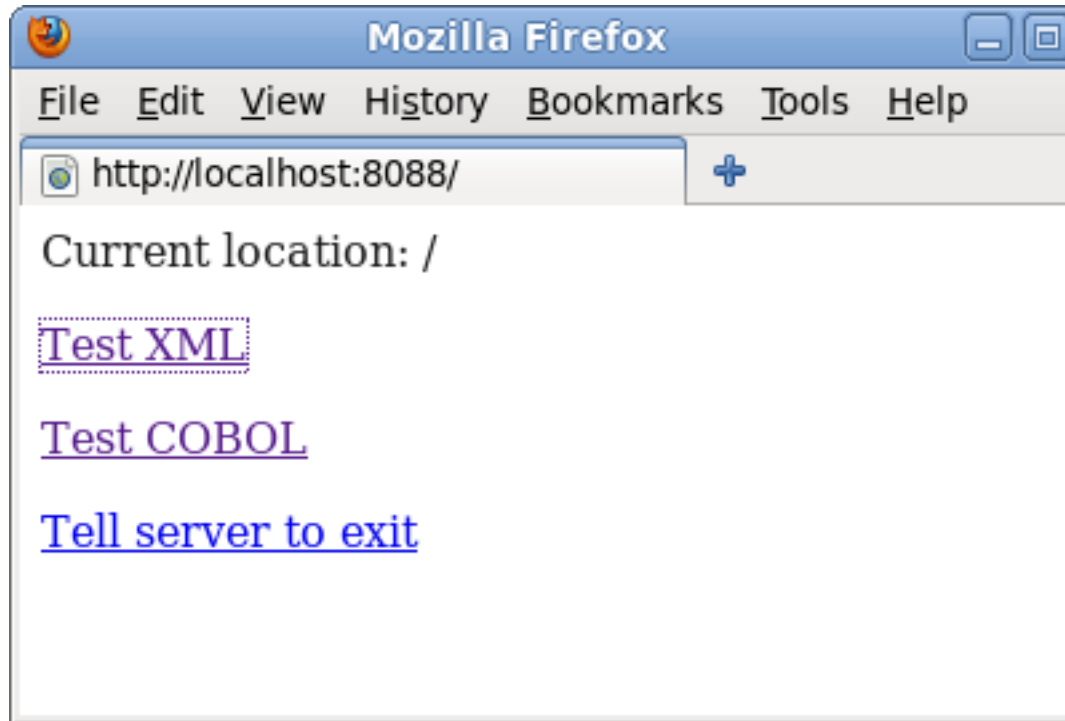
display "About to run server, ^C to terminate" end-display
call "CBL_OC_SOUPRUN" using
    by value soup-server
    returning result
end-call

goback.
end program ocsoup.

```

and a little bash

```
$ valac -c --pkg libsoup-2.4 --thread soupserver.vala
$ ... some warnings about unused methods ...
$ cobc -x ocsoup.cob soupserver.vala.o -lglib-2.0 -lsoup-2.4 -lgobject-2.0
$ ./ocsoup
Initialize soup HTTP server on port +0000008088
ss: 21CF060
Result: +0000000000 Server at: 0x00000000021cf060
About to run server, ^C to terminate
```



You get RESTful screen shots like...

The next steps are getting the add\_handler callbacks into COBOL, and then play with the template and replace model.

## 29.64 5.64 Is there a good SCM tool for OpenCOBOL?

In this author's opinion, yes. **Fossil**.

Where **SCM** is *Software Configuration Management*, and not simply *Source Code Management*, which Fossil does quite well.

See the **Fossil** site, snag a tar ball, make, and move the binary to /usr/bin.

Then, to start up your next OpenCOBOL COBOL project:

```
# Create the fossil distributed repository
$ mkdir ~/fossils
$ cd ~/fossils
$ fossil new nextbigthing.fossil

# Serve it up on the localhost port 8080
$ fossil server . &

# browse to the admin panel and do a little nicey nice config
$ opera http://localhost:8080/nextbigthing
```

```
# set up the working copy
$ cd ~/projects
$ mkdir nextbigthing
$ cd nextbigthing
$ fossil clone http://localhost:8080/nextbigthing nbt.fossil

# now look at the shiny copy of nextbig
$ ls
$ vi nextbigthing.cob
$ fossil add nextbigthing.cob
$ fossil ci -m "On to the next big thing"

# browse to the repo and create some wiki pages for morale boosting
$ opera http://localhost:8080/nextbigthing

# compile and run the next big thing
$ cobc -x nextbigthing.cob
$ ./nextbigthing

# browse again, and create the bug tickets
$ opera http://localhost:8080/nextbigthing/tktnew
```

Ahh, morale boosting bugs. :)

The fans of OpenCOBOL have posted a few Fossils at

<http://fossile.plpwebs.com/ocweb.cgi> for an experimental build of OpenCOBOL with support for ACCEPT var FROM HTTP-POST identifier and <http://fossile.plpwebs.com/ocsamples.cgi> where there is a stash of short OpenCOBOL source code samples. Access with:

```
$ fossil clone http://user:password@fossile.plpwebs.com/ocsamples.cgi my.fossil
$ fossil open my.fossil
```

If you don't have a username and password it'll simply be:

```
$ fossil clone http://fossile.plpwebs.com/ocsamples.cgi my.fossil
```



## 29.65 5.65 Does OpenCOBOL interface with FORTRAN?

Yes. Quite well in the GNU land.

Snuggled away at <http://fortranwiki.org/fortran/show/jucolor> is a color unit coverter; RGB to HLS, HSV to RGB, etc...

And with a simple Makefile ala

```
all: rgbcobol

libcolors.so: colors.for
    gfortran -ffree-form -shared -fPIC -o libcolors.so colors.for

rgbcobol: rgbcobol.cob libcolors.so
    cobc -g -debug -x rgbcobol.cob -lcolors -L .
```

and some COBOL

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20110411
*> Purpose:     Call a FORTRAN color unit converter, rgb, hsv, ...
*> Tectonics:   gfortran -ffree-form -shared -fPIC
*>              -o libcolors.so colors.for
*>              cobc -x rgbcobol.cob -lcolors -L .
*> *****
identification division.
program-id. rgbcobol.

data division.
working-storage section.
01 r usage float-short.
01 g usage float-short.
01 b usage float-short.

01 h usage float-short value 12.21.
01 l usage float-short value 21.12.
01 s usage float-short value 23.32.

01 st usage binary-long.

*> *****
procedure division.
move 000.0 to h
move 050.0 to l
move 100.0 to s
display "Calling FORTRAN with " h space l space s end-display
call "jucolor_" using 'hls', h, l, s, 'rgb', r, g, b, st end-call
display "Returned          " r space g space b end-display
display "Status of " st end-display
call "showit_" end-call
goback.
end program rgbcobol.
```

which produces:

```
[btiffin@home fortran]$ ./rgbcobol
Calling FORTRAN with 0.000000000000000000 50.0000000000000000 100.0000000000000000
```

```

inside jucolor_:  0.0000000  0.0000000  50.000000  0.0000000  100.00000
Returned        100.000000000000000000 0.000000000000000000 0.000000000000000000
Status of +0000000000
inside jucolor_:  0.0000000  0.0000000  50.000000  595.19684  100.00000
INPUT HLS PURE RED ==> OUTPUT RGB values are  100.00000  0.0000000  0.0000000
=====
inside jucolor_:  120.00000  100.00000  50.000000  0.0000000  100.00000
INPUT HLS PURE GREEN OUTPUT RGB values are  0.0000000  100.00000  0.0000000
=====
inside jucolor_:  240.00000  0.0000000  50.000000  100.00000  100.00000
INPUT HLS PURE BLUE OUTPUT RGB values are  0.0000000  0.0000000  100.00000
=====
inside jucolor_:  100.00000  0.0000000  0.0000000  0.0000000  0.0000000
INPUT RGB PURE RED OUTPUT HLS values are  0.0000000  50.000000  100.00000
=====
inside jucolor_:  0.0000000  0.0000000  100.00000  50.000000  0.0000000
INPUT RGB PURE GREEN OUTPUT HLS values are  120.00000  50.000000  100.00000
=====
inside jucolor_:  0.0000000  120.00000  0.0000000  50.000000  100.00000
INPUT RGB PURE BLUE OUTPUT HLS values are  240.00000  50.000000  100.00000
values are  240.00000  50.000000  100.00000
=====

```

The weird numbers on the second “inside jucolor\_” are uninitialized gfortran variables, displayed before being set, not great, but safe enough for a one off.

## 29.66 5.66 Does OpenCOBOL interface with APL?

See Does OpenCOBOL interface with J?

## 29.67 5.67 Does OpenCOBOL interface with J?

Yes, kinda. Jsoftware recently posted GPL 3 licensed source code for the J programming language. J is designed in part by one of the creators of APL, Eric Iverson. Initial tests have proven successful but there is more work before integration with **libj** in OpenCOBOL is ready for prime-time.

```

OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:    Brian Tiffin
*> Date:     20110711
*> Purpose:   Attempt calling a J sentence. APL in COBOL.
*> Tectonics: cobb -x callj.cob -lj
*> *****
identification division.
program-id. callj.

data division.
working-storage section.
77 jptr usage pointer.
77 result usage binary-long.

*> *****
procedure division.
call "JInit" returning jptr end-call

```

```

display jptr end-display

call "JDo"
  using by value jptr
  by content z"a =. 1 + 1"
  returning result
end-call
display result end-display

call "JDo"
  using by value jptr
  by content z"2 + 2"
  returning result
end-call
display result end-display

call "JDo"
  using by value jptr
  by content z "('Test Data',CR,LF) 1!:2 <'temp.dat'"
  returning result
end-call
display result end-display

call "JDo"
  using
    by value jptr
    by content z"load 'jgplsrc/test/test.ijs'"
  returning result
end-call
display result end-display

call "JDo"
  using
    by value jptr
    by content z"bad=: TEST ddall"
  returning result
end-call
display result end-display

call "JDo"
  using
    by value jptr
    by content z"BAD ddall"
  returning result
end-call
display result end-display
goback.
end program callj.

```

produces:

```

$ cobc -x callj.cob -lj
$ ./callj
0x00007f3b6ead7010
+0000000000
+0000000000
+0000000003
+0000000021

```

```
+0000000000
+0000000000
```

So libj inits, and can JDo J sentences, but there is a little more background effort to properly set J I/O and PATH settings into an array of callbacks. Doable, just have to ask the good folk at Jsoftware for a little assistance. More coming soon.

The GPL 3 J version 7.01b source code can be found at <http://www.jsoftware.com/> Compiling the sources took a little reading, but built clean on 64bit Fedora 14 after a quick edit of `jpglsrc/bin/jconfig`. Needed to set BITS to 64 and added readline support, as command line recall is more fun than no command line recall when running `jconsole`. After that `bin/build_libj bin/build_jconsole` all went smooth as silk. `libj.so` was copied to `/usr/lib64` and the above code compiled and linked just fine.

As did:

```
$ bin/build_defs
$ bin/build_tsdll
```

A test suite validates a J system. Read `test/test.ijs` and `test/tsu.ijs` for more info.

```
$ j/bin/jconsole
  load 'test/test.ijs'
  bad=: TEST ddall NB. run all tests
  BAD ddall NB. report tests that failed
```

with a full test suite pass, all successful. Once the callbacks are properly installed in the sample OpenCOBOL above, I'm sure the error 3 will be resolved for 1!:2 write to file as well as running the test suite from within JDo, which currently reports error 21. The above OpenCOBOL listing is the poor man's 10 minute guide to integrating J.

## 29.68 5.68 What is COBOLUnit?

A well documented, full featured Unit testing framework for COBOL, written in OpenCOBOL with a GPL license.

<http://sites.google.com/site/cobolunit/>

- Tutorials
- Installation instructions, with videos
- Open sources

Test suite configuration files look like:

```
<INIT>
  <SETNAME> SUITE-DELIVERY-COST
  <SETDESC> Tests Suite for delivery costs
<ADDSUITE>
  * Add a test
  <SETNAME> FRANCE-TO-ITALY
  <SETPROG> TS000011
  <SETDESC> IF FROM='FR' and TO='IT' then TAXES=120€
<ADDTEST>
<RUN>
```

and *with the scaffolding in place*, a success report looking like:

```
*****
COBOL UNIT : A COBOL FRAMEWORK FOR UNIT TESTS.
*****
```

```
COBOL UNIT Current release   : REL 1.00
COBOL UNIT Release date     : 2009-10-31
Language used for Logging    : EN
Verbosity Level of Log      : 1
End of the 'Testing Strategy Set up' Phase
Starting the 'Test Execution' Phase
|--- SUITE ' SUITE-DELIVERY-COST ' Running
|--- | TEST ' FRANCE-TO-ITALY ' Running
|    |- Assert ' FR => IT:TAX=120 ' success
|    |==> Test ' FRANCE-TO-ITALY ' * SUCCESS * ( 000000001 Assertions, 000000000 Failures, 0 errors)
|==> SUITE ' SUITE-DELIVERY-COST ' SUCCESS ( 000000000 test cases, 000000001 success, 000000000 fail)

*****
* SUCCESS * ( 000000001 Suites run, 000000001 succeed, 000000000 failed)
*****
( 00 min: 00 sec: 00 ms)
```

## 29.69 5.69 Can OpenCOBOL interface with Gambas?

Yes. See <http://code.google.com/p/gambascobolgui/downloads/list> for a working sample.

As a taster, the Gambas (<http://gambas.sourceforge.net/en/main.html>) sample calls OpenCOBOL coded as

OCOBOL

```
ENTRY "startGrid".
MOVE FCHIUSO TO GRID-FILE-STATE.
ACCEPT SOLODATA FROM DATE YYYYMMDD.
ACCEPT ORA FROM TIME.
MOVE DATAEORA TO STARTINGPOINT, PRMR-KEY-OF-LIGNE (GAP),
DATAEORA-KR.
PERFORM RWDWN.
MOVE 0 TO RETURN-CODE.
GOBACK.

ENTRY "fillrow" USING BY REFERENCE pRiga,
BY VALUE numRiga.
ADD 1 TO numRiga.
MOVE SUPER-LIGNE-PMP (numRiga) TO ROW-OUT.
SET pRiga TO ADDRESS OF ROW-OUT.
MOVE 0 TO RETURN-CODE.
GOBACK.
```

which this author found to be a pretty neat way of packaging OpenCOBOL *other language* callables.

