Hyp

OpenCOBOL FAQ

OpenCOB

Status

This is a 1.0 release candidate of the OpenCOBOL FAQ. Sourced at ocfaq.rst. Courtesy of ReStructuredText 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.

Authors: Brian Tiffin [btiffin],
John Ellis [jrls_swla],
Vincent Coen,
human [human],
Joseph James Frantz [aoirthoir],
Roger While [Roger],
Keisuke Nishida [Keisuke]

(with the invaluable assistance of many others)

Organization: The OpenCOBOL Project

Version: 1.0rc40, October 12, 2009 (work in progress)

Status: Release Candidate

ChangeLog: ChangeLog

Attention!

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?

FAQ Contents

1 OpenCOBOL

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 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?

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?

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 Can I run background processes using OpenCOBOL?

5 Features and extensions

5.1 How do I use OpenCOBOL for CGI?

5.2 What is ocdoc?

5.3 What is CBL_OC_DUMP?

5.4 Does OpenCOBOL support any SQL databases?

5.5 Does OpenCOBOL support ISAM?

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?

5.19 Can OpenCOBOL interface with S-Lang?

5.20 Can the GNAT Programming Studio be used with OpenCOBOL?

5.21 Does OpenCOBOL support SCREEN SECTION?

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?

5.30 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?

5.37 Does OpenCOBOL implement CONSTANT?

5.38 What source formats are accepted by OpenCOBOL?

5.39 Does OpenCOBOL support continuation lines?

5.40 Does OpenCOBOL support string concatenation?

5.41 Does OpenCOBOL support D indicator debug lines?

5.42 Does OpenCOBOL support mixed case source code?

5.43 What is the shortest OpenCOBOL program?

5.44 What is the shortest Hello World program in OpenCOBOL?

5.45 How do I get those nifty sequential sequence numbers in a source file?

5.46 Is there a way to count trailing spaces in data fields using OpenCOBOL?

5.47 Is there a way to left justify an edited numeric field?

5.48 Is there a way to detemermine when OpenCOBOL is running ASCII or EBCDIC?

5.49 Is there a way to determine when OpenCOBOL is running on 32 or 64 bits?

5.50 Does OpenCOBOL support recursion?

5.51 Does OpenCOBOL capture arithmetic overflow?

5.52 Can OpenCOBOL be used for plotting?

5.53 Does OpenCOBOL support the GIMP ToolKit, GTK+?

6 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 make check listing

7 Authors

8 Maintainers and Contributors

9 ChangeLog

1 OpenCOBOL

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.

1.2 What is COBOL?

COBOL is an acronym for COmmon Business Oriented Language. This author has always thought of it as "Common Business" Oriented more than Common "Business Oriented", but that emphasis is perhaps up to the readers 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.

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
- POSIX Systems including OpenSolaris
- OS/X

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.

Also check out kiska.net for binary builds on various platforms. Thanks to Sergey Kashyrin.

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 orginal 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.

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 to-
day 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 at last count. 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?

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*.

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

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 364 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 Er-
ror 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.

> **Attention!**
>
> Look into this and add some numbers

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.

1.14 Does OpenCOBOL support a GUI?

Yes, but not out of the box. There is not currently *(February 2009)* 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?.

The expectation is that GTK+ will be completely bound as a callable interface. That is currently *(February 2009)* not the case, with perhaps 2% of the GTK+ functionality wrapped (but with that 2%, fully functional graphical interfaces are possible).

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

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 2009)* 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 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 produc-
tion
environments since 2005.
(This includes projects that I person-
ally worked on plus other
  projects reported to me; these worldwide)

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

Reported on opencobol.org, The Nagasaki Prefecture, population 1.44 million and 30,000 civil employees is using OpenCOBOL in support of its payroll management system.

Another post from opencobol.org in April 2009, *reprinted with permission.*

```
OpenCOBOL viability

For those concerned about the viability of Open-
COBOL in a production
environment, I offer our situation as an example.

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

We then explored the CGI capabili-
ties and moved our home-brewed CGI handler
```

```
(written in HP3000 Cobol) over. We ended up chang-
ing only a few lines.

As Marcr's post indicates, we found a MySql wrap-
per and made some minor
changes to it.

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

Please accept our congratulations to the commu-
nity and our gratitude for the
help from the forum.

jimc
```

Attention!

Look into this - need more entries

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 2009)* 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 website is probably the best place to find out more about the OpenCOBOL system. add1tocobol.com is a place to find out about a few of the fan initiatives.

1.19 Can I help out with the OpenCOBOL project?

Absolutely. Visit the 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.

1.20 Is there an OpenCOBOL mailing list?

Yes. Visit 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 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 2009)* 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.

Attention!

Look into this

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.

1.23 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 disrepect 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

- What did COBOL reply to the executive? *Why 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.
```

  Brian Tiffin

- And how about a 5-7-5 haiku?

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

  Brian Tiffin

2 History

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 2009)* 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 today, 200,000,000,000 (two hundred thousand million) lines are COBOL. A full 2/3rds of the world's source code.

See the Wikipedia entry for COBOL for a lot more details.

2.2 What are the Official COBOL Standards?

Many thanks to William Klein for details on what wordings are to be used when referencing COBOL Standards:

```
There are several refer-
ences to "COBOL 85" and these are often
distinguished from "Intrinsic Functions".

The official (but really ob-
scure) term that should be used is "Amended
Third Standard COBOL". The "clearer" (and IMHO bet-
ter) term that should
be used is something like

 - "'85 Standard COBOL with its amendments"

By 1991 (actu-
ally 1993 for ISO rather than ANSI) there was no such thing
as "just '85 Standard COBOL". The only recognized Stan-
dard 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 Func-
tions Module" was
OPTIONAL in the ANSI and ISO COBOL Standards but was RE-
QUIRED (at the
```

```
HIGH level) for FIPS COBOL. As the "certifica-
tion tests" were aimed at
getting US government contracts, most ven-
dors (who were still doing
certification) actually treated Intrinsic Functions re-
quired not
optional for "High-
level" certification. (They were NOT included in the
FIPS intermediate certification process).

Bottom-Line:
 Although some intrinsic func-
tions were added in the '02 Standard (and
more are included in the draft revi-
sion), it is not proper (in my
opinion) to distinguish between supporting the '85 Stan-
dard and
supporting intrinsic functions.

P.S. The corrections amendment did make some techni-
cal changes but all
of these were included in the '02 Standard. There-
fore, 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.

Attention!

Details on official names of other standards still missing

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.

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 2009)* 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.

After a download

```
$ ./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.

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 2009)* 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

3.1 How do I install OpenCOBOL?

Installation instructions can be found at OpenCOBOL Install.

3.1.1 Debian

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

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

3.1.2 Windows

Build from sources under Cygwin or MinGW. Follow the instructions from the site listed above, or read the OC_GettingStarted_Windows document by William Klein available online at

- http://opencobol.add1tocobol.com/oc_gettingstarted_windows.html
- http://opencobol.add1tocobol.com/OC_GettingStarted_Windows.pdf

Also see What is the current version of OpenCOBOL?.

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 Open-
COBOL 1.1 to adapt to many kinds of systems.

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

To assign environment vari-
ables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descrip-
tions 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 spe-
cific to this package
      --help=recursive    dis-
play 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 [con-
figure 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 [EPRE-
FIX/sbin]
  --libexecdir=DIR        program executables [EPRE-
FIX/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 PRE-
FIX to installed program names
  --program-suffix=SUFFIX            append SUF-
FIX to installed program names
  --program-transform-name=PROGRAM   run sed PRO-
GRAM on installed program names

System types:
  --build=BUILD     configure for build-
ing 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 confus-
ing) 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 experimen-
tal code (Developers only!)
  --enable-param-
check    (OpenCOBOL) enable CALL parameter checking
  --enable-shared[=PKGS]
                          build shared libraries [de-
fault=yes]
  --enable-static[=PKGS]
                          build static libraries [de-
fault=yes]
  --enable-fast-install[=PKGS]
                          optimize for fast installa-
tion [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 com-
piler 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-db1               (OpenCOBOL) use Berke-
ley DB 1.85 (libdb-1.85)
  --with-db                (OpenCOBOL) use Berke-
ley DB 3.0 or later (libdb)(default)
  --with-lfs64             (Open-
COBOL) use large file system for file I/O (default)
  --with-dl                (OpenCOBOL) use system dy-
namic loader (default)
  --with-patch-
level      (OpenCOBOL) define a patch level (default 0)
  --with-varse          (OpenCOBOL) define variable se-
quential format (default 0)
  --with-gnu-ld            assume the C com-
piler uses GNU ld [default=no]
  --with-pic               try to use only PIC/non-
PIC objects [default=use
                           both]
  --with-tags[=TAGS]
                           include additional configura-
tions [automatic]
  --with-gnu-ld            assume the C com-
piler uses GNU ld default=no
  --with-libiconv-prefix[=DIR]  search for libi-
conv in DIR/include and DIR/lib
  --without-libiconv-prefix     don't search for libi-
conv in includedir and libdir
  --with-libintl-prefix[=DIR]  search for lib-
intl in DIR/include and DIR/lib
  --without-libintl-prefix     don't search for lib-
intl 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 <in-
clude dir>
  CPP         C preprocessor
  CXXCPP      C++ preprocessor

Use these variables to over-
ride the choices made by 'configure' or to help
it to find libraries and programs with nonstan-
dard names/locations.
```

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

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 2009)* 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.

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.

Example

```
$ cat hello.cob
```

Original source code;

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

OpenCOBOL stages. Preprocess

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

Preprocess only; For one thing, FIXED format becomes FREE format. For another COPY is processed. Displays

```
# 1 "hello.cob"

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

to standard out.
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**.
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         cobc -C hello.cob */

/* PROGRAM-ID : hello */

static unsigned char b_5[4] __at-
tribute__((aligned));       /* COB-CRT-STATUS */
static unsigned char b_1[4] __at-
tribute__((aligned));       /* RETURN-CODE */
static unsigned char b_2[4] __at-
tribute__((aligned));       /* SORT-RETURN */
static unsigned char b_3[4] __at-
tribute__((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, (un-
signed char *)"Hello World!", &a_2};

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

hello.c

```
/* 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         cobc -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 mod-
ule = { NULL, NULL, &f_5, NULL, cob_user_parameters, 0, '.', '$', ',', 1, 1, 1, 0};


  /* 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: */

  l_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 */
```

Generate assembler

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

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
```

Compile only; outputs assembly file. Produces **hello.s**.
Produce object code

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

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

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

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

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

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

Attention!

Need a little OS/X info here

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

Build 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?.

3.5 What is cobc?

cobc is the OpenCOBOL compiler. See What compiler options are supported? for more information.

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.

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.

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 in-
voked by the compiler
  -x                    Build an executable program
  -m                    Build a dynamically load-
able 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 Com-
patible
                          default     When not specified
                        See config/default.conf and con-
fig/*.conf
  -free                 Use free source format
  -
fixed                Use fixed source format (default)
  -O, -O2, -Os          Enable optimization
  -g                    Produce debugging informa-
tion in the output
  -debug                Enable all run-
time error checking
  -o <file>             Place the output into <file>
  -b                    Combine all in-
put files into a single
                        dynamically loadable module
  -E                    Preprocess only; do not com-
pile, assemble or link
  -C                    Translation only; con-
vert COBOL to C
  -
S                    Compile only; output assembly file
  -c                    Compile and assem-
ble, but do not link
  -t <file>             Generate and place a pro-
gram listing into <file>
```

```
  -I <directory>        Add <direc-
tory> to copy/include search path
  -L <directory>        Add <directory> to li-
brary search path
  -l <lib>              Link the library <lib>
  -D <define>           Pass <define> to the C compiler
  -conf=<file>          User defined dialect configura-
tion - See -std=
  --list-reserved       Display reserved words
  --list-intrinsics     Display intrinsic functions
  --list-mnemonics      Display mnemonic names
  -save-temps(=<dir>)   Save intermediate files (de-
fault current directory)
  -MT <target>          Set target file used in depen-
dency list
  -
MF <file>             Place dependency list into <file>
  -ext <extension>      Add default file extension

  -W                    Enable ALL warnings
  -Wall                 Enable all warnings ex-
cept as noted below
  -
Wobsolete             Warn if obsolete features are used
  -
Warchaic              Warn if archaic features are used
  -Wredefinition        Warn incompatible redefini-
tion of data items
  -Wconstant            Warn inconsistent constant
  -Wparentheses         Warn lack of parenthe-
ses 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 col-
umn 72, FIXED format (NOT set with -Wall)
  -
Wterminator           Warn lack of scope terminator END-
XXX (NOT set with -Wall)
  -Wtruncate            Warn possible field trunca-
tion (NOT set with -Wall)
  -Wlinkage             Warn dangling LINK-
AGE items (NOT set with -Wall)
  -Wunreachable         Warn unreachable state-
ments (NOT set with -Wall)

  -ftrace               Generate trace code (Exe-
```

```
cuted SECTION/PARAGRAPH)
  -ftraceall            Generate trace code (Exe-
cuted SECTION/PARAGRAPH/STATEMENTS)
  -fsyntax-only         Syntax error check-
ing only; don't emit any output
  -fdebugging-line      Enable debug-
ging lines ('D' in indicator column)
  -fsource-location     Generate source loca-
tion code (Turned on by -debug or -g)
  -fimplicit-init       Do automatic initializa-
tion of the Cobol runtime system
  -fsign-ascii          Numeric display sign ASCII (De-
fault on ASCII machines)
  -fsign-ebcdic         Numeric display sign EBCDIC (De-
fault on EBCDIC machines)
  -fstack-
check         PERFORM stack checking (Turned on by -
debug or -g)
  -ffold-copy-lower     Fold COPY sub-
ject to lower case (Default no transformation)
  -ffold-copy-upper     Fold COPY subject to up-
per case (Default no transformation)
  -fnotrunc             Do not truncate bi-
nary fields according to PICTURE
  -ffunctions-all       Allow use of intrinsic func-
tions without FUNCTION keyword
  -fmfcomment           '*' or '/' in col-
umn 1 treated as comment (FIXED only)
  -fnull-param          Pass extra NULL terminating point-
ers on CALL statements
```

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 compiles, 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.

3.10 What extensions are used if cobc is called with/without "-ext" for COPY

From Roger on opencobol.org

```
In the following order -
CPY, CBL, COB, cpy, cbl, cob and finally with no exten-
sion.

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

ie. They take precedence.
```

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 configura-
tion                                    -*- 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 en-
vironment variables.
# For example, given ASSIGN TO "DATAFILE", the ac-
tual 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 as-
sign 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               2
#                 5 -  9               4
#                10 - 18               8
#
# ‘1-2-4-8’       1 -  2               1
#                 3 -  4               2
#                 5 -  9               4
#                10 - 18               8
#
# ‘1--8’          1 -  2    1 -  2     1
#                 3 -  4    3 -  4     2
#                 5 -  6    5 -  7     3
#                 7 -  9    8 -  9     4
#                10 - 11   10 - 12     5
#                12 - 14   13 - 14     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 cur-
rently 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’, ‘ig-
nore’, ‘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
```

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 er-
rors, call vim in quickfix
%.q: %.cob
      cobc $(COBCWARN) -x $^ 2>errors.err || vi -q

# hack extension; make binary; capture warn-
ings, 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 list-
ing  (leaves tmps dir around)
%.lst: %.cob
```

```
	[ -d tmps ] || mkdir tmps
	cobc $(COBCWARN) --save-temps=tmps -c $^ -
o tmps/$*.o && ~/writ-
ing/add1/tools/cobxref/cobxref tmps/$*.i

# 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 quick-
fix 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.

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.

```
*> ** *>>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

```
*> ** *>>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 .
```

Fixed form in UPPERCASE

```
 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 .
```

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
- headfixuppper.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).

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: cobc -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
```

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.

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.

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
```

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: cobc -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 msg-
Pointer 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.
```

4 Reserved Words

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

Reserved Words

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

4.1.1 ACCEPT

```
ACCEPT variable FROM CONSOLE.
ACCEPT variable FROM ENVIRONMENT "path".
ACCEPT variable FROM COMMAND LINE.
ACCEPT variable AT 0101.
ACCEPT screen-variable.
```

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.
```

4.1.3 ACTIVE-CLASS

Not yet implemented. Object COBOL feature.

4.1.4 ADD

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

4.1.5 ADDRESS

```
SET pointer-variable TO ADDRESS OF linkage-store.
```

4.1.6 ADVANCING

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

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.

4.1.8 ALIGNED

4.1.9 ALL

A multipurpose reserved in context word.

```
INSPECT variable REPLACING ALL "123" WITH "456".
```

4.1.10 ALLOCATE

Allocates actual working storage for a BASED element.

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

4.1.11 ALPHABET

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

4.1.12 ALPHABETIC

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

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

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
```

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
```

4.1.15 ALPHANUMERIC

```
INITIALIZE data-record REPLACING ALPHANUMERIC BY literal-
value
```

4.1.16 ALPHANUMERIC-EDITED

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

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
```

4.1.18 ALTER

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

Yeah, just don't.

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.
```

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
```

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
```

4.1.22 ANYCASE

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".
```

4.1.24 AREA

Controls SORT, MERGE and RECORD data definitions.

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

4.1.25 AREAS

4.1.26 ARGUMENT-NUMBER

4.1.27 ARGUMENT-VALUE

Returns the next command line argument. This post from John on 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 "l".
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 "l"            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
```

4.1.28 ARITHMETIC

4.1.29 AS

4.1.30 ASCENDING

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 as-
sign 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 enviroment 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*.

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, col-
umn 10 for length of field <*
 ACCEPT name-var AT 0110 END-ACCEPT
```

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

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

4.1.43 BASED

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

A sample posted by [human]

```
*-----------------------------------------------------
----------
 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 ----------------------------
----------
*><*
```

4.1.44 BEEP

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

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 re-
sult 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| supported**, in the declaratives for REPORT SECTION control.

```
USE BEFORE REPORTING
 ...
```

4.1.46 BELL

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

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

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

4.1.55 BLINK

4.1.56 BLOCK

4.1.57 BOOLEAN

4.1.58 BOTTOM

4.1.59 BY

4.1.60 BYTE-LENGTH

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=UserMa

4.1.62 CANCEL

4.1.63 CD

4.1.64 CENTER

4.1.65 CF

4.1.66 CH

4.1.67 CHAIN

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
```

4.1.69 CHARACTER

4.1.70 CHARACTERS

4.1.71 CLASS

4.1.72 CLASS-ID

4.1.73 CLASSIFICATION

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
```

4.1.75 CODE

4.1.76 CODE-SET

4.1.77 COL

4.1.78 COLLATING

4.1.79 COLS

4.1.80 COLUMN

4.1.81 COLUMNS

4.1.82 COMMA

4.1.83 COMMAND-LINE

Provides access to command line arguments.

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

4.1.84 COMMIT

4.1.85 COMMON

4.1.86 COMMUNICATION

currently *(February 2009)* unsupported DIVISION, but see Does OpenCOBOL support Message Queues? for an alternative.

4.1.87 COMP

4.1.88 COMP-1

4.1.89 COMP-2

4.1.90 COMP-3

4.1.91 COMP-4

4.1.92 COMP-5

4.1.93 COMP-X

4.1.94 COMPUTATIONAL

4.1.95 COMPUTATIONAL-1

4.1.96 COMPUTATIONAL-2

4.1.97 COMPUTATIONAL-3

4.1.98 COMPUTATIONAL-4

4.1.99 COMPUTATIONAL-5

4.1.100 COMPUTATIONAL-X

4.1.101 COMPUTE

Computational arithmetic.

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

4.1.102 CONDITION

4.1.103 CONFIGURATION

4.1.104 CONSTANT

4.1.105 CONTAINS

4.1.106 CONTENT

4.1.107 CONTINUE

4.1.108 CONTROL

4.1.109 CONTROLS

4.1.110 CONVERTING

4.1.111 COPY

4.1.112 CORR

4.1.113 CORRESPONDING

4.1.114 COUNT

4.1.115 CRT

4.1.116 CURRENCY

4.1.117 CURSOR

4.1.118 CYCLE

4.1.119 DATA

4.1.120 DATA-POINTER

4.1.121 DATE

4.1.122 DAY

4.1.123 DAY-OF-WEEK

4.1.124 DE

4.1.125 DEBUGGING

4.1.126 DECIMAL-POINT

4.1.127 DECLARATIVES

4.1.128 DEFAULT

4.1.129 DELETE

4.1.130 DELIMITED

4.1.131 DELIMITER

4.1.132 DEPENDING

4.1.133 DESCENDING

4.1.134 DESTINATION

4.1.135 DETAIL

4.1.136 DISABLE

4.1.137 DISK

4.1.138 DISPLAY

Prints values to standard out, sets enviroment variables

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

4.1.139 DIVIDE

Highly precise arthimetic.

```
DIVIDE dividend BY divisor GIVING answer ROUNDED REMAINDER r
```

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.

