GNU Cobol Manual

for GNU Cobol 2.0

Keisuke Nishida / Roger While

Edition 2.0
Updated for GNU Cobol 2.0
19 June 2014

GNU Cobol is a free and open-source COBOL compiler, which translates COBOL programs to C code and compiles it using GCC or other native operating system C compiler.

This manual corresponds to GNU Cobol 2.0.

Table of Contents

1 Getting Started

1.1 Hello World!

This is a sample program that displays "Hello World":

```
---- hello.cob -------------------------
      * Sample COBOL program
       IDENTIFICATION DIVISION.
       PROGRAM-ID. hello.
       PROCEDURE DIVISION.
       DISPLAY "Hello World!".
       STOP RUN.
----------------------------------------
```

The compiler is `cobc`, which is executed as follows:

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

The executable file name (i.e., '`hello`' in this case) is determined by removing the extension from the source file name.

You can specify the executable file name by specifying the compiler option `-o` as follows:

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

Using more modern sources.

```
---- hellonew.cob ----------------
*> Sample GNU Cobol program
identification division.
program-id. hellonew.
procedure division.
display
   "Hello New World!"
end-display
goback.
---------------------------------
$ cobc -x -free hellonew.cob
$ ./hellonew
Hello New World!
```

Showing the use of free format, to end of line comments, the `goback` verb, and proper use of terminator with `end-display`.

2 Compile

This chapter describes how to compile COBOL programs using GNU Cobol.

2.1 Compiler Options

The compiler `cobc` accepts the options described in this section.
General syntax -
`cobc` [options] file [file ..]
A complete list of options can be displayed by using the help option.

2.1.1 Help Options

The following switches can be used for informational displays:

`--help` Display help screen (see Appendix A [cobc –help], page 18). `-h` will also display the help. No further actions will be taken except for further display options.

`--version`
Display compiler version, author package date and executable build date. `-V` will also display version. No further actions will be taken except for further display options.

`--info` Display build information (see Appendix B [cobc –info], page 21). No further actions will be taken except for further display options.

`-v` Verbosely displays the programs invoked during compilations.

`--list-reserved`
Display reserved words(see Appendix C [cobc –list-reserved], page 22). A Y/N field shows if the word is supported.[1] No further actions will be taken except for further display options.

`--list-intrinsics`
Display intrinsic functions (see Appendix D [cobc –list-intrinsics], page 34). A Y/N field shows if the function is implemented. No further actions will be taken except for further display options.

`--list-system`
Display system routines (see Appendix E [cobc –list-system], page 37). No further actions will be taken except for further display options.

`--list-mnemonics`
Display mnemonic names (see Appendix F [cobc –list-mnemonics], page 39). No further actions will be taken except for further display options.

2.1.2 Built Target

The `cobc` compiler can handle '`*.cob`', '`*.cbl`' as COBOL source code, '`*.c`' for C source code, '`*.o`' for object code, '`*.i`' for preprocessed code and '`*.so`' for dynamic modules and will do the right thing in terms of generation, compilation, or link.

The following options specify the target type produced by the compiler:

`-E` Preprocess only. Compiler directives are executed. Comment lines are removed. COPY statements are expanded. The output is saved in file '`*.i`'.

`-C` Translation only. COBOL source files are translated into C files. The output is saved in file '`*.c`'.

[1] Support may be partial or complete

`-S` Compile only. Translated C files are compiled by the C compiler to assembler code. The output is saved in file '`*.s`'.

`-c` Compile and assemble. This is equivalent to `cc -c`. The output is saved in file '`*.o`'.

`-m` Compile, assemble, and build a dynamically loadable module (i.e., a shared library). The output is saved in file '`*.so`'. This is the default behaviour if not other options are given.[2].

`-b` Compile, assemble, and combine all input files into a single dynamically loadable module. Unless `-o` is also used, the output is saved using the first filename as '`*.so`'.

`-x` Include the main function in the output, creating an executable image. The main entry point being the outermost `PROGRAM-ID`.

This option takes effect at the translation stage. If you give this option with `-C`, you will see the main function at the end of the generated C file.

`-I <directory>`
Add <directory> to copy/include search path

`-L <directory>`
Add <directory> to library search path

`-l <lib>` Link the library <lib>

`-D <define>`
Pass <define> to the COBOL compiler

`-o <file>` Place the output into <file>

Without any options above, the compiler builds a dynamically loadable module.

2.1.3 Source Format

GNU Cobol supports both fixed and free source format.

The default format is the fixed format. This can be explicitly overwritten by one of the following options:

`-free` Free format. The program-text area starts in column 1 and continues till the end of line. Effectively 255 characters in GNU Cobol.

`-fixed` Fixed format. Source code is divided into a 1-6 column sequence number area, column 7 indicator area, columns 8-72 program-text area, with columns 72-80 as a reference area. Historically this format is based on 80 character punch cards. FIXED format is the default used by the compiler unless overridden by compiler switch or source code directive, `>>SOURCE [FORMAT] [IS] {FIXED|FREE}`.

2.1.4 Warning Options

`-W` Enable every possible warning. This includes more information than -Wall would normally provide.

`-Wall` Enable all common warnings

`-Warchaic`
Warn if archaic features are used

`-Wcall-params`
Warn non 01/77 items for CALL params (NOT set with -Wall)

[2] The extension varies depending on your host.

`-Wcolumn-overflow`
: Warn if text after column 72 in FIXED format (NOT set with -Wall)

`-Wconstant`
: Warn inconsistent constant

`-Wimplicit-define`
: Warn implicitly defined data items

`-Wlinkage`
: Warn dangling LINKAGE items (NOT set with -Wall)

`-Wobsolete`
: Warn if obsolete features are used

`-Wparentheses`
: Warn lack of parentheses around AND within OR

`-Wredefinition`
: Warn incompatible redefinition of data items

`-Wstrict-typing`
: Warn type mismatch strictly

`-Wterminator`
: Warn lack of scope terminator END-XXX (NOT set with -Wall)

`-Wtruncate`
: Warn possible field truncation (NOT set with -Wall)

`-Wunreachable`
: Warn unreachable statements (NOT set with -Wall)

2.1.5 Configuration Options

`-std=<dialect>`
: Compiler uses the given dialect to determine certain compiler features and warnings.
See Appendix G [config/default.conf], page 41, and '`config/*.conf`'.

`-std=cobol2002`
: Cobol 2002

`-std=cobol85`
: Cobol 85

`-std=ibm` IBM Compatible

`-std=mvs` MVS Compatible

`-std=bs2000`
: BS2000 Compatible

`-std=mf` Micro Focus Compatible

`-std=default`
: When not specified

`-conf=<file>`
: User defined dialect configuration. See `-std=` above.
See Appendix G [config/default.conf], page 41, and '`config/*.conf`'.

2.1.6 Debug Switches

`-debug` Enable all run-time error checking

`-g` Produce debugging information in the output

`-O` Enable optimization of code size and execution speed. See `man gcc` for details.