The Gambas is nicely clean. Below being a snippet from the sample.

```
Extern cob_init(argc As Integer, argv As Integer) As Integer In "libcob"
Extern startGrid() As Integer In "SCONTO:69"
```

## 29.70 5.70 Does OpenCOBOL work with LLVM?

Yes. Almost first try for the February 2009 pre-release of 1.1. The compiler sources has a conditional use of a `-fno-gcse` switch that tripped warnings in clang causing some unit test failure reports. One change to compile out the `-fno-gcse` in `cobc/cobc.c`, and a simple:

```
$ sudo yum install llvm clang clang-analyzer clang-devel
$ export CC=clang
$ ./configure
OpenCOBOL Configuration:
  CC                clang
  COB_CC            clang
  CFLAGS           -O2
  COB_CFLAGS       -I/usr/local/include
  COB_EXTRA_FLAGS
  LDFLAGS
  COB_LDFLAGS
  COB_LIBS         -L${exec_prefix}/lib -lcob -lm -lgmp -lncurses -ldb
  COB_CONFIG_DIR   ${prefix}/share/open-cobol/config
  COB_COPY_DIR     ${prefix}/share/open-cobol/copy
  COB_LIBRARY_PATH ${exec_prefix}/lib/open-cobol
  COB_MODULE_EXT   so
  COB_SHARED_OPT   -shared
  COB_PIC_FLAGS    -fPIC -DPIC
  COB_EXPORT_DYN   -Wl,--export-dynamic
  COB_STRIP_CMD    strip --strip-unneeded
  Dynamic loading  System

$ scan-build make
scan-build: Removing directory '/tmp/scan-build-2012-05-23-2'
  because it contains no reports.

$ make check

# I had to make one change to cobc/cobc.c to remove -fno-gcse to avoid a
# bunch of make check 'failures' due to a warning about unused -fno-gcse

$ sudo make install
$ sudo ldconfig

# cobc is built with clang, and uses clang when compiling
# the .c generated from the .cob.

[btiffin@cobol]$ scan-build cobc -v -x hello.cob
scan-build: 'clang' executable not found in
  '/usr/lib64/clang-analyzer/scan-build/bin'.
scan-build: Using 'clang' from path: /usr/bin/clang
preprocessing hello.cob into /tmp/cob18158_0.cob
translating /tmp/cob18158_0.cob into /tmp/cob18158_0.c
clang -pipe -c -I/usr/local/include -Wno-unused -fsigned-char
-Wno-pointer-sign -o /tmp/cob18158_0.o /tmp/cob18158_0.c
clang -pipe -Wl,--export-dynamic -o hello /tmp/cob18158_0.o
-I/usr/local/lib -lcob -lm -lgmp -lncurses -ldb
scan-build: Removing directory '/tmp/scan-build-2012-05-23-2'
  because it contains no reports.

[btiffin@cobol]$ ./hello
Hello
```

```
[btiffin@cobol]$ ls -la hello
-rwxrwxr-x. 1 btiffin btiffin 9630 May 23 12:37 hello
```

And OpenCOBOL is good to go with clang and the LLVM universe. The above compiles OpenCOBOL with clang, and the installed `cobc` will use clang as the compiler after processing the COBOL sources. This is grand news in terms of anyone worried about OpenCOBOL viability into the future. The existant C ABI space and now the growing LLVM software pool. Nice.

## 29.71 5.71 Does OpenCOBOL interface with Python?

Yes. Embedding Python can be accomplished using only COBOL sources. Extending Python to allow calling COBOL modules, will usually require a small amount of glue code written in C.

Very high level Python embedding is pretty straight forward, been there, done that.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20130126
*> Purpose:     Embed Python
*> Tectonics:   cobc -x cobpy.cob -lpython2.6
*> *****
identification division.
program-id. cobpy.

procedure division.
call "Py_Initialize"
    on exception
        display "link cobpy with -lpython2.6" end-display
end-call
call "PyRun_SimpleString" using
    by reference
        "from time import time,ctime" & x"0a" &
        "print('Today is', ctime(time()))" & x"0a" & x"00"
    on exception continue
end-call
call "Py_Finalize" end-call
goback.
end program cobpy.
```

Giving:

```
$ cobc -x cobpy.cob -lpython2.6
$ ./cobpy
('Today is', 'Sat Jan 26 20:01:41 2013')
```

Python dutifully displayed the tuple. But what fun is Python if it is just for high level script side effects? Lots, but still.

Pure embedding.

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20130126
*> Purpose:     Embed Python
*> Tectonics:   cobc -x cobkat.cob -lpython2.6
*> NOTES:      leaks, no Py_DECREF macros called.
```

```
*> *****
identification division.
program-id. cobkat.

data division.
working-storage section.
77 python-name          usage pointer.
77 python-module        usage pointer.
77 python-dict          usage pointer.
77 python-func          usage pointer.
77 python-stringer     usage pointer.
77 python-args         usage pointer.
77 python-value        usage pointer.

01 cobol-buffer-pointer usage pointer.
01 cobol-buffer         pic x(80)          based.
01 cobol-string        pic x(80).

01 cobol-integer       usage binary-long.

01 command-line-args  pic x(80).

*> *****
procedure division.
call "Py_Initialize"
    on exception
        display "link cobpy with -lpython" end-display
end-call

*> Python likes module names in Unicode
call "PyUnicodeUCS4_FromString" using
    by reference "pythonfile" & x"00"
    returning python-name
    on exception
        display "unicode problem" end-display
end-call

*> import the module, using PYTHONPATH
call "PyImport_Import" using
    by value python-name
    returning python-module
    on exception
        display "this would be borked" end-display
end-call

if python-module equal null
    display "no pythonfile.py in PYTHONPATH" end-display
end-if

*> within the module, an attribute is "pythonfunction"
call "PyObject_GetAttrString" using
    by value python-module
    by reference "pythonfunction" & x"00"
    returning python-func
    on exception continue
end-call

*>
```



```

*> error handling now skimped out on
*>

*> pythonfunction takes a single argument
call "PyTuple_New" using
    by value 1
    returning python-args
end-call

*> of type long, hard coded to the ultimate answer
call "PyLong_FromLong" using
    by value 42
    returning python-value
end-call

*> set first (only) element of the argument tuple
call "PyTuple_SetItem" using
    by value python-args
    by value 0
    by value python-value
end-call

*> call the function, arguments marshalled for Python
call "PyObject_CallObject" using
    by value python-func
    by value python-args
    returning python-value
end-call

*> we know we get a long back, hopefully 1764
call "PyLong_AsLong" using
    by value python-value
    returning cobol-integer
end-call
display "Python returned: " cobol-integer end-display

*> ***** *
*> a function taking string and returning string
call "PyObject_GetAttrString" using
    by value python-module
    by reference "pythonstringer" & x"00"
    returning python-stringer
end-call

call "PyTuple_New" using
    by value 1
    returning python-args
end-call

*> Use the OpenCOBOL command argument
accept command-line-args from command-line end-accept
call "PyString_FromString" using
    by reference
        function concatenate(
            function trim(command-line-args)
            x"00")
    returning python-value
end-call

```

```
*> Set the function argument tuple to the cli args
call " PyTuple_SetItem " using
    by value python-args
    by value 0
    by value python-value
end-call

*> call the "pythonstringer" function
call " PyObject_CallObject " using
    by value python-stringer
    by value python-args
    returning python-value
end-call

*> return as String (with the MD5 hex digest tacked on)
call " PyString_AsString " using
    by value python-value
    returning cobol-buffer-pointer
end-call

*> one way of removing null while pulling data out of C
set address of cobol-buffer to cobol-buffer-pointer
string
    cobol-buffer delimited by x"00"
    into cobol-string
end-string
display "Python returned: " cobol-string end-display

*> and clear out <*>
call " Py_Finalize " end-call
goback.
end program cobkat.
```

with `pythonfile.py`

```
#
# Simple Python sample for OpenCOBOL embedding trial
#
def pythonfunction(i):
    return i * i

import hashlib
def pythonstringer(s):
    sum = hashlib.md5()
    sum.update(s)
    return s + ": " + sum.hexdigest()
```

Giving:

```
$ ./cobkat Python will use this for MD5 hash
no pythonfile.py in PYTHONPATH
Attempt to reference unallocated memory (Signal SIGSEGV)
Abnormal termination - File contents may be incorrect
```

Oops:

```
$ export PYTHONPATH=.
$ ./cobkat Python will use this for MD5 hash
```

Python returned: +0000001764

Python returned: Python will use this for MD5 hash: c5577e3ab8dea11adede20a1949b5fb3

*Oh, in case you're reading along, 1764 is the ultimate answer, squared.*

An OpenCOBOL source line like

```
set environment "PYTHONPATH" to "."
```

before `Py_Initialize` saves on the oops when you need to find current working directory Python scripts.

## 29.72 5.72 Can OpenCOBOL interface with Forth?

Yes, `ficl`, Forth Inspired Command Language embeds nicely.

*Ok, I said, easy, I meant almost easy, as I had to hunt down a `sysdep.h` file and could not get 4.10 to go, but 4.0.31 works the beauty, once the `sysdep.h` was put in place.*

First, the license compliance.

```

/*****
** f i c l . h
** Forth Inspired Command Language
** Author: John Sadler (john_sadler@alum.mit.edu)
** Created: 19 July 1997
** Dedicated to RHS, in loving memory
** $Id: //depot/gamejones/ficl/ficl.h#33 $
*****/
**
** Copyright (c) 1997-2001 John Sadler (john_sadler@alum.mit.edu)
** All rights reserved.
**
** Get the latest Ficl release at http://ficl.sourceforge.net
**
** I am interested in hearing from anyone who uses Ficl. If you have
** a problem, a success story, a defect, an enhancement request, or
** if you would like to contribute to the Ficl release, please
** contact me by email at the address above.
**
** L I C E N S E   a n d   D I S C L A I M E R
**
** Redistribution and use in source and binary forms, with or without
** modification, are permitted provided that the following conditions
** are met:
** 1. Redistributions of source code must retain the above copyright
**    notice, this list of conditions and the following disclaimer.
** 2. Redistributions in binary form must reproduce the above copyright
**    notice, this list of conditions and the following disclaimer in the
**    documentation and/or other materials provided with the distribution.
**
** THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
** ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
** IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
** ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
** FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
** DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
** OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
** HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT

```

```
** LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
** OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
** SUCH DAMAGE.
*/
```

And then the COBOL, **callficl.cob**

```
OCOBOL >>SOURCE FORMAT IS FIXED
*> *****
*> Author:      Brian Tiffin
*> Date:        20130220
*> Purpose:     Embed ficl
*> Tectonics:   cobb -x callficl.cob -lficl -L.
*>              LD_LIBRARY_PATH=. ./callficl
*> *****
identification division.
program-id. callficl.

data division.
working-storage section.
01 ficl-result  usage binary-long.
01 ficl-system  usage pointer.
01 ficl-vm      usage pointer.

*> *****
procedure division.
call "ficlSystemCreate" using
    by value 0
    returning ficl-system
end-call

display ficl-system end-display

call "ficlSystemCompileExtras" using
    by value ficl-system
end-call

call "ficlSystemCreateVm" using
    by value ficl-system
    returning ficl-vm
end-call

display ficl-vm end-display

call "ficlVmEvaluate" using
    by value ficl-vm
    by reference ".ver cr quit" & x"00"
    returning ficl-result
end-call

display ficl-result end-display

call "ficlVmEvaluate" using
    by value ficl-vm
    by reference
        ".( loading ooptest.fr ) cr load ooptest.fr" &
        x"0a" & " cr" & x"00"
    returning ficl-result
end-call
```

```

display ficl-result end-display

goback.
end program callficl.

```

and the test file **ooptest.fr**

```

\ OOP test stuff

only
also oop definitions

object subclass c-aggregate
c-byte obj: m0
c-byte obj: m1
c-4byte obj: m2
c-2byte obj: m3
end-class

object --> sub class1

cell: .a
cell: .b
: init
  locals| class inst |
    0 inst class --> .a !
    1 inst class --> .b !
;
end-class

class1 --> new c1inst

class1 --> sub class2
cell: .c
cell: .d

: init
  locals| class inst |
    inst class --> super --> init
    2 inst class --> .c !
    3 inst class --> .d !
;
end-class

class2 --> new c2inst

object subclass c-list
c-list ref: link
c-ref obj: payload
end-class

\ test stuff from ficl.html
.( metaclass methods ) cr
metaclass --> methods

cr .( c-foo class ) cr
object --> sub c-foo
cell: m_cell1

```

```

4 chars: m_chars
: init ( inst class -- )
  locals| class inst |
  0 inst class --> m_cell1 !
  inst class --> m_chars 4 0 fill
  ." initializing an instance of c_foo at " inst x. cr
;
end-class

```

```

.( c-foo instance methods... ) cr
c-foo --> new foo-instance
cr
foo-instance --> methods
foo-instance --> pedigree
cr
foo-instance 2dup
  --> methods
  --> pedigree
cr
c-foo --> see init
cr
foo-instance --> class --> see init

```

and finally, **the run**. The first two commands building up `ficl` and the `libficl` shared library, the next two for COBOL:

```

$ make -f Makefile.linux
$ make -f Makefile.linux main

```

```

$ cobc -g -debug -x callficl.cob -lficl -L .

```

```

$ LD_LIBRARY_PATH=./callficl

```

```

loading CORE EXT words
loading SEARCH & SEARCH-EXT words
loading Johns-Hopkins locals
loading MARKER
loading ficl O-O extensions
loading ficl utility classes
loading ficl string class
0x080569c0
0x08057928
Ficl version 4.0.31

-0000000056
loading ooptest.fr
metaclass methods
metaclassmethods:
debug see pedigree methods id offset-of sub
resume-class ref allot-array allot alloc-array alloc
new-array new array instance get-super get-wid
get-size .size .wid .super .do-instance
Dictionary: 24 words, 7786 cells used of 12288 total

```

```

c-foo class
c-foo instance methods...
initializing an instance of c_foo at 806043C

```

```

c-foomethods:
init m_chars m_cell1 .do-instance

```

Dictionary: 4 words, 7893 cells used of 12288 total

objectmethods:

```
debug  prev  next  index  methods size  pedigree  super
free   array-init  init  class  .do-instance
```

Dictionary: 13 words, 7893 cells used of 12288 total

c-foo object

c-foomethods:

```
init  m_chars m_cell1 .do-instance
```

Dictionary: 4 words, 7893 cells used of 12288 total

objectmethods:

```
debug  prev  next  index  methods size  pedigree  super
free   array-init  init  class  .do-instance
```

Dictionary: 13 words, 7893 cells used of 12288 total

c-foo object

: init

```
0  (link) (instruction 136)
1  2 (instruction 2)
2  (toLocal) (instruction 140), with argument 0 (0)
4  (toLocal) (instruction 140), with argument 1 (0x1)
6  0 (instruction 17)
7  (@local1) (instruction 146)
8  (@local0) (instruction 142)
9  s" m_cell1"
13 exec-method
14 ! (instruction 57)
15 (@local1) (instruction 146)
16 (@local0) (instruction 142)
17 s" m_chars"
21 exec-method
22 4 (instruction 4)
23 0 (instruction 17)
24 fill (instruction 111)
25 s" initializing an instance of c_foo at "
36 type
37 (@local1) (instruction 146)
38 x.
39 cr
40 (unlink) (instruction 137)
;
```

: init

```
0  (link) (instruction 136)
1  2 (instruction 2)
2  (toLocal) (instruction 140), with argument 0 (0)
4  (toLocal) (instruction 140), with argument 1 (0x1)
6  0 (instruction 17)
7  (@local1) (instruction 146)
8  (@local0) (instruction 142)
9  s" m_cell1"
13 exec-method
```

```

14  ! (instruction 57)
15  (@local1) (instruction 146)
16  (@local0) (instruction 142)
17  s" m_chars"
21  exec-method
22  4 (instruction 4)
23  0 (instruction 17)
24  fill (instruction 111)
25  s" initializing an instance of c_foo at "
36  type
37  (@local1) (instruction 146)
38  x.
39  cr
40  (unlink) (instruction 137)
;

-0000000257

```

Turns out that return codes -56 and -257 are ok codes, (from **ficl.h**):

```

#define FICL_VM_STATUS_QUIT      ( -56)    /* like FICL_VM_STATUS_ERROR_EXIT, but leave dataStack
#define FICL_VM_STATUS_OUT_OF_TEXT (-257) /* hungry - normal exit */

```

OpenCOBOL does Forth.