4.1.140 DIVISION

4.1.141 DOWN

4.1.142 DUPLICATES

4.1.143 DYNAMIC

4.1.144 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
```

4.1.145 EC

4.1.146 EGI

4.1.147 ELSE

4.1.148 EMI

4.1.149 ENABLE

4.1.150 END

4.1.151 END-ACCEPT

Explicit terminator for ACCEPT.

4.1.152 END-ADD

Explicit terminator for ADD.

4.1.153 END-CALL

Explicit terminator for CALL.

4.1.154 END-COMPUTE

Explicit terminator for COMPUTE.

4.1.155 END-DELETE

Explicit terminator for DELETE.

4.1.156 END-DISPLAY

Explicit terminator for DISPLAY.

4.1.157 END-DIVIDE

Explicit terminator for DIVIDE.

4.1.158 END-EVALUATE

Explicit terminator for EVALUATE.

4.1.159 END-IF

Explicit terminator for IF.

4.1.160 END-MULTIPLY

Explicit terminator for MULTIPLY.

4.1.161 END-OF-PAGE

4.1.162 END-PERFORM

Explicit terminator for PERFORM.

4.1.163 END-READ

Explicit terminator for READ.

4.1.164 END-RECEIVE

Explicit terminator for RECEIVE.

4.1.165 END-RETURN

Explicit terminator for RETURN.

4.1.166 END-REWRITE

Explicit terminator for REWRITE.

4.1.167 END-SEARCH

Explicit terminator for SEARCH.

4.1.168 END-START

Explicit terminator for START.

4.1.169 END-STRING

Explicit terminator for STRING.

4.1.170 END-SUBTRACT

Explicit terminator for SUBTRACT.

4.1.171 END-UNSTRING

Explicit terminator for UNSTRING.

4.1.172 END-WRITE

Explicit terminator for WRITE.

4.1.173 ENTRY

4.1.174 ENTRY-CONVENTION

4.1.175 ENVIRONMENT

Divisional name. And allows access to operating system environment variables.

4.1.176 ENVIRONMENT-NAME

4.1.177 ENVIRONMENT-VALUE

4.1.178 EO

4.1.179 EOL

4.1.180 EOP

4.1.181 EOS

4.1.182 EQUAL

Conditional expression to compare two data items for equality.

4.1.183 EQUALS

Conditional expression to compare two data items for equality.

4.1.184 ERASE

4.1.185 ERROR

4.1.186 ESCAPE

4.1.187 ESI

4.1.188 EVALUATE

4.1.189 EXCEPTION

4.1.190 EXCEPTION-OBJECT

4.1.191 EXCLUSIVE

4.1.192 EXIT

4.1.193 EXPANDS

4.1.194 EXTEND

4.1.195 EXTERNAL

4.1.196 FACTORY

4.1.197 FALSE

4.1.198 FD

4.1.199 FILE

4.1.200 FILE-CONTROL

4.1.201 FILE-ID

4.1.202 FILLER

4.1.203 FINAL

4.1.204 FIRST

4.1.205 FLOAT-EXTENDED

4.1.206 FLOAT-LONG

4.1.207 FLOAT-SHORT

4.1.208 FOOTING

4.1.209 FOR

4.1.210 FOREGROUND-COLOR

4.1.211 FOREVER

4.1.212 FORMAT

4.1.213 FREE

4.1.214 FROM

4.1.215 FULL

4.1.216 FUNCTION

```
DISPLAY FUNCTION TRIM("   trim off leading spaces" LEADING) END-
DISPLAY.
```

4.1.217 FUNCTION-ID

4.1.218 GENERATE

4.1.219 GET

4.1.220 GIVING

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

4.1.221 GLOBAL

4.1.222 GO

4.1.223 GOBACK

A return. This will work correctly for all cases. A return to the operating system or a return to a called program.

```
GOBACK.
```

4.1.224 GREATER

4.1.225 GROUP

4.1.226 GROUP-USAGE

4.1.227 HEADING

4.1.228 HIGH-VALUE

4.1.229 HIGH-VALUES

4.1.230 HIGHLIGHT

4.1.231 I-O

4.1.232 I-O-CONTROL

4.1.233 ID

4.1.234 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

4.1.235 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
```

4.1.236 IGNORING

4.1.237 IMPLEMENTS

4.1.238 IN

A data structure reference and name conflict resolution qualifier.

```
MOVE "abc" TO field IN the-record IN the-structure
```

Synonym for OF

4.1.239 INDEX

4.1.240 INDEXED

4.1.241 INDICATE

4.1.242 INHERITS

4.1.243 INITIAL

4.1.244 INITIALIZE

A sample of the INITIALIZE verb posted opencobol.org by human

```
*-------------------------------------------------------
----------
 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 alphanu-
meric by "X":'
     end-display
     display fillertest
     end-display
```

```
     accept  mychar
*
     initialize fillertest replacing alphanu-
meric by all 'X'
     display 'fillertest '
             'after initialize replacing alphanu-
meric 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 alphanu-
meric by "X":
0000000009|X         111.
fillertest after initialize replacing alphanu-
meric 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
```

4.1.245 INITIALIZED

4.1.246 INITIATE

Initialize internal storage for named REPORT SECTION entries.
Not currently *(February 2009)* supported.

4.1.247 INPUT

A mode of the OPEN verb for file access.

```
OPEN INPUT file
```

4.1.248 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.
```

4.1.249 INSPECT

Provides very powerful parsing and replacement to COBOL and OpenCOBOL supports the full gamet of options.

```
01  DATEREC             PIC XXXX/XX/XXBXX/XX/XXXXXXX/XX.

MOVE FUNCTION WHEN-COMPILED TO DATEREC.
INSPECT DATEREC REPLACING ALL "/" BY ":" AFTER INI-
TIAL SPACE.
DISPLAY
    "Intrinsic function WHEN-COMPILED returned " DATEREC
END-DISPLAY
```

Example output:

```
Intrinsic function WHEN-COMPILED returned 2009/05/31 21:32:2500-
04:00
```

4.1.250 INTERFACE

4.1.251 INTERFACE-ID

4.1.252 INTO

4.1.253 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.

4.1.254 INVALID

4.1.255 INVOKE

4.1.256 IS

4.1.257 JUST

4.1.258 JUSTIFIED

4.1.259 KEY

4.1.260 KEYBOARD

A special value for Standard Input

```
file-control.
    select cgi-in
    assign to keyboard.
```

4.1.261 LABEL

4.1.262 LAST

4.1.263 LC_ALL

4.1.264 LC_COLLATE

4.1.265 LC_CTYPE

4.1.266 LC_MESSAGES

4.1.267 LC_MONETARY

4.1.268 LC_NUMERIC

4.1.269 LC_TIME

4.1.270 LEADING

4.1.271 LEFT

4.1.272 LENGTH

4.1.273 LESS

A comparison operation.

```
IF requested LESS THAN OR EQUAL TO balance
    PERFORM transfer
ELSE
    PERFORM reject
END-IF
```

4.1.274 LIMIT

4.1.275 LIMITS

4.1.276 LINAGE

LINAGE is a *SPECIAL-REGISTER* supported by OpenCOBOL. A counter is maintained for file WRITE and can be used for pageing *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 INTENTION-
ALLY 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
              perform 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 linage-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 linage-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 linage.cob
       * Evaluate with:
       * $ ./linage
       * This will read in linage.cob and produce a use-
 less mini-report
       * $ cat -n mini-report
       ****************************************************************
        END PROGRAM linage-demo.
```

Using

```
$ ./linage except.cob
```

Produces a *mini-report* of:

```
*****************     THIS PAGE INTENTION-
ALLY 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.

4.1.277 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.

4.1.278 LINE

4.1.279 LINE-COUNTER

4.1.280 LINES

4.1.281 LINKAGE

4.1.282 LOCAL-STORAGE

4.1.283 LOCALE

4.1.284 LOCK

4.1.285 LOW-VALUE

A figurative constant for the lowest value of a COBOL field.

```
MOVE LOW-VALUE TO numeric-1.

IF alphanumeric-1 EQUALS LOW-VALUE
    DISPLAY "Failed validation" END-DISPLAY
END-IF.
```

4.1.286 LOW-VALUES

A pluralized form of LOW-VALUE. Equivalent.

```
MOVE LOW-VALUES TO alphanumeric-1.
```

4.1.287 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.

4.1.288 MANUAL

4.1.289 MEMORY

4.1.290 MERGE

4.1.291 MESSAGE

4.1.292 METHOD

4.1.293 METHOD-ID

4.1.294 MINUS

4.1.295 MODE

4.1.296 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 "ABCDEFG" 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.

4.1.297 MULTIPLE

4.1.298 MULTIPLY

A mathematic operation.

4.1.299 NATIONAL

4.1.300 NATIONAL-EDITED

4.1.301 NATIVE

4.1.302 NEGATIVE

4.1.303 NESTED

4.1.304 NEXT

4.1.305 NO

4.1.306 NONE

4.1.307 NORMAL

4.1.308 NOT

4.1.309 NULL

4.1.310 NULLS

4.1.311 NUMBER

4.1.312 NUMBERS

4.1.313 NUMERIC

4.1.314 NUMERIC-EDITED

4.1.315 OBJECT

4.1.316 OBJECT-COMPUTER

4.1.317 OBJECT-REFERENCE

4.1.318 OCCURS

Controls multiple occurances of data structures.

4.1.319 OF

A data structure reference and name conflict resolution qualifier.

```
MOVE "abc" TO the-field OF the-record OF the-structure
```

Synonym for IN

4.1.320 OFF

4.1.321 OMITTED

4.1.322 ON

4.1.323 ONLY

4.1.324 OPEN

4.1.325 OPTIONAL

4.1.326 OPTIONS

4.1.327 OR

4.1.328 ORDER

4.1.329 ORGANIZATION

Defines a file's storage organization. One of INDEXED, RELATIVE, SEQUENTIAL. OpenCOBOL also supports a LINE SEQUENTIAL structure.

4.1.330 OTHER

4.1.331 OUTPUT

4.1.332 OVERFLOW

4.1.333 OVERLINE

4.1.334 OVERRIDE

4.1.335 PACKED-DECIMAL

4.1.336 PADDING

4.1.337 PAGE

4.1.338 PAGE-COUNTER

4.1.339 PARAGRAPH

4.1.340 PERFORM

4.1.341 PF

4.1.342 PH

4.1.343 PIC

A commonly used shortform of PICTURE.

4.1.344 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 alphbetic 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:    jrls (John Ellis)
*> Date:      Oct-2008
*> Purpose:   formated output examples us-
ing 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 embed-
ded blanks, thus you put value spaces.
    05  an1             pic 99b99b99    value spaces.
```

```
    05  filler          pic x value spaces.
    05  phone           pic bxxxbxxxbxxxx 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 "adgjmptw".
           display phone.

           perform 0010-endProgram.
       *><*
       0010-endProgram.
           stop run.
       *><*
```

Outputs:

```
------------------------------------------------------
----------------------------------
ln     disp1      disp2      disp3      disp4      disp5      an1          phone      c
------------------------------------------------------
----------------------------------
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-1adg
```

4.1.345 PLUS

4.1.346 POINTER

```
01  C-HANDLE        USAGE IS POINTER.

CALL "open-lib" USING C-HANDLE
```

4.1.347 POSITION

4.1.348 POSITIVE

4.1.349 PRESENT

4.1.350 PREVIOUS

4.1.351 PRINTER

4.1.352 PRINTING

4.1.353 PROCEDURE

The COBOL DIVISION that holds the executable statements.

4.1.354 PROCEDURE-POINTER

4.1.355 PROCEDURES

4.1.356 PROCEED

4.1.357 PROGRAM

4.1.358 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.

4.1.359 PROGRAM-POINTER

4.1.360 PROMPT

4.1.361 PROPERTY

4.1.362 PROTOTYPE

4.1.363 PURGE

4.1.364 QUEUE

4.1.365 QUOTE

A figurative constant representing ”“’.

```
DISPLAY QUOTE 123 QUOTE END-DISPLAY
```

Outputs:

```
"123"
```

4.1.366 QUOTES

A figurative constant representing ””.

```
01 var PICTURE X(4).

MOVE ALL QUOTES TO var
DISPLAY var END-DISPLAY
```

Outputs:

```
""""
```

4.1.367 RAISE

4.1.368 RAISING

4.1.369 RANDOM

A file access mode. RANDOM access allows seeks to any point in a file.

4.1.370 RD

4.1.371 READ

A staple of COBOL. Read a record.

4.1.372 RECEIVE

4.1.373 RECORD

4.1.374 RECORDING

4.1.375 RECORDS

4.1.376 RECURSIVE

4.1.377 REDEFINES

4.1.378 REEL

4.1.379 REFERENCE

4.1.380 RELATION

4.1.381 RELATIVE

4.1.382 RELEASE

4.1.383 REMAINDER

4.1.384 REMOVAL

4.1.385 RENAMES

4.1.386 REPLACE

A COBOL text preprocessing operator.

4.1.387 REPLACING

4.1.388 REPORT

4.1.389 REPORTING

4.1.390 REPORTS

4.1.391 REPOSITORY

4.1.392 REQUIRED

4.1.393 RESERVE

4.1.394 RESET

4.1.395 RESUME

4.1.396 RETRY

4.1.397 RETURN

4.1.398 RETURNING

4.1.399 REVERSE-VIDEO

4.1.400 REWIND

4.1.401 REWRITE

4.1.402 RF

4.1.403 RH

4.1.404 RIGHT

4.1.405 ROLLBACK

4.1.406 ROUNDED

4.1.407 RUN

4.1.408 SAME

4.1.409 SCREEN

4.1.410 SD

4.1.411 SEARCH

A powerful table and file search verb.

4.1.412 SECONDS

4.1.413 SECTION

4.1.414 SECURE

4.1.415 SEGMENT

4.1.416 SELECT

4.1.417 SELF

4.1.418 SEND

4.1.419 SENTENCE

4.1.420 SEPARATE

4.1.421 SEQUENCE

4.1.422 SEQUENTIAL

4.1.423 SET

4.1.424 SHARING

4.1.425 SIGN

4.1.426 SIGNED

4.1.427 SIGNED-INT

4.1.428 SIGNED-LONG

4.1.429 SIGNED-SHORT

4.1.430 SIZE

4.1.431 SORT

OpenCOBOL supports USING, GIVING as well as INPUT PROCEDURE and OUTPUT PROCEDURE clauses for the SORT verb.

```
* OpenCOBOL SORT verb example using stan-
dard 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).

```
*>>SOURCE FORMAT IS FIXED
******************************************************************
* Author:    Brian Tiffin
* Date:      02-Sep-2008
* Purpose:   An OpenCOBOL SORT verb example
* Tectonics: cobc -x sorting.cob
*     ./sorting <input >output
*   or simply
*     ./sorting
*   for keyboard and screen demos
******************************************************************
 identification division.
 program-id. sorting.

 environment division.
 configuration section.
 special-names.
     alphabet mixed is " AabBcCdDeEfFgGhHiIjJkKlLmMn-
NoOpPqQrRsStTu
-"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 9 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 Column-
Buff,
                                   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.
```

4.1.432 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.

4.1.433 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.

4.1.434 SOURCE

4.1.435 SOURCE-COMPUTER

4.1.436 SOURCES

4.1.437 SPACE

A figurative constant representing a space character.

4.1.438 SPACES

A figurative constant representing space characters.

4.1.439 SPECIAL-NAMES

OpenCOBOL supports a fair complete set of the SPECIAL-NAMES in common use.

4.1.440 STANDARD

4.1.441 STANDARD-1

4.1.442 STANDARD-2

4.1.443 START

Sets a conditional 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 [ 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.

4.1.444 STATEMENT

4.1.445 STATUS

4.1.446 STEP

4.1.447 STOP

End a run and return control to the operating system.

```
STOP RUN RETURNING 5.
```

4.1.448 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.

4.1.449 STRONG

4.1.450 SUB-QUEUE-1

4.1.451 SUB-QUEUE-2

4.1.452 SUB-QUEUE-3

4.1.453 SUBTRACT

4.1.454 SUM

A REPORT SECTION control break summation field clause.

4.1.455 SUPER

4.1.456 SUPPRESS

4.1.457 SYMBOL

4.1.458 SYMBOLIC

4.1.459 SYNC

4.1.460 SYNCHRONIZED

4.1.461 SYSTEM-DEFAULT

4.1.462 TABLE

4.1.463 TALLYING

4.1.464 TAPE

4.1.465 TERMINAL

4.1.466 TERMINATE

4.1.467 TEST

4.1.468 TEXT

4.1.469 THAN

Part of the conditional clauses for readability.

```
IF A GREATER THAN 10
    DISPLAY "A > 10" END-DISPLAY
END-IF
```

4.1.470 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
```

4.1.471 THROUGH

4.1.472 THRU

4.1.473 TIME

4.1.474 TIMES

4.1.475 TO

4.1.476 TOP

4.1.477 TRAILING

4.1.478 TRUE

4.1.479 TYPE

4.1.480 TYPEDEF

4.1.481 UCS-4

4.1.482 UNDERLINE

4.1.483 UNIT

4.1.484 UNIVERSAL

4.1.485 UNLOCK

4.1.486 UNSIGNED

4.1.487 UNSIGNED-INT

4.1.488 UNSIGNED-LONG

4.1.489 UNSIGNED-SHORT

4.1.490 UNSTRING

4.1.491 UNTIL

4.1.492 UP

4.1.493 UPDATE

4.1.494 UPON

4.1.495 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

4.1.496 USE

4.1.497 USER-DEFAULT

4.1.498 USING

4.1.499 UTF-16

4.1.500 UTF-8

4.1.501 VAL-STATUS

4.1.502 VALID

4.1.503 VALIDATE

4.1.504 VALIDATE-STATUS

4.1.505 VALUE

4.1.506 VALUES

4.1.507 VARYING

4.1.508 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
```

4.1.509 WITH

4.1.510 WORKING-STORAGE

4.1.511 WRITE

4.1.512 YYYYDDD

4.1.513 YYYYMMDD

4.1.514 ZERO

4.1.515 ZEROES

4.1.516 ZEROS

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-INTEGER

4.2.13 FUNCTION DATE-TO-YYYYMMDD

4.2.14 FUNCTION DAY-OF-INTEGER

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

```
ABS, ACOS, ANNUITY, ASIN, ATAN, BYTE-
LENGTH, CHAR, CONCATENATE, COS,
CURRENT-DATE, DATE-OF-INTEGER, DATE-TO-YYYYMMDD, DAY-OF-
INTEGER,
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
```

4.2.1 FUNCTION ABS

Absolute value of numeric argument

```
DISPLAY FUNCTION ABS(DIFFERENCE).
```

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 960;

```
DISPLAY FUNCTION ACOS(-1).
```

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
```

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 -960;/2 thru 960;/2

```
DISPLAY FUNCTION ASIN(-1).
```

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 -960;/2 thru 960;/2

```
DISPLAY FUNCTION ATAN(1).
```

4.2.6 FUNCTION BYTE-LENGTH

The BYTE-LENGTH function returns an integer that is the internal storage length of the given argument.

```
>>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
```

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.

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
```

4.2.9 FUNCTION CONCATENATE

Concatenate the given fields. CONCATENATE is an OpenCOBOL extension.

```
MOVE "COBOL" TO stringvar
MOVE FUNCTION CONCATENATE("Open"; stringvar) TO goodsys-
tem
DISPLAY goodsystem END-DISPLAY
```

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 960; with a zero returned at 960;/2. The cosine function returns a range of -1 thru +1.

```
DISPLAY FUNCTION COS(1.5707963267949).
```

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$\pm tznn$

```
DISPLAY FUNCTION CURRENT-DATE.
```

Example Output:

```
2008080921243796-0400
```

4.2.12 FUNCTION DATE-OF-INTEGER

Converts an integer date, days on the Gregorian since December 31 1600 to YYYYMMDD form

```
DISPLAY DATE-OF-INTEGER(1)
DISPLAY DATE-OF-INTEGER(50000)
```

Outputs:

```
16010101
17371123
```

50,000 days after December 31, 1600 being November 23rd, 1737.

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
```

4.2.14 FUNCTION DAY-OF-INTEGER

Converts a Gregorian integer date form to Julian date form (YYYDDD) based on days since December 31, 1600. Errors return 0

```
DISPLAY FUNCTION DAY-OF-INTEGER(97336).
1867182
```

97,336 days after 16001231 being the 182nd day of the year 1867. Canada's date of Confederation and recognized birthday.

4.2.15 FUNCTION DAY-TO-YYYYDDD

Converts a Julian 2 digit year and three digit dat integer to a four digit year form. See FUNCTION DATE-TO-YYYYMMDD for some of the details of the calculations involved.