`-O2` Optimize even more.

`-Os` Optimize for size. Optimizer will favour code size over execution speed.

`-ftrace` Generate trace code (Executed SECTION/PARAGRAPH)

`-ftraceall`
Generate trace code (Executed SECTION/PARAGRAPH/STATEMENTS)

`-fsyntax-only`
Syntax error checking only; don't emit any output

`-fdebugging-line`
Enable debugging lines ('D' in indicator column)

`-fsource-location`
Generate source location code (Turned on by -debug or -g)

`-fimplicit-init`
Do automatic initialization of the Cobol runtime system

`-fstack-check`
PERFORM stack checking (Turned on by -debug or -g)

`-fnotrunc`
Do not truncate binary fields according to PICTURE

2.1.7 Miscellaneous

`-P` Generate and place a program listing into '`*.lst`'

`-ext <extension>`
Add default file extension

`-fmfcomment`
'*' or '/' in column 1 treated as comment (FIXED only)

`-fsign=ASCII`
Numeric display sign ASCII (Default on ASCII machines)

`-fsign=EBCDIC`
Numeric display sign EBCDIC (Default on EBCDIC machines)

`-ffunctions-all`
Allow use of intrinsic functions without FUNCTION keyword

`-ffold-copy=LOWER`
Fold COPY subject to lower case (Default no transformation)

`-ffold-copy=UPPER`
Fold COPY subject to upper case (Default no transformation)

`-save-temps(=<dir>)`
Save intermediate files (default current directory)

2.2 Multiple Sources

A program often consists of multiple source files. This section describes how to compile multiple source files.

This section also describes how to build a shared library that can be used by any COBOL programs and how to use external libraries from COBOL programs.

2.2.1 Static Linking

The easiest way of combining multiple files is to compile them into a single executable.

One way is to specify all files on the command line:

```
$ cobc -x -o prog main.cob subr1.cob subr2.cob
```

Another way is to compile each file with the option `-c`, and link them at the end. The top-level program must be compiled with the option `-x`:

```
$ cobc -c subr1.cob
$ cobc -c subr2.cob
$ cobc -c -x main.cob
$ cobc -x -o prog main.o subr1.o subr2.o
```

You can link C routines as well using either method:

Method 1:

```
$ cobc -o prog main.cob subrs.c
```

Method 2:

```
$ cobc -c subrs.c
$ cobc -c -x main.cob
$ cobc -x -o prog main.o subrs.o
```

Any number of functions can be contained in a single C file.

The linked programs will be called dynamically; that is, the symbol will be resolved at run time. For example, the following COBOL statement

```
CALL "subr" USING X.
```

will be converted into an equivalent C code like this:

```
int (*func)() = cob_resolve("subr");
if (func != NULL)
  func (X);
```

With the compiler options `-fstatic-call`, more efficient code will be generated like this:

```
subr(X);
```

Note that this option is effective only when the called program name is a literal (like `CALL "subr".`). With a data name (like `CALL SUBR.`), the program is still called dynamically.

2.2.2 Dynamic Linking

There are two methods to achieve this. Method 1 (Using driver program). Compile all programs with the option `-m`:

```
$ cobc -m main.cob subr.cob
```

This creates shared object files '`main.so subr.so`'[3].

Before running the main program, install the module files in your library directory:

```
$ cp subr.so /your/cobol/lib
```

Set the environment variable `COB_LIBRARY_PATH` to your library directory, and run the main program:

[3] The extension varies depending on your host.

```
$ export COB_LIBRARY_PATH=/your/cobol/lib
```

Note: You may set the variable to directly point to the directory where you compiled the sources.

Now execute your program:

```
$ cobcrun main
```

Method 2. The main program and subprograms can be compiled separately.

The main program is compiled as usual:

```
$ cobc -x -o main main.cob
```

Subprograms are compiled with the option `-m`:

```
$ cobc -m subr.cob
```

This creates a module file '`subr.so`'[4].

Before running the main program, install the module files in your library directory:

```
$ cp subr.so /your/cobol/lib
```

Now, set the environment variable `COB_LIBRARY_PATH` to your library directory, and run the main program:

```
$ export COB_LIBRARY_PATH=/your/cobol/lib
$ ./main
```

2.2.3 Building Library

You can build a shared library by combining multiple COBOL programs and even C routines:

```
$ cobc -c subr1.cob
$ cobc -c subr2.cob
$ cc -c subr3.c
$ cc -shared -o libsubrs.so subr1.o subr2.o subr3.o
```

2.2.4 Using Library

You can use a shared library by linking it with your main program.

Before linking the library, install it in your system library directory:

```
$ cp libsubrs.so /usr/lib
```

or install it somewhere else and set `LD_LIBRARY_PATH`:

```
$ cp libsubrs.so /your/cobol/lib
$ export LD_LIBRARY_PATH=/your/cobol/lib
```

Then, compile the main program, linking the library as follows:

```
$ cobc -x main.cob -L/your/cobol/lib -lsubrs
```

2.3 C Interface

This chapter describes how to combine C programs with COBOL programs.

2.3.1 Writing Main Program in C

Include '`libcob.h`' in your C program. Call `cob_init` before using any COBOL module:

```
#include <libcob.h>

int
main (int argc, char **argv)
```

[4] The extension varies depending on your host.

```
{
  /* initialize your program */
  ...

  /* initialize the COBOL run-time library */
  cob_init (argc, argv);

  /* rest of your program */
  ...

  /* Clean up and terminate - This does not return */
  cob_stop_run (return_status);

}
```

You can write `cobc_init(0, NULL);` if you do not want to pass command line arguments to COBOL.

You can compile your C program as follows:

```
cc -c ‘cob-config --cflags‘ main.c
```

The compiled object must be linked with libcob as follows:

```
cc -o main main.o ‘cob-config --libs‘
```

2.3.2 Static linking with COBOL programs

Let's call the following COBOL module from a C program:

```
---- say.cob ---------------------------
       IDENTIFICATION DIVISION.
       PROGRAM-ID. say.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       LINKAGE SECTION.
       01 HELLO PIC X(6).
       01 WORLD PIC X(6).
       PROCEDURE DIVISION USING HELLO WORLD.
       DISPLAY HELLO WORLD.
       EXIT PROGRAM.
----------------------------------------
```

This program accepts two arguments, displays them, and exits.

From the viewpoint of C, this is equivalent to a function having the following prototype:

```
extern int say(char *hello, char *world);
```

So, your main program will look like as follows:

```
---- hello.c ---------------------------
#include <libcob.h>

extern int say(char *hello, char *world);

int
main()
{
  int ret;
  char hello[7] = "Hello ";
```

```
  char world[7] = "World!";

  cob_init(0, NULL);

  ret = say(hello, world);

  return ret;
}
---------------------------------------
```

Compile these programs as follows:

```
$ cc -c `cob-config --cflags` hello.c
$ cobc -c -static say.cob
$ cobc -x -o hello hello.o say.o
$ ./hello
Hello World!
```

2.3.3 Dynamic linking with COBOL programs

You can find a COBOL module having a specific PROGRAM-ID by using a C function `cob_resolve`, which takes the module name as a string and returns a pointer to the module function.