<http://ficl.sourceforge.net/>

*p.s. One small note. The ficl load word, load ooptest.fr needed a newline after the filename. Normally Forth uses a straight up space delimited word parser, but ficl accounts for filenames with spaces in them. Nice feature.*

## 29.73 5.73 Can OpenCOBOL interface with Shakespeare?

Yes. The refernece implementation of the Shakespeare Programming Language builds into OpenCOBOL applications that can CALL SPL modules.

Technicals: I downloaded Marlowe which fixes the reference implementation problem with Roman Numerals.

<https://bitbucket.org/kcartmell/marlowe/downloads>

Then inside a working dir (`_/lang/cobol/cobill/_` for instance) create `spl` and `untar`, and make SPL. *I assume the spl/ sub directory in the Makefile listed below.* What is happening here isn't runtime link loading, it is just building the SPL engine into COBOL, and then CALL the result of `spl2c`.

This first cut **lacks art**. Lacks. Sad, so verily verily sad.

**cobill.cob**

```

OCOBOL*> *****
*> Author:      Brian Tiffin
*> Date:        20130224
*> Purpose:     COBOL meets Shakespeare
*> Tectonics:   cbc -x -Ispl cobill.cob ocshake.c
*>              spl/libspl.c spl/strutils.c
*> pre-req:     spl2c ocshake.spl and an spl/ distribution
*> *****
identification division.
program-id. cobill.
procedure division.
call "ocshake" end-call

```



```
goback.  
end program cobill.
```

Then some cowardly SPL, **ocshake.spl**

The derp in SPL from OpenCOBOL.

Ajax, the loud mouth.

Dorcas, the d.  
Escalus, the e.  
Rosalind, the r.  
Prospero, the p.  
The Archbishop of Canterbury, the new line.

Act I: derping.

Scene I: derp.

[Enter Ajax and Dorcas]

Ajax:

You amazing beautiful fine charming gentle delicious door.  
You are as honest as the sum of a bold brave hard proud noble stone wall and thyself.  
You are as trustworthy as the sum of a proud rich tree and thyself.  
Speak your mind.

[Exit Dorcas]

[Enter Escalus]

Ajax:

You bluest peaceful smooth lovely warm embroidered summer's day.  
You are as beautiful as the sum of a fine honest fair sweet gentle wind and thyself.  
You are as lovely as the sum of a reddest sunny flower and thyself.  
You are as mighty as the sum of a sky and thyself.  
Speak your mind.

[Exit Escalus]

[Enter Rosalind]

Ajax:

You fair reddest sweet rich smooth blossoming red rose.  
You are as rich as the difference between thyself and a golden gentle clearest wind.  
You are as rich as the difference between thyself and a proud white lantern.  
You are as rich as the difference between thyself and a honest morning.  
Speak your mind.

[Exit Rosalind]

[Enter Prospero]

Ajax:

You proud prompt pretty loving gentle warm purple pony.  
You are as bold as the difference between thyself and an amazing cute delicious pretty purse.  
Speak your mind.

[Exeunt]

Scene II: a new line.

[Enter Ajax and The Archbishop of Canterbury]

Ajax:

You are nothing.  
You are a bold beautiful blossoming wind.  
You are as cunning as the sum of thyself and a tiny thing.  
Speak your mind!

[Exeunt]

**A Makefile of:**

```
cobill: ocshake.spl cobill.cob
    spl/spl2c <ocshake.spl >ocshake.c
    sed -i 's/int main(void)/int ocshake(void)/' ocshake.c
    cobc -x -Ispl cobill.cob ocshake.c spl/libspl.c spl/strutils.c
```

**Then a run of:**

```
$ make
spl/spl2c <ocshake.spl >ocshake.c
sed -i 's/int main(void)/int ocshake(void)/' ocshake.c
cobc -x -Ispl cobill.cob ocshake.c spl/libspl.c spl/strutils.c
$ ./cobill
derp
$
```

**derp**, in a 20K binary, from 2K of source.

*I am kinda proud of Scene II, that one reads well. The rest needs some Fahrenheit 451*

## 6 NOTES

### Notes

- 6.1 big-endian
- 6.2 little-endian
- 6.3 ASCII
- 6.4 currency symbol
- 6.5 DSO
- 6.6 errno
- 6.7 gdb
- 6.8 GMP
- 6.9 ISAM
- 6.10 line sequential
- 6.11 APT
- 6.12 ROBODoc Support
- 6.13 cobol.vim
- 6.14 make check listing
- 6.15 ABI
- 6.16 Tectonics
- 6.17 Setting Locale
- 6.18 GNU
- 6.19 Performing FOREVER?
- 6.20 POSIX

Notes

### 30.1 6.1 big-endian

Binary values stored with the most significant byte at the lowest memory address.

**Big End First.**

See <http://en.wikipedia.org/wiki/Endianness> for more details.

The OpenCOBOL compiler *default* storage format for `USAGE BINARY` and `COMP`.

## 30.2 6.2 little-endian

Binary values stored with the most significant byte at the highest memory address.

### Little End First.

<http://en.wikipedia.org/wiki/Endianness> for more details.

This is the common Intel architecture form, and **USAGE** clauses of COMPUTATIONAL-5, BINARY-CHAR, BINARY-SHORT, BINARY-LONG, BINARY-DOUBLE are a true performance boost on this hardware. See [http://www.opencobol.org/modules/bwiki/index.php?cmd=read&page=UserManual%2F4#content\\_1\\_0](http://www.opencobol.org/modules/bwiki/index.php?cmd=read&page=UserManual%2F4#content_1_0) for some details.

## 30.3 6.3 ASCII

American Symbolic Code for Information Interchange.

The character encoding common to personal computers and the early Internet Age, therefore OpenCOBOL. OpenCOBOL also supports the **EBCDIC** character encoding so some data transfers and keyboard handling or console display programs may need programmer attention to detail. Although this is a rare case as OpenCOBOL operates using an intelligent choice of encoding for each platform build.

See [http://en.wikipedia.org/wiki/American\\_Standard\\_Code\\_for\\_Information\\_Interchange](http://en.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange) for more info.

---

**Note:** Unicode? OpenCOBOL supports PIC N, a two-byte character field.

---

## 30.4 6.4 currency symbol

COBOL allows a **SPECIAL-NAMES** clause that determines the currency symbol. This effects both source codes and input/output **PICTURE** definitions.

```
CONFIGURATION SECTION.  
SPECIAL-NAMES.  
CURRENCY SIGN IS "#".
```

## 30.5 6.5 DSO

Dynamic Shared Objects.

Similar to but subtly different from *share libraries*.

## 30.6 6.6 errno

OpenCOBOL and C are fairly closely related as OpenCOBOL produces intermediate C source code and passes this off to another compiler.

Some C functions had no easy way to report out-of-bound errors so a global int **errno** is defined in the standard C library as a thread safe variable. Conscientious programmers will reset and test this variable for any and all functions documented as setting **errno**.

This is not straight forward for OpenCOBOL, but a small wrapper along the lines of

```
/* set/get errno */
#include <errno.h>

int reset_errno() {
    errno = 0;
    return errno;
}

int get_errno() {
    return errno;
}
/**/
```

exposes this critical run-time variable.

Usage:

```
$ cobc -c geterrno.c
$ cobc -x program.cob geterrno.o
```

and then something like

```
CALL "reset_errno" END-CALL
MOVE FUNCTION SQRT(-1) TO root
CALL "get_errno" RETURNING result END-CALL
IF result NOT EQUAL ZERO
    CALL "perror" USING NULL END-CALL
END-IF
```

Outputs:

```
Numerical argument out of domain
```

## 30.7 6.7 gdb

The GNU symbolic debugger. Big, deep, wide.

```
$ info gdb for the details.
```

or visit <http://www.gnu.org/software/gdb/documentation/>

## 30.8 6.8 GMP

GNU MP libgmp. GNU Library for decimal arithmetic. See <http://gmplib.org/> for complete details on the library advertised as *Arithmetic without limitations*.

## 30.9 6.9 ISAM

Indexed Sequential Access Method. A system to allow a variety of access methods for data records in file storage.

See <http://en.wikipedia.org/wiki/ISAM> for more details.

### 30.9.1 6.9.1 OpenCOBOL FILE STATUS codes

From [http://oldsite.add1tocobol.com/tiki-list\\_file\\_gallery.php?galleryId=1](http://oldsite.add1tocobol.com/tiki-list_file_gallery.php?galleryId=1) **statcodes.cpy** courtesy of John Ellis.

```
01  status-code          pic x(2) value spaces.
   88  SUCCESS           value '00'.
   88  SUCCESS_DUPLICATE value '02'.
   88  SUCCESS_INCOMPLETE value '04'.
   88  SUCCESS_OPTIONAL  value '05'.
   88  SUCCESS_NO_UNIT   value '07'.
   88  END_OF_FILE       value '10'.
   88  OUT_OF_KEY_RANGE  value '14'.
   88  KEY_INVALID       value '21'.
   88  KEY_EXISTS        value '22'.
   88  KEY_NOT_EXISTS    value '23'.
   88  PERMANENT_ERROR   value '30'.
   88  INCONSISTENT_FILENAME value '31'.
   88  BOUNDARY_VIOLATION value '34'.
   88  NOT_EXISTS        value '35'.
   88  PERMISSION_DENIED value '37'.
   88  CLOSED_WITH_LOCK  value '38'.
   88  CONFLICT_ATTRIBUTE value '39'.
   88  ALREADY_OPEN     value '41'.
   88  NOT_OPEN          value '42'.
   88  READ_NOT_DONE     value '43'.
   88  RECORD_OVERFLOW   value '44'.
   88  READ_ERROR        value '46'.
   88  INPUT_DENIED      value '47'.
   88  OUTPUT_DENIED     value '48'.
   88  I_O_DENIED        value '49'.
   88  RECORD_LOCKED    value '51'.
   88  END_OF_PAGE       value '52'.
   88  I_O_LINAGE        value '57'.
   88  FILE_SHARING      value '61'.
   88  NOT_AVAILABLE   value '91'.
```

Download and then in your WORKING-STORAGE SECTION use

```
COPY "statcodes.cpy".
```

## 30.10 6.10 line sequential

An access method for newline terminated files. OpenCOBOL reads each line and strips off carriage returns and line feeds. Filling the record buffer with the current line and padding with spaces.

## 30.11 6.11 APT

Advanced Package Tool. One of the strengths of the Debian GNU/Linux system. Allows for dependency checked binary packages.

## 30.12 6.12 ROBODoc Support

Below is a sample of a configuration file for using ROBODoc with OpenCOBOL programs.

```

# robodoc.rc for OpenCOBOL
#
items:
    NAME
    AUTHOR
    DATE
    PURPOSE
    TECTONICS
    SYNOPSIS
    INPUTS
    OUTPUTS
    SIDE EFFECTS
    HISTORY
    BUGS
    EXAMPLE
    SOURCE
ignore items:
    HISTORY
    BUGS
item order:
    PURPOSE
    SYNOPSIS
    INPUTS
    OUTPUTS
source items:
    SYNOPSIS
preformatted items:
    INPUTS
    OUTPUTS
format items:
    PURPOSE
    SIDE EFFECTS
options:
#   --src ./
#   --doc ./doc
    --html
    --syntaxcolors
#   --singledoc
#   --multidoc
    --index
    --tabsize 4
headertypes:
    J "Projects"           robo_projects    2
    F "Files"              robo_files      1
    e "Makefile Entries"  robo_mk_entries
    x "System Tests"      robo_sys_tests
    q Queries              robo_queries
ignore files:
    README
    CVS
    *.bak
    *~
    "a test_*"
accept files:
    *.cob
    *.COB
    *.cbl
    *.CBL

```

```

*.cpy
*.CPY
header markers:
*>****
remark markers:
*>
end markers:
*>****
header separate characters:
'
header ignore characters:
[
remark begin markers:
*>+
remark end markers:
*>-
source line comments:
*>
# OpenCOBOL keywords *><*
keywords:
accept
access
active-class
add
address
advancing
after
aligned
all
allocate
alphabet
alphabetic
alphabetic-lower
alphabetic-upper
alphanumeric
alphanumeric-edited
also
alter
alternate
and
any
anycase
are
area
areas
argument-number
argument-value
arithmetic
as
ascending
assign
at
attribute
auto
auto-skip
automatic
autoterminate
b-and
```



b-not  
b-or  
b-xor  
background-color  
based  
beep  
before  
bell  
binary  
binary-c-long  
binary-char  
binary-double  
binary-long  
binary-short  
bit  
blank  
blink  
block  
boolean  
bottom  
by  
byte-length  
call  
cancel  
cd  
center  
cf  
ch  
chain  
chaining  
character  
characters  
class  
class-id  
classification  
close  
code  
code-set  
col  
collating  
cols  
column  
columns  
comma  
command-line  
commit  
common  
communication  
comp  
comp-1  
comp-2  
comp-3  
comp-4  
comp-5  
comp-x  
computational  
computational-1  
computational-2

computational-3  
computational-4  
computational-5  
computational-x  
compute  
condition  
configuration  
constant  
contains  
content  
continue  
control  
controls  
converting  
copy  
corr  
corresponding  
count  
crt  
currency  
cursor  
cycle  
data  
data-pointer  
date  
day  
day-of-week  
de  
debugging  
decimal-point  
declaratives  
default  
delete  
delimited  
delimiter  
depending  
descending  
destination  
detail  
disable  
disk  
display  
divide  
division  
down  
duplicates  
dynamic  
ebcdic  
ec  
egi  
else  
emi  
enable  
end  
end-accept  
end-add  
end-call  
end-compute

end-delete  
end-display  
end-divide  
end-evaluate  
end-if  
end-multiply  
end-of-page  
end-perform  
end-read  
end-receive  
end-return  
end-rewrite  
end-search  
end-start  
end-string  
end-subtract  
end-unstring  
end-write  
entry  
entry-convention  
environment  
environment-name  
environment-value  
eo  
eol  
eop  
eos  
equal  
equals  
erase  
error  
escape  
esi  
evaluate  
exception  
exception-object  
exclusive  
exit  
expands  
extend  
external  
factory  
false  
fd  
file  
file-control  
file-id  
filler  
final  
first  
float-extended  
float-long  
float-short  
footing  
for  
foreground-color  
forever  
format