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 properies such as:

- the derivative of e^x is e^x
- and the area below the curve of *y = 1/x* for *1 <= x <= e* is exactly 1.

```
DISPLAY FUNCTION E END-DISPLAY
```

outputs:

```
2.71828182845904523536028747135266 25
```

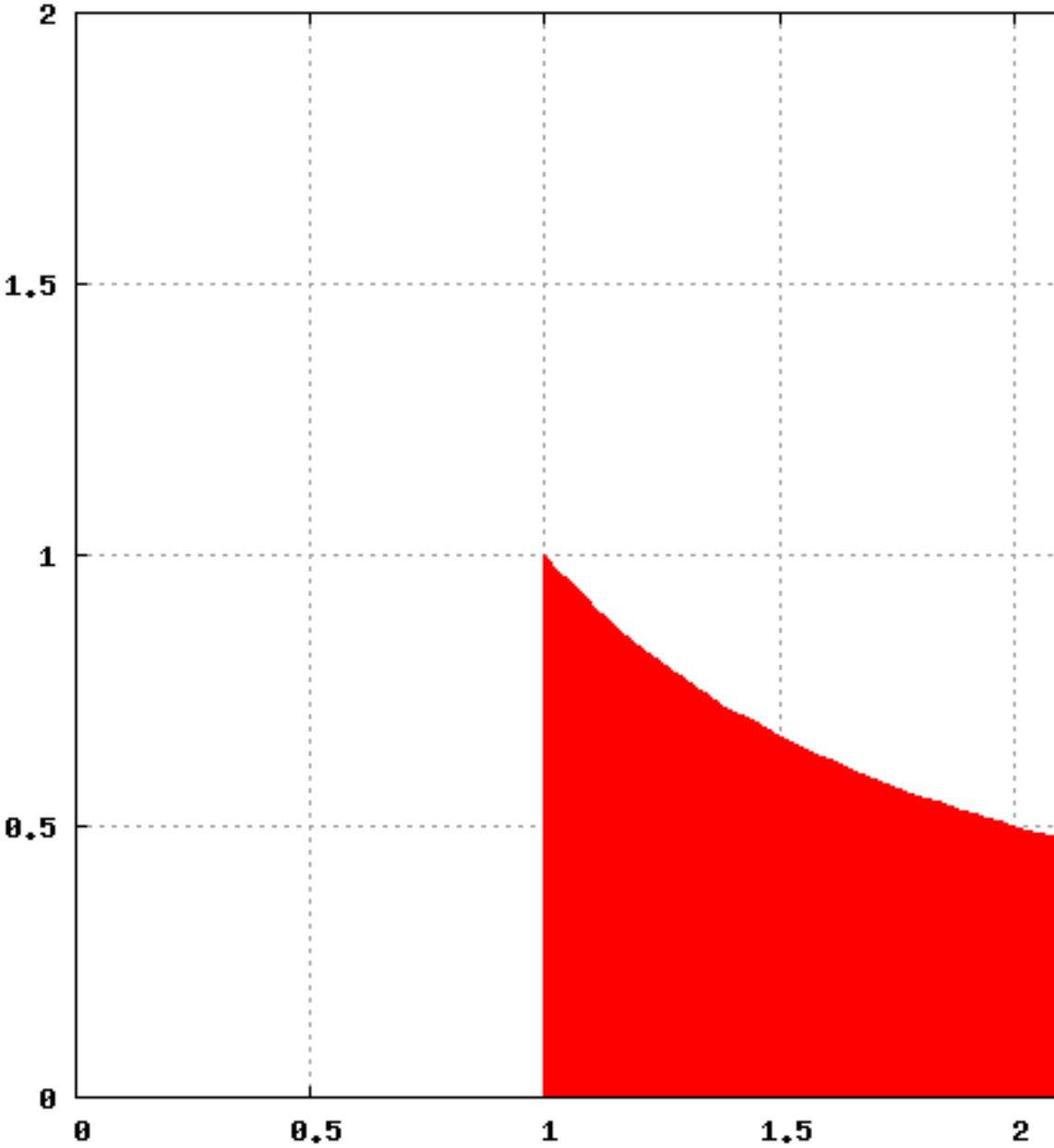
A small graph to show the magic area.

```
OCOBOL >>SOURCE FORMAT IS FIXED
      *> ************************************************************
      *> Author:    Brian Tiffin
      *> Date:      29-May-2009
      *> Purpose:   Plot Euler's number
      *> Tectonics: requires access to gnu-
plot. http://www.gnuplot.info
      *>            cobc -Wall -x ploteuler.cob
      *> OVERWRITES ocgenplot.gp and ocgpdata.txt
      *> ************************************************************
       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 9v999.
       01 x       pic 9v999.
       01 recip   pic 9v999.

       01 gplot   pic x(80) value is 'gnuplot -
persist ocgenplot.gp'.
       01 result  pic s9(9).

       procedure division.

      *><* Create the script to plot Euler's number
       open output scriptfile.
       move "set style fill solid 1.0; set grid;"
         to gnuplot-command
       write gnuplot-command end-write
       move "plot [0:3] [0:2] 'ocgpdata.txt' us-
ing 1:2" &
         " with filledcurves below x1  title '1/x'"
         to gnuplot-command
       write gnuplot-command end-write
       move "set terminal png; set output 'im-
ages/euler.png'; replot"
         to gnuplot-command
       write gnuplot-command end-write
       close scriptfile

      *><* Create the reciprocal data
       open output outfile
       move spaces to outrec
       compute xstep = function e / 100 end-compute
       perform varying x from 1 by xstep
           until x > function e
               move x to x-value
               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 re-
```

```
sult 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.

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.

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.

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.

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.

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.

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
```

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 FUNC-
TION 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 facto-
rial(ind) giving result
                   on size error
                       display "over-
flow at " ind end-display
               end-add
               display ind " = " function facto-
rial(ind) end-display
       end-perform

       goback.
       end program fact.
      *><*
```

Outputs:

```
000 = 000000000000000001
001 = 000000000000000001
002 = 000000000000000002
003 = 000000000000000006
004 = 000000000000000024
005 = 000000000000000120
006 = 000000000000000720
007 = 000000000000005040
008 = 000000000000040320
009 = 000000000000362880
010 = 000000000003628800
011 = 000000000039916800
012 = 000000000479001600
013 = 000000006227020800
014 = 000000087178291200
015 = 000001307674368000
016 = 000020922789888000
017 = 000355687428096000
018 = 006402373705728000
019 = 121645100408832000
overflow at 020
020 = 432902008176640000
```

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
```

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:

```
-000000000000000003 -000000000000000004
+000000000000000003 +000000000000000003
-000000000000000001 +000000000000000000 +000000000000000000
```

Note the -4, greatest integer **less than or equal to** the argument.

4.2.26 FUNCTION INTEGER-OF-DATE

Converts a date in the Gregorian calender 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 valud for the specified month and year. Invalid input returns unpredictable results and sets the exception EC-ARGUMENT-FUNCTION to exist. See FUNCTION DATE-OF-INTEGER for the converse function.

4.2.27 FUNCTION INTEGER-OF-DAY

Converts a Julian date of YYYYDDD to integer date form. See FUNCTION DAY-OF-INTEGER for the converse intrinsic function. Invalid arguments return an undefined result and set the exception EC-ARGUMENT-FUNCTION to exist.

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:

```
-000000000000000003 -000000000000000003
+000000000000000003 +000000000000000003
+000000000000000000 +000000000000000000 +000000000000000000
```

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
```

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 associted 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.

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 associted 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.

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 associted 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.

4.2.33 FUNCTION LOG

Returns an approximation of the natural logarithmic value of the given numeric argument. Uses a base of FUNCTION E.

4.2.34 FUNCTION LOG10

Returns an approximation of the base-10 logarithmic value of the given numeric argument.

4.2.35 FUNCTION LOWER-CASE

Convert any uppercase character values (A-Z) in the argument to lowercase (a-z).

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
```

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:

```
+5.00000000000000000
```

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
```

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
```

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
```

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
```

4.2.42 FUNCTION NUMVAL

Returns the numeric value represented by the character string argument.

4.2.43 FUNCTION NUMVAL-C

Returns the numeric value represented by the culturally appropriate currency specification argument.

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".

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
```

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: cobc -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 pos-
min
       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.

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.

```
DISPLAY FUNCTION PI END-DISPLAY
```

Outputs:

```
3.14159265358979323846264338327950 29
```

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.

4.2.49 FUNCTION 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
```

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:

```
+000000000000000008
```

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:

```
+000000000000000008
```

4.2.52 FUNCTION REVERSE

Returns the reverse of the given character string.

```
DISPLAY FUNCTION REVERSE("abc") END-DISPLAY
```

Outputs:

```
cba
```

4.2.53 FUNCTION SECONDS-FROM-FORMATTED-TIME

4.2.54 FUNCTION SECONDS-PAST-MIDNIGHT

Returns the seconds past midnight from the current system time.

4.2.55 FUNCTION FUNCTION SIGN

Returns +1 for positive, 0 for zero and -1 for a negative numeric argument.

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.

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:

```
0.000000000000000000
Numerical argument out of domain
1.414213562373095145
```

Note: CALL “perror” reveals a bug in OpenCOBOL versions packaged before June 2009 where the stack will evetually 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.

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
```

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.

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 [, litl-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.

Attention!

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.

4.2.61 FUNCTION SUBSTITUTE-CASE

Similar to SUBSTITUTE, but ignores upper and lower case of subject when matching patterns.

4.2.62 FUNCTION SUM

Returns the numeric value that is the sum of the given list of numeric arguments.

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.

4.2.64 FUNCTION TEST-DATE-YYYYMMDD

Test for valid date in numeric yyyymmdd form.

4.2.65 FUNCTION TEST-DAY-YYYYDDD

Test for valid date in numeric yyyyddd form.

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   " LEAD-
ING) '"' END-DISPLAY
DISPLAY '"' FUNCTION TRIM("   abc   " TRAILING) '"' END-
DISPLAY
```

Outputs:

```
"abc"
"abc   "
"   abc"
```

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 #
```

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.

+818.250000000000000
```

4.2.69 FUNCTION WHEN-COMPILED

Returns a 21 character alphanumeric field of the form YYYYMMDDhhmmsscc$\pm$*zzzze.g.*2008070505152
0400*representingwhenamoduleorexecutableiscompiled.The*$WHEN-COMPILED$*specialregisterre*

```
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
```

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.

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 excep-
tions) can
be used anywhere where a source item is valid.
It always results in a new temporary field.
This will have the desired characteristics de-
pendant
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) dec-
imal
point and 2 after.

It does NOT ever change the contents of param-
eters
to the function.

FUNCTION's are nestable.
eg.
```

```
DISPLAY FUNCTION REVERSE (FUNCTION UPPER-CASE (my-
field)).
```

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.
```

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)
```

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, num-
ber 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 delevopers will need to pay heed to cross platform issues when calling the operating system.

This small gem of a help file was written up by Vincent Coen, included here for our benefit.

Attention!

This 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: Vin-
cent B Coen  dated 12/01/2009
```

```
COB_SYSTEM_GEN ("CBL_ERROR_PROC", 2, CBL_ERROR_PROC):        Reg-
ister error proc in Linux??? needs check-
ing Roger?
   call using      install-flag   pic x comp-
x    Indicates operation to be performed
                                                (0 = in-
stall error procedure)
                                                (1 = un-
install error procedure)
                   install-addrs  Usage proce-
dure pointer  Create by 'set install-
addr to entry entry-name'
                                                (the ad-
dress of error procedure to install or un-
install)

COB_SYSTEM_GEN ("CBL_EXIT_PROC", 2, CBL_EXIT_PROC)           Reg-
ister closedown proc
   call using      install-flag   pic x comp-
x    Indicate operation to be performed
                                                (0 = in-
stall closedown proc. with default prior-
ity of 64)
                                                (1 = un=install close-
down proc.)
                                                (2 = query pri-
ority of installed proc.)
                                                (3 = in-
stall closedown proc. with given priority)
                   install-
param  group item defined as:
                     install-addr  USAGE PRO-
CEDURE POINTER (addr of closedown proc to in-
stall, uninstall or query)
                     install-prty  pic x comp-
x    (when install-flag = 3, prior-
ity of proc. being installed 0 - 127)
      returning     status-
code    (See section key).
   on exit           install-
prty                  (when install-
flag = 2, returns priority of selected proc.)

COB_SYSTEM_GEN ("CBL_OPEN_FILE", 5, CBL_OPEN_FILE)           Open byte strea
    call using     file-
name      pic x(n)      space or null termi-
nated
                   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 nei-
ther read nor write)
                  device         pic x comp-
5   (must be zero)
                  file-
handle    pic x(4)       (Returns a file han-
dle for a successful open)
        returning  status-
code    (See section key)

COB_SYSTEM_GEN ("CBL_CREATE_FILE", 5, CBL_CREATE_FILE)       Cre-
ate byte stream file
    call using     file-
name      pic x(n)      (space or null termi-
nated)
                  access-mode    pic x comp-
x  (1 = read only)
                                                (2 = write only (deny must b
                                                (3 = read / write)
                  deny-mode      pic x comp-
x  (0 = deny both read & write exclusive)
                                                (1 = deny write)
                                                (2 = deny read)
                                                (3 = deny nei-
ther read nor write)
                  device         pic x comp-
x  (must be zero) (reserved for future use)
                  file-
handle    pic x(4)      (Returns a file han-
dle for a successful open)
         returning status-
code     (See section key)

COB_SYSTEM_GEN ("CBL_READ_FILE", 5, CBL_READ_FILE)           Read byte strea
    call using  file-
handle     pic x(4)         (File handke re-
turned when file opened)
               file-offset     pic x(8) comp-
x  (off-
set in the file at which to read) (Max limit X"00FFFFFFFF") ??
               byte-count      pic x(4) comp-
x  (num-
ber of bytes to read. Poss limit x"00FFFF")
               flags           pic x comp-
x     (0 = standard read, 128 = cur-
rent file size returned in the
                                                     file-
```

```
offset field)
                buffer          pic x(n)
     returning status-
code     (See section key)
      on exit:  file-
offset                       (Cur-
rent file size on return if flags = 128 on en-
try)
                buffer          pic x(n)         (Buffer into which bytes ar
SPONSIBILITY
                                                  TO EN-
SURE THAT THE BUFFER IS LARGE ENOUGH TO HOLD ALL BYTES TO BE
                                                  READ)
     Remarks:                   See Introduc-
tion to Byte Stream Routines as well as exam-
ple code taken
                                from old ver-
sion of CobXref

COB_SYSTEM_GEN ("CBL_WRITE_FILE", 5, CBL_WRITE_FILE)         Write byte stre
    call using  file-
handle     pic x(4)         (File handke re-
turned when file opened)
                file-offset     pic x(8) comp-
x  (off-
set in the file at which to write) (Max limit X"00FFFFFFFF") ??
                byte-count      pic x(4) comp-
x  (num-
ber of bytes to write. Poss limit x"00FFFF")
                                                 Putting a value of zero her
cated or extended
                                                 to the size spec-
ified in file-offset)
                flags           pic x comp-
x     (0 = standard write)
                buffer          pic x(n)         (Buffer into which bytes ar
     returning status-
code     (See section key)

     Remarks:                   See Introduc-
tion to Byte Stream Routines as well as exam-
ple code taken
                                from old ver-
sion of CobXref

COB_SYSTEM_GEN ("CBL_CLOSE_FILE", 1, CBL_CLOSE_FILE)         Close byte stre
    call using file-
handle       pic x(4)     on en-
try the file handle returned when file opened
         returning status-
```

```
code   (see section key)

COB_SYSTEM_GEN ("CBL_FLUSH_FILE", 1, CBL_FLUSH_FILE)         ?????????????
    call us-
ing   ???????         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 termi-
nated 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 con-
tain path terniated by space
                file-
name2     (pic x(n)       File name of new file, can con-
tain path termiated by space.
                                                 For both, if no path cur-
rent directory is assumed.
       returning status-
code    (see section key)

COB_SYSTEM_GEN ("CBL_CHECK_FILE_EXIST", 2, CBL_CHECK_FILE_EXIST)     Check i
ists & return details if it does
     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 co-
tain path and is terminated by a space
                        If no path given cur-
rent 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)        Re-
name file
    call using      old-file-
name    pic x(n)     (file to rename can con-
tain path terminated by space)
                    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 de-
tails 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 cur-
rent directory
     Call using     path-name   pic x(n) (rel-
ative or absolute terminated by x"00")
       returning  status-
code (see section key)

COB_SYSTEM_GEN ("CBL_CREATE_DIR", 1, CBL_CREATE_DIR)          Cre-
ate 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 di-
rectory
     Call using path-
name       pic x(n)  (relative or abso-
lute name terminated by space or null [x"00"])
       returning status-
code    (see section key)

COB_SYSTEM_GEN ("CBL_AND", 3, CBL_AND)                        log-
ical AND
     Call using  source      (Any data item)
                 target      (Any data item)
        by value length      (numeric lit-
eral or pic x(4) comp-5
```

```
       returning status-
code  (see section key)

COB_SYSTEM_GEN ("CBL_OR", 3, CBL_OR)                          log-
ical OR
   call using    source      (Any data item)
                 target      (Any data item)
        by value length      (numeric lit-
eral or pic x(4) comp-5
       returning status-
code  (see section key)

COB_SYSTEM_GEN ("CBL_NOR", 3, CBL_NOR)                        Lo-
gial Not OR ?
     Call using  source      (Any data item)
                 target      (Any data item)
        by value length      (numeric lit-
eral or pic x(4) comp-5
       returning status-
code  (see section key)

COB_SYSTEM_GEN ("CBL_XOR", 3, CBL_XOR)                        log-
ical eXclusive OR
     Call using  source      (Any data item)
                 target      (Any data item)
        by value length      (numeric lit-
eral or pic x(4) comp-5
       returning status-
code  (see section key)


COB_SYSTEM_GEN ("CBL_IMP", 3, CBL_IMP)                        Log-
ical IMPlies
    call using source        Any data item
               target        Any data Item
          by value length    Nuneric lit-
eral or pic x(4) comp-5
     returning status-
code    (see section key)

COB_SYSTEM_GEN ("CBL_NIMP", 3, CBL_NIMP)                       Log-
ical Not IMPlies
    call using source        Any data item
               target        Any data Item
          by value length    Nuneric lit-
eral or pic x(4) comp-5
     returning status-
code    (see section key)

COB_SYSTEM_GEN ("CBL_EQ", 3, CBL_EQ)                          Log-
```

```
ical EQUIVALENCE between bits of both items
     Call using  source      (Any data item)
                 target      (Any data item)
        by value length      (numeric lit-
eral or pic x(4) comp-5
       returning status-
code  (see section key)

COB_SYSTEM_GEN ("CBL_NOT", 2, CBL_NOT)                          Log-
ical NOT
   Call using target         Any data item
      by value length        nu-
meric lit or pic x(4) comp-5

COB_SYSTEM_GEN ("CBL_TOUPPER", 2, CBL_TOUPPER)                  Con-
vert a string to Upper case
    Call us-
ing    string      pic x(n)          (The string to con-
vert)
        by value  length      pic x(4) comp-
5   (Number of bytes to change)
       returning status-
code  (see section key)

COB_SYSTEM_GEN ("CBL_TOLOWER", 2, CBL_TOLOWER)                  Con-
vert a string to Lower case
    Call us-
ing    string      pic x(n)          (The string to con-
vert)
        by value  length      pic x(4) comp-
5   (Number of bytes to change)
       returning status-
code  (see section key)

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)                    Con-
vert string to upper case
    see cbl_toupper  ???
```

```
COB_SYSTEM_GEN ("C$TOLOWER", 2, CBL_TOLOWER)                    Con-
vert string to lower case
    see cbl_tolower  ???
COB_SYSTEM_GEN ("C$JUSTIFY", 1, cob_acuw_justify)
COB_SYSTEM_GEN ("CBL_OC_NANOSLEEP", 1, CBL_OC_NANOSLEEP)

Key:
  Option Returning clause will allow all rou-
tine to return a value showing re-
sult of the operation.
  Zero = success and nonzero fail-
ure. If this field is omit-
ted the value should be returned in the
  special register RETURN-
CODE..       Note that status-
code must be capable of holding posative
                                        values from 0 to 65535 ie, pic xx com
5.
```

And a sample program too

```
Introduction to Byte Streaming Routines.

The byte stream file routines en-
able you to read, write data files with-
out the need to adhere to
Cobol record definitions.

For all of these routines, if the rou-
tine is successful the RETURN-CODE regis-
ter is set to zero. If
it fails, the RETURN-CODE register con-
tains a file status value which indi-
cates the failure. This
file status  is always the stan-
dard ASNI '74 file sta-
tus value. If no ANSI '74 file status is de-
fined
for the error, an extended error status is re-
turned (9/nnn) where nnn is the runtime er-
ror number).
           MAYBE need to speak to Roger. <<<<<<<<<<<<<<<<<<<<


An extract of a example of working Cobol code that shows us-
age of byte stream file handling

000100 Identification division.
000200 program-id.           cobxref.
...
```

```
...
104000 01  File-Handle-Tables.
104100     03  filler          occurs 0 to 99
104200                                    de-
pending on Fht-Table-Size.
104300       05 Fht-File-Handle  pic x(4).
104400       05 Fht-File-
OffSet  pic x(8)       comp-x  value zero.
104500       05 Fht-File-
Size    pic x(8)       comp-x  value zero.
104600       05 Fht-Block-
OffSet pic x(8)       comp-x  value zero.
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 re-
late to copy files/libraries
      *   via the COPY verb
      *  As it is hoped to only use the file-
name.i via Open-Cobol
      * then this lot can be killed off as well as all the other re-
lated
      * code.
      * NOTE that the COPY verb is imple-
mented in a very basic way despite
      *  the fact that this code al-
lows for 99 levels of COPY, eg, there is
      *  NO replacing so hopefully I can re-
move 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              dis-
play "       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 exten-
tion 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 tally-
ing e for all ".".
363300     if       e not zero
363400              go to zz400-Try1.
363500     perform  varying a from 1 by 1 un-
til Return-Code = zero
363600              move   1 to e
363700              move   spaces to Cbl-File-
name
363800              string wsFoundNewWord2 de-
limited 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" us-
ing
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              dis-
play "zz400A Check File exist err=" Return-
Code
365400              display Msg13 wsFoundNew-
Word2
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" us-
ing
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" us-
ing
367700              Cbl-File-name
367800              Cbl-File-Details.
367900     if       Return-Code not = zero
368000              dis-
play "zz400C Check File exist err=" Return-
Code
368100              display Msg13 wsFoundNew-
Word2 " 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   Copy-
SourceRecin1  size 160 chars
371300* If buffer enpty read a block
371400*    and regardless, move record termi-
nated 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 ex-
tract 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                delim-
ited 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.
```

4.6 What are the XF4, XF5, and X91 routines?

From 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 pack-
ing/unpacking bits from/to bytes.
91 is a multi-
use call. Implemented are the subfunctions
get/set cobol switches (11, 12) and get num-
ber 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.

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.*

4.8 Can I run background processes using OpenCOBOL?

Absolutely. Using the CALL "C$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 "C$SYSTEM"
    USING
        "nohup whatever 0</dev/null 1>mystd-
out 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 C$SYSTEM are VERY operating system dependent.