`cob_resolve` returns NULL if there is no module. In this case, the function `cob_resolve_error` returns the error message.

Let's see an example:

```
---- hello-dynamic.c -------------------
#include <libcob.h>

static int (*say)(char *hello, char *world);

int
main()
{
  int ret;
  char hello[7] = "Hello ";
  char world[7] = "World!";

  cob_init(0, NULL);

  /* find the module with PROGRAM-ID "say". */
  say = cob_resolve("say");

  /* if there is no such module, show error and exit */
  if (say == NULL) {
    fprintf(stderr, "%s\n", cob_resolve_error ());
    exit(1);
  }

  /* call the module found and exit with the return code */
  ret = say(hello, world);

  return ret;
}
---------------------------------------
```

Compile these programs as follows:

```
$ cc -c `cob-config --cflags` hello-dynamic.c
$ cobc -x -o hello hello-dynamic.o
$ cobc -m say.cob
$ export COB_LIBRARY_PATH=.
$ ./hello
Hello World!
```

2.3.4 Static linking with C programs

Let's call the following C function from COBOL:

```
---- say.c ------------------------------
int
say(char *hello, char *world)
{
  int i;
  for (i = 0; i < 6; i++)
    putchar(hello[i]);
  for (i = 0; i < 6; i++)
    putchar(world[i]);
  putchar('\n');
  return 0;
}
-----------------------------------------
```

This program is equivalent to the foregoing '`say.cob`'.

Note that, unlike C, the arguments passed from COBOL programs are not terminated by the null character (i.e., `\0`).

You can call this function in the same way you call COBOL programs:

```
---- hello.cob -------------------------
       IDENTIFICATION DIVISION.
       PROGRAM-ID. hello.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 HELLO PIC X(6) VALUE "Hello ".
       01 WORLD PIC X(6) VALUE "World!".
       PROCEDURE DIVISION.
       CALL "say" USING HELLO WORLD.
       STOP RUN.
-----------------------------------------
```

Compile these programs as follows:

```
$ cc -c say.c
$ cobc -c -static -x hello.cob
$ cobc -x -o hello hello.o say.o
$ ./hello
Hello World!
```

2.3.5 Dynamic linking with C programs

You can create a dynamic-linking module from a C program by passing an option `-shared` to the C compiler:

```
$ cc -shared -o say.so say.c
$ cobc -x hello.cob
$ export COB_LIBRARY_PATH=.
$ ./hello
Hello World!
```

3 Customize

3.1 Customizing Compiler

These settings are effective at compile-time.

Environment variables (default value):

`COB_CC` C compiler ("gcc")

`COB_CFLAGS`
: Flags passed to the C compiler ("-I$(PREFIX)/include")

`COB_LDFLAGS`
: Flags passed to the C compiler ("")

`COB_LIBS` Standard libraries linked with the program ("-L$(PREFIX)/lib -lcob")

`COB_LDADD`
: Additional libraries linked with the program ("")

3.2 Customizing Library

These settings are effective at run-time.

Environment variables (default value):

`COB_LIBRARY_PATH`
: Dynamic-linking module path (".:$(PREFIX)/lib/gnu-cobol")

4 Optimize

4.1 Optimize Options

There are three compiler options for optimization: `-O`, `-Os` and `-O2`. These options enable optimization at both translation (from COBOL to C) and compilation (C to assembly) levels.

Currently, there is no difference between these optimization options at the translation level.

The option `-O`, `-Os` or `-O2` is passed to the C compiler as it is and used for C level optimization.

4.2 Optimize Call

When a CALL statement is executed, the called program is linked at run time. By specifying the compiler option `-fstatic-call`, you can statically link the program at compile time and call it efficiently. (see Section 2.2.1 [Static Linking], page 6)

4.3 Optimize Binary

By default, data items of usage binary or comp are stored in the big-endian form. On those machines whose native byte order is little-endian, this is not quite efficient.

If you prefer, you can store binary items in the native form of your machine. Set the config option `binary-byteorder` to `native` in your config file (see Chapter 3 [Customize], page 12).

In addition, setting the option `binary-size` to `2-4-8` or `1-2-4-8` is more efficient than others.

5 Debug

5.1 Debug Options

The compiler option `-debug` can be used during the development of your programs. It enables all run-time error checking, such as subscript boundary checks and numeric data checks, and displays run-time errors with source locations.

6 System routines

6.1 CBL_OC_GETOPT

CBL_OC_GETOPT realises the quite well-known option parser getopt for GNU Cobol. The usage of this system routine is described by the following example.

```
identification division.
program-id. prog.

data division.
working-storage section.
    78 shortoptions value "jkl".

    01 longoptions.
        05 optionrecord occurs 2 times.
            10 optionname   pic x(25).
            10 has-value    pic 9.
            10 valpoint     pointer value NULL.
            10 return-value pic x(4).

    01 longind     pic 99.
    01 long-only   pic 9 value 1.

    01 return-char pic x(4).
    01 opt-val     pic x(10).

    01 counter     pic 9 value 0.
```

We first need to define the necessary fields for getopt's shortoptions (so), longoptions (lo), longoption index (longind), long-only-option (long-only) and also the fields for return values return-char and opt-val (arbitrary size with trimming, see return codes).
The shortoptions are written down as an alphanumeric field (string with arbitrary size) as follows:

```
"ab:c::d"
```

This means we want getopt to look for shortoptions named a, b, c or d and we demand an option value for b and we are accepting an optional one for c.

The longoptions are defined as a table of records with oname, has-value, valpoint and val. The field oname defines the name of a longoption, has-value defines if an option value is demanded(has-val = 1), optional(2) or not required(0).
The longoption structure is immutable! You can vary the amount of records only. The pointer valpoint is used to specify an address to save getopt's return value to. The pointer is optional. If it is NULL, getopt returns a value as usual. If you use the pointer it has to point to a PIC X(4) field.

The field val is a PIC X(4) character which is returned if the longoption was recognized.

Now we have the tools to run CBL_OC_GETOPT within the procedure division.

```
procedure division.
    move "version" to optionname(1).
    move 0 to has-value(1).
    move "v" to return-value(1).

    move "verbose" to optionname(2).
    move 0 to has-value(2).
    move "V" to return-value(2).

    perform with test before until counter > 5
        call 'CBL_OC_GETOPT' using
        by reference shortoptions longoptions longind
        by value long-only
        by reference return-char opt-val
        end-call

        display return-char end-display
        display opt-val end-display

        add 1 to counter end-add
    end-perform
    stop run.
```

The example shows how we initialize all parameters and call the routine. We call getopt 6 times as we have 5 options to recognize and one additional call just to see that getopt returns '-1' in this case.