free  
from  
full  
function  
function-id  
generate  
get  
giving  
global  
go  
goback  
greater  
group  
group-usage  
heading  
high-value  
high-values  
highlight  
i-o  
i-o-control  
id  
identification  
if  
ignoring  
implements  
in  
index  
indexed  
indicate  
inherits  
initial  
initialize  
initialized  
initiate  
input  
input-output  
inspect  
interface  
interface-id  
into  
intrinsic  
invalid  
invoke  
is  
just  
justified  
key  
label  
last  
lc\_all  
lc\_collate  
lc\_ctype  
lc\_messages  
lc\_monetary  
lc\_numeric  
lc\_time  
leading  
left

length  
less  
limit  
limits  
linage  
linage-counter  
line  
line-counter  
lines  
linkage  
local-storage  
locale  
lock  
low-value  
low-values  
lowlight  
manual  
memory  
merge  
message  
method  
method-id  
minus  
mode  
move  
multiple  
multiply  
national  
national-edited  
native  
negative  
nested  
next  
no  
none  
normal  
not  
null  
nulls  
number  
numbers  
numeric  
numeric-edited  
object  
object-computer  
object-reference  
occurs  
of  
off  
omitted  
on  
only  
open  
optional  
options  
or  
order  
organization

other  
output  
overflow  
overline  
override  
packed-decimal  
padding  
page  
page-counter  
paragraph  
perform  
pf  
ph  
pic  
picture  
plus  
pointer  
position  
positive  
present  
previous  
printer  
printing  
procedure  
procedure-pointer  
procedures  
proceed  
program  
program-id  
program-pointer  
prompt  
property  
prototype  
purge  
queue  
quote  
quotes  
raise  
raising  
random  
rd  
read  
receive  
record  
recording  
records  
recursive  
redefines  
reel  
reference  
relation  
relative  
release  
remainder  
removal  
renames  
replace  
replacing

report  
reporting  
reports  
repository  
required  
reserve  
reset  
resume  
retry  
return  
returning  
reverse-video  
rewind  
rewrite  
rf  
rh  
right  
rollback  
rounded  
run  
same  
screen  
sd  
search  
seconds  
section  
secure  
segment  
select  
self  
send  
sentence  
separate  
sequence  
sequential  
set  
sharing  
sign  
signed  
signed-int  
signed-long  
signed-short  
size  
sort  
sort-merge  
source  
source-computer  
sources  
space  
spaces  
special-names  
standard  
standard-1  
standard-2  
start  
statement  
status  
step

stop  
string  
strong  
sub-queue-1  
sub-queue-2  
sub-queue-3  
subtract  
sum  
super  
suppress  
symbol  
symbolic  
sync  
synchronized  
system-default  
table  
tallying  
tape  
terminal  
terminate  
test  
text  
than  
then  
through  
thru  
time  
times  
to  
top  
trailing  
true  
type  
typedef  
ucs-4  
underline  
unit  
universal  
unlock  
unsigned  
unsigned-int  
unsigned-long  
unsigned-short  
unstring  
until  
up  
update  
upon  
usage  
use  
user-default  
using  
utf-16  
utf-8  
val-status  
valid  
validate  
validate-status



```

value
values
varying
when
with
working-storage
write
yyyyddd
yyyymmdd
zero
zeroes
zeros

```

To be used with

```
$ robodoc --src program.cob --doc program --singlefile --rc robocob.rc
```

Producing a nice HTML file documenting the program using embedded ROBODoc comment line directives. See [ROBODoc](#) for more information.

## 30.13 6.13 cobol.vim

Many thanks to the good people at [www.vim.org](http://www.vim.org):

```

" Vim syntax file
" Language: COBOL
" Maintainers: Davyd Ondrejko
" (formerly Sitaram Chamarty
" James Mitchell
" Last change: 2001 Sep 02

" For version 5.x: Clear all syntax items
" For version 6.x: Quit when a syntax file was already loaded

" Stephen Gennard
" - added keywords - AS, REPOSITORY
" - added extra cobolCall bits

if version < 600
  syntax clear
elseif exists("b:current_syntax")
  finish
endif

" MOST important - else most of the keywords wont work!
if version < 600
  set isk=@,48-57,-
else
  setlocal isk=@,48-57,-
endif

syn case ignore

if exists("cobol_legacy_code")
  syn match cobolKeys "^\\a\\{1,6\\}" contains=cobolReserved
else
  syn match cobolKeys "" contains=cobolReserved

```

endif

```

syn keyword cobolReserved contained ACCEPT ACCESS ADD ADDRESS ADVANCING AFTER ALPHABET ALPHABETIC
syn keyword cobolReserved contained ALPHABETIC-LOWER ALPHABETIC-UPPER ALPHANUMERIC ALPHANUMERIC-EDITED
syn keyword cobolReserved contained ALTERNATE AND ANY ARE AREA AREAS ASCENDING ASSIGN AT AUTHOR BEFORE
syn keyword cobolReserved contained BLANK BLOCK BOTTOM BY CANCEL CBLD CD CF CH CHARACTER CHARACTERS
syn keyword cobolReserved contained CLOCK-UNITS CLOSE COBOL CODE CODE-SET COLLATING COLUMN COMMA COMMA-SEPARATED
syn keyword cobolReserved contained COMMUNICATIONS COMPUTATIONAL COMPUTE CONFIGURATION CONTENT CONTINUE
syn keyword cobolReserved contained CONTROL CONVERTING CORR CORRESPONDING COUNT CURRENCY DATA DATE DATE-
syn keyword cobolReserved contained DATE-WRITTEN DAY DAY-OF-WEEK DE DEBUG-CONTENTS DEBUG-ITEM DEBUG-ITEMS
syn keyword cobolReserved contained DEBUG-NAME DEBUG-SUB-1 DEBUG-SUB-2 DEBUG-SUB-3 DEBUGGING DECIMAL-
syn keyword cobolReserved contained DELARATIVES DELETE DELIMITED DELIMITER DEPENDING DESCENDING DESTROY
syn keyword cobolReserved contained DETAIL DISABLE DISPLAY DIVIDE DIVISION DOWN DUPLICATES DYNAMIC EQUATION
syn keyword cobolReserved contained ENABLE END-ADD END-COMPUTE END-DELETE END-DIVIDE END-EVALUATE END-
syn keyword cobolReserved contained END-MULTIPLY END-OF-PAGE END-PERFORM END-READ END-RECEIVE END-RECEIVED
syn keyword cobolReserved contained END-REWRITE END-SEARCH END-START END-STRING END-SUBTRACT END-UNSTRING
syn keyword cobolReserved contained END-WRITE ENVIRONMENT EQUAL ERROR ESI EVALUATE EVERY EXCEPTION
syn keyword cobolReserved contained EXTEND EXTERNAL FALSE FD FILE FILE-CONTROL FILLER FINAL FIRST FOOTING
syn keyword cobolReserved contained GENERATE GIVING GLOBAL GREATER GROUP HEADING HIGH-VALUE HIGH-VALUE-
syn keyword cobolReserved contained I-O-CONTROL IDENTIFICATION IN INDEX INDEXED INDICATE INITIAL INITIALS
syn keyword cobolReserved contained INITIATE INPUT INPUT-OUTPUT INSPECT INSTALLATION INTO IS JUST
syn keyword cobolReserved contained JUSTIFIED KEY LABEL LAST LEADING LEFT LENGTH LOCK MEMORY
syn keyword cobolReserved contained MERGE MESSAGE MODE MODULES MOVE MULTIPLE MULTIPLY NATIVE NEGATIVE
syn keyword cobolReserved contained NUMBER NUMERIC NUMERIC-EDITED OBJECT-COMPUTER OCCURS OF OFF OMITTED
syn keyword cobolReserved contained OPTIONAL OR ORDER ORGANIZATION OTHER OUTPUT OVERFLOW PACKED-DECIMAL
syn keyword cobolReserved contained PAGE PAGE-COUNTER PERFORM PF PH PIC PICTURE PLUS POSITION POSITIVE
syn keyword cobolReserved contained PRINTING PROCEDURE PROCEDURES PROCEED PROGRAM PROGRAM-ID PURGE QUANTITY
syn keyword cobolReserved contained RANDOM RD READ RECEIVE RECORD RECORDS REDEFINES REEL REFERENCE REPEAT
syn keyword cobolReserved contained RELATIVE RELEASE REMAINDER REMOVAL REPLACE REPLACING REPORT REPORTS
syn keyword cobolReserved contained REPORTS RERUN RESERVE RESET RETURN RETURNING REVERSED REWIND REWRITE
syn keyword cobolReserved contained RIGHT ROUNDED SAME SD SEARCH SECTION SECURITY SEGMENT SEGMENT-LIMIT
syn keyword cobolReserved contained SELECT SEND SENTENCE SEPARATE SEQUENCE SEQUENTIAL SET SIGN SIZE SIGNED
syn keyword cobolReserved contained SORT-MERGE SOURCE SOURCE-COMPUTER SPECIAL-NAMES STANDARD
syn keyword cobolReserved contained STANDARD-1 STANDARD-2 START STATUS STRING SUB-QUEUE-1 SUB-QUEUE-2
syn keyword cobolReserved contained SUB-QUEUE-3 SUBTRACT SUM SUPPRESS SYMBOLIC SYNC SYNCHRONIZED TABLE
syn keyword cobolReserved contained TAPE TERMINAL TERMINATE TEST TEXT THAN THEN THROUGH THRU TIME TIME-
syn keyword cobolReserved contained TRAILING TRUE TYPE UNIT UNSTRING UNTIL UP UPON USAGE USE USING VARYING
syn keyword cobolReserved contained VARYING WHEN WITH WORDS WORKING-STORAGE WRITE

```

" new

```
syn keyword cobolReserved contained AS LOCAL-STORAGE LINKAGE SCREEN ENTRY
```

" new - btiffin

```
syn keyword cobolReserved contained END-ACCEPT END-DISPLAY
```

" new

```
syn keyword cobolReserved contained environment-name environment-value argument-number
```

```
syn keyword cobolReserved contained call-convention identified pointer
```

```
syn keyword cobolReserved contained external-form division wait national
```

" new -- oo stuff

```
syn keyword cobolReserved contained repository object class method-id method object static
```

```
syn keyword cobolReserved contained class-id class-control private inherits object-storage
```

```
syn keyword cobolReserved contained class-object protected delegate
```

```
syn keyword cobolReserved contained try catch raise end-try super property
```

```
syn keyword cobolReserved contained override instance equals
```

```

" new - new types
syn match cobolTypes "condition-value"hs=s,he=e
syn match cobolTypes "binary-char"hs=s,he=e
syn match cobolTypes "binary-c-long"hs=s,he=e
syn match cobolTypes "binary-long"hs=s,he=e
syn match cobolTypes "binary-short"hs=s,he=e
syn match cobolTypes "binary-double"hs=s,he=e
syn match cobolTypes "procedure-pointer"hs=s,he=e
syn match cobolTypes "object reference"hs=s,he=e

syn match cobolReserved contained "\<CONTAINS\>"
syn match cobolReserved contained "\<\ (IF\|ELSE\|INVALID\|END\|EOP\)\>"
syn match cobolReserved contained "\<ALL\>"

syn keyword cobolConstant SPACE SPACES NULL ZERO ZEROES ZEROS LOW-VALUE LOW-VALUES

syn keyword cobolReserved contained fold folder

if exists("cobol_legacy_code")
  syn match cobolMarker "^.\{6\}"
  syn match cobolBadLine "^.\{6\}[\^ D\-*$/].*"hs=s+6
  " If comment mark somehow gets into column past Column 7.
  syn match cobolBadLine "^.\{6\}\s\+.*"
endif

syn match cobolNumber "\<-\=\d*\.\=\d\+\>" contains=cobolMarker,cobolComment
syn match cobolPic "\<S*9\+\>" contains=cobolMarker,cobolComment
syn match cobolPic "\<$*\.\.=9\+\>" contains=cobolMarker,cobolComment
syn match cobolPic "\<Z*\.\.=9\+\>" contains=cobolMarker,cobolComment
syn match cobolPic "\<V9\+\>" contains=cobolMarker,cobolComment
syn match cobolPic "\<9\+V\>" contains=cobolMarker,cobolComment
syn match cobolPic "\<-\+[Z9]\+\>" contains=cobolMarker,cobolComment
syn match cobolTodo "todo" contained

if exists("cobol_mf_syntax")
  syn region cobolComment start="*>" end="*$" contains=cobolTodo,cobolMarker
endif

syn keyword cobolGoTo GO GOTO
syn keyword cobolCopy COPY

" cobolBAD: things that are BAD NEWS!
syn keyword cobolBAD ALTER ENTER RENAMES

" cobolWatch: things that are important when trying to understand a program
syn keyword cobolWatch OCCURS DEPENDING VARYING BINARY COMP REDEFINES
syn keyword cobolWatch REPLACING THROW
syn match cobolWatch "COMP-[123456XN]"

" new - btiffin, added Intrinsics
syn keyword cobolWatch ABS ACOS ANNUITY ASIN ATAN BYTE-LENGTH CHAR
syn keyword cobolWatch COS CURRENT-DATE DATE-OF-INTEGGER DATE-TO-YYYYMMDD
syn keyword cobolWatch DAY-OF-INTEGGER DAY-TO-YYYYDDD E EXCEPTION-FILE
syn keyword cobolWatch EXCEPTION-LOCATION EXCEPTION-STATEMENT
syn keyword cobolWatch EXCEPTION-STATUS EXP EXP10 FACTORIAL FRACTION-PART
syn keyword cobolWatch INTEGGER INTEGGER-OF-DATE INTEGGER-OF-DAY INTEGGER-PART
syn keyword cobolWatch LENGTH LOCALE-DATE LOCALE-TIME LOG LOG10 LOWER-CASE
syn keyword cobolWatch MAX MEAN MEDIAN MIDRANGE MIN MOD NUMVAL NUMVAL-C