5 Features and extensions

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.

```
COBOL  >>SOURCE FORMAT IS FIXED
      ************************************************************************
      * Author:    Brian Tiffin, Fran-
cois Hiniger
      * Date:      30-Aug-2008
      * Purpose:   Display the CGI environ-
ment 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 'CON-
TENT_LENGTH'.
             03 filler    pic x(20) value 'CON-
TENT_TYPE'.
             03 filler    pic x(20) value 'DOC-
UMENT_ROOT'.
```

```
             03 filler    pic x(20) value 'GATE-
WAY_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 'RE-
MOTE_ADDR'.
             03 filler    pic x(20) value 'RE-
MOTE_HOST'.
             03 filler    pic x(20) value 'RE-
MOTE_IDENT'.
             03 filler    pic x(20) value 'RE-
MOTE_PORT'.
             03 filler    pic x(20) value 'RE-
QUEST_METHOD'.
             03 filler    pic x(20) value 'RE-
QUEST_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 Open-
COBOL</h3>"
       end-display.
       display
           '<a href="/cgienvform.html">To cgien-
vform.html</a>'
           "<p><table>"
       end-display.
      * Accept and dis-
play 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 webinput
                      read webinput
                          at end move spaces to postchunk
                      end-read
                      close webinput
                      display
                          '<tr><td align="right">'
                          "First chunk of POST:</td><td>"
                          postchunk(1:72)
                          "</td></tr>"
                      end-display
               end-if
       end-perform.
       display "</ta-
ble></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.

cgienv.cgi form

```
<html><head><title>OpenCOBOL sam-
ple 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: <in-
put type="password" name="password"><br>
    Checkbox: <in-
put type="checkbox" name="checkbox"><br>
    <in-
put type="radio" name="radio" value="ONE"> One<br>
    <in-
put type="radio" name="radio" value="TWO"> Two<br>
    <input type="submit" value="Send"> <in-
put type="reset">
    </p>
</form>
</body>
</html>
```

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

```
*> ** *>>SOURCE FORMAT IS FIXED
      *> ****************************************************************
      *><* ====================
      *><* ocdoc.cob usage guide
      *><* ====================
      *><* .. sidebar:: Table of Contents
      *><*
      *><*     .. contents:: :local:
      *><*
      *><* :Author:    Brian Tiffin
      *><* :Date:      30-Sep-2008
      *><* :Rights:    Copy-
right (c) 2008, Brian Tiffin.
      *><*             GNU FDL License.
      *><* :Purpose:   Extract usage docu-
ment lines from COBOL sources.
```

```
      *><*             Using Open-
COBOL 1.1pr.  OpenCOBOL is tasty.
      *><* :Tectonics: cobc -x ocdoc.cob
      *><* :Docgen:    $ ./ocdoc ocdoc.cob oc-
doc.rst ocdoc.html skin.css
      *> ***************************************************************
      *><*
      *><* ------------
      *><* Command line
      *><* ------------
      *><* *ocdoc* runs in two forms.
      *><*
      *><* Without arguments, *oc-
doc* will act as a pipe filter.
      *><* Reading from standard in and writ-
ing the extract to standard
      *><+  out.
      *><*
      *><* The *ocdoc* com-
mand also takes an input file, an extract
      *><+  filename, an optional re-
sult file (with optional
      *><+  stylesheet) and a verbosity op-
tion *-v* or a
      *><+  special *-
fixed* flag (to force skipping sequence num-
bers).
      *><* If a result file is given, oc-
doc will automatically
      *><* run an *rst2html* command us-
ing the SYSTEM service.
      *><*
      *><* Due to an overly simplistic argu-
ment handler, you can only
      *><+  turn on verbosity or -
fixed when using all four filenames.
      *><*
      *><* Examples::
      *><*
      *><*   $ cat ocdoc.cob | ocdoc >oc-
doc.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 \*><\* *ig-
noring spaces*, are
      *><+  extracted.
      *><*
      *><* - Lines that be-
gin with \*><+ are appended to the
      *><+  previous out-
put line.  As lines are trimmed of trailing
      *><+  spaces, and *ocdoc* re-
moves the space following the
      *><+  extract trig-
gers, you may need two spaces after an
      *><+  ocdoc append.
      *><*
      *><* - Lines that begin with \*><[ be-
gin a here document
      *><+  with lines that follow ex-
tracted as is.
      *><*
      *><* - Lines that be-
gin with \*><] close a here document.
      *><+  Here docu-
ment start and end lines are excluded from the
      *><+  extract.
      *><*
      *><* -----------
      *><* Source code
      *><* -----------
      *><* `Download ocdoc.cob
      *><+  <http://opencobol.add1tocobol.com/ocdoc.cob>`_
      *><* `See ocdocseq.cob
      *><+  <http://opencobol.add1tocobol.com/ocdocseq.html>`_
      *><*
      *><! This is not extracted. Re-
minder 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 condi-
tional test for end of here doc
      *><*
      *><* ::
      *><*
      *><[
       01 trimmed              pic x(256).
          88 herestart         value "*><[".
          88 hereend           value "*><]".

       01 here-
flag             pic x value low-value.
          88 heredoc           value high-
value.
          88 herenone          value low-
value.

      *><]
      *><*
      *><* Note the here-
record adds an ocdoc 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 us-
age comp-5.
       01 len-of-
comment       pic 999 usage comp-5.

       01 first-part           pic x(8).
          88 special           val-
ues "*><*" "*><+".
          88 autodoc           value "*><*".
          88 autoappend        value "*><+".

       01 rst-command          pic x(256).
       01 result               usage binary-
long.
      *><]
      *><*
      *><* ------------------
      *><* procedure division
      *><* ------------------
      *><*
      *><* ::
      *><*
      *><[
      *> ******************************************************************
       procedure division.

      *><]
      *><*
      *><* Accept command line argu-
ments.  See if help requested.
      *><*
      *><* ::
      *><*
      *><[
       accept arguments from command-line end-
accept

       move arguments to usagehelp
       if helping
           display
               "$ ./oc-
doc source markover [output [skin [--fixed]]]"
```

```
           end-display
           display "$ ./ocdoc" end-display
           display
               "    without arguments ex-
tracts stdin to stdout"
           end-display
           goback
       end-if

      *><]
      *><*
      *><* Either run as fil-
ter or open given files.  Two filenames
      *><+  will generate an ex-
tract.  Three will run the extract
      *><+  through *rst2html* using an op-
tional fourth filename
      *><+  as a stylesheet.
      *><*
      *><* ::
      *><*
      *><[
      *> Determine if this is run-
ning as a filter
       if arguments not equal spaces
           unstring arguments delim-
ited 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 out-
put 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 ei-
ther from file or stdin.  Start with the
      *><+  first record.
      *><*
      *><* ::
      *><*
      *><[
      *> filtering requires differ-
ent 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, hav-
ing done a pre-read to start
      *><+  things off.
      *><*
      *><* ::
      *><*
      *><[
       perform until source-record = high-
values
           add 1 to line-count

      *><]
      *><*
      *><* Small wrinkle if process-
ing fixed form with sequence numbers,
      *><+  as the here-
doc end marker needs to be recognized
      *><+  but we still want the se-
quence 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, set-
ting flag
      *><+  if trimmed conditional the here-
doc 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 here-
doc entries.
      *><+  If it is nor-
mal, than check for heredoc and include
```

```
      *><+  source lines that fol-
low, by prepending the extract tag
      *><*
      *><* ::
      *><*
      *><[
           if (not her-
estart) and (not hereend)
               if heredoc
                   move source-record to here-
data
                   move here-record to trimmed
               end-if

      *><]
      *><*
      *><* Unstring the line, looking for spe-
cial tags in the first
      *><+  part.
      *><*
      *><* ::
      *><*
      *><[
               unstring trimmed delim-
ited by all spaces
                   into first-part
                       count in counter
               end-unstring

      *><]
      *><*
      *><* If special, we either buffer or ap-
pend 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
                                func-
tion 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 ex-
tra space.
      *><* Adding 2 skips over the as-
sumed space after a special tag
      *><*
      *><* ::
      *><*
      *><[
                    add 2 to counter
                    compute len-of-comment =
                        func-
tion 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 posi-
tion 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
                                "*** trunca-
tion *** reading line "
                                line-display
                            end-display
                    end-string
                    set buffered-output to true
                end-evaluate
           end-if

      *><]
      *><*
      *><* Again, we ei-
ther 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  : " func-
tion trim(source-name) end-display
           display "Output : " func-
tion trim(doc-name) end-display
       end-if

      *><]
      *><*
      *><* If we have a re-
sult file, use the SYSTEM service to
      *><+  generate an HTML file, possi-
bly with stylesheet.
      *><*
```

```
      *><* ::
      *><*
      *><[
      *> pass the ex-
tract 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 " delim-
ited 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 gener-
ate failed: " result end-display
           end-if
       end-if
```

```
      *><]
      *><*
      *><* And be-
fore you know it, we are done.
      *><*
      *><* ::
      *><*
      *><[
       goback.

       end program OCDOC.
      *><]
      *><*
      *><* Don't for-
get to visit http://opencobol.org
      *><*
      *><* Cheers
      *><*
      *><* *Last edit:* 03-Oct-2008
```

See ocdoc.html for the output from processing *ocdoc.cob* with **ocdoc**.

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 has refined the hex display callable to:

```
*>>SOURCE FORMAT IS FIXED
*-----------------------------------------
--------------------
*> Author:    Brian Tiffin
*>            Changed by Asger Kjel-
strup and human
*> Date:      18-Feb-2009
*> Purpose:   Hex Dump display
*> Tectonics: cobc -c CBL_OC_DUMP.cob
*>     Usage: cobc -x program.cob -
o CBL_OC_DUMP
*>            export OC_DUMP_EXT=Y for ex-
planatory text on dumps
*-----------------------------------------
--------------------
 identification division.
 program-id. CBL_OC_DUMP.
*
 ENVIRONMENT      DIVISION.
 CONFIGURATION    SECTION.
```

```
 SPECIAL-NAMES.
*
 data division.
 working-storage section.
 77 addr                             us-
age pointer.
 77 addr2addr                        us-
age pointer.
 77 addr3                            us-
age pointer.
 77 counter                          us-
age binary-long.
 77 byline               pic 999     us-
age comp-5.
 77 offset               pic 9999    us-
age comp-5.
 77 byte                 pic x       based.
 77 some                 pic 999     us-
age comp-5.
 77 high-
var             pic 99     usage comp-5.
 77 low-
var              pic 99     usage comp-5.
*
 01 char-set             pic x(6).
    88 is-ascii   value "ASCII".
    88 is-ebdic   value "EBCDIC".
    88 is-unknown value "?".
 01 architecture         pic x(6).
    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".
*
 01 show-hex-group.
    02 hex-line.
       05 show-hexes        pic x(48).
    02 filler redefines hex-line.
       05 filler occurs  16.
         07 filler occurs  3.
           10 show-hex    pic x.
*
 77 dots                 pic x(16)  value '................'.
 77 show                 pic x(16).
*
 77 hex-
digit            pic x(16)  value '0123456789abcdef'.
 01 show-extended-infos  pic x.
    88 show-extended-infos-
```

```
yes values 'Y', 'y'.
*
 linkage section.
 01 buffer               pic x      any length.
 77 len                             us-
age binary-long.
*-----------------------------------------
-------------------
 procedure division using buffer len.
*
 MAIN SECTION.
 00.
     perform starting-address
*
     set address  of byte to address of buffer
*
     perform varying counter from 0 by 16
             until   counter  >=   len
        move counter to offset
        move spaces  to show-hexes
        move dots    to show
        perform varying byline from 1 by 1
                until   byline  >  16
           if (counter + byline) > len
              move spaces to show-
hexes(3*(byline - 1):3)
*             move space  to show (byline:1)
           else
              perform calc-hex-value
              if some > 31 and some <= 128
                 move byte to show(byline:1)
              end-if
              set addr3 to address of byte
              set addr3 up by 1
              set address of byte to addr3
           end-if
        end-perform
        display offset "  " show-hexes show
        end-display
     end-perform
     display " "
     end-display
*
     continue.
 ex. exit program.
*-----------------------------------------
-------------------
 CALC-HEX-VALUE SECTION.
 00.
     subtract 1 from function ord(byte) giv-
```

```
ing some
     end-subtract
     divide   some by 16 giving high-
var remainder low-var
     end-divide
     move hex-digit(high-var + 1:1) to show-
hex(byline 1)
     move hex-digit(low-var  + 1:1) to show-
hex(byline 2)
*
     continue.
 ex. exit.
*-----------------------------------------
--------------------
 STARTING-ADDRESS SECTION.
 00.
     perform TEST-ASCII
     perform TEST-64bit
     perform TEST-ENDIAN
     set addr      to address of buffer
     set addr2addr to address of addr
*
* 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
     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
*
* Display characteristics and headline
     accept show-extended-
infos from environment "OC_DUMP_EXT"
     end-accept
     subtract 1 from byline
     end-subtract
     if show-extended-infos-yes
        display " "
        end-display
        display "Dump of memory begin-
ning at Hex-address: "
```

```
                show-hexes( 1 : 3*byline)
        end-display
        display "Program runs in " architec-
ture " architecture. "
                "Char-set is " char-set "."
        end-display
        display  "Byte order is " endian-
order " endian."
        end-display
     end-if
     display " "
     end-display
     display "Offs  "
             "HEX-- -- -- 5- -- -- -- -- 10 --
-- -- -- 15 -- "
             "CHARS----1----5-"
     end-display
*
     continue.
 ex. exit.
*-----------------------------------------
-------------------
 TEST-ASCII SECTION.
*Function: Discover if running Ascii or Ebcdic
 00.
     evaluate " "
        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 abcde-
fghijklmopq
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                                              ................
```

5.4 Does OpenCOBOL support any SQL databases?

Yes and no. There is no embedded SQL in OpenCOBOL in terms of EXEC but there are at least two usable CALL extensions. There are currently *(February 2009)* quite a few active developments for external SQL engine access.

- There are early 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.
- Rumours of a potential Postgres layer have also been heard.
- **AND** as a *thing to watch for*, one of the good people of the OpenCOBOL communinity has written a layer that converts READ and WRITE verbage to SQL calls *at run time*. More on this as it progresses.

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 acronymn for Indexed Sequential Access Method.

OpenCOBOL has fairly full support of all standard specified ISAM compile and runtime semantics.

For example

```
*>>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
    alter-
nate record key is splitkey of indexing-record
        with duplicates
    .

*> ** OpenCOBOL does not yet sup-
port 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 sepa-
rate 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 ex-
ample
 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

*> us-
ing read key and then next key / last key com-
pare
 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-prime-key
 perform read-previous-by-key
     until no-more-records

*> and with that we are done with index-
ing sample
 close indexing

 goback.
*> ****************************************************************

*><* Write paragraph
 write-indexing-record.
     write indexing-record
         invalid key
             display
                 "rewrite key: " key-
field 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 index-
ing 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 old-
key 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: " key-
field 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 or reference para-
graph
 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 para-
graph
 start-at-key.
     display "Seek-
ing >= " 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 or 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.
*><*
*><* 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.

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 it's 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?.

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.

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.

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 re-
ferred
to in the USING phrase of the PROCEDURE DIVI-
SION.
It also only has relevance when the pro-
gram is CALL'ed
from another prog.
This means that the addressabil-
ity of these items must
be programmed (usually with SET ADDRESS) be-
fore reference.

Per default, the item loses it's addressabil-
ity on exit
from the program. This option causes the mod-
ule to retain
the item's address between CALL invocations of the pro-
gram.
```

With some rumours that this may become the default in future releases of OpenCOBOL, and the *-fstatic-linkage* option may be deprecated.

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 Mes-
sage Queues                  */
/* Author: Brian Tif-
fin                                    */
/* Date:   Au-
gust, 2008                                  */
/* Build:  gcc -
c ocmq.c                                  */
/* Usage:  cobc -x -
lrt program.cob ocmq.o                 */

#include <fcntl.h>           /* For O_* con-
stants */
#include <sys/stat.h>        /* For mode con-
stants */
#include <errno.h>           /* Access to er-
ror values */
#include <mqueue.h>          /* The mes-
sage queues */
#include <signal.h>          /* for notifica-
tion */
#include <time.h>            /* for the timed ver-
sions */
#include <stdio.h>
#include <string.h>          /* For str-
error */

#include <lib-
cob.h>          /* for cob_resolve */

/* Forward declarations */
static void ocmq_handler(int, sig-
info_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 argu-
ments, permissions and attributes */
int MQCREATE(char *mq-
name, 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 MQSE-
TATTR(int mqid, char *mqattr, char *oldattr) {
    mqd_t mqres;

    errno = 0;
    mqres = mq_setattr((mqd_t)mqid, (struct mq_attr *)mqattr, (struct mq_att
dattr);
    return (int)mqres;
}

/* Send a message to the queue */
int MQSEND(int mqid, char *mes-
sage, int length, unsigned int mqprio) {
    mqd_t mqres;

    errno = 0;
    mqres = mq_send((mqd_t)mqid, mes-
```

```
sage, (size_t)length, mqprio);
    return (int)mqres;
}

/* Read the highest priority message */
int MQRECEIVE(int mqid, char *msgbuf, int bu-
flen, int *retprio) {
    ssize_t retlen;

    errno = 0;
    retlen = mq_receive((mqd_t)mqid, msg-
buf, buflen, retprio);
    return (int)retlen;
}

/* Timeout send */
int MQTIMEDSEND(int mqid, char *mes-
sage, int length, un-
signed int mqprio, int secs, long nanos) {

    mqd_t mqres;
    struct timespec mqtimer;
    struct timeval curtime;

    /* Expect sec-
onds and nanos to wait, not abso-
lute.  Add the OpenCOBOL values */
    gettimeofday(&curtime, NULL);
    mqtimer.tv_sec = cur-
time.tv_sec + (time_t)secs;
    mqtimer.tv_nsec = nanos;

    errno = 0;
    mqres = mq_timedsend((mqd_t)mqid, mes-
sage, (size_t)length, mqprio, &mqtimer);
    return (int)mqres;
}

/* Read the highest priority message */
int MQTIMEDRECEIVE(int mqid, char *msg-
buf, int buflen, int *ret-
prio, int secs, long nanos) {
    ssize_t retlen;
    struct timespec mqtimer;

    struct timeval curtime;

    /* Expect sec-
onds and nanos to wait, not abso-
lute.  Add the OpenCOBOL values */
```

```
    gettimeofday(&curtime, NULL);
    mqtimer.tv_sec = cur-
time.tv_sec + (time_t)secs;
    mqtimer.tv_nsec = nanos;

    errno = 0;
    retlen = mq_timedreceive((mqd_t)mqid, msg-
buf, 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 no-
tify signal - fill in a
     * sigaction struc-
ture and pass it to sigaction(). Because the
     * handler needs the siginfo struc-
ture 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, &ocsigac-
tion, NULL) == -1) {
        fprintf(stderr, "%s\n", "Error post-
ing sigaction");
        return -1;
    }

    /* Set up notifica-
tion: fill in a sigevent structure and pass it
     * to mq_notify(). The queue ID is passed as an ar-
gument to the
     * signal handler.
     */
    ocsigevent.sigev_signo      = SIGUSR1;
    oc-
sigevent.sigev_notify     = SIGEV_SIGNAL;
    oc-
sigevent.sigev_value.sival_int = (int)mqid;

    if (mq_notify((mqd_t)mqid, &oc-
sigevent) == -1) {
```

```
        fprintf(stderr, "%s\n", "Error post-
ing 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, sig-
info_t *pInfo, void *pSigContext) {
    struct sigevent ocnotify;
    mqd_t mqid;

    /* Get the ID of the mes-
sage queue out of the siginfo struc-
ture.     */
    mqid = (mqd_t) pInfo->si_value.sival_int;

    /* The MQPROCESSOR is a hardcoded Open-
COBOL 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 resolv-
ing MQPROCESSOR");
        return;
    }
```

```
    /* Request notification again; it re-
sets 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 post-
ing notify");
        return;
    }

    /* Call the cobol module with the mes-
sage 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 Open-
COBOL 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 protec-
tion/permission bits
       01 MQS-
IREAD            constant as 256.
       01 MQS-
IWRITE           constant as 128.

      * Need a better way of displaying new-
lines
       01 new-
line              pic x value x'0a'.

      * Message Queues re-
turn an ID, maps to int
       01 mqid                 usage binary-
long.
       01 mqres                usage binary-
long.
      * Queue names end up in an mqueue vir-
tual 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 'Open-
COBOL is awesome'.
       01 user-message         pic x(80).
       01 send-length          usage binary-
long.

       01 urgent-
message       pic x(20) value 'Urgent Open-
COBOL msg'.

      * Data members for ac-
cess 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 Mes-
sage Queue entry limit of 8K
       01 msgbuf               pic x(8192).
       01 msglen               usage binary-
long value 8192.
      * Priori-
ties range from 0 to 31 on many sys-
tems, can be more
       01 msgprio              usage binary-
long.
      * MQ at-
tributes.  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 MQCRE-
ATE and MQOPEN.
      * This example uses non block-
ing, non exclusive create
      *   read/write by owner and default at-
tributes
       compute
           mqopenflags = MQO-RDWR + MQO-
CREAT + MQO-NONBLOCK
       end-compute.
       compute
           mqpermissions = MQS-IREAD + MQS-
IWRITE
       end-compute.
```

```
      * Sam-
ple shows the two types of open, but only eval-
uates create
       if zero = zero
       call "MQCREATE" using mqname
                             by value mqopen-
flags
                             by value mqpermis-
sions
                             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 cre-
ate
       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 re-
moved on close
      * call "MQUNLINK" using mqname
      *                 returning mqres
      * end-call.
      * move "unlink" to operation.
      * perform show-error.

      * Use the command line argu-
ments 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 (prior-
ity 31)
       call "MQSEND" using by value mqid
                           by refer-
ence 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 mes-
sage (16)
       inspect urgent-message
           replacing leading "Urgent" by "Mid-
dle".
       call "MQSEND" using by value mqid
                           by refer-
ence 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 mes-
sage off queue
       call "MQRECEIVE" using by value mqid
                              by reference msg-
```

```
buf
                              by value msglen
                              by reference msg-
prio
                          returning mqres
       end-call.
       display
           newline "re-
cieve len: " mqres " prio: " msgprio
       end-display.
       if mqres > 0
           display
               "priority 31 message: " msg-
buf(1:mqres)
           end-display
       end-if.
       move "receive" to operation.
       perform show-error.

      * Pull the middling priority mes-
sage off queue
       call "MQRECEIVE" using by value mqid
                              by reference msg-
buf
                              by value msglen
                              by reference msg-
prio
                          returning mqres
       end-call.
       display
           newline "re-
cieve len: " mqres " prio: " msgprio
       end-display.
       if mqres > 0
           display
               "priority 16 message: " msg-
buf(1:mqres)
           end-display
       end-if.
       move "receive" to operation.
       perform show-error.

      * ** INTENTIONAL ERROR ms-
glen param too small **
      * Pull message off queue
       call "MQRECEIVE" using by value mqid
                              by reference msg-
buf
                              by value 1024
                              by reference msg-
```

```
prio
                        returning mqres
       end-call.
       display
           newline "re-
cieve 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 mes-
sage off queue, in blocking mode
       move MQO-NONBLOCK to mqflags of mqattr.
       call "MQSETATTR" using by value mqid
                              by refer-
ence mqattr
                              by reference ol-
dattr
                        returning mqres
       end-call
       move "setattr" to operation.
       perform show-error.
       perform show-attributes.

       call "MQRECEIVE" using by value mqid
                              by reference msg-
buf
                              by value msglen
                              by reference msg-
prio
                        returning mqres
       end-call.
       display
           newline "re-
cieve len: " mqres " prio: " msgprio
       end-display.
       if mqres > 0
           display
               "priority 1 message: " msg-
buf(1:mqres)
           end-display
       end-if.
       move "receive" to operation.
       perform show-error.
```

```
       perform varying loopy from 1 by 1
           until loopy > 5
               dis-
play "Sleeper call " loopy end-display
               call "CBL_OC_NANOSLEEP" us-
ing 50000000000
                                       return-
ing mqres
               end-call
       end-perform.

      * Close the queue.  As it is set un-
linked, 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 re-
moved on close
       call "MQUNLINK" using mqname
                       returning mqres
       end-call.
       move "unlink" to operation.
       perform show-error.

       goback.

      ******************************************************************
      * Information display of the Mes-
sage Queue attributes.
       show-attributes.
       call "MQGETATTR" using by value mqid
                              by refer-
ence 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  new-
line
           "max msg:  " mq-
maxmsg  of mqattr  newline
           "mqs size: " mqmsg-
```

```
size of mqattr  newline
           "cur msgs: " mqcurmsqs of mqattr
       end-display
       .

      * The C global errno error display para-
graph
       show-error.
           call "ERRORNUMBER" return-
ing mqres end-call
           if mqres > 0
               display
                   operation " errno: " mqres
               end-display
               call "ERRORSTRING" using errstr
                                        by value length errstr
                                   return-
ing mqres end-call
               if mqres > 0
                   display
                       "     str-
error: " errstr(1:mqres)
                   end-display
               end-if
           end-if
       .
       end program mqsample.

      ******************************************************************
      * Author:    Brian Tiffin
      * Date:      August 2008
      * Purpose:   Demonstration of Open-
COBOL 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 proce-
dure with " mqid end-display.
       perform
           with test after
           until mqres <= 0
      * Pull highest priority mes-
sage off queue
           call "MQRECEIVE" us-
ing by value mqid
                                    by refer-
ence msgbuf
                                    by value ms-
glen
                                    by refer-
ence msgprio
                               returning mqres
           end-call
           display
               "re-
cieve len: " mqres " prio: " msgprio
           end-display
           if mqres > 0
               display
                   "priority 31 message: " ms-
gbuf(1:mqres)
               end-display
           end-if
      *    move "receive" to operation
      *    perform show-error
       end-perform.

       goback.
       end program MQSIGNAL.
```