The return-char might contain the following:

- regular character if an option was recognized
- '?' if we have got an undefined option
- '1' if got a non-option
- '0' if valpoint != NULL and we are writing the return value to the specified address
- '-1' if we don't have any more options

The return-codes of CBL_OC_GETOPT are:

- '1' if we've got a non-option
- '0' if valpoint != NULL and we are writing the return value to the specified address
- '-1' if we don't have any more options
- '2' if we have got an truncated option value in opt-val (because opt-val was too small)
- '3' if we got a regular answer from getopt

Appendix A cobc --help

```
Usage: cobc [options] file ...

Options:
  -help                 Display this message
  -version, -V          Display compiler version
  -info, -i             Display compiler build information
  -v                    Display the commands invoked by the compiler
  -x                    Build an executable program
  -m                    Build a dynamically loadable module (default)
  -std=<dialect>        Warnings/features for a specific dialect :
                          cobol2002   Cobol 2002
                          cobol85     Cobol 85
                          ibm         IBM Compatible
                          mvs         MVS Compatible
                          bs2000      BS2000 Compatible
                          mf          Micro Focus Compatible
                          default     When not specified
                        See config/default.conf and config/*.conf
  -free                 Use free source format
  -fixed                Use fixed source format (default)
  -O, -O2, -Os          Enable optimization
  -g                    Enable C compiler debug / stack check / trace
  -debug                Enable all run-time error checking
  -o <file>             Place the output into <file>
  -b                    Combine all input files into a single
                        dynamically loadable module
  -E                    Preprocess only; do not compile or link
  -C                    Translation only; convert COBOL to C
  -S                    Compile only; output assembly file
  -c                    Compile and assemble, but do not link
  -P(=<dir or file>)    Generate preprocessed program listing (.lst)
  -Xref                 Generate cross reference through 'cobxref'
                        (V. Coen's 'cobxref' must be in path)
  -I <directory>        Add <directory> to copy/include search path
  -L <directory>        Add <directory> to library search path
  -l <lib>              Link the library <lib>
  -A <options>          Add <options> to the C compile phase
  -Q <options>          Add <options> to the C link phase
  -D <define>           DEFINE <define> to the COBOL compiler
  -K <entry>            Generate CALL to <entry> as static
  -conf=<file>          User defined dialect configuration - See -std=
  -list-reserved        Display reserved words
  -list-intrinsics      Display intrinsic functions
  -list-mnemonics       Display mnemonic names
  -list-system          Display system routines
  -save-temps(=<dir>)   Save intermediate files
                        - Default : current directory
  -ext <extension>      Add default file extension

  -W                    Enable ALL warnings
```

```
-Wall                  Enable all warnings except as noted below
-Wobsolete             Warn if obsolete features are used
-Warchaic              Warn if archaic features are used
-Wredefinition         Warn incompatible redefinition of data items
-Wconstant             Warn inconsistent constant
-Woverlap              Warn overlapping MOVE items
-Wparentheses          Warn lack of parentheses around AND within OR
-Wstrict-typing        Warn type mismatch strictly
-Wimplicit-define      Warn implicitly defined data items
-Wcorresponding        Warn CORRESPONDING with no matching items
-Wexternal-value       Warn EXTERNAL item with VALUE clause
-Wcall-params          Warn non 01/77 items for CALL params
                       - NOT set with -Wall
-Wcolumn-overflow      Warn text after column 72, FIXED format
                       - NOT set with -Wall
-Wterminator           Warn lack of scope terminator END-XXX
                       - NOT set with -Wall
-Wtruncate             Warn possible field truncation
                       - NOT set with -Wall
-Wlinkage              Warn dangling LINKAGE items
                       - NOT set with -Wall
-Wunreachable          Warn unreachable statements
                       - NOT set with -Wall

-fsign=<value>         Define display sign representation
                       - ASCII or EBCDIC (Default : machine native)
-ffold-copy=<value>    Fold COPY subject to value
                       - UPPER or LOWER (Default : no transformation)
-ffold-call=<value>    Fold PROGRAM-ID, CALL, CANCEL subject to value
                       - UPPER or LOWER (Default : no transformation)
-fdefaultbyte=<value> Initialize fields without VALUE to decimal value
                       - 0 to 255 (Default : initialize to picture)
-fintrinsics=<value>   Intrinsics to be used without FUNCTION keyword
                       - ALL or intrinsic function name (,name,...)
-ftrace                Generate trace code
                       - Executed SECTION/PARAGRAPH
-ftraceall             Generate trace code
                       - Executed SECTION/PARAGRAPH/STATEMENTS
                       - Turned on by -debug
-fsyntax-only          Syntax error checking only; don't emit any output
-fdebugging-line       Enable debugging lines
                       - 'D' in indicator column or floating >>D
-fsource-location      Generate source location code
                       - Turned on by -debug/-g/-ftraceall
-fimplicit-init        Automatic initialization of the Cobol runtime system
-fstack-check          PERFORM stack checking
                       - Turned on by -debug or -g
-fsyntax-extension     Allow syntax extensions
                       - eg. Switch name SW1, etc.
-fwrite-after          Use AFTER 1 for WRITE of LINE SEQUENTIAL
                       - Default : BEFORE 1
-fmfcomment            '*' or '/' in column 1 treated as comment
```

```
                       - FIXED format only
  -fnotrunc            Allow numeric field overflow
                       - Non-ANSI behaviour
  -fodoslide           Adjust items following OCCURS DEPENDING
                       - Requires implicit/explicit relaxed syntax
  -fsingle-quote       Use a single quote (apostrophe) for QUOTE
                       - Default : double quote
  -frecursive-check    Check recursive program call
  -frelax-syntax       Relax syntax checking
                       - eg. REDEFINES position
  -foptional-file      Treat all files as OPTIONAL
                       - unless NOT OPTIONAL specified
```

Appendix B cobc --info

```
cobc (GNU Cobol) 2.0.0
Copyright (C) 2001,2002,2003,2004,2005,2006,2007 Keisuke Nishida
Copyright (C) 2006-2012 Roger While
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Built     Jan 26 2012 12:30:55
Packaged  Nov 26 2011 16:18:14 UTC
C version "4.3.2 [gcc-4_3-branch revision 141291]"
Build information
Build environment        : x86_64-unknown-linux-gnu
CC                       : gcc -std=gnu99
CPPFLAGS                 :
CFLAGS                   : -g -O2 -pipe -finline-functions -fsigned-char
                           -Wall -Wwrite-strings -Wmissing-prototypes
                           -Wno-format-y2k
LD                       : /usr/x86_64-suse-linux/bin/ld -m elf_x86_64
LDFLAGS                  : -Wl,-z,relro,-z,now

GNU Cobol information
COB_CC                   : gcc -std=gnu99
COB_CFLAGS               : -I/usr/local/include -pipe
COB_LDFLAGS              :
COB_LIBS                 : -L/usr/local/lib -lcob -lm -lgmp -lncursesw
                           -ldb
COB_CONFIG_DIR           : /usr/local/share/gnu-cobol/config
COB_COPY_DIR             : /usr/local/share/gnu-cobol/copy
COB_LIBRARY_PATH         : /usr/local/lib/gnu-cobol
COB_MODULE_EXT           : so
COB_EXEEXT               :
Dynamic loading          : System
"CBL_" param check       : Disabled
Variable format          : 0
BINARY-C-LONG            : 8 bytes
Sequential handler       : Internal
ISAM handler             : BDB
```