```

```

syn keyword cobolWatch ORD ORD-MAX ORD-MIN PI PRESENT-VALUE RANDOM RANGE
syn keyword cobolWatch REM REVERSE SECONDS-FROM-FORMATTED-TIME
syn keyword cobolWatch SECONDS-PAST-MIDNIGHT SIGN SIN SQRT
syn keyword cobolWatch STANDARD-DEVIATION STORED-CHAR-LENGTH SUM TAN
syn keyword cobolWatch SUBSTITUTE SUBSTITUTE-CASE
syn keyword cobolWatch TEST-DATE-YYMMDD TEST-DAY-YYYYDDD TRIM UPPER-CASE
syn keyword cobolWatch VARIANCE WHEN-COMPILED YEAR-TO-YYYY

syn region cobolEXECs contains=cobolLine start="EXEC " end="END-EXEC"

syn match cobolComment "^.\{6\}\*.*"hs=s+6 contains=cobolTodo,cobolMarker
syn match cobolComment "^.\{6\}/.*"hs=s+6 contains=cobolTodo,cobolMarker
syn match cobolComment "^.\{6\}C.*"hs=s+6 contains=cobolTodo,cobolMarker

if exists("cobol_legacy_code")
  syn match cobolCompiler "^.\{6\}$.*"hs=s+6
  syn match cobolDecl "^.\{6\} \{1,8\}\(0\=1\|77\|78\) "hs=s+7,he=e-1 contains=cobolMarker
  syn match cobolDecl "^.\{6\} \+[1-8]\d "hs=s+7,he=e-1 contains=cobolMarker
  syn match cobolDecl "^.\{6\} \+0\=[2-9] "hs=s+7,he=e-1 contains=cobolMarker
  syn match cobolDecl "^.\{6\} \+66 "hs=s+7,he=e-1 contains=cobolMarker
  syn match cobolWatch "^.\{6\} \+88 "hs=s+7,he=e-1 contains=cobolMarker
else
  syn match cobolWhiteSpace "^*\[ \t]"
  syn match cobolCompiler "$.*"hs=s,he=e contains=cobolWhiteSpace,cobolTypes
  syn match cobolDecl "0\=[1-9] *$"hs=s,he=e-1 contains=cobolWhiteSpace,cobolTypes
  syn match cobolDecl "66 *$"hs=s,he=e-1 contains=cobolWhiteSpace,cobolTypes
  syn match cobolWatch "88 *$"hs=s,he=e-1 contains=cobolWhiteSpace,cobolTypes
endif

syn match cobolBadID "\k\+--\($|[^\-A-Z0-9]\)"

syn keyword cobolCALLs CALL CANCEL GOBACK INVOKE PERFORM END-PERFORM END-CALL RUN
syn match cobolCALLs "STOP \+RUN"
syn match cobolCALLs "EXIT \+PROGRAM"
syn match cobolCALLs "EXIT \+PROGRAM \+RETURNING"
syn match cobolCALLs "EXIT \+PERFORM"
syn match cobolCALLs "EXIT \+METHOD"
syn match cobolCALLs "EXIT \+SECTION"
syn match cobolCALLs "STOP " contains=cobolString

syn match cobolExtras /\<VALUE \+d\+\.\/hs=s+6,he=e-1

" zero terminated strings eg: pic x(10) value z"My C String"
if exists("cobol_mf_syntax")
  syn match cobolString /z"[^"]*\("\|\$\)/
endif

syn match cobolString /"[^"]*\("\|\$\)/
syn match cobolString /'[^']*\('\|\$\)/

" new - btiffin, added libcob calls
syn match cobolWatch /\(["']\)SYSTEM\1/
syn match cobolWatch /["']CBL_ERROR_PROC["']/
syn match cobolWatch /["']CBL_EXIT_PROC["']/
syn match cobolWatch /["']CBL_OPEN_FILE["']/
syn match cobolWatch /["']CBL_CREATE_FILE["']/
syn match cobolWatch /["']CBL_READ_FILE["']/
syn match cobolWatch /["']CBL_WRITE_FILE["']/

```

```

syn match cobolWatch /["']CBL_CLOSE_FILE["']/
syn match cobolWatch /["']CBL_FLUSH_FILE["']/
syn match cobolWatch /["']CBL_DELETE_FILE["']/
syn match cobolWatch /["']CBL_COPY_FILE["']/
syn match cobolWatch /["']CBL_CHECK_FILE_EXIST["']/
syn match cobolWatch /["']CBL_RENAME_FILE["']/
syn match cobolWatch /["']CBL_GET_CURRENT_DIR["']/
syn match cobolWatch /["']CBL_CHANGE_DIR["']/
syn match cobolWatch /["']CBL_CREATE_DIR["']/
syn match cobolWatch /["']CBL_DELETE_DIR["']/
syn match cobolWatch /["']CBL_AND["']/
syn match cobolWatch /["']CBL_OR["']/
syn match cobolWatch /["']CBL_NOR["']/
syn match cobolWatch /["']CBL_XOR["']/
syn match cobolWatch /["']CBL_IMP["']/
syn match cobolWatch /["']CBL_NIMP["']/
syn match cobolWatch /["']CBL_EQ["']/
syn match cobolWatch /["']CBL_NOT["']/
syn match cobolWatch /["']CBL_TOUPPER["']/
syn match cobolWatch /["']CBL_TOLOWER["']/
syn match cobolWatch /["']\\364["']/
syn match cobolWatch /["']\\365["']/
syn match cobolWatch /["']\\221["']/
syn match cobolWatch /["']C$NARG["']/
syn match cobolWatch /["']C$PARAMSIZE["']/
syn match cobolWatch /["']C$MAKEDIR["']/
syn match cobolWatch /["']C$CHDIR["']/
syn match cobolWatch /["']C$SLEEP["']/
syn match cobolWatch /["']C$COPY["']/
syn match cobolWatch /["']C$FILEINFO["']/
syn match cobolWatch /["']C$DELETE["']/
syn match cobolWatch /["']C$TOUPPER["']/
syn match cobolWatch /["']C$TOLOWER["']/
syn match cobolWatch /["']C$JUSTIFY["']/
syn match cobolWatch /["']CBL_OC_NANOSLEEP["']/

if exists("cobol_legacy_code")
  syn region cobolCondFlow contains=ALLBUT,cobolLine start="\<\(IF\|INVALID\|END\|EOP\)\>"
    skip=\/(\/\|"\)[^"]\{-\}\(\/\|'\|$\)/ end="\." keepend
  syn region cobolLine start="^\.\{6\} " end="$" contains=ALL
endif

if exists("cobol_legacy_code")
  " catch junk in columns 1-6 for modern code
  syn match cobolBAD "^\{0,5\}[^ ].*"
endif

" many legacy sources have junk in columns 1-6: must be before others
" Stuff after column 72 is in error - must be after all other "match" entries
if exists("cobol_legacy_code")
  syn match cobolBadLine "^\.\{6\}[^*./].\{66,\}"
endif

" Define the default highlighting.
" For version 5.7 and earlier: only when not done already
" For version 5.8 and later: only when an item doesn't have highlighting yet
if version >= 508 || !exists("did_cobol_syntax_inits")
  if version < 508

```

```
    let did_cobol_syntax_inits = 1
    command -nargs=+ HiLink hi link <args>
else
    command -nargs=+ HiLink hi def link <args>
endif
HiLink cobolBAD Error
HiLink cobolBadID Error
HiLink cobolBadLine Error
HiLink cobolMarker Comment
HiLink cobolCALLs Function
HiLink cobolComment Comment
HiLink cobolKeys Comment
HiLink cobolCompiler PreProc
HiLink cobolEXECs PreProc
HiLink cobolCondFlow Special
HiLink cobolCopy PreProc
HiLink cobolDecl Type
HiLink cobolTypes Type
HiLink cobolExtras Special
HiLink cobolGoTo Special
HiLink cobolConstant Constant
HiLink cobolNumber Constant
HiLink cobolPic Constant
HiLink cobolReserved Statement
HiLink cobolString Constant
HiLink cobolTodo Todo
HiLink cobolWatch Special
delcommand HiLink
endif

let b:current_syntax = "cobol"

" vim: ts=6 nowrap
```

## 30.14 6.14 make check listing

A make check from February 2009:

```
## ----- ##
## OpenCOBOL 1.1 test suite: Syntax Tests. ##
## ----- ##
 1: COPY: file not found                ok
 2: COPY: replacement order             ok
 3: COPY: separators                    ok
 4: COPY: partial replacement           ok
 5: COPY: recursive replacement         ok
 6: Invalid PROGRAM-ID                  ok
 7: Invalid PROGRAM-ID type clause (1) ok
 8: Invalid PROGRAM-ID type clause (2) ok
 9: Undefined data name                 ok
10: Undefined group name                ok
11: Undefined data name in group        ok
12: Reference not a group name           ok
13: Incomplete 01 definition            ok
14: Same labels in different sections   ok
15: Redefinition of 01 items            ok
```

---

16: Redefinition of 01 and 02 items	ok
17: Redefinition of 02 items	ok
18: Redefinition of 77 items	ok
19: Redefinition of 01 and 77 items	ok
20: Redefinition of 88 items	ok
21: Ambiguous reference to 02 items	ok
22: Ambiguous reference to 02 and 03 items	ok
23: Ambiguous reference with qualification	ok
24: Unique reference with ambiguous qualifiers	ok
25: Undefined procedure name	ok
26: Redefinition of section names	ok
27: Redefinition of section and paragraph names	ok
28: Redefinition of paragraph names	ok
29: Ambiguous reference to paragraph name	ok
30: Non-matching level numbers (extension)	ok
31: Ambiguous AND/OR	ok
32: START on SEQUENTIAL file	ok
33: Subscripted item requires OCCURS clause	ok
34: The number of subscripts	ok
35: OCCURS with level 01, 66, 77, and 88	ok
36: OCCURS with variable-occurrence data item	ok
37: Nested OCCURS clause	ok
38: OCCURS DEPENDING followed by another field	ok
39: OCCURS DEPENDING without TO clause	ok
40: REDEFINES: not following entry-name	ok
41: REDEFINES: level 02 by 01	ok
42: REDEFINES: level 03 by 02	ok
43: REDEFINES: level 66	ok
44: REDEFINES: level 88	ok
45: REDEFINES: lower level number	ok
46: REDEFINES: with OCCURS	ok
47: REDEFINES: with subscript	ok
48: REDEFINES: with variable occurrence	ok
49: REDEFINES: with qualification	ok
50: REDEFINES: multiple redefinition	ok
51: REDEFINES: size exceeds	ok
52: REDEFINES: with VALUE	ok
53: REDEFINES: with intervention	ok
54: REDEFINES: within REDEFINES	ok
55: Numeric item (integer)	ok
56: Numeric item (non-integer)	ok
57: Numeric item with picture P	ok
58: Signed numeric literal	ok
59: Alphabetic item	ok
60: Alphanumeric item	ok
61: Alphanumeric group item	ok
62: Numeric-edited item	ok
63: Alphanumeric-edited item	ok
64: MOVE SPACE TO numeric or numeric-edited item	ok
65: MOVE ZERO TO alphabetic item	ok
66: MOVE alphabetic TO x	ok
67: MOVE alphanumeric TO x	ok
68: MOVE alphanumeric-edited TO x	ok
69: MOVE numeric (integer) TO x	ok
70: MOVE numeric (non-integer) TO x	ok
71: MOVE numeric-edited TO x	ok
72: Operands must be groups	ok
73: MOVE: misc	ok

```
74: Category check of Format 1          ok
75: Category check of Format 2          ok
76: Category check of literals          ok
77: SET: misc                            ok
```

```
## ----- ##
## Test results. ##
## ----- ##
```

All 77 tests were successful.

PASS: ./syntax

```
## ----- ##
## OpenCOBOL 1.1 test suite: Run Tests. ##
## ----- ##
```

```
1: DISPLAY literals                      ok
2: DISPLAY literals, DECIMAL-POINT is COMMA ok
3: Hexadecimal literal                   ok
4: DISPLAY data items with VALUE clause  ok
5: DISPLAY data items with MOVE statement ok
6: GLOBAL at same level                  ok
7: GLOBAL at lower level                 ok
8: non-numeric subscript                 ok
9: The range of subscripts                ok
10: Subscript out of bounds (1)           ok
11: Subscript out of bounds (2)          ok
12: Value of DEPENDING ON N out of bounds (lower)ok
13: Value of DEPENDING ON N out of bounds (upper)ok
14: Subscript bounds with ODO (lower)    ok
15: Subscript bounds with ODO (upper)    ok
16: Subscript bounds with ODO            ok
17: Subscript by arithmetic expression   ok
18: Separate sign positions              ok
19: Static reference modification         ok
20: Dynamic reference modification        ok
21: Static out of bounds                  ok
22: Offset underflow                     ok
23: Offset overflow                       ok
24: Length underflow                     ok
25: Length overflow                       ok
26: ACCEPT                                ok
27: INITIALIZE group entry with OCCURS   ok
28: INITIALIZE OCCURS with numeric edited ok
29: INITIALIZE complex group (1)         ok
30: INITIALIZE complex group (2)         ok
31: INITIALIZE with REDEFINES            ok
32: Source file not found                 ok
33: Comma separator without space         ok
34: LOCAL-STORAGE                        ok
35: EXTERNAL data item                   ok
36: EXTERNAL AS data item                 ok
37: cobcrun validation                    ok
38: MOVE to itself                        ok
39: MOVE with refmod                      ok
40: MOVE with refmod (variable)          ok
41: MOVE with group refmod                ok
42: MOVE indexes                          ok
43: MOVE X'00'                            ok
44: Level 01 subscripts                   ok
```



---

45: Class check with reference modification	ok
46: Index and parenthesized expression	ok
47: Alphanumeric and binary numeric	ok
48: Dynamic call with static linking	ok
49: CALL m1. CALL m2. CALL m1.	ok
50: CALL binary literal parameter/LENGTH OF	ok
51: INSPECT REPLACING LEADING ZEROS BY SPACES	ok
52: INSPECT: No repeat conversion check	ok
53: INSPECT: REPLACING figurative constant	ok
54: INSPECT: TALLYING BEFORE	ok
55: INSPECT: TALLYING AFTER	ok
56: INSPECT REPLACING TRAILING ZEROS BY SPACES	ok
57: INSPECT REPLACING complex	ok
58: SWITCHES	ok
59: Nested PERFORM	ok
60: EXIT PERFORM	ok
61: EXIT PERFORM CYCLE	ok
62: EXIT PARAGRAPH	ok
63: EXIT SECTION	ok
64: 88 with FILLER	ok
65: Non-overflow after overflow	ok
66: PERFORM ... CONTINUE	ok
67: STRING with subscript reference	ok
68: UNSTRING DELIMITED ALL LOW-VALUE	ok
69: READ INTO AT-END sequence	ok
70: First READ on empty SEQUENTIAL INDEXED file	ok
71: REWRITE a RELATIVE file with RANDOM access	ok
72: SORT: table sort	ok
73: SORT: EBCDIC table sort	ok
74: SORT nonexistent file	ok
75: PIC ZZZ-, ZZZ+	ok
76: Larger REDEFINES lengths	ok
77: PERFORM type OSVS	ok
78: Sticky LINKAGE	ok
79: COB_PRE_LOAD test	ok
80: COB_LOAD_CASE=UPPER test	ok
81: 88 level with FALSE IS clause	ok
82: ALLOCATE/FREE with BASED item	ok
83: INITIALIZATE with reference modification	ok
84: CALL with OMITTED parameter	ok
85: ANY LENGTH	ok
86: BASED item non-ALLOCATED (debug)	ok
87: COMP-5	ok
88: Hexadecimal numeric literal	ok
89: Semi-parenthesized condition	ok
90: ADDRESS OF	ok
91: LENGTH OF	ok
92: WHEN-COMPILED	ok
93: Complex OCCURS DEPENDING ON	ok
94: MOVE NON-INTEGGER TO ALPHA-NUMERIC	ok
95: CALL USING file-name	ok
96: CALL unusual PROGRAM-ID.	ok
97: Case independent PROGRAM-ID	ok
98: PROGRAM-ID AS clause	ok
99: Quoted PROGRAM-ID	ok
100: ASSIGN MF	ok
101: ASSIGN IBM	ok
102: ASSIGN mapping	ok