5.11 Can OpenCOBOL interface with Lua?

Yes. Lua can be embedded in OpenCOBOL applications.

```
*>>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 Gen-
eral Public License
*><*             | No warranty ex-
pressed or implied
*><* :Tectonics: | cobc -c -
I/usr/include/lua5.1/ oclua.c
*><*             | cobc -x -
llua5.1 luacaller.cob oclua.o
*><*             | ./ocdoc lua-
caller.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, Pontifi-
cal Catholic University
*><*             of Rio de Janeiro in Brazil.
*><*             http://www.lua.org/authors.html
*><* :Sources:   | http://opencobol.add1tocobol.com/luacaller.cob
*><*             | http://opencobol.add1tocobol.com/oclua.c
*><*             | http://opencobol.add1tocobol.com/oclua.lua
*><*             | http://opencobol.add1tocobol.com/oclua.rst
*><*             | http://opencobol.add1tocobol.com/ocfaq.rss
*><*
*> ****************************************************************
 identification division.
 program-id. luacaller.

 data division.
 working-storage section.
 01 luastate             usage pointer.
 01 lu-
ascript            pic x(10) value 'oclua.lua' & x"00".
 01 luacommand          pic x(64).
 01 luaresult           pic x(32).
 01 lualength           usage binary-long.

 01 items               pic 9 us-
age computational-5.
 01 luastack.
    03 luaitem         pic x(32) oc-
curs 5 times.
 01 depth               usage binary-long.
```

```
*> ****************************************************************
 procedure division.

 call "OCLUA_OPEN" returning luastate end-call

 move 'return "Open-
COBOL " .. 1.0 + 0.1' & x"00" to luacommand
 call "OCLUA_DOSTRING"
     using
         by value luastate
         by reference luacommand
         by reference luaresult
         by value function length(luaresult)
     returning depth
 end-call
 display
     "OpenCOBOL displays: " depth " |" luare-
sult "|"
 end-display

 call "OCLUA_DOFILE"
     using
         by value luastate
         by reference luascript
         by reference luaresult
         by value 32
     returning depth
 end-call
 display
     "OpenCOBOL displays: " depth " |" luare-
sult "|"
 end-display

 call "OCLUA_DOFILE"
     using
         by value luastate
         by reference luascript
         by reference luaresult
         by value 32
     returning depth
 end-call
 display
     "OpenCOBOL displays: " depth " |" luare-
sult "|"
 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 luas-
tate end-call

 goback.
 end program luacaller.
*> ****************************************************************
*><* ++++++++
*><* Overview
*><* ++++++++
*><* The OpenCOBOL Lua interface is de-
fined at a very high level.
```

```
*><*
*><* The objective is to provide easy ac-
cess to Lua through
*><* script files or strings to be evaluated.
*><*
*><* Com-
mand strings and script file names passed to Lua MUST be
*><* termi-
nated with a null byte, as per C Language con-
ventions.
*><*
*><* A Lua en-
gine is started with a call to OCLUA_OPEN, which
*><* returns an Open-
COBOL POINTER that is used to reference
*><* the Lua state for all further calls.
*><*
*><* A Lua en-
gine is run down with a call to OCLUA_CLOSE.
*><*
*><* .. Attention::
*><*   Calls to Lua with-
out a valid state will cause
*><*   undefined behaviour and crash the ap-
plication.
*><*
*><* Lua uses a stack and re-
sults of the Lua RETURN reserved
*><* word are placed on this stack.  Multi-
ple values can be
*><* returned from Lua.
*><*
*><* The developer is responsi-
ble for stack overflow conditions
*><* and the size of the stack (default 20 el-
ements) is
*><* controlled with OCLUA_STACK using an in-
teger that
*><* determines the numbers of slots to re-
serve.
*><*
*><* 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 luas-
tate END-CALL
*><*
*><* -----------
*><* OCLUA_STACK
*><* -----------
*><* Check and possibly re-
size the Lua data stack.  Returns 0 if
*><* Lua cannot expand the stack to the re-
quested 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 alphanu-
meric field as a Lua program
*><* producing any top of stack entry and re-
turning the depth of
*><* stack after evaluation.
*><*
*><* Takes a luastate, a null terminated com-
mand string,
*><* a result field and length and re-
turns an integer depth.
*><*
*><* .. Attention::
*><*   The Lua stack is NOT popped while re-
turning the top of stack entry.
*><*
```

```
*><* ::
*><*
*><*   01 luacommand pic x(64).
*><*   01 luaresult  pic x(32).
*><*   01 depth      usage binary-long.
*><*
*><*   move 'return "Open-
COBOL " .. 1.0 + 0.1' & x"00" to luacommand
*><*   call "OCLUA_DOSTRING"
*><*       using
*><*           by value luastate
*><*           by reference luacommand
*><*           by reference luaresult
*><*           by value func-
tion length(luaresult)
*><*       returning depth
*><*   end-call
*><*   display
*><*       "OpenCOBOL dis-
plays: " depth " |" luaresult "|"
*><*   end-display
*><*
*><* Outputs::
*><*
*><*   OpenCOBOL displays: +0000000001 |Open-
COBOL 1.1                   ||
*><*
*><* ------------
*><* OCLUA_DOFILE
*><* ------------
*><* Evaluate a script using a null termi-
nated alphanumeric field
*><* naming a Lua program source file, re-
trieving any top of
*><* stack entry and return-
ing the depth of stack after evaluation.
*><*
*><* Takes a luastate, a null terminated file-
name,
*><* a result field and length and re-
turns an integer depth.
*><*
*><* .. Attention::
*><*   The Lua stack is NOT popped while re-
turning the top of
*><*   stack entry.
*><*
*><* ::
*><*
*><*   01 lu-
```

```
ascript  pic x(10) value 'oclua.lua' & x"00".
*><*   01 luaresult  pic x(32).
*><*
*><*   call "OCLUA_DOFILE"
*><*       using
*><*           by value luastate
*><*           by reference luascript
*><*           by reference luaresult
*><*           by value func-
tion length(luaresult)
*><*       returning depth
*><*   end-call
*><*   display
*><*       "OpenCOBOL dis-
plays: " 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 dis-
plays: +0000000002 |Hello Open-
COBOL from Lua        ||
*><*
*><* and on re-
turn from Lua, there is *math.pi* and the
*><* Hello string remain-
ing on the Lua state stack.
*><*
*><* -----------
*><* OCLUA_DEPTH
*><* -----------
*><* Returns the current number of ele-
ments 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, re-
turning the length
*><* of the retrieved item.
*><*
*><* An example that populates and dis-
plays an OpenCOBOL table::
*><*
*><*   01 items                pic 9 us-
age computational-5.
*><*   01 luastack.
*><*      03 luaitem           pic x(32) oc-
curs 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 func-
tion length(luaresult)
*><*               returning lualength
*><*           end-call
*><*           move luaresult to lu-
aitem(items)
*><*   end-perform
*><*
*><*   perform varying items from 1 by 1
*><*       until items > depth
*><*           display
*><*              "Item " items ": " lu-
aitem(items)
*><*           end-display
*><*   end-perform
*><*
*><* Lua numbers the in-
dexes of stacked items from 1, first
*><* item to n, last item (cur-
rent top of stack). Negative
*><* indexes may also be used as docu-
mented 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 ele-
ments off of the Lua stack
*><* returning the depth of the stack af-
ter 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 unpre-
dictable and may well
*><*    lead to a SIGSEGV crash.
*><*
*><* ::
*><*
*><*   call "OCLUA_CLOSE" using by value luas-
tate end-call
*><*
```

The above code use 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 <lauxlib.h>
#include <lualib.h>

/* Open the Lua engine and load all the de-
fault 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, un-
signed char *cobol, int coblen) {
    int result;
    int stacked;
    const char *retstr;
    int retlen;

    memset(cobol, ' ', coblen);
    re-
sult = ((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) {
        /* popu-
late cobol field with top of stack */
        ret-
str = lua_tolstring(L, stacked, &retlen);
        memcpy(cobol, ret-
str, (coblen > retlen) ? retlen : coblen);
    }
    /* return number of items on the stack */
    return stacked;
    }
}

/* by filename */
int OCLUA_DOFILE(lua_State *L, const char *file-
name, unsigned char *cobol, int coblen) {
    return OCLUA_DO(L, 1, file-
name, cobol, coblen);
}
```

```
/* by string */
int OCLUA_DOSTRING(lua_State *L, const char *string, un-
signed char *cobol, int coblen) {
    re-
turn OCLUA_DO(L, 0, string, cobol, coblen);
}

/* retrieve stack item as string */
int OCLUA_GET(lua_State *L, int element, un-
signed char *cobol, int coblen) {
    const char *retstr;
    int retlen;

    /* popu-
late cobol field with top of stack */
    memset(cobol, ' ', coblen);
    retstr = lua_tolstring(L, ele-
ment, &retlen);
    if (retstr == NULL) {
    return -1;
    } else {
    memcpy(cobol, ret-
str, (coblen > retlen) ? retlen : coblen);
    return retlen;
    }
}

/* check the stack, resize if needed, re-
turns 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 uses a sample Lua script

```
-- Start
-- Script: oclua.lua
print("Lua prints hello")

hello = "Hello OpenCOBOL from Lua"
return math.pi, hello
-- End
```

5.12 Can OpenCOBOL use ECMAScript?

Yes. Using the SpiderMonkey engine. See Can OpenCOBOL use JavaScript?

5.13 Can OpenCOBOL use JavaScript?

Yes. A wrapper for the SpiderMonkey engine allows OpenCOBOL access to core JavaScript.

```
/* OpenCOBOL with embedded spidermon-
key 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 environ-
ment type */
#define XP_UNIX

#if (defined(XP_WIN) || de-
fined(XP_UNIX) || defined(XP_BEOS) || de-
fined(XP_OS2))
#include "jsapi.h"
#else
#error "Must de-
fine 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 val-
ues */
    if (rtsize == 0) { rtsize = 0x100000; }
    if (cxsize == 0) { cxsize = 0x1000; }

    /* Initialize a runtime space */
    rt = JS_NewRuntime(rtsize);
    if (rt == NULL) { re-
turn OCJS_ERROR_RUNTIME; }
    /* Attach a context */
    cx = JS_NewContext(rt, cxsize);
    if (cx == NULL) { re-
turn OCJS_ERROR_CONTEXT; }
    /* And a default global */
    global = JS_NewObject(cx, &global_class, NULL, NULL);
    if (global == NULL) { re-
turn OCJS_ERROR_GLOBAL; }
    /* Load standard classes */
    ok = JS_InitStandardClasses(cx, global);

    /* Return success or stan-
dard class load error */
    re-
turn (ok == JS_TRUE) ? 0 : OCJS_ERROR_STANDARD;
}
```

```
/* Evaluate script */
int ocjsEvaluate(char *script, char *re-
sult, int length) {
    jsval rval;
    JSString *str;
    int reslen = OCJS_ERROR_EVALUATE;

    JSBool ok;

    /* filename and line number, not re-
ported */
    char *filename = NULL;
    int lineno = 0;

    /* clear the result field */
    memset(result, ' ', length);

    /* Evaluate javascript */
    ok = JS_EvaluateScript(cx, global, script, strlen(script), file-
name, 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; }
        /* con-
vert down to char and move to OpenCOBOl re-
sult field */
        mem-
cpy(result, JS_GetStringBytes(str), reslen);
    }
    return reslen;
}

/* Evaluate script from file */
int ocjsFromFile(char *filename, char *re-
sult, int length) {
    FILE *fin;
    int bufsize = 10240;
    char inbuf[bufsize];
    int reslen;

    fin = fopen(filename, "r");
    if (fin == NULL) { re-
turn OCJS_ERROR_EVALUATE; }
    //while (fread(inbuf, sizeof(char), buf-
size, fin) > 0) {
    if (fread(inbuf, 1, bufsize, fin) > 0) {
        reslen = ocjsEvaluate(inbuf, re-
```

```
sult, 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, re-
lease engine */
int ocjsString(char *script, char *re-
sult, int length) {
    int reslen;

    reslen = ocjsInitialize(0, 0);
    if (reslen < 0) { return reslen; }
    reslen = ocjsEvaluate(script, re-
sult, length);
    ocjsRunDown();
    return reslen;
}
/**/
```

A sample OpenCOBOL application:

```
*>>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, re-
turn 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 recieved    : " re-
sult 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 offest) dol-
lar 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: " func-
tion 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 recieved    : " re-
sult 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: " func-
tion 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 recieved    : " re-
sult newline end-display

*><* erroneous script
 move spaces to javascript
 string
     'an error of some kind;' delim-
```

```
ited by size
     x"00" delimited by size
     into javascript
 end-string

 display
     "Script: " func-
tion 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 recieved    : " re-
sult 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: " func-
tion 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 recieved    : " re-
sult newline end-display

*><* Rundown the js engine
 call "ocjsRunDown" returning result

*><* take first name last name, re-
turn last "," first
 move spaces to javascript
 string
     "re = /(\w+)\s(\w+)/; " delimited by size
     'str = "John Smith"; ' delimited by size
     'new-
str = str.replace(re, "$2, $1"); ' delim-
ited by size
     "newstr;" delimited by size
     x"00" delimited by size
     into javascript
 end-string

 display
     "Script: " func-
tion 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 recieved    : " re-
sult newline end-display

*><* split a string using numbers return ar-
ray (as js string form)
 move spaces to javascript
 string
     'myString = "Hello 1 word. Sentence num-
ber 2."; '
         delimited by size
     'splits = myString.split(/(\d)/); ' de-
limited by size
     'splits;'  delimited by size
     x"00" delimited by size
```

```
     into javascript
 end-string

 display
     "Script: " func-
tion 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 recieved    : " re-
sult newline end-display

*><* Get javascript date
 move "new Date()" & x"00" to javascript

 display
     "Script: " func-
tion 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 recieved    : " re-
sult end-display

 goback.
 end program jscaller.
```

And with a sample script:

Attention!

Need something for ocjsscript.js in the public domain that is only Core js

Sample output:

```
js> 123 * 456 + 789
OpenCOBOL result-field: 56877
OpenCOBOL recieved    : +000000005

Script: pat = /\$\d+\.\d+\s\d+/; a = "----+---
-1----+-$56.78 90----3----+----
4"; a.search(pat);
OpenCOBOL result-field: 16
OpenCOBOL recieved    : +000000002

Script: a;
OpenCOBOL result-field: ----+----1----+-
$56.78 90----3----+----4
OpenCOBOL recieved    : +000000040

Script: an error of some kind;
 *** script problem ***
OpenCOBOL result-field:
OpenCOBOL recieved    : -000000005

Script: re = /(\w+)\s(\w+)/; str = "John Smith"; new-
str = str.replace(re, "$2, $1"); newstr;
OpenCOBOL result-field: Smith, John
OpenCOBOL recieved    : +000000011

Script: myString = "Hello 1 word. Sen-
tence number 2."; splits = myS-
tring.split(/(\d)/); splits;
OpenCOBOL result-
field: Hello ,1, word. Sentence number ,2,.
OpenCOBOL recieved    : +000000036

Script: new Date()
OpenCOBOL result-
field: Mon Sep 15 2008 04:16:06 GMT-0400 (EDT)
OpenCOBOL recieved    : +000000039
```

5.14 Can OpenCOBOL interface with Scheme?

Yes, directly embedded with Guile and libguile.

callguile.cob

```
*>>SOURCE FORMAT IS FIXED
*> ************************************************************
*> Author:    Brian Tiffin
*> Date:      20090215
*> Purpose:   Demonstrate libguile Scheme in-
teractions
```

```
*> Tectonics: cobc -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 subto-
tal             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" us-
ing "script.scm" & x"00" end-call
 display "OC:" end-display

 display "OC: evaluate one of the de-
fined 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 calcula-
tion" 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 inte-
ger 10" end-display
 call "scm_from_int32"
     using by value size is 4 10 return-
ing radix-scm
 end-call

 display "OC: have guile convert num-
ber, 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 nu-
meric 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: " an-
swer 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 pack-
ages, charge the minimum
  (if (and (< weight 20) (< length 5))
      0.95

      ;; Otherwise for long pack-
ages, 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.

5.15 Can OpenCOBOL interface with Tcl/Tk?

Yes. OpenCOBOL supports the Tcl/Tk embedding engine developed by Rildo Pragna as part of the TinyCOBOL project. We have been given permission by Rildo to embed his engine in OpenCOBOL.

5.16 Can OpenCOBOL interface with Falcon PL?

Not yet, but work with Giancarlo to allow embedding of Falcon scripts is in progress.

FalconPL has some nice features.

```
saying = List("Have", "a", "nice", "day")

for elem in saying
   >> elem
   formiddle: >> " "
   forlast: > "!"
end
```

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) re-
turn 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, Inter-
faces.C;
```

```
use  Ada.Text_IO, Ada.Integer_Text_IO, Inter-
faces.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 Open-
COBOL");
   Put("OpenCOBOL passed: ");
   Put(value);
   New_Line;
   New_Line;
end ingress;

function echo(message : in char_array) re-
turn integer is
begin
    Put(To_Ada(message, true));
    return To_Ada(message, true)'length;
end echo;

end HelloAda;
```

The adacaller.cob source file

```
****************** ada-
caller.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
```

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 enviroment.

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

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.

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.

5.19.2 Keyboard control

This sample only show S-Lang terminal input. A very sophisticated terminal output control interface is also available.

```
*>>SOURCE FORMAT IS FIXED
*> ****************************************************************
*> Author:    Brian Tiffin
*> Date:      20090503
*> Purpose:   Experimental S-Lang interface
*> Tectonics: cobc -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 prior-
ity (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 spe-
cial mode" x"0d" end-display
 end-if

*> install an exit handler to put termi-
nal 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 proce-
dure" end-display
 end-if

*> Not sure?  Have SLang han-
dle ^C or let OpenCOBOL take over?
 call "SLang_set_abort_signal" us-
ing by value 0 end-call

*> The demo.  Fetch a key, then fetch a key-
code.  4 times.
*>   SLang terminals display newline as new-
line.  Need explicit
*>   CR to get a carriage re-
turn.  Hence the x"0d".
*>   Plus, output is buffered until line ter-
minators.
 display
     "Tap a normal key, then tap a 'spe-
cial' key, ie F1, 4 times"
     x"0d"
 end-display
 perform 4 times
     call "SLang_getkey" returning thekey end-
call
     display thekey space with no advanc-
ing end-display
     call "SLkp_getkey" returning thekm end-
call
     display thekm x"0d" end-display
 end-perform

*> Exit handler will take care of reset-
ting terminal
 goback.


*> ****************************************************************
*> Exit procedure to ensure terminal prop-
erly 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 'spe-
cial' 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.

5.19.3 Scripting

S-Lang also provides a very comprehensive scripting language, which is very easy to embed.