Appendix C cobc --list-reserved

Reserved Words	Implemented (Y/N)
ACCEPT	Y
ACCESS	Y
ACTIVE-CLASS	N
ADD	Y
ADDRESS	Y
ADVANCING	Y
AFTER	Y
ALIGNED	N
ALL	Y
ALLOCATE	Y
ALPHABET	Y
ALPHABETIC	Y
ALPHABETIC-LOWER	Y
ALPHABETIC-UPPER	Y
ALPHANUMERIC	Y
ALPHANUMERIC-EDITED	Y
ALSO	Y
ALTER	Y
ALTERNATE	Y
AND	Y
ANY	Y
ANYCASE	N
ARE	Y
AREA	Y
AREAS	Y
ARGUMENT-NUMBER	Y
ARGUMENT-VALUE	Y
ARITHMETIC	N (Context sensitive)
AS	Y
ASCENDING	Y
ASCII	Y (Context sensitive)
ASSIGN	Y
AT	Y
ATTRIBUTE	Y (Context sensitive)
AUTO	Y
AUTO-SKIP	Y
AUTOMATIC	Y
AUTOTERMINATE	Y
AWAY-FROM-ZERO	Y (Context sensitive)
B-AND	N
B-NOT	N
B-OR	N
B-XOR	N
BACKGROUND-COLOR	Y
BACKGROUND-COLOUR	Y
BASED	Y
BEEP	Y

```
BEFORE                          Y
BELL                            Y
BINARY                          Y
BINARY-C-LONG                   Y
BINARY-CHAR                     Y
BINARY-DOUBLE                   Y
BINARY-INT                      Y
BINARY-LONG                     Y
BINARY-LONG-LONG                Y
BINARY-SHORT                    Y
BIT                             N
BLANK                           Y
BLINK                           Y
BLOCK                           Y
BOOLEAN                         N
BOTTOM                          Y
BY                              Y
BYTE-LENGTH                     Y (Context sensitive)
CALL                            Y
CANCEL                          Y
CAPACITY                        N (Context sensitive)
CD                              N (85 obsolete)
CENTER                          N (Context sensitive)
CF                              Y
CH                              Y
CHAIN                           N
CHAINING                        Y
CHARACTER                       Y
CHARACTERS                      Y
CLASS                           Y
CLASS-ID                        N
CLASSIFICATION                  Y (Context sensitive)
CLOSE                           Y
CODE                            Y
CODE-SET                        Y
COL                             Y
COLLATING                       Y
COLS                            Y
COLUMN                          Y
COLUMNS                         Y
COMMA                           Y
COMMAND-LINE                    Y
COMMIT                          Y
COMMON                          Y
COMMUNICATION                   N (85 obsolete)
COMP                            Y
COMP-1                          Y
COMP-2                          Y
COMP-3                          Y
COMP-4                          Y
COMP-5                          Y
COMP-6                          Y
```

```
COMP-X                          Y
COMPUTATIONAL                   Y
COMPUTATIONAL-1                 Y
COMPUTATIONAL-2                 Y
COMPUTATIONAL-3                 Y
COMPUTATIONAL-4                 Y
COMPUTATIONAL-5                 Y
COMPUTATIONAL-X                 Y
COMPUTE                         Y
CONDITION                       Y
CONFIGURATION                   Y
CONSTANT                        Y
CONTAINS                        Y
CONTENT                         Y
CONTINUE                        Y
CONTROL                         Y
CONTROLS                        Y
CONVERSION                      Y (Context sensitive)
CONVERTING                      Y
COPY                            Y
CORR                            Y
CORRESPONDING                   Y
COUNT                           Y
CRT                             Y
CRT-UNDER                       Y
CURRENCY                        Y
CURSOR                          Y
CYCLE                           Y (Context sensitive)
DATA                            Y
DATA-POINTER                    N
DATE                            Y
DAY                             Y
DAY-OF-WEEK                     Y
DE                              Y
DEBUGGING                       Y
DECIMAL-POINT                   Y
DECLARATIVES                    Y
DEFAULT                         Y
DELETE                          Y
DELIMITED                       Y
DELIMITER                       Y
DEPENDING                       Y
DESCENDING                      Y
DESTINATION                     N
DETAIL                          Y
DISABLE                         N
DISC                            Y (Context sensitive)
DISK                            Y (Context sensitive)
DISPLAY                         Y
DIVIDE                          Y
DIVISION                        Y
DOWN                            Y
```

```
DUPLICATES                      Y
DYNAMIC                         Y
EBCDIC                          Y (Context sensitive)
EC                              Y
EGI                             N (85 obsolete)
ELSE                            Y
EMI                             N (85 obsolete)
EMPTY-CHECK                     Y
ENABLE                          N (85 obsolete)
END                             Y
END-ACCEPT                      Y
END-ADD                         Y
END-CALL                        Y
END-CHAIN                       N
END-COMPUTE                     Y
END-DELETE                      Y
END-DISPLAY                     Y
END-DIVIDE                      Y
END-EVALUATE                    Y
END-IF                          Y
END-MULTIPLY                    Y
END-OF-PAGE                     Y
END-PERFORM                     Y
END-READ                        Y
END-RECEIVE                     N (85 obsolete)
END-RETURN                      Y
END-REWRITE                     Y
END-SEARCH                      Y
END-START                       Y
END-STRING                      Y
END-SUBTRACT                    Y
END-UNSTRING                    Y
END-WRITE                       Y
ENTRY                           Y
ENTRY-CONVENTION                N (Context sensitive)
ENVIRONMENT                     Y
ENVIRONMENT-NAME                Y
ENVIRONMENT-VALUE               Y
EO                              N
EOL                             Y (Context sensitive)
EOP                             Y
EOS                             Y (Context sensitive)
EQUAL                           Y
EQUALS                          Y
ERASE                           Y
ERROR                           Y
ESCAPE                          Y
ESI                             N (85 obsolete)
EVALUATE                        Y
EXCEPTION                       Y
EXCEPTION-OBJECT                N
EXCLUSIVE                       Y
```

```
EXIT                            Y
EXPANDS                         N (Context sensitive)
EXTEND                          Y
EXTERNAL                        Y
FACTORY                         N
FALSE                           Y
FD                              Y
FILE                            Y
FILE-CONTROL                    Y
FILE-ID                         Y
FILLER                          Y
FINAL                           Y
FIRST                           Y
FLOAT-BINARY-128                N
FLOAT-BINARY-32                 N
FLOAT-BINARY-64                 N
FLOAT-DECIMAL-16                Y
FLOAT-DECIMAL-34                Y
FLOAT-EXTENDED                  N
FLOAT-INFINITY                  N
FLOAT-LONG                      Y
FLOAT-NOT-A-NUMBER              N (Context sensitive)
FLOAT-SHORT                     Y
FOOTING                         Y
FOR                             Y
FOREGROUND-COLOR                Y
FOREGROUND-COLOUR               Y
FOREVER                         Y
FORMAT                          N
FREE                            Y
FROM                            Y
FULL                            Y
FUNCTION                        Y
FUNCTION-ID                     Y
FUNCTION-POINTER                N
GENERATE                        Y
GET                             N
GIVING                          Y
GLOBAL                          Y
GO                              Y
GOBACK                          Y
GREATER                         Y
GROUP                           Y
GROUP-USAGE                     N
HEADING                         Y
HIGH-VALUE                      Y
HIGH-VALUES                     Y
HIGHLIGHT                       Y
I-O                             Y
I-O-CONTROL                     Y
ID                              Y
IDENTIFICATION                  Y
```

```
IF                              Y
IGNORE                          Y
IGNORING                        Y
IMPLEMENTS                      N (Context sensitive)
IN                              Y
INDEX                           Y
INDEXED                         Y
INDICATE                        Y
INDIRECT                        N (Context sensitive)
INHERITS                        N
INITIAL                         Y
INITIALISE                      Y
INITIALISED                     Y
INITIALIZE                      Y
INITIALIZED                     Y
INITIATE                        Y
INPUT                           Y
INPUT-OUTPUT                    Y
INSPECT                         Y
INTERFACE                       N
INTERFACE-ID                    N
INTERMEDIATE                    N (Context sensitive)
INTO                            Y
INTRINSIC                       Y (Context sensitive)
INVALID                         Y
INVOKE                          N
IS                              Y
JUST                            Y
JUSTIFIED                       Y
KEPT                            Y
KEY                             Y
KEYBOARD                        Y (Context sensitive)
LABEL                           Y
LAST                            Y
LC_ALL                          N (Context sensitive)
LC_COLLATE                      N (Context sensitive)
LC_CTYPE                        N (Context sensitive)
LC_MESSAGES                     N (Context sensitive)
LC_MONETARY                     N (Context sensitive)
LC_NUMERIC                      N (Context sensitive)
LC_TIME                         N (Context sensitive)
LEADING                         Y
LEFT                            Y
LEFT-JUSTIFY                    N
LEFTLINE                        Y
LENGTH                          Y
LENGTH-CHECK                    Y
LESS                            Y
LIMIT                           Y
LIMITS                          Y
LINAGE                          Y
LINAGE-COUNTER                  Y
```

```
LINE                          Y
LINE-COUNTER                  Y
LINES                         Y
