Chapter

# 61

# *SDL C Compiler Driver (SCCD)*

**The SDL C Compiler Driver (SCCD) is a utility intended to simplify C level debugging of code generated by the SDL suite and hand-written code. It can be invoked from the makefiles used by the SDL suite or stand-alone from the OS command line.**

# Introduction

The SDL C Compiler Driver (SCCD) is intended to simplify C level debugging by generating an intermediate C file placed in a user defined directory. This C file has all its macros expanded and is optionally "beautified" (pretty-printed). For ease of use SCCD is used as a C compiler driver called from the SDL suite generated or handwritten makefiles. SCCD may also be used as a command line option when compiling C code.

The simplest way to introduce this facility for use in the standard SDL suite environment is to modify the `makeoptions` or `make.opt` file associated with the predefined run-time library to be used. To enable this feature in a modified the SDL suite run-time library, the directory and file structure must be similar to the pre-defined one.

The `makeoptions` or `make.opt` file is found in the `$sdtdir/SCT[Kernel name]` directory. The only modification **required** is to change the line

```
sctCC = cc      /* or some other compiler executable name */
```

to

```
sctCC = sccd cc     /* or some other compiler executable name */
```

SCCD must of course be visible in your path.

To customize the behavior of SCCD, modify the configuration file `sccd_<your_compiler_type>.cfg` variables described in the following sections.

# Syntax for Invoking

The following syntax can be used to invoke SCCD:

To execute SCCD:

```
sccd <C compiler command line>
```

The C compiler command line should be the normal command line used to compile a C file: it should include the name of your C compiler, possible compiler options, and finally the name of the C file.

To print the values of the variables in the configuration file `sccd.cfg`:

```
sccd
```

To print the variables with extra help information:

```
sccd -h
```

# Return Codes

0: Success.

1: After printing “help”.

2: No `.c` file.

3: Could not open `InFile.c`.

4: Could not open `InFile.i` for write.

5: `sccdMOVE` or `sccdOUTFILEREDIR` needs to be defined.

6: Could not open `InFile.i` for read.

7: Could not open/create `TmpDir/InFile.c`.

# Actions Performed by SCCD

1. Execute an optional first-user defined command (`sccdUSER_CMD1`).
2. Create a sub-directory for temporary files and the pre-processed `.c` files.
3. Execute an optional second user-defined command (`sccdUSER_CMD2`).
4. Run a C pre-processor pass to expand all macros.
5. Execute an optional third user-defined command (`sccdUSER_CMD3`).
6. Pretty print the file.
7. Optional clean-up of the sub-directory.
8. Optionally copy `.hs` files to the sub-directory.
9. Execute an optional fourth user-defined command (`sccdUSER_CMD4`). You should use this command if you wish to invoke the "indent" utility from SCCD (see "C Beautifier" on page 2947 for more information).
10. Optionally compile, i.e. run the original command line.
11. Optional clean-up of the sub-directory, but leave the pre-processed `.c` file(s) for debugging purposes.

# Configuration File

The behavior of SCCD is defined using a number of variables, each starting with "sccd". The variables are defined in a configuration file, `sccd_<your_compiler_type>.cfg`. Select the configuration file that corresponds to your C compiler and copy this file as `sccd.cfg`. If run from the SDL suite, SCCD uses `$sctdir/sccd.cfg` as the configuration file; otherwise SCCD searches for `sccd.cfg` in the current directory, `$SCCD`, and **(on UNIX)** `$HOME`.

> **Note:**
>
> If `sccd.cfg` is not found, hard-coded defaults suitable for the Gnu C compiler (gcc) are used.

Below are the variables that control the behavior of SCCD. Note that all characters in variable values are significant, including white spaces.

`sccdNAME = "Default"`

Compiler name as defined in `scttypes.h`.

`sccdINFILESUFFIX = ".c"`

The expected file name suffix of the In-file(s), Default ".c".

`sccdCPP = ""`

The name of the C pre-processor. If this is left empty, CPP is used. Default "".