```
*>>SOURCE FORMAT IS FIXED
*> ****************************************************************
*> Author:    Brian Tiffin
*> Date:      20090505
*> Purpose:   Experimental S-Lang interface
*> Tectonics: cobc -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 initializ-
ing 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 intrin-
sic" 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 af-
ter COBOL add = %d\n", slint);' & x"00"
     returning result
 end-call
 if result equal -1
     display "error with printf af-
ter cobol add" end-display
 end-if

*> Let's get out of here and do the Dil-
bert 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
```

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>

    <Key-
words>^(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>
    <Key-
words>communication|report|screen|</Keywords>
    <Keywords>section|declaratives|</Keywords>
    <Keywords>end|</Keywords>
    <Keywords>perform|end-
perform|until|times|varying|</Keywords>
    <Key-
words>add|subtract|multiply|divide|compute|</Keywords>
    <Keywords>end-add|end-subtract|end-
multiply|end-divide|end-compute|</Keywords>
    <Key-
words>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>
      <Con-
stant_Character>&apos;</Constant_Character>
      <Can_Indent>True</Can_Indent>
      <Syn-
tax_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; ***************************************************************
```

```
      *&amp;gt; Author:    Brian Tiffin
      *&amp;gt; Date:      %D
      *&amp;gt; Purpose:   %_
      *&amp;gt; Tectonics: make
      *&amp;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.
      *&amp;gt; ****************************************************************

       end program %(pid).
    </text>
  </alias>

  <Language>
    <Name>Vala</Name>
    <Spec_Suffix>.vala</Spec_Suffix>

    <Key-
words>^(bool|char|constpointer|double|float|size_t|ssize_t|string|unichar|vo
    <Key-
words>int|int8|int16|int32|int64|long|short|</Keywords>
    <Key-
words>uint|uint8|uint16|uint32|uint64|ulong|ushort|</Keywords>
    <Key-
words>class|delegate|enum|errordomain|interface|namespace|struct|</Keywords>
    <Key-
words>break|continue|do|for|foreach|return|while|</Keywords>
```

```
    <Keywords>else|if|switch|</Keywords>
    <Keywords>case|default|</Keywords>
    <Key-
words>abstract|const|dynamic|ensures|extern|inline|internal|override|</Keywo
    <Key-
words>private|protected|public|requires|signal|static|virtual|volatile|weak|
    <Keywords>false|null|true|</Keywords>
    <Key-
words>try|catch|finally|throw|</Keywords>
    <Key-
words>as|base|construct|delete|get|in|is|lock|new|out|params|ref|</Keywords>
    <Key-
words>sizeof|set|this|throws|typeof|using|value|var|yield|yields)\b</Keyword

    <Context>
      <New_Line_Comment_Start>//</New_Line_Comment_Start>
      <Comment_Start>/*</Comment_Start>
      <Comment_End>*/</Comment_End>
      <String_Delimiter>&quot;</String_Delimiter>
      <Con-
stant_Character>&apos;</Constant_Character>
      <Can_Indent>True</Can_Indent>
      <Syn-
tax_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" in-
dex="opencobol">
    <language>OpenCOBOL</language>
    <initial-cmd-line>-m</initial-cmd-line>
      <switches lines="3" columns="2">
         <title line="1" column="1" >Code gen-
eration</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" >Debug-
ging</title>
         <title line="3" column="2" >Syn-
tax</title>

         <radio>
            <radio-entry label="Build dy-
namic 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 la-
bel="Translation only, COBOL to C" switch="-
C" />
            <radio-entry la-
bel="Compile only, output assem-
bly file" switch="-S" />
         </radio>
         <check label="Syntax check-
ing only" switch="-fsyntax-only"
                tip="Syntax error check-
ing only; no output emitted" />

         <combo label="Optimization" switch="-
O" nodigit="1" noswitch="0"
                tip="Controls the optimiza-
tion level">
            <combo-
entry label="No optimization" value="0" />
            <combo-
entry label="Simple optimization" value="1" />
            <combo-entry label="Some more op-
timization" value="s" />
            <combo-
entry label="Full optimization" value="2" />
         </combo>

         <field label="Generate List-
ing to " switch="-t" separator=" " as-
file="true"
                tip="Generate a list-
ing file to given filename" />
         <field label="Save Gener-
ated files to " switch="-save-temps" separa-
tor="=" as-directory="true"
                tip="Save tempo-
rary files to given directory" />
```

```
         <radio line="2" column="1">
            <radio-
entry label="Format FIXED" switch="-fixed"
                tip="Standards mandate de-
fault is fixed format source code" />
            <radio-entry la-
bel="Format FREE (FIXED is default)" switch="-
free"
                tip="Assume free for-
mat source code" />
         </radio>
         <check label="MF com-
ment (may lead to ambiguous source)" switch="-
fmfcomment" line="2" column="1"
                tip="Allow * or / in col-
umn 1 as FIXED format line comment" />
         <check label="FUNCTION im-
plied" switch="-ffunctions-all" line="2" col-
umn="1"
                tip="Allow use of intrin-
sic functions without FUNCTION keyword" />
         <check la-
bel="Fold Copy LOWER" switch="-ffold-copy-
lower" line="2" column="1"
                tip="Fold COPY sub-
ject to lower case" />
         <check label="Fold Copy UP-
PER" switch="-ffold-copy-upper" line="2" col-
umn="1"
                tip="Fold COPY subject to up-
per case" />
         <check label="Full Warn-
ings" switch="-W" line="2" column="1"
                tip="ALL possible warnings" />
         <popup label="Warnings" line="2" col-
umn="1">
            <check label="All (excep-
tions listed below)" switch="-Wall" />
            <check label="Obsolete" switch="-
Wobsolete"
                   tip="Warn if obsolete fea-
tures used" />
            <check label="Archaic" switch="-
Warchaic"
                   tip="Warn if archaic fea-
tures used" />
            <check la-
bel="Redefinition" switch="-Wredefinition"
                   tip="Warn of incompati-
```

```
ble redefinition of data items" />
            <check label="Constant" switch="-
Wconstant"
                    tip="Warn of inconsis-
tent constant" />
            <check la-
bel="Parentheses" switch="-Wparentheses"
                    tip="Warn of lack of paren-
theses around AND within OR" />
            <check label="Strict typ-
ing" switch="-Wstrict-typing"
                    tip="Warn of type mis-
match, strictly" />
            <check label="Implicit de-
fine" switch="-Wimplicit-define"
                    tip="Warn of implicitly de-
fined data items" />
            <check la-
bel="Call params (Not set for All)" switch="-
Wcall-params"
                    tip="Warn of non 01/77 items for CALL" />
            <check label="Column over-
flow (Not set for All)" switch="-Wcolumn-
overflow"
                    tip="Warn for FIXED for-
mat text past column 72" />
            <check la-
bel="Terminator (Not set for All)" switch="-
Wterminator"
                    tip="Warn when miss-
ing scope terminator (END-xxx)" />
            <check la-
bel="Truncate (Not set for All)" switch="-
Wtruncate"
                    tip="Warn of possi-
ble field truncation" />
            <check la-
bel="Linkage (Not set for All)" switch="-
Wlinkage"
                    tip="Warn of dangling LINK-
AGE items" />
            <check la-
bel="Unreachable (Not set for All)" switch="-
Wunreachable"
                    tip="Warn of unreach-
able statements" />
        </popup>

        <check label="Internal run-
time error checks" switch="-debug" column="2"
```

```
                tip="generate extra inter-
nal tests" />
         <check label="Implicit initial-
ize" switch="-fimplicit-init" column="2"
                tip="Do automatic initializa-
tion of the Cobol runtime system" />
         <check label="No trunca-
tion" switch="-fnotrunc" column="2"
                tip="Do not truncate bi-
nary fields according to PICTURE" />
         <check label="Sign ASCII" switch="-
fsign-ascii" column="2"
                tip="Numeric dis-
play sign ASCII (Default on ASCII ma-
chines)" />
         <check label="Sign EBCDIC" switch="-
fsign-ebcdic" column="2"
                tip="Numeric dis-
play sign EBCDIC (Default on EBCDIC ma-
chines)" />
         <check label="Stack checking for PER-
FORM" switch="-fstack-check" column="2"
                tip="Generate code to ver-
ify that you do not go beyond the bound-
ary of the stack" />
         <check label="Pass ex-
tra NULL" switch="-fnull-param" column="2"
                tip="Pass extra NULL terminat-
ing pointers on CALL statements" />

         <check label="Enable Debug-
ging lines" switch="-fdebugging-
line" line="2" column="2"
                tip="Enable col-
umn 7 D (FIXED FORMAT) de-
bug lines and &gt;&gt;D inline compiler direc-
tive" />
         <check label="Object Debug Informa-
tion" switch="-g" line="2" column="2"
                tip="Link level debug informa-
tion" />
         <check label="Trace (SEC-
TION/PARAGRAPH)" switch="-
ftrace" line="2" column="2"
                tip="Enable out-
put of trace statements for SECTION and PARA-
GRAPH" />
         <check label="Trace all (SEC-
TION/PARAGRAPH/STATEMENT)" switch="-
ftraceall" line="2" column="2"
```

```
                tip="Enable out-
put of trace statements for SECTION, PARA-
GRAPH and STATEMENTS" />
         <check label="Source loca-
tions" switch="-fsource-
location" line="2" column="2"
                tip="Generate source loca-
tion code (Turned on by -debug or -g)" />

         <check label="COBOL2002" switch="-
std=cobol2002" line="3" column="2"
                tip="Override the com-
piler's default, and config-
ure for COBOL 2002" />
         <check label="COBOL 85" switch="-
std=cobol85" line="3" column="2"
                tip="Override the com-
piler's default, and config-
ure for COBOL 85" />
         <check label="Micro Focus" switch="-
std=mf" line="3" column="2"
                tip="Override the com-
piler's default, and Micro Focus compatibil-
ity" />
      </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 ar-
guments" "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 ar-
guments" "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>
```

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 proceeded by a DISPLAY. The extra DISPLAY won't hurt, but is not necessary.

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.

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.

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 posting by Warren announcing release 0.95:

```
CobCurses is a package designed to allow Open-
Cobol
programmers to create screens on open sys-
tem platforms,
or those (like Windows) that can use PD-
Curses. Since
handcrafting screens is te-
dious work, this package
includes a "Screen Designer" utility.

All User Guides and Programmer Guide documen-
tation can
be found on the source forge (see link at bot-
tom).

==== RELEASE NOTES ====

A large number of internal changes were imple-
mented in
this re-
lease, but first let's cover the user visible
improvements:

1. MENUS! Popup menus are now sup-
ported, and are available
 in sdesign with every Ac-
tion field. In fact, any sdesign
 field that is marked with a dia-
mond graphic, has the
 ability to popup a menu with F1 (or ^O).

2. To support menus, FUNC-
TION 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 back-
ground. It is now
 possible using sdesign to draw alternate-
charset
 graphics in your screen back-
ground. See the notes in
 the opening help screen for the "Paint" func-
tion.

4. TRACE facilities. CobCurses now includes an
 environment variable that can enable captur-
ing of
```

```
 trace information to a file for debug-
ging. A routine
 named NC_TRACE_MSG can also be used to add cus-
tom
 messages to the trace file.

INTERNAL CHANGES:

The main two major internal changes were:

1. The terminal support has been virtual-
ized, so that
 the CobCurses routines deal with a "terminal"
 object (not curses routines). This will even-
tually
 lead to other possible windowing inter-
faces like
 perhaps graphic X Window or native Win-
dows support.

 The other motivation for this was to al-
low CobCurses
 to have one consistent set of con-
stants for colours,
 attributes and character sets. Previ-
ously, these
 values were different depend-
ing upon the platform
 and implementation of curses used.

2. Menu support has been provided indepen-
dently of curses.
 This is important for portability since PD-
Curses and
 some platforms do not pro-
vide a curses menu library.
 This also guaran-
tees that CobCurses menus will behave
 consistently on all platforms (and over-
come menu paging
 bugs in ncurses).

PLANNED FOR THE NEXT RELEASE:

Please avoid writ-
ing much code that works with colour pairs.
In the next re-
lease, it is planned to hide the colour pair
value altogether by using a TDC (Termi-
nal Drawing Context).
```

```
This TDC will tie together at-
tributes and colours, and
perhaps other "drawing con-
texts" so that you won't have to
manage colour pairs (this will be transpar-
ent). This will
also pave the way for graphical inter-
faces where a selected
font and line styles etc. may also be sup-
ported.

NOTES:

HPUX users will need to link with ncurses,
instead of the native HPUX curses li-
braries. I didn't
have time to fully investi-
gate this, but the native
include files de-
fine 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
```

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 2009)* 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
```

Example **truncated** to 72 and using the ocdoc.cob OpenCOBOL program for source code:

```
$ cobc -save-temps ocdoc.cob
$ cobxref ocdoc.i -L
$ cut -c1-72 ocdoc.lst
```

```
ACS Cobol Cross Refer-
ence Xref v0.95.27 (04/01/2009@11:27) Dictio-
nary Fi

Symbols of Module: ocdoc (ocdoc)
--------------------------------

Data Section (FILE)               Defn    Lo-
cations
---------------------------------+------------
--------------------------

doc-
output                          000124F  000252 000499
doc-
record                          000125F  000269 000381 000387 000390 00
                                         000478 000482 000485
source-
input                        000122F  000251 000287 000458 000500
source-
record                       000123F  000285 000288 000300 000316 00
                                         000324 000355 000456 000459
standard-
input                       000117F  000256 000282 000453 000497
standard-
output                      000119F  000257 000496
stdin-
record                        000118F  000283 000285 000454 000456
stdout-
record                       000120F  000387 000388 000475 000476

ACS Cobol Cross Refer-
ence Xref v0.95.27 (04/01/2009@11:27) Dictio-
nary Fi

Symbols of Module: ocdoc (ocdoc)
--------------------------------

Data Section (WORKING-
STORAGE)    Defn    Locations
---------------------------------+------------
--------------------------

arguments                           000128W  000219 000221 000244 000245
autoappend                          000187W  000380
autodoc                             000186W  000385
buffer-
empty                        000178W  000267 000380 000398 000472
buffer-flag                         000177W
buffer-
```

```
offset                    000176W  000268 000382 000399 000433 00
buffered-
output                  000179W  000385 000441 000471
counter                          000181W  000369 000410 000412 000416
data-field1                      000193W
data-field2                      000194W
data-field3                      000197W
data-record                      000192W
data-subfield1                   000195W
data-
subfield2                  000196W  000218
doc-
buffer                      000175W  000417 000419 000430
doc-
name                        000130W  000246 000505 000522 000532
filter-flag                      000138W
filtering                        000139W  000254 000281 000386 000452 00
first-
part                       000184W  000368
helping                          000137W  000222
here-
data                        000169W  000355
here-
record                      000167W  000356
heredoc                          000156W  000315 000337 000354
hereend                          000153W  000340 000353
hereflag                         000155W
herenone                         000157W  000341
herestart                        000152W  000336 000353
len-of-
comment                   000182W  000411 000415 000416
line-
count                       000141W  000270 000301 000435
line-
display                     000142W  000435 000438
result                           000190W  000548 000551 000552
result-
name                      000131W  000247 000518 000524 000534
rst-
command                      000189W  000517 000525 000535 000542 00
seq-
data                         000173W  000317
seq-
record                       000171W  000318
skipseqnum                       000135W  000314
source-
name                      000129W  000246 000504
special                          000185W  000379
style-
name                        000132W  000247 000519 000530
```

```
trimmed                          000151W  000316 000321 000324 000356 00
usagehelp                        000136W  000221
verbose                          000134W  000392 000480 000503 000539
verbosity                        000133W  000248

ACS Cobol Cross Refer-
ence Xref v0.95.27 (04/01/2009@11:27) Dictio-
nary Fi

Variable Tested [S]            Symbol (88-
Conditions)
-----------------------------------------------
----------------

buffer-flag                    buffer-empty
buffer-flag                    buffered-output
filter-flag                    filtering
first-part                     special
first-part                     autodoc
first-part                     autoappend
hereflag                       heredoc
hereflag                       herenone
trimmed                        herestart
trimmed                        hereend
usagehelp                      helping
verbosity                      verbose
verbosity                      skipseqnum

ACS Cobol Cross Refer-
ence Xref v0.95.27 (04/01/2009@11:27) Dictio-
nary Fi

Variable Tested                Symbol (88-
Conditions) [S]
-----------------------------------------------
----------------

first-part                     autoappend
first-part                     autodoc
buffer-flag                    buffer-empty
buffer-flag                    buffered-output
filter-flag                    filtering
usagehelp                      helping
hereflag                       heredoc
trimmed                        hereend
hereflag                       herenone
trimmed                        herestart
verbosity                      skipseqnum
first-part                     special
verbosity                      verbose
```

```
ACS Cobol Cross Refer-
ence Xref v0.95.27 (04/01/2009@11:27) Dictio-
nary Fi

Procedure                          Defn    Lo-
cations
-----------------------------------+------------
-------------------------

trim                               000324P  000394 000430 000482 000504 00

ACS Cobol Cross Refer-
ence Xref v0.95.27 (04/01/2009@11:27) Dictio-
nary Fi

Unreferenced Working Storage Symbols

buffer-flag                        000177W
data-field1                        000193W
data-field2                        000194W
data-field3                        000197W
data-record                        000192W
data-subfield1                     000195W
filter-flag                        000138W
hereflag                           000155W

ACS Cobol Cross Refer-
ence Xref v0.95.27 (04/01/2009@11:27) Dictio-
nary Fi

Unreferenced Procedures

None
```

CobXRef produces 132 column output by default and the commands used here limit the width to 72 characters in order to fit the FAQ file.

5.26 Does OpenCOBOL implement Report Writer?

Not at this time. *July, 2008*

But it does support LINAGE. See Does OpenCOBOL implement LINAGE?

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.

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.

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. **-savetemps**[**=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 Founda-
tion, Inc.
License GPLv3+: GNU GPL ver-
sion 3 or later <http://gnu.org/licenses/gpl.html>
This is free soft-
ware: you are free to change and redis-
tribute it.
There is NO WARRANTY, to the extent permit-
ted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu"...
Using host libthread_db li-
brary "/lib/i686/cmov/libthread_db.so.1".
(gdb) break 106
Breakpoint 1 at 0xOB-
FUSCA: file hello.c, line 106.
(gdb) break 109
Breakpoint 2 at 0xTETH-
ESY: file hello.c, line 109.
(gdb) run
Starting pro-
gram: /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 SEC-
TION", "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.*

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

```
D    IF 1 = 1
D       DISPLAY "Debug Line"  END-DISPLAY
D    ELSE
        DISPLAY "Normal Line" END-DISPLAY
D    END-IF
```

For using the environment Just define

```
01  debugmode pic x.
    88  debugmode-on values 'O', 'Y', 'J', 'o', 'y', 'j', '1'.
```

put an

```
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

```
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

```
D 01 debugmode pic x.
D    88 debugmode-
on values 'O', 'Y', 'J', 'o', 'y', 'j', '1'.

...

D    accept debugmode from Environment "DEBUG-
MODE"
D    end-accept

...
```

```
D    IF debugmode-on
D       DISPLAY "Debug Line"  END-DISPLAY
D    ELSE
        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.

5.30 Is there a C interface to OpenCOBOL?

Most definitely. See http://www.opencobol.org/modules/bwiki/index.php?cmd=read&page=Use
for details.

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

```
* 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

```
* Change nulls to spaces
 INSPECT current-url
     REPLACING ALL X"00" WITH SPACE.
```

Or there is also modified references in OpenCOBOL

```
* Assume IND is the first trail-
ing 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).
```

[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 it’s nature, passes all arguments by reference. That can be overridden with the **BY VALUE** clause and the **BY CONTENT** clause.

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.

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.

5.34 What is PI in OpenCOBOL?

With OpenCOBOL 1.1

```
DISPLAY FUNCTION PI.
3.14159265358979323846264338327950 29

DISPLAY FUNCTION E.
2.71828182845904523536028747135266 25
```

Thats 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.

5.35 Does OpenCOBOL support the Object features of the 2002 standard?

Not yet. *July 2008*

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.

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.

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.

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 be-
tween the words THIS and STRING, as
      -"continuation lines always fill to col-
umn 72.".
       procedure division.
       display longstr.
       display length longstr.
       display func-
tion length(function trim(longstr trailing)).
       display otherstr(1:72).
       display otherstr(73:75).
       display length otherstr.
       display func-
tion length(function trim(otherstr trailing)).
       goback.
```

Compiled with:

```
$ cobc longcont.cob
$ cobcrun longcont
```

produces:

```
This will all be one string in FIXED for-
mat source code
80
00000055
this                           string will have spaces be-
tween the words
THIS and STRING, as   continuation lines al-
ways 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.

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 for-
mat source code
$ cobc -free longstr.cob
$ cobcrun longstr
This will all be one string in both FIXED and FREE for-
mat source code
```

And for an Intrinsic FUNCTION unique to OpenCOBOL, see FUNCTION CONCATENATE.

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

```
If you put a D in column 7 OC han-
dles this as a comment. These lines are
only compiled if you run cobc with -
fdebugging-line.

By using this you can put some test mes-
sages 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?

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' ambigu-
ous; 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.

5.43 What is the shortest OpenCOBOL program?

All that is needed is a program-id. Doesn't do much.

```
program-id. a.
```

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 forums.

Please note: This is **not good** COBOL form, and is only shown as an example of the possibilities.

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.

Attention!

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.

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 -
                FUNC-
TION 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 forum post.

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 LEAD-
ING 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 LEAD-
ING 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.

5.48 Is there a way to detemermine when OpenCOBOL is running ASCII or EBCDIC?

OpenCOBOL supports both ASCII and EBCDIC character encodings. A simple test such as

```
01  MYSPACE   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.

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.

5.50 Does OpenCOBOL support recursion?

Yes. Not completely to standard currently *(February 2009)*, as there are no restrictions on calling programs in a recursive manner, but yes.

A made up example using a factorial called program

```
*> ** *> ****************************************************************
      *> Author:    Brian Tiffin
      *> Date:      29-Dec-2008
      *> Purpose:   Horsing around with recur-
sion
      *> 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 re-
turning 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 re-
turning 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: +000000001
ret: +000000001
ret: +000000002
ret: +000000006
ret: +000000024
4! = +0000000024
```

Of course the *Intrinsic FUNCTION FACTORIAL* might be a more efficient and much easier way at getting factorials.