103: ASSIGN expansion	ok
104: ASSIGN with COB_FILE_PATH	ok
105: NUMBER-OF-CALL-PARAMETERS	ok
106: PROCEDURE DIVISION USING BY ...	ok
107: PROCEDURE DIVISION CHAINING ...	ok
108: STOP RUN RETURNING	ok
109: ENTRY	ok
110: LINE SEQUENTIAL write	ok
111: LINE SEQUENTIAL read	ok
112: ASSIGN to KEYBOARD/DISPLAY	ok
113: Environment/Argument variable	ok
114: DECIMAL-POINT is COMMA (1)	ok
115: DECIMAL-POINT is COMMA (2)	ok
116: DECIMAL-POINT is COMMA (3)	ok
117: DECIMAL-POINT is COMMA (4)	ok
118: DECIMAL-POINT is COMMA (5)	ok
119: 78 Level (1)	ok
120: 78 Level (2)	ok
121: 78 Level (3)	ok
122: Unreachable statement	ok
123: RETURN-CODE moving	ok
124: RETURN-CODE passing	ok
125: RETURN-CODE nested	ok
126: FUNCTION ABS	ok
127: FUNCTION ACOS	ok
128: FUNCTION ANNUITY	ok
129: FUNCTION ASIN	ok
130: FUNCTION ATAN	ok
131: FUNCTION CHAR	ok
132: FUNCTION COMBINED-DATETIME	ok
133: FUNCTION CONCATENATE	ok
134: FUNCTION CONCATENATE with reference modding	ok
135: FUNCTION COS	ok
136: FUNCTION DATE-OF-INTEGERS	ok
137: FUNCTION DATE-TO-YYYYMMDD	ok
138: FUNCTION DAY-OF-INTEGERS	ok
139: FUNCTION DAY-TO-YYYYDDD	ok
140: FUNCTION E	ok
141: FUNCTION EXCEPTION-FILE	ok
142: FUNCTION EXCEPTION-LOCATION	ok
143: FUNCTION EXCEPTION-STATEMENT	ok
144: FUNCTION EXCEPTION-STATUS	ok
145: FUNCTION EXP	ok
146: FUNCTION FACTORIAL	ok
147: FUNCTION FRACTION-PART	ok
148: FUNCTION INTEGER	ok
149: FUNCTION INTEGER-OF-DATE	ok
150: FUNCTION INTEGER-OF-DAY	ok
151: FUNCTION INTEGER-PART	ok
152: FUNCTION LENGTH	ok
153: FUNCTION LOCALE-DATE	ok
154: FUNCTION LOCALE-TIME	ok
155: FUNCTION LOCALE-TIME-FROM-SECONDS	ok
156: FUNCTION LOG	ok
157: FUNCTION LOG10	ok
158: FUNCTION LOWER-CASE	ok
159: FUNCTION LOWER-CASE with reference modding	ok
160: FUNCTION MAX	ok

```

161: FUNCTION MEAN ok
162: FUNCTION MEDIAN ok
163: FUNCTION MIDRANGE ok
164: FUNCTION MIN ok
165: FUNCTION MOD ok
166: FUNCTION NUMVAL ok
167: FUNCTION NUMVAL-C ok
168: FUNCTION ORD ok
169: FUNCTION ORD-MAX ok
170: FUNCTION ORD-MIN ok
171: FUNCTION PI ok
172: FUNCTION PRESENT-VALUE ok
173: FUNCTION RANGE ok
174: FUNCTION REM ok
175: FUNCTION REVERSE ok
176: FUNCTION REVERSE with reference modding ok
177: FUNCTION SECONDS-FROM-FORMATTED-TIME ok
178: FUNCTION SECONDS-PAST-MIDNIGHT ok
179: FUNCTION SIGN ok
180: FUNCTION SIN ok
181: FUNCTION SQRT ok
182: FUNCTION STANDARD-DEVIATION ok
183: FUNCTION STORED-CHAR-LENGTH ok
184: FUNCTION SUBSTITUTE ok
185: FUNCTION SUBSTITUTE with reference modding ok
186: FUNCTION SUBSTITUTE-CASE ok
187: FUNCTION SUBSTITUTE-CASE with reference mod ok
188: FUNCTION TAN ok
189: FUNCTION TRIM ok
190: FUNCTION TRIM with reference modding ok
191: FUNCTION UPPER-CASE ok
192: FUNCTION UPPER-CASE with reference modding ok
193: FUNCTION VARIANCE ok
194: FUNCTION WHEN-COMPILED ok

```

```

## ----- ##
## Test results. ##
## ----- ##

```

```

All 194 tests were successful.
PASS: ./run

```

```

## Run time tests with -O option ##

```

```

## ----- ##
## OpenCOBOL 1.1 test suite: Run Tests. ##
## ----- ##
  1: DISPLAY literals ok
  2: DISPLAY literals, DECIMAL-POINT is COMMA ok
  3: Hexadecimal literal ok
  4: DISPLAY data items with VALUE clause ok
  5: DISPLAY data items with MOVE statement ok
  6: GLOBAL at same level ok
  7: GLOBAL at lower level ok
  8: non-numeric subscript ok
  9: The range of subscripts ok
 10: Subscript out of bounds (1) ok
 11: Subscript out of bounds (2) ok

```

12: Value of DEPENDING ON N out of bounds (lower)ok  
13: Value of DEPENDING ON N out of bounds (upper)ok  
14: Subscript bounds with ODO (lower) ok  
15: Subscript bounds with ODO (upper) ok  
16: Subscript bounds with ODO ok  
17: Subscript by arithmetic expression ok  
18: Separate sign positions ok  
19: Static reference modification ok  
20: Dynamic reference modification ok  
21: Static out of bounds ok  
22: Offset underflow ok  
23: Offset overflow ok  
24: Length underflow ok  
25: Length overflow ok  
26: ACCEPT ok  
27: INITIALIZE group entry with OCCURS ok  
28: INITIALIZE OCCURS with numeric edited ok  
29: INITIALIZE complex group (1) ok  
30: INITIALIZE complex group (2) ok  
31: INITIALIZE with REDEFINES ok  
32: Source file not found ok  
33: Comma separator without space ok  
34: LOCAL-STORAGE ok  
35: EXTERNAL data item ok  
36: EXTERNAL AS data item ok  
37: cobcrun validation ok  
38: MOVE to itself ok  
39: MOVE with refmod ok  
40: MOVE with refmod (variable) ok  
41: MOVE with group refmod ok  
42: MOVE indexes ok  
43: MOVE X'00' ok  
44: Level 01 subscripts ok  
45: Class check with reference modification ok  
46: Index and parenthesized expression ok  
47: Alphanumeric and binary numeric ok  
48: Dynamic call with static linking ok  
49: CALL m1. CALL m2. CALL m1. ok  
50: CALL binary literal parameter/LENGTH OF ok  
51: INSPECT REPLACING LEADING ZEROS BY SPACES ok  
52: INSPECT: No repeat conversion check ok  
53: INSPECT: REPLACING figurative constant ok  
54: INSPECT: TALLYING BEFORE ok  
55: INSPECT: TALLYING AFTER ok  
56: INSPECT REPLACING TRAILING ZEROS BY SPACES ok  
57: INSPECT REPLACING complex ok  
58: SWITCHES ok  
59: Nested PERFORM ok  
60: EXIT PERFORM ok  
61: EXIT PERFORM CYCLE ok  
62: EXIT PARAGRAPH ok  
63: EXIT SECTION ok  
64: 88 with FILLER ok  
65: Non-overflow after overflow ok  
66: PERFORM ... CONTINUE ok  
67: STRING with subscript reference ok  
68: UNSTRING DELIMITED ALL LOW-VALUE ok  
69: READ INTO AT-END sequence ok

---

70: First READ on empty SEQUENTIAL INDEXED file	ok
71: REWRITE a RELATIVE file with RANDOM access	ok
72: SORT: table sort	ok
73: SORT: EBCDIC table sort	ok
74: SORT nonexistent file	ok
75: PIC ZZZ-, ZZZ+	ok
76: Larger REDEFINES lengths	ok
77: PERFORM type OSVS	ok
78: Sticky LINKAGE	ok
79: COB_PRE_LOAD test	ok
80: COB_LOAD_CASE=UPPER test	ok
81: 88 level with FALSE IS clause	ok
82: ALLOCATE/FREE with BASED item	ok
83: INITIALIZER with reference modification	ok
84: CALL with OMITTED parameter	ok
85: ANY LENGTH	ok
86: BASED item non-ALLOCATED (debug)	ok
87: COMP-5	ok
88: Hexadecimal numeric literal	ok
89: Semi-parenthesized condition	ok
90: ADDRESS OF	ok
91: LENGTH OF	ok
92: WHEN-COMPILED	ok
93: Complex OCCURS DEPENDING ON	ok
94: MOVE NON-INTEGERS TO ALPHA-NUMERIC	ok
95: CALL USING file-name	ok
96: CALL unusual PROGRAM-ID.	ok
97: Case independent PROGRAM-ID	ok
98: PROGRAM-ID AS clause	ok
99: Quoted PROGRAM-ID	ok
100: ASSIGN MF	ok
101: ASSIGN IBM	ok
102: ASSIGN mapping	ok
103: ASSIGN expansion	ok
104: ASSIGN with COB_FILE_PATH	ok
105: NUMBER-OF-CALL-PARAMETERS	ok
106: PROCEDURE DIVISION USING BY ...	ok
107: PROCEDURE DIVISION CHAINING ...	ok
108: STOP RUN RETURNING	ok
109: ENTRY	ok
110: LINE SEQUENTIAL write	ok
111: LINE SEQUENTIAL read	ok
112: ASSIGN to KEYBOARD/DISPLAY	ok
113: Environment/Argument variable	ok
114: DECIMAL-POINT is COMMA (1)	ok
115: DECIMAL-POINT is COMMA (2)	ok
116: DECIMAL-POINT is COMMA (3)	ok
117: DECIMAL-POINT is COMMA (4)	ok
118: DECIMAL-POINT is COMMA (5)	ok
119: 78 Level (1)	ok
120: 78 Level (2)	ok
121: 78 Level (3)	ok
122: Unreachable statement	ok
123: RETURN-CODE moving	ok
124: RETURN-CODE passing	ok
125: RETURN-CODE nested	ok
126: FUNCTION ABS	ok
127: FUNCTION ACOS	ok

128:	FUNCTION ANNUITY	ok
129:	FUNCTION ASIN	ok
130:	FUNCTION ATAN	ok
131:	FUNCTION CHAR	ok
132:	FUNCTION COMBINED-DATETIME	ok
133:	FUNCTION CONCATENATE	ok
134:	FUNCTION CONCATENATE with reference modding	ok
135:	FUNCTION COS	ok
136:	FUNCTION DATE-OF-INTEGER	ok
137:	FUNCTION DATE-TO-YYYYMMDD	ok
138:	FUNCTION DAY-OF-INTEGER	ok
139:	FUNCTION DAY-TO-YYYYDDD	ok
140:	FUNCTION E	ok
141:	FUNCTION EXCEPTION-FILE	ok
142:	FUNCTION EXCEPTION-LOCATION	ok
143:	FUNCTION EXCEPTION-STATEMENT	ok
144:	FUNCTION EXCEPTION-STATUS	ok
145:	FUNCTION EXP	ok
146:	FUNCTION FACTORIAL	ok
147:	FUNCTION FRACTION-PART	ok
148:	FUNCTION INTEGER	ok
149:	FUNCTION INTEGER-OF-DATE	ok
150:	FUNCTION INTEGER-OF-DAY	ok
151:	FUNCTION INTEGER-PART	ok
152:	FUNCTION LENGTH	ok
153:	FUNCTION LOCALE-DATE	ok
154:	FUNCTION LOCALE-TIME	ok
155:	FUNCTION LOCALE-TIME-FROM-SECONDS	ok
156:	FUNCTION LOG	ok
157:	FUNCTION LOG10	ok
158:	FUNCTION LOWER-CASE	ok
159:	FUNCTION LOWER-CASE with reference modding	ok
160:	FUNCTION MAX	ok
161:	FUNCTION MEAN	ok
162:	FUNCTION MEDIAN	ok
163:	FUNCTION MIDRANGE	ok
164:	FUNCTION MIN	ok
165:	FUNCTION MOD	ok
166:	FUNCTION NUMVAL	ok
167:	FUNCTION NUMVAL-C	ok
168:	FUNCTION ORD	ok
169:	FUNCTION ORD-MAX	ok
170:	FUNCTION ORD-MIN	ok
171:	FUNCTION PI	ok
172:	FUNCTION PRESENT-VALUE	ok
173:	FUNCTION RANGE	ok
174:	FUNCTION REM	ok
175:	FUNCTION REVERSE	ok
176:	FUNCTION REVERSE with reference modding	ok
177:	FUNCTION SECONDS-FROM-FORMATTED-TIME	ok
178:	FUNCTION SECONDS-PAST-MIDNIGHT	ok
179:	FUNCTION SIGN	ok
180:	FUNCTION SIN	ok
181:	FUNCTION SQRT	ok
182:	FUNCTION STANDARD-DEVIATION	ok
183:	FUNCTION STORED-CHAR-LENGTH	ok
184:	FUNCTION SUBSTITUTE	ok
185:	FUNCTION SUBSTITUTE with reference modding	ok

```

186: FUNCTION SUBSTITUTE-CASE          ok
187: FUNCTION SUBSTITUTE-CASE with reference mod ok
188: FUNCTION TAN                      ok
189: FUNCTION TRIM                    ok
190: FUNCTION TRIM with reference modding ok
191: FUNCTION UPPER-CASE              ok
192: FUNCTION UPPER-CASE with reference modding ok
193: FUNCTION VARIANCE                ok
194: FUNCTION WHEN-COMPILED          ok

```

```

## ----- ##
## Test results. ##
## ----- ##

```

All 194 tests were successful.

PASS: ./run-O

```

## ----- ##
## OpenCOBOL 1.1 test suite: Data Representation. ##
## ----- ##
  1: BINARY: 2-4-8 big-endian          ok
  2: BINARY: 2-4-8 native              ok
  3: BINARY: 1-2-4-8 big-endian       ok
  4: BINARY: 1-2-4-8 native           ok
  5: BINARY: 1--8 big-endian          ok
  6: BINARY: 1--8 native              ok
  7: BINARY: full-print                ok
  8: DISPLAY: Sign ASCII               ok
  9: DISPLAY: Sign ASCII (2)          ok
10: DISPLAY: Sign EBCDIC               ok
11: PACKED-DECIMAL dump                ok
12: PACKED-DECIMAL display             ok
13: PACKED-DECIMAL move                ok
14: PACKED-DECIMAL arithmetic (1)     ok
15: PACKED-DECIMAL arithmetic (2)     ok
16: PACKED-DECIMAL numeric test       ok
17: POINTER: display                   ok

```

```

## ----- ##
## Test results. ##
## ----- ##

```

All 17 tests were successful.