`sccdCPPFLAGS = ""`

Enable CPP and do not remove comments. This is C compiler dependent. Default for gcc is "-P -E -C", and for cc "-C -P".

`sccdMACROPREFIX = "-D"`

CPP command-line define MACRO prefix. Default "-D".

`sccdINCLUDE1 = "-I"`

CPP command-line include-path prefix. Default "-I".

`sccdINCLUDE2 = ""`

Alternative CPP command-line include-path prefix. Default "".

`sccdOUTFILEREDIR = "-o "`

Character sequence to control CPP output file name. If empty, use `sccdFMOVE` instead.

`sccdFMOVE = ""`

OS forced file move or copy command. Used instead of `sccdOUTFILEREDIR`. Default: "".

`sccdDELETE = "rm -f"`

OS forced delete file command. Default: "rm -f".

`sccdCOPY = "cp"`

OS normal copy command. Default: "cp".

`sccdCOMPILE = "ON"`

Controls whether the final compilation pass should be run or not. Values are: "OFF" and default is "ON".

`sccdDEBUG = "OFF"`

Enable execution. Values are: “ON” and default is “OFF”.

`sccdPURGE = "ON"`

Purge temporary files. Values are: “OFF” and default is “ON”.

`sccdUSE_HS = "OFF"`

When set “ON”, the `.hs` files are not included until the compilation pass. Values are: “ON” and default is “OFF”.

`sccdSILENT = "OFF"`

Enable trace printout. Values are: “ON” and default is “OFF”.

`sccdTMPDIR = "sccdtmp"`

Temporary directory for the pre-processing. Default is `sccdtmp`. Setting `sccdTMPDIR` to “” or “.” in the configuration file suppresses temporary directory creation.

```
sccdUSER_CMD1 = ""
sccdUSER_CMD2 = ""
sccdUSER_CMD3 = ""
sccdUSER_CMD4 = ""
```

User-defined commands (see “Actions Performed by SCCD” on page 2944). The following pseudo variables can be used in all but the first one (`sccdUSER_CMD1`):

- `%f` expands to In-file name without extension.
- `%p` expands to In-file path.
- `%d` expands to the value of `sccdTMPDIR`.

Example: `echo \"Pre-processed C-file = %p/%d/%f.c\"`

To include ‘#’ in `sccdUSER_CMDx` and `sccdTMPDIR`, enter **`\#`**
To include ‘"’ in `sccdUSER_CMDx` and `sccdTMPDIR`, enter **`\"`**
To include ‘\’ in `sccdUSER_CMDx` and `sccdTMPDIR`, enter **`\\`**

# C Beautifier

If you need a C beautifier to further format the C code from the SDL suite, do as follows.

Try the `indent` utility (courtesy of Joseph Arcaneaux). The `.zip` file contains source code and an executable compiled as a win32 application; the `.tar` file contains the original indent-1.9.1 source code for UNIX. The UNIX version must be compiled for the appropriate platform but this work is normally simplified if you read the documentation in the `.tar` file and use the enclosed make file.

**The indent executable must be placed in your path.**

In order to easily invoke indent from SCCD, insert the following statement in `sccdUSER_CMD4` in the appropriate `sccd.cfg` file(s), assuming no other changes have been made to the `sccd.cfg` file(s):

For UNIX compiler environments:

```
sccdUSER_CMD4 = "indent -kr -l70 -i2 %p/%d/%f.c"
```

For “DOS”-like compiler environments:

```
sccdUSER_CMD4 = "indent -kr -l70 -i2 %p\\%d\\%f.c"
```

This setup gives a `.c` source file formatted according to rules very like the ones used in the Kernighan & Richie “C” book. It will also try to force lines to be shorter than 70 characters and will use 2 positions indentation in if/while/... statements.

A slightly more elaborate example:

```
sccdUSER_CMD4 = "indent -kr -l70 -br -nce -nlp -ci3
-i2 %p/%d/%f.c"
```

# Chapter 61 SDL C Compiler Driver (SCCD)