5.51 Does OpenCOBOL capture arithmetic overflow?

Yes. Here is one sample using *ADD* with *ON SIZE ERROR*.

```
*> ** *> ***************************************************************
      *> Author:    Brian Tiffin
      *> Date:      04-Feb-2009
      *> Purpose:   Factorial and overflow
      *> Tectonics: cobc -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 facto-
rial(fact) to zero giving answer
                   on size error
                       display
                           "over-
flow at: " fact " is " answer
                           " with-
out test " function factorial(fact)
                       end-display
                   not on size error
                       display fact ": " an-
swer end-display
               end-add
       end-perform
       .

       00-leave.
       goback.

       end program overflowing.
      *> ****************************************************************
```

which outputs:

```
+0000000001: +0000000000000000001
+0000000002: +0000000000000000002
+0000000003: +0000000000000000006
+0000000004: +0000000000000000024
+0000000005: +0000000000000000120
+0000000006: +0000000000000000720
+0000000007: +0000000000000005040
+0000000008: +0000000000000040320
+0000000009: +0000000000000362880
+0000000010: +0000000000003628800
+0000000011: +0000000000039916800
+0000000012: +0000000000479001600
+0000000013: +0000000006227020800
+0000000014: +0000000087178291200
+0000000015: +0000001307674368000
+0000000016: +0000020922789888000
+0000000017: +0000355687428096000
+0000000018: +0006402373705728000
+0000000019: +0121645100408832000
overflow at: +0000000020 is +0121645100408832000 with-
out test 432902008176640000
```

```
overflow at: +0000000021 is +00121645100408832000 with-
out test 197454024290336768
```

5.52 Can OpenCOBOL be used for plotting?

Yes? One way is with an external call to *gnuplot*.

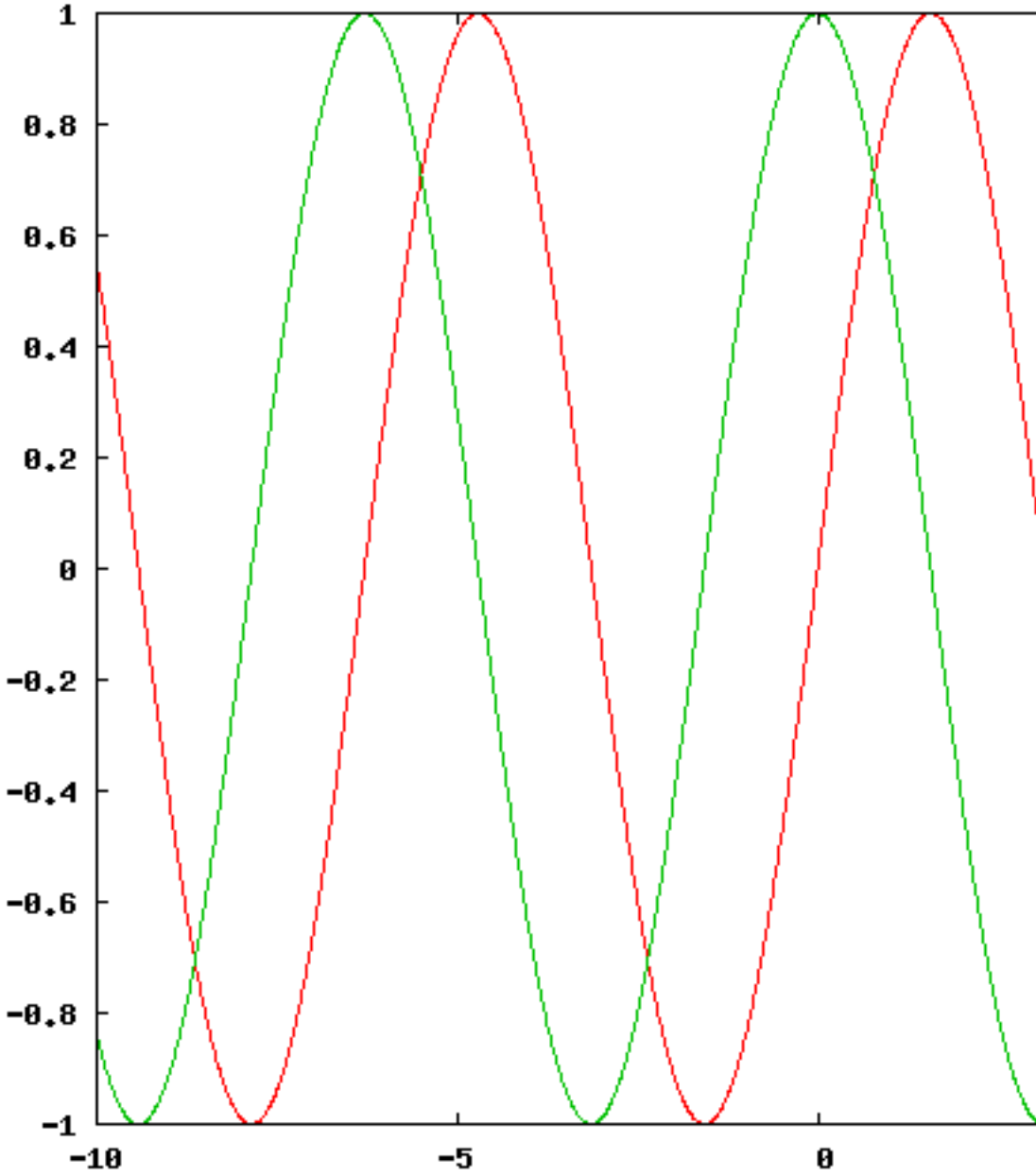
```
COBOL  >>SOURCE FORMAT IS FIXED
      ************************************************************
      * Author:    Brian Tiffin
      * Date:      29-July-2008
      * Purpose:   Plot trig and a random in-
come/expense/worth report
      * Tectonics: requires access to gnu-
plot. http://www.gnuplot.info
      *            cobc -Wall -x plotworth.cob
      *  OVERWRITES ocgenplot.gp ocgp-
data.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).
```

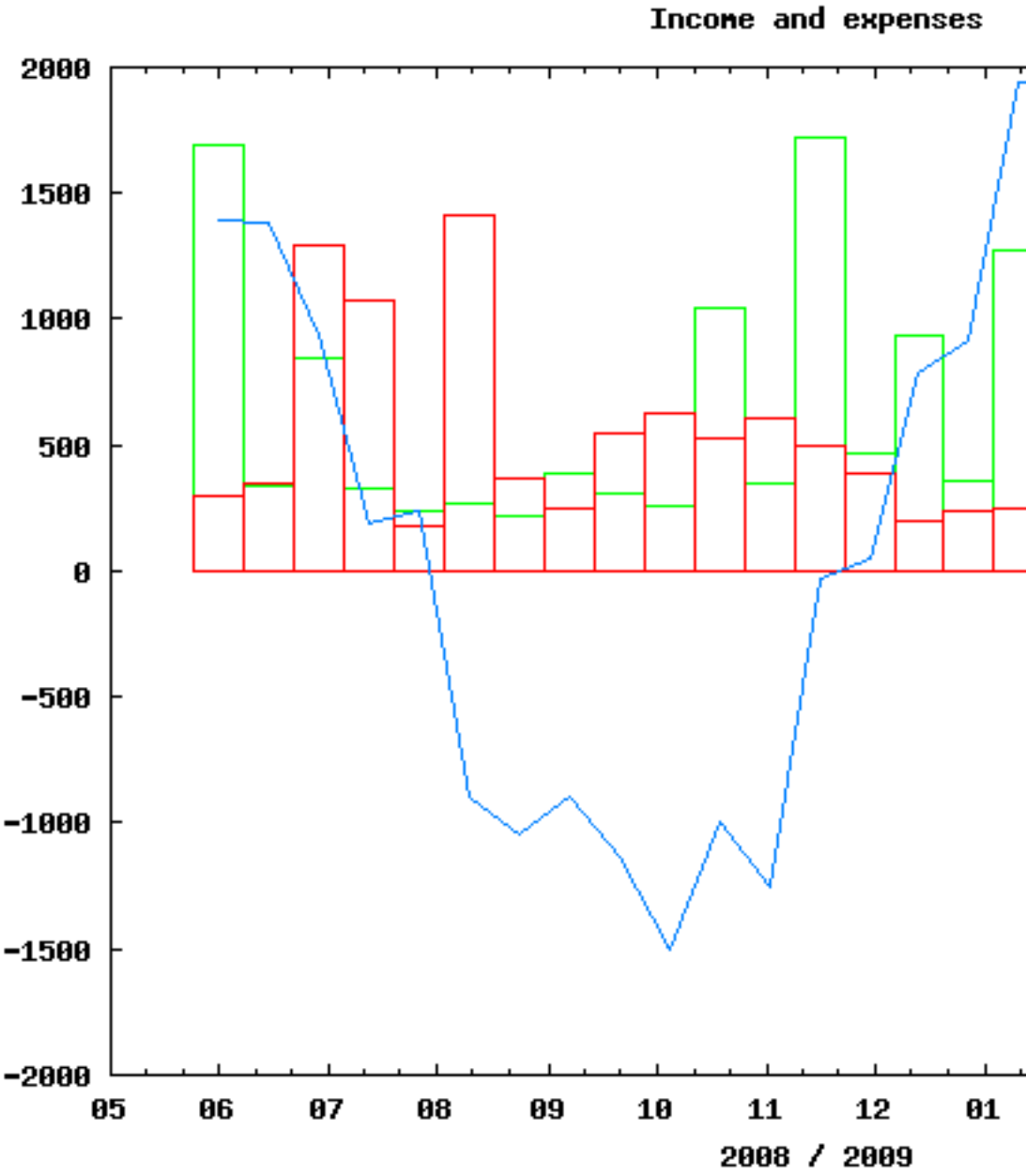
```
           03 filler    pic x.
           03 income    pic -zzzzzz9.99.
           03 filler    pic x.
           03 expense   pic -zzzzzz9.99.
           03 filler    pic x.
           03 networth  pic -zzzzzz9.99.

       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 'gnu-
plot -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' us-
ing 1:2 with lines title 'sin(x)'"
      - to gnuplot-command.
       write gnuplot-command.
       move "replot 'ocgpdata.txt' us-
ing 1:3 with lines title 'cos(x)'"
      - to gnuplot-command.
       write gnuplot-command.
       move "set terminal png; set out-
put '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 ran-
dom networth
       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 ex-
penses"' to gnuplot-command.
       write gnuplot-command.
       move 'set xla-
bel "2008 / 2009"' to gnuplot-command.
       write gnuplot-command.
       move 'plot "ocgpdata.txt" us-
ing 1:2 with boxes title "Income"
      -' linecolor rgb "green"' to gnuplot-
command.
       write gnuplot-command.
       move 'replot "ocgpdata.txt" us-
ing 1:3 with boxes title "Expense"
      -' linecolor rgb "red"' to gnuplot-
command.
       write gnuplot-command.
       move 'replot "ocgpdata.txt" us-
ing 1:4 with lines title "Worth"'
      -    to gnuplot-command.
       write gnuplot-command.
       move 'set terminal png; set out-
put "plotworth.png"; replot'
      -    to gnuplot-command.
       write gnuplot-command.
       close scriptfile.

      * Generate a bi-
weekly dataset with date, income, ex-
pense, 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 ran-
dom() * 2000
              compute worth = worth + amount
              move amount to income
              compute amount  = function ran-
dom() * 1800
              compute worth = worth - amount
              move amount to expense
              move worth to networth
              write moneyrec
       end-perform.
       close moneyfile.

      * Invoke gnu-
plot 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:


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-10
-5
0



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.

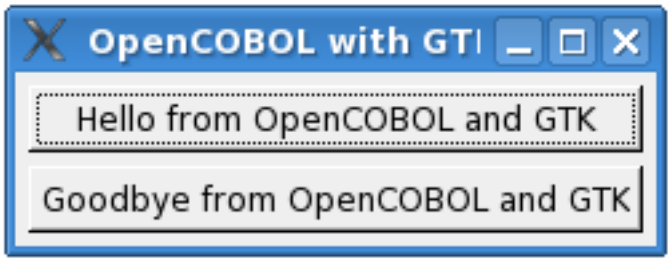
What does GIMP stand for?

GIMP is an acronym for the *GNU Image Manipulation Program*, a very complete and robust grapic design tool. See the GIMP site for more information.

GTK+ is the GIMP ToolKit. See the GTK site for more information.

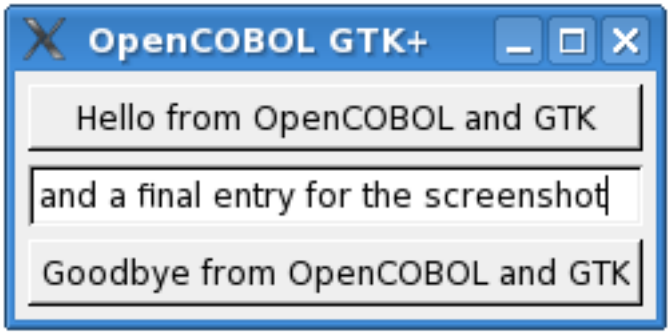
Simple buttons

```
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 - editted             , +0000000021
text: then a clear                      , +0000000012
text:                                   , +0000000000
text: and a final entry for the screen, +0000000032
```



Sample OpenCOBOL that generated the above

```
*> ** *>>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             us-
age procedure-pointer.
       01 params               usage pointer.

      *> ****************************************************************
       procedure division.

      *> Initialize GTK
       CALL "CBL_OC_GTK_INIT_CHECK" return-
ing result END-CALL
    >>D   display "init: " result end-display

      *> Create a toplevel window
       CALL "CBL_OC_GTK_WINDOW_NEW" return-
ing 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 de-
stroy, quit main loop handler
       set callback to entry "CBL_OC_destroy"
       CALL "CBL_OC_G_SIGNAL_CONNECT"
           using by value gtk-window
               by refer-
ence "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 Open-
COBOL and GTK" & x"00"
       returning gtk-hello
       END-CALL
    >>D   display "button: " gtk-hello end-
display

      *> Connect the hello but-
ton 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
```

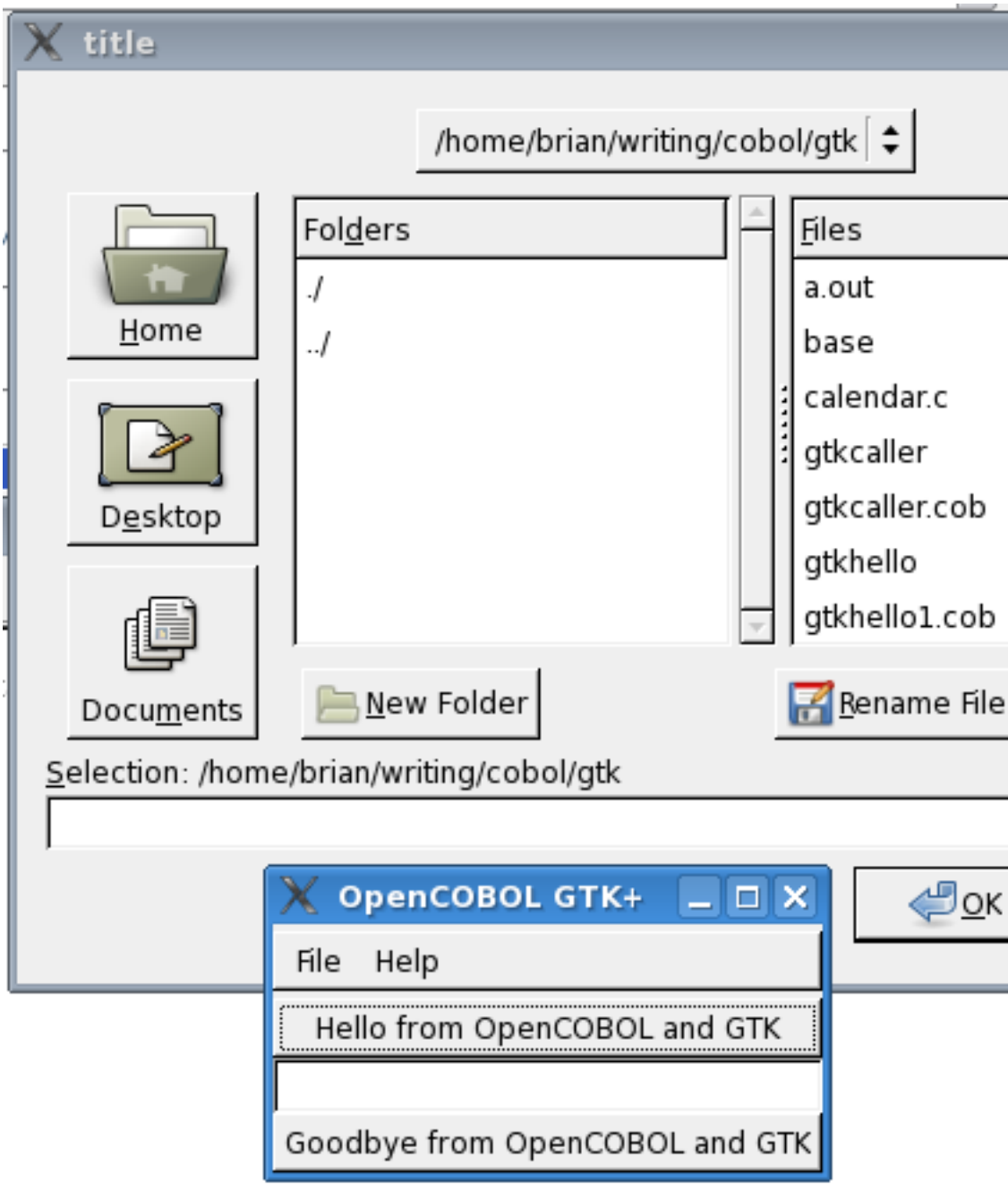
```
       END-CALL

      *> Pack the but-
ton 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, pass-
ing 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 bot-
tom
       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 Open-
```

```
COBOL and GTK" & x"00"
           returning gtk-goodbye
       END-CALL
    >>D   display "button: " gtk-goodbye end-
display

      *> Connect the bye but-
ton 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, un-
der 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 re-
turned when GTK main exits
       CALL "CBL_OC_GTK_MAIN" END-CALL

      *> Something termi-
nated the GTK main loop, sys-close or bye or
       display "ending..." end-display
```

```
       goback.
       end program gtkhello.
      *> ****************************************************************

      *> **** window shutdown call-
back *******************************
       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 call-
back ****************************
       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 call-
back ************************
      *> This procedure called from GTK on en-
ter 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
       dis-
play "text: " textfield ", " textlen end-
display

       goback.
       end program CBL_OC_activate.
       *><*
```

A screenshot with added menu and file dialog after hitting File -> Open



6 Notes

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.

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=UserMan
for some details.

6.3 ASCII

American Symbolic Code for Information Interchange.

The character encoding common to personal computers and the 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 for more info.

Attention!

Unicode?

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 "#".
```

6.5 DSO

Dynamic Shared Objects.

Similar to but subtlet different from *share libraries*.

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
```

6.7 gdb

The GNU symbolic debugger. Big, deep, wide.