PASS: ./data-rep

## Data representation tests with -O option ##

```

## ----- ##
## OpenCOBOL 1.1 test suite: Data Representation. ##
## ----- ##
  1: BINARY: 2-4-8 big-endian          ok
  2: BINARY: 2-4-8 native              ok
  3: BINARY: 1-2-4-8 big-endian       ok
  4: BINARY: 1-2-4-8 native           ok
  5: BINARY: 1--8 big-endian          ok
  6: BINARY: 1--8 native              ok
  7: BINARY: full-print                ok
  8: DISPLAY: Sign ASCII               ok
  9: DISPLAY: Sign ASCII (2)          ok

```

```
10: DISPLAY: Sign EBCDIC           ok
11: PACKED-DECIMAL dump           ok
12: PACKED-DECIMAL display        ok
13: PACKED-DECIMAL move           ok
14: PACKED-DECIMAL arithmetic (1) ok
15: PACKED-DECIMAL arithmetic (2) ok
16: PACKED-DECIMAL numeric test   ok
17: POINTER: display              ok
```

```
## ----- ##
## Test results. ##
## ----- ##
```

```
All 17 tests were successful.
PASS: ./data-rep-0
=====
All 5 tests passed
=====
```

## 30.15 6.15 ABI

Application Binary Interface. An acronym that covers the way object code is managed and the expectations of the run-time system. OpenCOBOL is at home in the “C” ABI.

- Link names are as expected.
- CALL arguments are stacked as expected for C programming.
- etc...

The C application binary interface allows OpenCOBOL to link with many existant libraries, more than enough, but does mean that small wrapper access code may be required for access to C++ runtimes.

## 30.16 6.16 Tectonics

I use the expression **tectonics** using the definition below as a basis for the nerd slang describing the code building process. Using a lookup from the **dict://** protocol bank of open servers:

```
"Tectonics" gcide "The Collaborative International Dictionary of English v.0.48"
Tectonics \Tec*ton"ics\, n.
1. The science, or the art, by which implements, vessels,
dwellings, or other edifices, are constructed, both
agreeably to the end for which they are designed, and in
conformity with artistic sentiments and ideas.
[1913 Webster]
```

Trying to infer that building with OpenCOBOL is rock solid and artistically pleasing. Ok fine, I mean **wicked cool!**.

## 30.17 6.17 Setting Locale

OpenCOBOL supports LC\_ locale settings, during builds and with generated programs.



## 30.18 6.18 GNU

GNU is Not Unix, one of the original recursive acronyms. GNU software leads the Free Software movement, and with the Linux kernel is a critical piece in the GNU/Linux operating system. See <http://www.gnu.org/> for more details.

The developers of OpenCOBOL follow, as closely as possible, the GNU coding standards. <http://www.gnu.org/prep/standards/>

## 30.19 6.19 Performing FOREVER?

I asked on [opencobol.org](http://opencobol.org) for some input, and an interesting conversation ensued. I've included most of the forum thread archive, to give a sense of various programmer styles and group thought processing. See [FOREVER](#).

Subject: FOREVER and a small request for involvement

I just updated the FAQ and was wondering if anyone could come up with a better/different short sample program than the one I use in

<http://opencobol.add1tocobol.com/#forever>

The one I have also demonstrates the CYCLE clause of EXIT PERFORM, but reading it, it seems a little, umm, lame for what is a pretty powerful program flow construct.

[i]Plus I'd like to show off a little more community involvement and spread some credit around.[/i]

Cheers,  
Brian

-----  
I think it's fine and think you should leave it as it is...

human

-----  
human;

I know it's "fine", kinda, but I'm also trying to get some of the lurkers out into the open. :-)

Hoping that some small steps will lead to bigger bolder steps.

Plus, the post was a thinly veiled self promotion and the, [i]as always[/i], greater desire to inform that OpenCOBOL supports FOREVER along with EXIT PERFORM CYCLE.

As I add reserved words to the FAQ in the future, I may post up more of these challenges [i]in a thinly veiled disguise to highlight the feature[/i].

Cheers,  
Brian

-----  
As one of the "lurkers", may I offer an excuse. I think that many of us who do not make a contribution, are ordinary cobol people who know nothing of C or web

based extensions or GUI or database extensions. Much of the discussion here seems pretty esoteric. There is no place where one feels that it would be appropriate to post ordinary basic cobol programs or even tips. I think this is a pity, but I don't have any solutions. Going way back to the computer language cobol group in the pre YK2 years, it was apparent that cobol programmers were a most ungenerous lot. "Do your own homework", and "I do this for money not for free" were common responses with a few exceptions like WM Klein and J McLendon. Perhaps the decline of cobol might have made people more open. Even though cobol is the accounting language, you can't I think find books with debtors, creditors, stock payroll and general ledger. You can find them in basic, but not cobol. I think that if there was a place where low level people could contribute, perhaps they might. It is not appropriate to clutter up this forum, but it would need to be a place which is just as simple to write to, else most of us would be unable to join in.

John.

-----  
Thanks for the post John.

Exactly the catch-22 I wanted to break here. OpenCOBOL is for sharing. And yes, old school COBOL is/was very much "top-secret, tight lipped programming". We can change that.

No need to feel you have to talk C bindings, or GUI or highfalutin issues.

A nice challenge on a short sample of

```
PERFORM FOREVER
  do some thing
  now get me outta here
  do some other thing
END-PERFORM
```

was what I wanted to start up.

A sample on a neat INSPECT trick, or a blurb on preferred section/paragraph naming. Anything. OpenCOBOL doesn't have to be closed like the olden days. [i]And to be honest, it is to great credit that most COBOLers kept their tight lips, when I just know that some of them wanted to help, or point out mistakes, or show off, but couldn't, due to the nature of the work they were/are doing.[/i] We can, and we should, flap some loose lips. :-)

Do that here on opencobol.org. I'd read the posts, and feel better for the reading, and the learning, of all the old and new techniques.

I blather on with samples and bindings to show what OpenCOBOL is capable of, but a pure COBOL discussion would be more than welcome. It'd be appreciated.

Unless it sounds like actual homework and it'd hurt more than help, there won't be many "Do your own homework" remarks...umm, I hope ([i]no, I'm pretty sure[/i]).

[b]To everyone[/b]; join in, the water's fine. ;-)

In the FAQ as it stands, there are over 500 reserved words in section 4 and only a mere hundred or so have code samples. I'd gladly read submissions here, get permission and then include them (with or without credit at author's desire) for everyone's benefit.

If we start to overwhelm the forum and people want to direct compiler questions to Roger, we can work out a way to keep his perception of the signal to noise ratio high enough for productive usage of time.

Cheers,  
Brian

-----  
Did not know that existed.

```
>>SOURCE FORMAT IS FREE
id division.
program-id. read_forever.
environment division.
input-output section.
file-control.
    select my-file
        organization line sequential
        assign to "myfile".

data division.
file section.
fd my-file.
01 my-record          pic x(80).

procedure division.
    open input my-file
    perform forever
        read my-file
        at end
            exit perform
        end-read
        display my-record
    end-perform
    close my-file
    goback.
```

Cool. No need for a goto, a file status, or any working-storage at all.

Too bad it's apparently not standard.

-----  
Yep, no standard - but a real nice extension.

```
If you want to do this the standard way do [code]
[...]
```

```
perform until 0 = 1
    read my-file
    at end
        exit perform
    end-read
    display my-record
    end-display
end-perform
[...]
```

OpenCOBOL may supports PERFORM UNTIL EXIT, too (this is a MF extension, if I remember this correct).

human  
-----

OK Brian here is how we did this in the original dialects of COBOL.

In an effort to show how the language has changed, I offer the following version of Brian's program. While many styles can be effectively used in COBOL programming, this program is an example of the style used in programming shops where I worked.

The first six columns of each source line were reserved for the source code sequence number (usually page and line number). We generally used the first three columns to represent the ascending page number and the last three for the line number on the page. Skipping ten numbers between each original line allowed us to insert additional lines when needed. You can see that an insertion was made at 001045. These sequence numbers were desirable in that the program was punched on cards with one card for each line. If the source card deck was accidentally dropped the sequence numbers allowed us to get the source deck back into order.

You will also notice that the code is all in uppercase. Quite simply, early line printers could not print lowercase. Take a look at line 001080. While even early compilers would have allowed us to write "VALUE 0" we would spell out the word zero since the difference in appearance between an alphabetic letter O and a numeric zero was easy to miss when reading the program.

All of the environment division has been left out of this program, although it was almost always necessary. The numbers after "FOREVERLOOP" on line 001070 were the version number of the program. It was our habit to keep a journal (in comment lines) at the beginning of the program describing modifications that were made to the program.

The variable names start with "WS-". This allowed the reader of the program to understand that the variable in question was in the WORKING-STORAGE instead of being part of a file descriptor, thus making it easier to find.

Numeric fields were almost always signed, both for efficiency at run-time and to allow for the possibility of a value going negative even if it should not. COMP asked the compiler to use the most efficient method to store the value on the architecture on which the program was going to run.

You will see that the display statements start their display with "I) ". We used this to make reading console output easier. "I)" was for normal information, "W)" was for warnings, and "T)" was for terminal conditions.

From a syntactical standpoint this code was written to the COBOL-68 standard. Structured programming constructs were not available.

Paragraphs were numbered in ascending sequence in order to make finding a paragraph easier.

Sentences were kept short and periods were used as often as we could use them.

```
001010 IDENTIFICATION DIVISION.  
001020 PROGRAM-ID. FOREVERLOOP.  
001030*
```

```

001040 DATA DIVISION.
001050 WORKING-STORAGE SECTION.
001060 01 WS-PROGRAM-NAME                PIC X(16)
001070                                     VALUE "FOREVERLOOP 001".
001080 01 WS-COBOL                       PIC S9 COMP VALUE ZERO.
001090 01 WS-C                             PIC S9 COMP VALUE 1.
001100 01 WS-FORTRAN                     PIC S9 COMP VALUE 2.
001110 01 WS-ED1S                       PIC Z-.
001110*
001010 PROCEDURE DIVISION.
001020     DISPLAY "I) PROGRAM ", WS-PROGRAM-NAME, " BEGINNING".
001030 0100-LOOP.
001040     ADD 1 TO WS-COBOL.
001045     MOVE WS-COBOL TO WS-ED1S.
001050     DISPLAY "I) COBOL AT ", WS-ED1S.
001060     IF WS-COBOL IS GREATER THAN WS-FORTRAN
001070         THEN GO TO 0800-ENDER.
001080     IF WS-COBOL IS EQUAL TO 1
001090         THEN DISPLAY "I) COBOL STILL CREEPING UP ON C".
001100     GO TO 0100-LOOP.
001110*
001120 0800-ENDER.
001130     DISPLAY "I) COBOL SURPASSED C AND FORTRAN".
001140     DISPLAY "I) PROGRAM ", WS-PROGRAM-NAME, " TERMINATED".
001150*
001160     STOP RUN.

```

The run-time output is below:

```

[code]
I) PROGRAM FOREVERLOOP 001 BEGINNING
I) COBOL AT 1
I) COBOL STILL CREEPING UP ON C
I) COBOL AT 2
I) COBOL AT 3
I) COBOL SURPASSED C AND FORTRAN
I) PROGRAM FOREVERLOOP 001 TERMINATED
[/code]

```

Please note that I am not advocating this style. However it is a good example of traditional methods.

-----

You made one "syntax" error for duplicating "old-style" (required for Standard conformance) programming.

You hae DISPLAY statement immediately following the PROCEDURE DIVISON header. Up until "more recent" Standards, you were required to have either a section or paragraph header and could NOT have statements "outside" of a named procedure.

P.S. In the days of "numbered lines" and all upper-case, you probably would have also had a REMARKS paragraph, but that was optional.

-----

As is usually the case, Mr. Klein is correct. :-)

Chalk it up to CRS (Can't Remember Stuff).

Yes the "old-style" relied a lot more on the environment division, including the ability to specify both a source computer and an object computer. This would allow the compilers that supported it to output different object code depending on the object computer specified.

A compile of a simple listing program done on a four tape 1401 would take about 15 minutes and then you had to run the result through the Autocoder macro assembler.

The 360's would generally compile directly (without the Autocoder step) and would get the job done in a few minutes but if you were not authorized to be in the computer room you had to wait until someone in production saw fit to run your compile for you.

-----  
Like OMG! I learned COBOL on the 1401. And I remember pops letting me practice on the week ends on the 360.

Good times... But the PC is so much more convenient!  
-----

Now thats what I'm talking about.

John, Jim, Frank, Bill, human; If you don't mind, I'd like to include nearly this entire thread in the FAQ, (under what heading I'm not sure, but this is some wicked good COBOL technical [i]and cultural[/i] wisdom).

Damon; not to worry, I plan on including as many of your snippets as the future will bear. ;-)

More of this please...[i]he said, hinting towards the anonymous readers[/i].

Cheers,  
Brian  
-----

I added a more contemporary method of doing the same thing for the COBOL newbies.

```
001010 IDENTIFICATION DIVISION.
001020 PROGRAM-ID. FOREVERLOOP.
001030*
021611*****
021611*
021611* This program will demonstrate various techniques and
021611* coding styles.
021611*
021611* Version 001--Shows a COBOL68 technique
021611* 02/15/2011--J C Currey
021611*
021611* Version 002--Shows an OpenCOBOL 1.1 technique
021611* 02/16/2011--J C Currey
021611*
021611*****
001040 DATA DIVISION.
001050 WORKING-STORAGE SECTION.
001060 01 WS-PROGRAM-NAME PIC X(16)
021611 VALUE "FOREVERLOOP 002".
001080 01 WS-COBOL PIC S9 COMP VALUE ZERO.
```

```

001090 01  WS-C                               PIC S9 COMP VALUE 1.
001100 01  WS-FORTRAN                         PIC S9 COMP VALUE 2.
001110 01  WS-ED1S                            PIC Z-.
001110*
001010 PROCEDURE DIVISION.
001020     DISPLAY "I) PROGRAM ", WS-PROGRAM-NAME, " BEGINNING".
021611*
021611*     THIS CODE SHOWS HOW WE WOULD DO IT WITH COBOL68
021611*
001030 0100-LOOP.
001040     ADD 1 TO WS-COBOL.
001045     MOVE WS-COBOL TO WS-ED1S.
001050     DISPLAY "I) COBOL AT ", WS-ED1S.
001060     IF WS-COBOL IS GREATER THAN WS-FORTRAN
001070         THEN GO TO 0800-ENDER.
001080     IF WS-COBOL IS EQUAL TO 1
001090         THEN DISPLAY "I) COBOL STILL CREEPING UP ON C".
001100     GO TO 0100-LOOP.
001110*
001120 0800-ENDER.
001130     DISPLAY "I) COBOL SURPASSED C AND FORTRAN".
021611     DISPLAY " ".
021611*
021611*     Now we will do the same thing a newer way
021611*
021611     perform with test after
021611         varying ws-cobol from 1 by 1
021611         until ws-cobol is greater than ws-fortran
021611         move ws-cobol to ws-ed1s
021611         display "I) COBOL at ", ws-ed1s
021611         evaluate ws-cobol
021611             when 1
021611                 display "I) COBOL still creeping up on C"
021611             when 3
021611                 display "I) COBOL surpassed C and FORTRAN"
021611         end-evaluate
021611     end-perform.
021611*
001140     DISPLAY "I) PROGRAM ", WS-PROGRAM-NAME, " TERMINATED".
001150*
001160     STOP RUN.