LINKAGE                       Y
LOCAL-STORAGE                 Y
LOCALE                        Y
LOCK                          Y
LOW-VALUE                     Y
LOW-VALUES                    Y
LOWER                         Y (Context sensitive)
LOWLIGHT                      Y
MANUAL                        Y
MEMORY                        Y
MERGE                         Y
MESSAGE                       N (85 obsolete)
METHOD                        N
METHOD-ID                     N
MINUS                         Y
MODE                          Y
MOVE                          Y
MULTIPLE                      Y
MULTIPLY                      Y
NAME                          Y (Context sensitive)
NATIONAL                      Y
NATIONAL-EDITED               Y
NATIVE                        Y
NEAREST-AWAY-FROM-ZERO        Y (Context sensitive)
NEAREST-EVEN                  Y (Context sensitive)
NEAREST-TOWARD-ZERO           Y (Context sensitive)
NEGATIVE                      Y
NESTED                        N
NEXT                          Y
NO                            Y
NO-ECHO                       Y
NONE                          N (Context sensitive)
NORMAL                        Y (Context sensitive)
NOT                           Y
NULL                          Y
NULLS                         Y
NUMBER                        Y
NUMBERS                       Y
NUMERIC                       Y
NUMERIC-EDITED                Y
OBJECT                        N
OBJECT-COMPUTER               Y
OBJECT-REFERENCE              N
OCCURS                        Y
OF                            Y
OFF                           Y
OMITTED                       Y
ON                            Y
ONLY                          Y
```

```
OPEN                          Y
OPTIONAL                      Y
OPTIONS                       N
OR                            Y
ORDER                         Y
ORGANISATION                  Y
ORGANIZATION                  Y
OTHER                         Y
OUTPUT                        Y
OVERFLOW                      Y
OVERLINE                      Y
OVERRIDE                      N
PACKED-DECIMAL                Y
PADDING                       Y
PAGE                          Y
PAGE-COUNTER                  Y
PARAGRAPH                     Y (Context sensitive)
PERFORM                       Y
PF                            Y
PH                            Y
PIC                           Y
PICTURE                       Y
PLUS                          Y
POINTER                       Y
POSITION                      Y
POSITIVE                      Y
PREFIXED                      N (Context sensitive)
PRESENT                       Y
PREVIOUS                      Y
PRINTER                       Y (Context sensitive)
PRINTING                      Y
PROCEDURE                     Y
PROCEDURE-POINTER             Y
PROCEDURES                    Y
PROCEED                       Y
PROGRAM                       Y
PROGRAM-ID                    Y
PROGRAM-POINTER               Y
PROHIBITED                    Y (Context sensitive)
PROMPT                        Y
PROPERTY                      N
PROTOTYPE                     N
PURGE                         N (85 obsolete)
QUEUE                         N (85 obsolete)
QUOTE                         Y
QUOTES                        Y
RAISE                         N
RAISING                       N
RANDOM                        Y
RD                            Y
READ                          Y
RECEIVE                       N (85 obsolete)
```

```
RECORD                          Y
RECORDING                       Y
RECORDS                         Y
RECURSIVE                       Y (Context sensitive)
REDEFINES                       Y
REEL                            Y
REFERENCE                       Y
REFERENCES                      Y
RELATION                        N (Context sensitive)
RELATIVE                        Y
RELEASE                         Y
REMAINDER                       Y
REMOVAL                         Y
RENAMES                         Y
REPLACE                         Y
REPLACING                       Y
REPORT                          Y
REPORTING                       Y
REPORTS                         Y
REPOSITORY                      Y
REQUIRED                        Y
RESERVE                         Y
RESET                           Y
RESUME                          N
RETRY                           N
RETURN                          Y
RETURNING                       Y
REVERSE-VIDEO                   Y
REVERSED                        Y
REWIND                          Y
REWRITE                         Y
RF                              Y
RH                              Y
RIGHT                           Y
RIGHT-JUSTIFY                   N
ROLLBACK                        Y
ROUNDED                         Y
ROUNDING                        N (Context sensitive)
RUN                             Y
SAME                            Y
SCREEN                          Y
SCROLL                          Y (Context sensitive)
SD                              Y
SEARCH                          Y
SECONDS                         N (Context sensitive)
SECTION                         Y
SECURE                          Y
SEGMENT                         N (85 obsolete)
SEGMENT-LIMIT                   Y
SELECT                          Y
SELF                            N
SEND                            N (85 obsolete)
```

```
SENTENCE                        Y
SEPARATE                        Y
SEQUENCE                        Y
SEQUENTIAL                      Y
SET                             Y
SHARING                         Y
SIGN                            Y
SIGNED                          Y
SIGNED-INT                      Y
SIGNED-LONG                     Y
SIGNED-SHORT                    Y
SIZE                            Y
SORT                            Y
SORT-MERGE                      Y
SOURCE                          Y
SOURCE-COMPUTER                 Y
SOURCES                         N
SPACE                           Y
SPACE-FILL                      N
SPACES                          Y
SPECIAL-NAMES                   Y
STANDARD                        Y
STANDARD-1                      Y
STANDARD-2                      Y
STANDARD-BINARY                 N (Context sensitive)
STANDARD-DECIMAL                N (Context sensitive)
START                           Y
STATEMENT                       N (Context sensitive)
STATIC                          Y (Context sensitive)
STATUS                          Y
STDCALL                         Y (Context sensitive)
STEP                            Y
STOP                            Y
STRING                          Y
STRONG                          N (Context sensitive)
SUB-QUEUE-1                     N (85 obsolete)
SUB-QUEUE-2                     N (85 obsolete)
SUB-QUEUE-3                     N (85 obsolete)
SUBTRACT                        Y
SUM                             Y
SUPER                           N
SUPPRESS                        Y
SYMBOL                          N (Context sensitive)
SYMBOLIC                        Y
SYNC                            Y
SYNCHRONISED                    Y
SYNCHRONIZED                    Y
SYSTEM-DEFAULT                  Y
TABLE                           N
TALLYING                        Y
TAPE                            Y (Context sensitive)
TERMINAL                        N (85 obsolete)
```

```
TERMINATE                       Y
TEST                            Y
TEXT                            N (85 obsolete)
THAN                            Y
THEN                            Y
THROUGH                         Y
THRU                            Y
TIME                            Y
TIME-OUT                        Y (Context sensitive)
TIMEOUT                         Y (Context sensitive)
TIMES                           Y
TO                              Y
TOP                             Y
TOWARD-GREATER                  Y (Context sensitive)
TOWARD-LESSER                   Y (Context sensitive)
TRAILING                        Y
TRAILING-SIGN                   N
TRANSFORM                       Y
TRUE                            Y
TRUNCATION                      Y (Context sensitive)
TYPE                            Y
TYPEDEF                         N
UCS-4                           N (Context sensitive)
UNDERLINE                       Y
UNIT                            Y
UNIVERSAL                       N
UNLOCK                          Y
UNSIGNED                        Y
UNSIGNED-INT                    Y
UNSIGNED-LONG                   Y
UNSIGNED-SHORT                  Y
UNSTRING                        Y
UNTIL                           Y
UP                              Y
UPDATE                          Y
UPON                            Y
UPPER                           Y (Context sensitive)
USAGE                           Y
USE                             Y
USER                            Y (Context sensitive)
USER-DEFAULT                    Y
USING                           Y
UTF-16                          N (Context sensitive)
UTF-8                           N (Context sensitive)
VAL-STATUS                      N
VALID                           N
VALIDATE                        N
VALIDATE-STATUS                 N
VALUE                           Y
VALUES                          Y
VARYING                         Y
WAIT                            Y
```

```
WHEN                            Y
WITH                            Y
WORDS                           Y
WORKING-STORAGE                 Y
WRITE                           Y
YYYYDDD                         Y (Context sensitive)
YYYYMMDD                        Y (Context sensitive)
ZERO                            Y
ZERO-FILL                       N
ZEROES                          Y
ZEROS                           Y

Extra (obsolete) context sensitive words
AUTHOR
DATE-COMPILED
DATE-MODIFIED
DATE-WRITTEN
INSTALLATION
REMARKS
SECURITY

Extra internal registers        Definition
RETURN-CODE                     USAGE BINARY-LONG
SORT-RETURN                     USAGE BINARY-LONG
NUMBER-OF-CALL-PARAMETERS       USAGE BINARY-LONG
COB-CRT-STATUS                  PIC 9(4)
'LENGTH OF' phrase              USAGE BINARY-LONG
```