```
$ info gdb for the details.
```

or visit http://www.gnu.org/software/gdb/documentation/

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*.

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.

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.

6.11 APT

Advanced Package Tool. One of the strengths of the Debian GNU/Linux system. Allows for dependency checked binary packages.

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_syst_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.

6.13 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 or-
der                       ok
  3: COPY: separa-
tors                                ok
  4: COPY: partial replace-
ment                     ok
  5: COPY: recursive replace-
ment                    ok
  6: Invalid PROGRAM-
ID                              ok
  7: Invalid PROGRAM-
ID type clause (1)              ok
  8: Invalid PROGRAM-
ID type clause (2)              ok
  9: Unde-
fined data name                                  ok
 10: Unde-
fined group name                                 ok
 11: Unde-
fined data name in group                        ok
 12: Refer-
ence not a group name                          ok
 13: Incomplete 01 defini-
tion                        ok
 14: Same labels in different sec-
tions             ok
 15: Redefini-
tion of 01 items                           ok
 16: Redefini-
tion of 01 and 02 items                    ok
 17: Redefini-
tion of 02 items                           ok
 18: Redefini-
tion of 77 items                           ok
 19: Redefini-
tion of 01 and 77 items                    ok
 20: Redefini-
tion of 88 items                           ok
 21: Ambiguous refer-
ence to 02 items                 ok
 22: Ambiguous refer-
ence to 02 and 03 items          ok
 23: Ambiguous reference with qualifica-
tion       ok
 24: Unique reference with ambiguous quali-
fiers   ok
 25: Undefined proce-
dure name                        ok
 26: Redefinition of sec-
tion names                  ok
```

```
 27: Redefinition of section and para-
graph names  ok
 28: Redefinition of para-
graph names              ok
 29: Ambiguous reference to para-
graph name        ok
 30: Non-
matching level numbers (extension)       ok
 31: Ambigu-
ous AND/OR                              ok
 32: START on SEQUEN-
TIAL file                     ok
 33: Subscripted item requires OC-
CURS clause      ok
 34: The number of sub-
scripts                     ok
 35: OC-
CURS with level 01, 66, 77, and 88         ok
 36: OCCURS with variable-
occurrence data item    ok
 37: Nested OC-
CURS clause                          ok
 38: OCCURS DEPENDING followed by an-
other field   ok
 39: OCCURS DEPENDING with-
out TO clause           ok
 40: REDEFINES: not following entry-
name          ok
 41: REDE-
FINES: level 02 by 01                    ok
 42: REDE-
FINES: level 03 by 02                    ok
 43: REDE-
FINES: level 66                          ok
 44: REDE-
FINES: level 88                          ok
 45: REDEFINES: lower level num-
ber                ok
 46: REDEFINES: with OC-
CURS                        ok
 47: REDEFINES: with sub-
script                     ok
 48: REDEFINES: with variable occur-
rence          ok
 49: REDEFINES: with qualifica-
tion                ok
 50: REDEFINES: multiple redefini-
tion             ok
 51: REDEFINES: size ex-
ceeds                       ok
```

```
 52: REDE-
FINES: with VALUE                          ok
 53: REDEFINES: with interven-
tion                 ok
 54: REDEFINES: within REDE-
FINES                  ok
 55: Numeric item (inte-
ger)                       ok
 56: Numeric item (non-
integer)                   ok
 57: Numeric item with pic-
ture P                  ok
 58: Signed numeric lit-
eral                       ok
 59: Alpha-
betic item                                ok
 60: Alphanu-
meric item                              ok
 61: Alphanu-
meric 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 alpha-
betic item                 ok
 66: MOVE alpha-
betic TO x                           ok
 67: MOVE alphanu-
meric TO x                         ok
 68: MOVE alphanumeric-
edited TO x                ok
 69: MOVE numeric (inte-
ger) 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 For-
mat 1                   ok
 75: Category check of For-
mat 2                   ok
 76: Category check of liter-
als                  ok
 77: SET: misc                                      ok
```

```
## ------------- ##
## Test results. ##
## ------------- ##

All 77 tests were successful.
PASS: ./syntax
## ---------------------------------- ##
## OpenCOBOL 1.1 test suite: Run Tests. ##
## ---------------------------------- ##
  1: DISPLAY liter-
als                              ok
  2: DISPLAY literals, DECIMAL-
POINT is COMMA     ok
  3: Hexadecimal lit-
eral                           ok
  4: DIS-
PLAY data items with VALUE clause         ok
  5: DISPLAY data items with MOVE state-
ment       ok
  6: GLOBAL at same level                          ok
  7: GLOBAL at lower level                         ok
  8: non-
numeric subscript                         ok
  9: The range of sub-
scripts                       ok
 10: Sub-
script out of bounds (1)                  ok
 11: Sub-
script out of bounds (2)                  ok
 12: Value of DEPEND-
ING ON N out of bounds (lower)ok
 13: Value of DEPEND-
ING ON N out of bounds (upper)ok
 14: Sub-
script bounds with ODO (lower)            ok
 15: Subscript bounds with ODO (up-
per)            ok
 16: Sub-
script bounds with ODO                    ok
 17: Subscript by arithmetic expres-
sion          ok
 18: Separate sign posi-
tions                        ok
 19: Static reference modifica-
tion               ok
 20: Dynamic reference modifica-
tion              ok
 21: Static out of bounds                         ok
 22: Offset under-
flow                             ok
```

```
 23: Offset over-
flow                                ok
 24: Length under-
flow                               ok
 25: Length over-
flow                                ok
 26: AC-
CEPT                                          ok
 27: INITIALIZE group entry with OC-
CURS           ok
 28: INITIALIZE OCCURS with nu-
meric edited        ok
 29: INITIALIZE com-
plex group (1)                  ok
 30: INITIALIZE com-
plex group (2)                  ok
 31: INITIALIZE with REDE-
FINES                      ok
 32: Source file not found                           ok
 33: Comma separator with-
out space                  ok
 34: LOCAL-
STORAGE                                    ok
 35: EXTER-
NAL data item                               ok
 36: EXTER-
NAL AS data item                            ok
 37: cobcrun valida-
tion                               ok
 38: MOVE to it-
self                                   ok
 39: MOVE with ref-
mod                                ok
 40: MOVE with refmod (vari-
able)                   ok
 41: MOVE with group ref-
mod                          ok
 42: MOVE in-
dexes                                      ok
 43: MOVE X'00'                                      ok
 44: Level 01 sub-
scripts                            ok
 45: Class check with reference modifica-
tion      ok
 46: Index and parenthesized expres-
sion           ok
 47: Alphanumeric and binary nu-
meric              ok
 48: Dynamic call with static link-
ing             ok
```

```
 49: CALL m1. CALL m2. CALL m1.                    ok
 50: CALL binary literal parame-
ter/LENGTH OF      ok
 51: INSPECT REPLACING LEADING ZE-
ROS BY SPACES    ok
 52: INSPECT: No repeat conver-
sion check          ok
 53: INSPECT: REPLACING figurative con-
stant       ok
 54: INSPECT: TALLYING BE-
FORE                      ok
 55: INSPECT: TALLYING AF-
TER                       ok
 56: INSPECT REPLACING TRAILING ZE-
ROS BY SPACES   ok
 57: INSPECT REPLACING com-
plex                     ok
 58: SWITCHES                                      ok
 59: Nested PER-
FORM                                  ok
 60: EXIT PER-
FORM                                    ok
 61: EXIT PERFORM CY-
CLE                              ok
 62: EXIT PARA-
GRAPH                                 ok
 63: EXIT SEC-
TION                                    ok
 64: 88 with FILLER                                ok
 65: Non-
overflow after overflow                   ok
 66: PERFORM ... CON-
TINUE                            ok
 67: STRING with subscript refer-
ence              ok
 68: UNSTRING DELIMITED ALL LOW-
VALUE             ok
 69: READ INTO AT-
END sequence                      ok
 70: First READ on empty SEQUENTIAL IN-
DEXED file  ok
 71: REWRITE a RELATIVE file with RANDOM ac-
cess   ok
 72: SORT: ta-
ble sort                                ok
 73: SORT: EBCDIC ta-
ble sort                         ok
 74: SORT nonexis-
tent file                           ok
 75: PIC ZZZ-
```

```
, ZZZ+                                  ok
 76: Larger REDE-
FINES lengths                        ok
 77: PER-
FORM type OSVS                                ok
 78: Sticky LINK-
AGE                                  ok
 79: COB_PRE_LOAD test                                    ok
 80: COB_LOAD_CASE=UPPER test                             ok
 81: 88 level with FALSE IS clause                        ok
 82: ALLO-
CATE/FREE with BASED item                   ok
 83: INITIZIALIZE with reference modifica-
tion     ok
 84: CALL with OMITTED parame-
ter                    ok
 85: ANY LENGTH                                           ok
 86: BASED item non-
ALLOCATED (debug)               ok
 87: COMP-
5                                       ok
 88: Hexadecimal numeric lit-
eral                    ok
 89: Semi-
parenthesized condition                  ok
 90: AD-
DRESS OF                                    ok
 91: LENGTH OF                                            ok
 92: WHEN-
COMPILED                                 ok
 93: Complex OCCURS DEPEND-
ING ON                     ok
 94: MOVE NON-INTEGER 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: AS-
SIGN MF                                     ok
101: AS-
SIGN IBM                                    ok
102: ASSIGN map-
ping                                   ok
```

```
103: ASSIGN expan-
sion                                ok
104: AS-
SIGN with COB_FILE_PATH                        ok
105: NUMBER-OF-CALL-
PARAMETERS                        ok
106: PROCEDURE DIVISION US-
ING BY ...                   ok
107: PROCEDURE DIVISION CHAIN-
ING ...                  ok
108: STOP RUN RETURN-
ING                               ok
109: EN-
TRY                                            ok
110: LINE SEQUEN-
TIAL write                             ok
111: LINE SEQUEN-
TIAL read                              ok
112: ASSIGN to KEY-
BOARD/DISPLAY                       ok
113: Environment/Argument vari-
able                   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 state-
ment                            ok
123: RETURN-
CODE moving                                ok
124: RETURN-
CODE passing                               ok
125: RETURN-
CODE nested                                ok
126: FUNC-
TION ABS                                      ok
127: FUNC-
TION ACOS                                     ok
128: FUNCTION ANNU-
ITY                                 ok
129: FUNC-
```

```
TION ASIN                                  ok
130: FUNC-
TION ATAN                                  ok
131: FUNC-
TION CHAR                                  ok
132: FUNCTION COMBINED-
DATETIME                    ok
133: FUNCTION CONCATE-
NATE                          ok
134: FUNCTION CONCATENATE with reference mod-
ding  ok
135: FUNC-
TION 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: FUNC-
TION E                                     ok
141: FUNCTION EXCEPTION-
FILE                        ok
142: FUNCTION EXCEPTION-
LOCATION                    ok
143: FUNCTION EXCEPTION-
STATEMENT                   ok
144: FUNCTION EXCEPTION-
STATUS                      ok
145: FUNC-
TION EXP                                   ok
146: FUNCTION FACTO-
RIAL                            ok
147: FUNCTION FRACTION-
PART                         ok
148: FUNCTION INTE-
GER                               ok
149: FUNCTION INTEGER-OF-
DATE                       ok
150: FUNCTION INTEGER-OF-
DAY                        ok
151: FUNCTION INTEGER-
PART                          ok
152: FUNC-
TION LENGTH                                ok
153: FUNCTION LOCALE-
DATE                          ok
154: FUNCTION LOCALE-
```

```
TIME                          ok
155: FUNCTION LOCALE-TIME-FROM-
SECONDS            ok
156: FUNC-
TION LOG                                    ok
157: FUNC-
TION LOG10                                  ok
158: FUNCTION LOWER-
CASE                            ok
159: FUNCTION LOWER-
CASE with reference modding   ok
160: FUNC-
TION MAX                                    ok
161: FUNC-
TION MEAN                                   ok
162: FUNCTION ME-
DIAN                                  ok
163: FUNC-
TION MIDRANGE                               ok
164: FUNC-
TION MIN                                    ok
165: FUNC-
TION MOD                                    ok
166: FUNCTION NUM-
VAL                                   ok
167: FUNCTION NUMVAL-
C                               ok
168: FUNC-
TION ORD                                    ok
169: FUNCTION ORD-
MAX                                ok
170: FUNCTION ORD-
MIN                                ok
171: FUNC-
TION PI                                     ok
172: FUNCTION PRESENT-
VALUE                         ok
173: FUNC-
TION RANGE                                  ok
174: FUNC-
TION REM                                    ok
175: FUNCTION RE-
VERSE                                 ok
176: FUNCTION REVERSE with reference mod-
ding      ok
177: FUNCTION SECONDS-FROM-FORMATTED-
TIME         ok
178: FUNCTION SECONDS-PAST-
MIDNIGHT               ok
179: FUNC-
```

```
TION SIGN                                  ok
180: FUNC-
TION SIN                                   ok
181: FUNC-
TION SQRT                                  ok
182: FUNCTION STANDARD-
DEVIATION                   ok
183: FUNCTION STORED-CHAR-
LENGTH                  ok
184: FUNCTION SUBSTI-
TUTE                             ok
185: FUNCTION SUBSTITUTE with reference mod-
ding   ok
186: FUNCTION SUBSTITUTE-
CASE                      ok
187: FUNCTION SUBSTITUTE-
CASE with reference mod  ok
188: FUNC-
TION TAN                                   ok
189: FUNC-
TION TRIM                                  ok
190: FUNCTION TRIM with reference mod-
ding         ok
191: FUNCTION UPPER-
CASE                             ok
192: FUNCTION UPPER-
CASE with reference modding   ok
193: FUNCTION VARI-
ANCE                               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 liter-
als                                ok
  2: DISPLAY literals, DECIMAL-
POINT is COMMA     ok
  3: Hexadecimal lit-
eral                            ok
```

```
  4: DIS-
PLAY data items with VALUE clause         ok
  5: DISPLAY data items with MOVE state-
ment       ok
  6: GLOBAL at same level                          ok
  7: GLOBAL at lower level                         ok
  8: non-
numeric subscript                        ok
  9: The range of sub-
scripts                      ok
 10: Sub-
script out of bounds (1)                  ok
 11: Sub-
script out of bounds (2)                  ok
 12: Value of DEPEND-
ING ON N out of bounds (lower)ok
 13: Value of DEPEND-
ING ON N out of bounds (upper)ok
 14: Sub-
script bounds with ODO (lower)            ok
 15: Subscript bounds with ODO (up-
per)            ok
 16: Sub-
script bounds with ODO                    ok
 17: Subscript by arithmetic expres-
sion           ok
 18: Separate sign posi-
tions                       ok
 19: Static reference modifica-
tion               ok
 20: Dynamic reference modifica-
tion               ok
 21: Static out of bounds                         ok
 22: Offset under-
flow                              ok
 23: Offset over-
flow                               ok
 24: Length under-
flow                              ok
 25: Length over-
flow                               ok
 26: AC-
CEPT                                      ok
 27: INITIALIZE group entry with OC-
CURS           ok
 28: INITIALIZE OCCURS with nu-
meric edited        ok
 29: INITIALIZE com-
plex group (1)                 ok
 30: INITIALIZE com-
```

```
plex group (2)                 ok
 31: INITIALIZE with REDE-
FINES                     ok
 32: Source file not found                          ok
 33: Comma separator with-
out space                 ok
 34: LOCAL-
STORAGE                                  ok
 35: EXTER-
NAL data item                             ok
 36: EXTER-
NAL AS data item                          ok
 37: cobcrun valida-
tion                              ok
 38: MOVE to it-
self                                  ok
 39: MOVE with ref-
mod                               ok
 40: MOVE with refmod (vari-
able)                   ok
 41: MOVE with group ref-
mod                           ok
 42: MOVE in-
dexes                                    ok
 43: MOVE X'00'                                     ok
 44: Level 01 sub-
scripts                            ok
 45: Class check with reference modifica-
tion      ok
 46: Index and parenthesized expres-
sion           ok
 47: Alphanumeric and binary nu-
meric               ok
 48: Dynamic call with static link-
ing             ok
 49: CALL m1. CALL m2. CALL m1.                     ok
 50: CALL binary literal parame-
ter/LENGTH OF      ok
 51: INSPECT REPLACING LEADING ZE-
ROS BY SPACES    ok
 52: INSPECT: No repeat conver-
sion check          ok
 53: INSPECT: REPLACING figurative con-
stant       ok
 54: INSPECT: TALLYING BE-
FORE                      ok
 55: INSPECT: TALLYING AF-
TER                       ok
 56: INSPECT REPLACING TRAILING ZE-
ROS BY SPACES   ok
```

```
 57: INSPECT REPLACING com-
plex                     ok
 58: SWITCHES                                          ok
 59: Nested PER-
FORM                                  ok
 60: EXIT PER-
FORM                                    ok
 61: EXIT PERFORM CY-
CLE                               ok
 62: EXIT PARA-
GRAPH                                 ok
 63: EXIT SEC-
TION                                    ok
 64: 88 with FILLER                                    ok
 65: Non-
overflow after overflow                    ok
 66: PERFORM ... CON-
TINUE                             ok
 67: STRING with subscript refer-
ence               ok
 68: UNSTRING DELIMITED ALL LOW-
VALUE              ok
 69: READ INTO AT-
END sequence                       ok
 70: First READ on empty SEQUENTIAL IN-
DEXED file  ok
 71: REWRITE a RELATIVE file with RANDOM ac-
cess   ok
 72: SORT: ta-
ble sort                                ok
 73: SORT: EBCDIC ta-
ble sort                          ok
 74: SORT nonexis-
tent file                            ok
 75: PIC ZZZ-
, ZZZ+                                  ok
 76: Larger REDE-
FINES lengths                       ok
 77: PER-
FORM type OSVS                               ok
 78: Sticky LINK-
AGE                                   ok
 79: COB_PRE_LOAD test                                 ok
 80: COB_LOAD_CASE=UPPER test                          ok
 81: 88 level with FALSE IS clause                     ok
 82: ALLO-
CATE/FREE with BASED item                  ok
 83: INITIZIALIZE with reference modifica-
tion     ok
 84: CALL with OMITTED parame-
```

```
ter                    ok
 85: ANY LENGTH                                        ok
 86: BASED item non-
ALLOCATED (debug)              ok
 87: COMP-
5                                        ok
 88: Hexadecimal numeric lit-
eral                    ok
 89: Semi-
parenthesized condition                  ok
 90: AD-
DRESS OF                                    ok
 91: LENGTH OF                                         ok
 92: WHEN-
COMPILED                                 ok
 93: Complex OCCURS DEPEND-
ING ON                    ok
 94: MOVE NON-INTEGER 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: AS-
SIGN MF                                     ok
101: AS-
SIGN IBM                                    ok
102: ASSIGN map-
ping                                   ok
103: ASSIGN expan-
sion                                 ok
104: AS-
SIGN with COB_FILE_PATH                     ok
105: NUMBER-OF-CALL-
PARAMETERS                        ok
106: PROCEDURE DIVISION US-
ING BY ...                ok
107: PROCEDURE DIVISION CHAIN-
ING ...                ok
108: STOP RUN RETURN-
ING                               ok
109: EN-
TRY                                         ok
110: LINE SEQUEN-
```

```
TIAL write                          ok
111: LINE SEQUEN-
TIAL read                           ok
112: ASSIGN to KEY-
BOARD/DISPLAY                    ok
113: Environment/Argument vari-
able                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 state-
ment                          ok
123: RETURN-
CODE moving                            ok
124: RETURN-
CODE passing                           ok
125: RETURN-
CODE nested                            ok
126: FUNC-
TION ABS                                  ok
127: FUNC-
TION ACOS                                 ok
128: FUNCTION ANNU-
ITY                               ok
129: FUNC-
TION ASIN                                 ok
130: FUNC-
TION ATAN                                 ok
131: FUNC-
TION CHAR                                 ok
132: FUNCTION COMBINED-
DATETIME                    ok
133: FUNCTION CONCATE-
NATE                           ok
134: FUNCTION CONCATENATE with reference mod-
ding  ok
135: FUNC-
TION 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: FUNC-
TION E                                    ok
141: FUNCTION EXCEPTION-
FILE                       ok
142: FUNCTION EXCEPTION-
LOCATION                   ok
143: FUNCTION EXCEPTION-
STATEMENT                  ok
144: FUNCTION EXCEPTION-
STATUS                     ok
145: FUNC-
TION EXP                                  ok
146: FUNCTION FACTO-
RIAL                           ok
147: FUNCTION FRACTION-
PART                        ok
148: FUNCTION INTE-
GER                             ok
149: FUNCTION INTEGER-OF-
DATE                      ok
150: FUNCTION INTEGER-OF-
DAY                       ok
151: FUNCTION INTEGER-
PART                         ok
152: FUNC-
TION LENGTH                               ok
153: FUNCTION LOCALE-
DATE                         ok
154: FUNCTION LOCALE-
TIME                         ok
155: FUNCTION LOCALE-TIME-FROM-
SECONDS            ok
156: FUNC-
TION LOG                                  ok
157: FUNC-
TION LOG10                                ok
158: FUNCTION LOWER-
CASE                            ok
159: FUNCTION LOWER-
CASE with reference modding   ok
160: FUNC-
TION MAX                                  ok
161: FUNC-
TION MEAN                                 ok
```

```
162: FUNCTION ME-
DIAN                                ok
163: FUNC-
TION MIDRANGE                                ok
164: FUNC-
TION MIN                                     ok
165: FUNC-
TION MOD                                     ok
166: FUNCTION NUM-
VAL                                 ok
167: FUNCTION NUMVAL-
C                              ok
168: FUNC-
TION ORD                                     ok
169: FUNCTION ORD-
MAX                                 ok
170: FUNCTION ORD-
MIN                                 ok
171: FUNC-
TION PI                                      ok
172: FUNCTION PRESENT-
VALUE                          ok
173: FUNC-
TION RANGE                                   ok
174: FUNC-
TION REM                                     ok
175: FUNCTION RE-
VERSE                                ok
176: FUNCTION REVERSE with reference mod-
ding      ok
177: FUNCTION SECONDS-FROM-FORMATTED-
TIME         ok
178: FUNCTION SECONDS-PAST-
MIDNIGHT               ok
179: FUNC-
TION SIGN                                    ok
180: FUNC-
TION SIN                                     ok
181: FUNC-
TION SQRT                                    ok
182: FUNCTION STANDARD-
DEVIATION                   ok
183: FUNCTION STORED-CHAR-
LENGTH                   ok
184: FUNCTION SUBSTI-
TUTE                             ok
185: FUNCTION SUBSTITUTE with reference mod-
ding   ok
186: FUNCTION SUBSTITUTE-
CASE                      ok
```

```
187: FUNCTION SUBSTITUTE-
CASE with reference mod  ok
188: FUNC-
TION TAN                                  ok
189: FUNC-
TION TRIM                                 ok
190: FUNCTION TRIM with reference mod-
ding         ok
191: FUNCTION UPPER-
CASE                           ok
192: FUNCTION UPPER-
CASE with reference modding   ok
193: FUNCTION VARI-
ANCE                             ok
194: FUNCTION WHEN-
COMPILED                         ok

## ------------ ##
## Test results. ##
## ------------ ##

All 194 tests were successful.
PASS: ./run-0
## ------------------------------------------
--- ##
## OpenCOBOL 1.1 test suite: Data Representa-
tion. ##
## ------------------------------------------
--- ##
  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: DIS-
PLAY: Sign ASCII                           ok
  9: DIS-
PLAY: Sign ASCII (2)                       ok
 10: DIS-
PLAY: 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: dis-
play                              ok

## ------------- ##
## Test results. ##
## ------------- ##

All 17 tests were successful.
PASS: ./data-rep

## Data representation tests with -O option ##

## ------------------------------------------
--- ##
## OpenCOBOL 1.1 test suite: Data Representa-
tion. ##
## ------------------------------------------
--- ##
  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: DIS-
PLAY: Sign ASCII                                ok
  9: DIS-
PLAY: Sign ASCII (2)                            ok
 10: DIS-
PLAY: 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: dis-
play                              ok

## ------------ ##
## Test results. ##
## ------------ ##

All 17 tests were successful.
PASS: ./data-rep-0
==================
All 5 tests passed
==================
```

[Keisuke] Keisuke Nishida
Initial developer and creator of OpenCOBOL. From the 1990s through 2004 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 2009)* 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 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 OpenCOBOL forum.

[btiffin] Brian Tiffin
Initial FAQ. sample programs for OpenCOBOL 1.1.

[aoirthoir] Joseph James Frantz
Hosting, support.

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.

[jrls_swla] John Ellis
Samples and how-to's and ...

[human] human
Samples and style

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 reorg. 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.