```

The explanation was then updated

In an effort to show how the language has changed, I offer the following version of Brian's program. While many styles can be effectively used in COBOL programming, this program is an example of the style used in programming shops where I worked.

The first six columns of each source line were reserved for the source code sequence number (usually page and line number). We generally used the first three columns to represent the ascending page number and the last three for the line number on the page. Skipping ten numbers between each original line allowed us to insert additional lines when needed. You can see that an insertion was made at 001045. These sequence numbers were desirable in that the program was punched on cards with one card for each line. If the source card deck was accidentally dropped the sequence numbers allowed us to get the source deck back into order.

You will also notice that the code is all in uppercase. Quite simply, early line printers could not print lowercase. Take a look at line 001080. While even early compilers would have allowed us to write "VALUE 0" we would spell out the word zero since the difference in appearance between an alphabetic letter O and a numeric zero was easy to miss when reading the program.

All of the environment division has been left out of this program, although it was almost always necessary. The numbers after "FOREVERLOOP" on line 001070 were the version number of the program. It was our habit to keep a journal (in comment lines) at the beginning of the program describing modifications that were made to the program.

The variable names start with "WS-". This allowed the reader of the program to understand that the variable in question was in the WORKING-STORAGE instead of being part of a file descriptor, thus making it easier to find.

Numeric fields were almost always signed, both for efficiency at run-time and to allow for the possibility of a value going negative even if it should not. COMP asked the compiler to use the most efficient method to store the value on the architecture on which the program was going to run.

You will see that the display statements start their display with "I) ". We used this to make reading console output easier. "I)" was for normal information, "W)" was for warnings, and "T)" was for terminal conditions.

From a syntactical standpoint this code was written to the COBOL-68 standard. Structured programming constructs were not available.

Paragraphs were numbered in ascending sequence in order to make finding a paragraph easier.

\*\*\*\*\*

Version 002 shows how one might code the application with OpenCOBOL 1.1.

A modification log has been added via comments at the beginning of the program.

Note that the sequence numbers are now being used to store the date that the new or changed code was made. By looking at the modification date and then referring to the modification log, one can determine what changed from version to version.

Structured programming constructs have been used.

I expect that there may be some discussion as to which method is easier to read and understand.

jimc

-----  
This is a variation of the 'perform forever' program.

```
>>SOURCE FORMAT IS FREE
program-id. "readForever".
*>
```



```

*>
*> Author.          rkeane
*> Written:         16 Feb 2011
*> Purpose:         A variation of submitted "read-forever"
*>
environment division.
input-output section.
file-control.
    select my-file assign to "myFile"
        organization line sequential.

data division.
file section.
fd myFile.
01 myRecord                      pic x(80).
working-storage section.
*>
procedure division.
main.
    open input myFile

    perform forever
        read myFile
            not at end
                display myRecord

        at end
            perform finish
            goback
        end-read
    end-perform
exit.

finish.
    close my-file
exit.

```

Using non-structured statements:

```

procedure division.
main.
    open input myFile.
0100-loop.
    read myFile next record
        at end close myFile
            stop run.
    display myRecord.
    go to 0100-loop.

```

-----

I don't know if anyone else is getting this sensation, but is COBOL becoming cool enough for the internet generation now? Thanks to open folk and OpenCOBOL? [i]Or did I just jinx the tide?[/i] :-)

Cheers,  
Brian

-----

I found the thread a nice read. And to top it off, for me, Roger added a nice idiom in a separate thread for avoiding

paragraphs and sections. Not FOREVER related, but a nice use for an “empty” inline PERFORM.

```
Yep,  
One thing that I saw on earlier posts to  
the newsgroup cobol was -  
What is the need/justification for an  
empty inline perform group.  
ie.  
PERFORM  
...  
END-PERFORM
```

None of the discussions then realized that  
there is a -  
EXIT PERFORM [CYCLE]

Therefore, it is a method to to  
define an exit condition without having paragraphs.

```
ie. (very simply)  
PERFORM  
  READ xxx  
  AT END  
    EXIT PERFORM  
  END-READ  
  MOVE something TO somewhere  
END-PERFORM
```

```
.. test xxx status and somewhere
```

There are, of course, other variations.  
Basically, it means that you code without  
using section/paragraphs.  
(Recommended, if only from performance point of view)

Note that the CYCLE option offers interesting possibilities.

Roger

## 30.20 6.20 POSIX

An acronym first suggested by Richard Stallman for the IEEE specification for maintaining compatibility between operating systems. IEEE Std 1003.1-1988.

**POSIX** Portable Operating System Interface

## 7 AUTHORS



## **8 MAINTAINERS AND CONTRIBUTORS**



## 9 CHANGELOG

- 02-Jul-2008, 06-Jul-2008, 07-Jul-2008, 11-Jul-2008, 13-Jul-2008** Experimental version for comment. First 0.0 pre-alpha release. Second 0.0 pre-alpha. Last 0.0 pre-alpha. Checked in for diffs. Last-last 0.0 pre-alpha. Verify DIFF functionality.
- 17-Jul-2008, 20-Jul-2008, 24-Jul-2008, 28-Jul-2008** Last-last-last 0.0 pre-alpha. Second DIFF. Corrections pass. Expanded the SCREEN SECTION questions. Another correction pass, with clarifications from Roger While
- 10-Aug-2008, 21-Aug-2008, 28-Aug-2008, 29-Aug-2008, 30-Aug-2008** Started in on the intrinsic functions. Dropped the pre from the alpha designation. Still some Look into this entries. Move to add1tocobol.com Publish link to 1.0rc Skeleton of the reserved words list Let the tweaking begin
- 23-Sep-2008** Adds and a trial skin
- 13-Oct-2008, 15-Oct-2008, 19-Oct-2008, 22-Oct-2008, 29-Oct-2008** Added a few samples. Added TABLE SORT sample. Added configure script information. Added dialect configuration information.
- 28-Nov-2008** OpenCOBOL passes the NIST test suite.
- 12-Dec-2008, 16-Dec-2008, 21-Dec-2008** Added new links to OpenCOBOL 1.1 binary builds by Sergey. Updated header templates. Added a few keywords.
- 28-Dec-2008, 29-Dec-2008, 30-Dec-2008** Added info on CobXRef, some debugging tricks and an entry on recursion.
- 01-Jan-2009, 10-Jan-2009, 12-Jan-2009, 22-Jan-2009** Lame attempt at clarifying (excusing) poor use of Standards references. Small corrections and additions to SQL entry. Added a few RESERVED entries and Vincent's STOCK library expansion. Typos.
- 02-Feb-2009, 06-Feb-2009, 09-Feb-2009, 11-Feb-2009** Coloured Source codes. Added info on COB\_PRE\_LOAD, added LINAGE sample, fixed colours (kinda). Added Haiku, disclaimer about no claim to Standards conformance. Updated look.
- 16-Feb-2009, 18-Feb-2009** Added JavaScript, Lua, Guile embedding samples and mention Tcl/Tk, GTK. Added CBL\_OC\_DUMP sample by Asger Kjelstrup and human
- 09-Mar-2009, 31-Mar-2009** Added Vala and a few more RESERVED word entries. Added -ext clarification.
- 17-Apr-2009, 18-Apr-2009, 19-Apr-2009** Clarified -fsource-location option. Added a production use posting. Added START and ISAM sample.
- 01-May-2009, 09-May-2009, 28-May-2009, 31-May-2009** Started a structural and TOC reorg. Mention S-Lang. Continue re-org. Added some FUNCTION samples. Getting close to a complete Intrinsic list.
- 01-Jun-2009, 03-Jun-2009, 05-Jun-2009, 28-Jun-2009** Added errno, makefile, a few samples and some reserved word explanations. Added filter.cob the stdin stdout sample. Added some reserved word blurbs and the message queue sample. human assisted corrections. Many thanks to human.
- 29-Jul-2009** more human assisted corrections.

- 13-Sep-2009** Some printing information.
- 12-Oct-2009** Added some links, credits.
- 15-Feb-2010, 20-Feb-2010, 25-Feb-2010, 27-Feb-2010, 28-Feb-2010** Added advocacy, and a few tweaks. Added Jim's PRTCBL. Added Angus' ocsort. Added cobol.vim and Easter Day programs. Updated CBL\_OC\_DUMP source code listing. Added a REPLACE text preprocessor sample. Added pgcob.cob PostgreSQL sample.
- 01-Mar-2010, 28-Mar-2010** Added Oracle procob news. Added FILE STATUS codes to ISAM note. Mention TP-COBOL-DEBUGGER. Updated INSPECT sample and COB\_SCREEN\_ESC entry. Added ocgtk.c
- 04-Apr-2010, 05-Apr-2010, 11-Apr-2010, 15-Apr-2010** Fixed up the source code listings. Added telco benchmark. Added print to PDF. Added COB\_LIBRARY\_PATH info. Expanded the Tcl/Tk entry. Added Mac install instructions from Ganymede. REXX.
- 05-May-2010, 06-May-2010** Added the SEARCH and SORT sample. Updated REXX. Image for GNAT GPS.
- 13-Jun-2010** Reorganized table of contents boxes. Split SEARCH sample source code.
- 18-Oct-2010, 19-Oct-2010, 24-Oct-2010, 30-Oct-2010, 31-Oct-2010** Added some working Vala code samples. Added DamonH's AJAX code to the CGI section. Updated the CBL\_OC\_DUMP listings. Added a few minor reserved word entries. Added translation help request note. Added mkfifo sample. Added call Genie sample. Added CBL\_OC\_GTKHTML sample. Updated the PI and PRESENT-VALUE entries. Updated CHARACTERS entry.
- 01-Nov-2010, 06-Nov-2010, 18-Nov-2010, 20-Nov-2010, 27-Nov-2010** Added a Genie sample. Some small touch-ups. Restored borked colouring. Added DECLARATIVES entry and a few small tweaks. Added a few RESERVED words entries. Added ROOT/CINT info. Expanded install instructions.
- 12-Dec-2010, 31-Dec-2010** Added libsoup HTTP server sample. Changed EOP file status 52 copy sample. Updated Falcon entry.
- 02-Jan-2011, 23-Jan-2011** Added errorproc.cob sample. Added some vim and Fossil info.
- 13-Feb-2011** Fixed an unnecessary css import, small corrections. Added REPOSITORY, CYCLE and FOREVER entries.
- 07-May-2011** Added gfortran sample.
- 25-Jun-2011, 26-Jun-2011** Added sourceforge link. Updated shortest program entries. Updated a few reserved words.
- 01-Jul-2011, 02-Jul, 10-Jul, 11-Jul, 12-Jul, 20-Jul, 23-Jul** Updated CALL reserved word entry to show off ON EXCEPTION CONTINUE. Updated a few more reserved words; DATE, DAY, DEBUGGING. D's are done. Fleshed out a few reserved words, E's done. Added links to the Doxygen API passes. Started on some future 2.0 entries with the Directives. Added blurb about LD\_RUN\_PATH. Added initial entry on APL/J linkage. Into the Fs. Done A thru K. Done L.
- 01-Aug-2011, 05-Aug, 06-Aug, 07-Aug, 08-Aug, 09-Aug, 13-Aug, 14-Aug, 22-Aug** Done M, N. Done O. Fixed the colours after a Pygments update. P's in. Q, R done. Doing S. Just passed 750000 bytes of FAQ. Done S to Z. Started documenting the GNU build tool options available. Fixed a DSO misnomer.
- 26-Aug-2011** Finished the last FUNCTION.
- 03-Sep-2011, 25-Sep, 28-Sep** Fixed the ocgtk.c files, getting rid of void returns. Updated list of platforms with 1.1pre-rel running. Added COBOLUnit.
- 06-Dec-2011, 26-Dec-2011** Added Gambas interface link. Fixed INDEXED entry. Added INITIAL source sample.
- 12-Jan-2012, 14-Jan, 15-Jan, 20-Jan** Added a criticism of easter.cob. Updated CURSOR and FOREVER entries. Version to 1.1rc01. FOREVER thread listing moved. cupsPrintFile documented. Rid of the >< comment output. LOCALE-DATE update. Removed Organization from attributions, there is no official group.
- 05-Feb-2012, 29-Feb** Added to DIVIDE, put in some lists in the RESERVED words. Added Public Accounting.



**07-Mar-2012, 19-Mar** Added carpe diem farberistic joke.

**22-Apr-2012** Typos. Added the size listing for hello.

**27-May-2012** Added LLVM and clang reference.

**03-Jun-2012** Added site favicon.ico from Silk/help.png, credited Mark James. Fleshed out telco benchmark entry.

**08-Feb-2013, 09-Feb, 25-Feb** Moving to Sphinx, started documenting new SourceForge project site. Fixed cobxref listing, no truncated lines. Added 'nosidebar' to the sphinx-doc theme settings in conf.py. Added some Computus, and Latin. Updating current version information. Added Python embedding. Added ficl Forth notes. Added Shakespeare.



# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*



# BIBLIOGRAPHY

[Keisuke] Keisuke Nishida

Initial developer and creator of OpenCOBOL. From the 1990s through 2004 and still active was the primary developer and OpenCOBOL project lead. His efforts are greatly appreciated by the userbase of OpenCOBOL.

[Roger] Roger While

OpenCOBOL 1.1 is currently (*February 2013*) in development, and Roger is the lead programmer. From early 2004 up till today, and tomorrow, Roger has been very active on the [opencobol.org](http://opencobol.org) website, and is open to feature requests and clarifications to the implementation. Roger has, since January 2008, actively monitored an OpenCOBOL 1.1 wishlist on the [opencobol.org](http://opencobol.org) OpenCOBOL forum.

[btiffin] Brian Tiffin

Initial FAQ. sample programs for OpenCOBOL 1.1.

[aoirthoir] Joseph James Frantz

Hosting, support.

[jrls\_wla] John Ellis

Samples and how-to's and ...

[human] human

Samples and style

[wmklein] Bill Klein

Keeper of the [COBOL FAQ](#) and all round COBOL myth buster.