Appendix D cobc --list-intrinsics

Intrinsic Function	Implemented	Parameters
ABS	Y	1
ACOS	Y	1
ANNUITY	Y	2
ASIN	Y	1
ATAN	Y	1
BOOLEAN-OF-INTEGER	N	2
BYTE-LENGTH	Y	1
CHAR	Y	1
CHAR-NATIONAL	N	1
COMBINED-DATETIME	Y	2
CONCATENATE	Y	Variable
COS	Y	1
CURRENCY-SYMBOL	Y	0
CURRENT-DATE	Y	0
DATE-OF-INTEGER	Y	1
DATE-TO-YYYYMMDD	Y	Variable
DAY-OF-INTEGER	Y	1
DAY-TO-YYYYDDD	Y	Variable
DISPLAY-OF	N	Variable
E	Y	0
EXCEPTION-FILE	Y	0
EXCEPTION-FILE-N	N	0
EXCEPTION-LOCATION	Y	0
EXCEPTION-LOCATION-N	N	0
EXCEPTION-STATEMENT	Y	0
EXCEPTION-STATUS	Y	0
EXP	Y	1
EXP10	Y	1
FACTORIAL	Y	1
FORMATTED-CURRENT-DATE	N	1
FORMATTED-DATE	N	2
FORMATTED-DATETIME	N	Variable
FORMATTED-TIME	N	Variable
FRACTION-PART	Y	1
HIGHEST-ALGEBRAIC	Y	1
INTEGER	Y	1
INTEGER-OF-BOOLEAN	N	1
INTEGER-OF-DATE	Y	1
INTEGER-OF-DAY	Y	1
INTEGER-OF-FORMATTED-DATE	N	2
INTEGER-PART	Y	1
LENGTH	Y	1
LENGTH-AN	Y	1
LOCALE-COMPARE	Y	Variable
LOCALE-DATE	Y	2
LOCALE-TIME	Y	2
LOCALE-TIME-FROM-SECONDS	Y	2
LOG	Y	1

LOG10	Y	1
LOWER-CASE	Y	1
LOWEST-ALGEBRAIC	Y	1
MAX	Y	Variable
MEAN	Y	Variable
MEDIAN	Y	Variable
MIDRANGE	Y	Variable
MIN	Y	Variable
MOD	Y	2
MODULE-CALLER-ID	Y	0
MODULE-DATE	Y	0
MODULE-FORMATTED-DATE	Y	0
MODULE-ID	Y	0
MODULE-PATH	Y	0
MODULE-SOURCE	Y	0
MODULE-TIME	Y	0
MONETARY-DECIMAL-POINT	Y	0
MONETARY-THOUSANDS-SEPARATOR	Y	0
NATIONAL-OF	N	Variable
NUMERIC-DECIMAL-POINT	Y	0
NUMERIC-THOUSANDS-SEPARATOR	Y	0
NUMVAL	Y	1
NUMVAL-C	Y	2
NUMVAL-F	Y	1
ORD	Y	1
ORD-MAX	Y	Variable
ORD-MIN	Y	Variable
PI	Y	0
PRESENT-VALUE	Y	Variable
RANDOM	Y	Variable
RANGE	Y	Variable
REM	Y	2
REVERSE	Y	1
SECONDS-FROM-FORMATTED-TIME	Y	2
SECONDS-PAST-MIDNIGHT	Y	0
SIGN	Y	1
SIN	Y	1
SQRT	Y	1
STANDARD-COMPARE	N	Variable
STANDARD-DEVIATION	Y	Variable
STORED-CHAR-LENGTH	Y	1
SUBSTITUTE	Y	Variable
SUBSTITUTE-CASE	Y	Variable
SUM	Y	Variable
TAN	Y	1
TEST-DATE-YYYYMMDD	Y	1
TEST-DAY-YYYYDDD	Y	1
TEST-FORMATTED-DATETIME	N	2
TEST-NUMVAL	Y	1
TEST-NUMVAL-C	Y	2
TEST-NUMVAL-F	Y	1
TRIM	Y	2

```
UPPER-CASE                    Y               1
VARIANCE                      Y               Variable
WHEN-COMPILED                 Y               0
YEAR-TO-YYYY                  Y               Variable
```

Appendix E cobc --list-system

System routine	Parameters
SYSTEM	1
CBL_AND	3
CBL_CHANGE_DIR	1
CBL_CHECK_FILE_EXIST	2
CBL_CLOSE_FILE	1
CBL_COPY_FILE	2
CBL_CREATE_DIR	1
CBL_CREATE_FILE	5
CBL_DELETE_DIR	1
CBL_DELETE_FILE	1
CBL_EQ	3
CBL_ERROR_PROC	2
CBL_EXIT_PROC	2
CBL_FLUSH_FILE	1
CBL_GET_CSR_POS	1
CBL_GET_CURRENT_DIR	3
CBL_GET_SCR_SIZE	2
CBL_IMP	3
CBL_NIMP	3
CBL_NOR	3
CBL_NOT	2
CBL_OC_NANOSLEEP	1
CBL_OPEN_FILE	5
CBL_OR	3
CBL_READ_FILE	5
CBL_RENAME_FILE	2
CBL_TOLOWER	2
CBL_TOUPPER	2
CBL_WRITE_FILE	5
CBL_XOR	3
C$CALLEDBY	1
C$CHDIR	2
C$COPY	3
C$DELETE	2
C$FILEINFO	2
C$GETPID	0
C$JUSTIFY	1
C$MAKEDIR	1
C$NARG	1
C$PARAMSIZE	1
C$PRINTABLE	1
C$SLEEP	1
C$TOLOWER	2
C$TOUPPER	2
X"91"	2
X"E4"	0
X"E5"	0

```
X"F4"                          2
X"F5"                          2
```

Appendix F cobc --list-mnemonics

```
Mnemonic names
SYSIN           Device name
SYSIPT          Device name
STDIN           Device name
SYSOUT          Device name
SYSLIST         Device name
SYSLST          Device name
STDOUT          Device name
PRINTER         Device name
SYSERR          Device name
STDERR          Device name
CONSOLE         Device name
C01             Feature name
C02             Feature name
C03             Feature name
C04             Feature name
C05             Feature name
C06             Feature name
C07             Feature name
C08             Feature name
C09             Feature name
C10             Feature name
C11             Feature name
C12             Feature name
CSP             Feature name
FORMFEED        Feature name
CALL-CONVENTION Feature name
SWITCH-0        Switch name
SWITCH-1        Switch name
SWITCH-2        Switch name
SWITCH-3        Switch name
SWITCH-4        Switch name
SWITCH-5        Switch name
SWITCH-6        Switch name
SWITCH-7        Switch name
SWITCH-8        Switch name
SWITCH-9        Switch name
SWITCH-10       Switch name
SWITCH-11       Switch name
SWITCH-12       Switch name
SWITCH-13       Switch name
SWITCH-14       Switch name
SWITCH-15       Switch name

Extended mnemonic names (with -fsyntax-extension)
SW0             Switch name
SW1             Switch name
SW2             Switch name
SW3             Switch name
```

```
SW4             Switch name
SW5             Switch name
SW6             Switch name
SW7             Switch name
SW8             Switch name
SW9             Switch name
SW10            Switch name
SW11            Switch name
SW12            Switch name
SW13            Switch name
SW14            Switch name
SW15            Switch name
```

Appendix G config/default.conf

```
# GNU Cobol compiler configuration
#
# Copyright (C) 2001,2002,2003,2004,2005,2006,2007 Keisuke Nishida
# Copyright (C) 2007-2012 Roger While
#
# This file is part of GNU Cobol.
#
# The GNU Cobol compiler is free software: you can redistribute it
# and/or modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation, either version 3 of the
# License, or (at your option) any later version.
#
# GNU Cobol 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.
#
# You should have received a copy of the GNU General Public License
# along with GNU Cobol.  If not, see <http://www.gnu.org/licenses/>.


# Value: any string
name: "GNU Cobol"

# Value: enum
standard-define                 0
#        CB_STD_OC = 0,
#        CB_STD_MF,
#        CB_STD_IBM,
#        CB_STD_MVS,
#        CB_STD_BS2000,
#        CB_STD_85,
#        CB_STD_2002

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

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

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

```
#
filename-mapping:               yes

# Alternate formatting of numeric fields
pretty-display:                 yes

# Allow complex OCCURS DEPENDING ON
complex-odo:                    no

# Allow REDEFINES to other than last equal level number
indirect-redefines:             no

# Binary byte size - defines the allocated bytes according to PIC
# Value:         signed  unsigned  bytes
#                ------  --------  -----
# '2-4-8'         1 -  4    same        2
#                 5 -  9    same        4
#                10 - 18    same        8
#
# '1-2-4-8'       1 -  2    same        1
#                 3 -  4    same        2
#                 5 -  9    same        4
#                10 - 18    same        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

# Numeric truncation according to ANSI
binary-truncate:                yes

# Binary byte order
# Value: 'native', 'big-endian'
binary-byteorder:               big-endian

# Allow larger REDEFINES items
larger-redefines-ok:            no

# Allow certain syntax variations (eg. REDEFINES position)
relaxed-syntax-check:           no

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

```
# If yes, linkage-section items remain allocated
# between invocations.
sticky-linkage:                 no

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

# If yes, allow reserved words from the 85 standard
cobol85-reserved:               no

# Allow Hex 'F' for NUMERIC test of signed PACKED DECIMAL field
hostsign:                       no

# not-reserved:
# Value: Word to be taken out of the reserved words list
# (case independent)
# Words that are in the (proposed) standard but may conflict

# Dialect features
# Value: 'ok', 'archaic', 'obsolete', 'skip', 'ignore', 'unconformable'

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