ICS 212
Jason Favrod

An Introduction to SCCS

The Original Version Control System

What is SCCS?

SCCS stands for *S*ource *C*ode *C*ontrol *S*ystem; it is a suite of programs that are used together to manage the development of source code which is eventually compiled into usable software. SCCS can be used to manage other classes of text files (not only source code), but the reason it was created and its predominate use is to mange changes made to source code files.

What is meant by development/changes?

Source code is generally recorded in plain (non-formatted) text files called source files and collectively *the source code* for a particular piece of software. The source code itself must then be transformed from text into the binary that a CPU understands. This transformation is done via compilation, as is the case with C and Java, or it is translated as is true of PHP and Python source code.

If a programmer wants to fix, add features, or otherwise improve the functionality of her program she would open her source code with a standard text editor, make the desired changes, and then compile (or run through a translator) the source code. Once every thing is working correctly she can then release a new version of the program with the improvement. This is essentially software development in a nut shell.

Okay, so where does SCCS come in?

As stated, SCCS is used to mange the process described in the last section. The most simple case of software development is a single programmer maintaining and improving a single program. Again the general procedure would be:

1. Create/Edit a source file.
2. Save edits to that file.
3. Compile source back into the executable program.

If all edits were done in a single sitting, and each edit make was completely correct, then development would happen linearly and management would be similarly straightforward. But making improvements to source code very often requires more than a single editing session. New bugs are regularly found. And making things more complicated, code added as an improvement sometimes introduces new bugs. Because of this the need arises to keep track of changes.

For example, a developer can be working on a program and he makes an couple of improvements. Then he discovers one of the changes made actually crashes the program under certain circumstances. With SCCS, or pretty much any other version control system, the programmer can view a history of the changes made to track down when the bug was introduced. Necessary changes to the source code can then be made to implement a fix, or if required, the entire set of changes that were made can be removed from the source code.

This use of SCCS, or more generally version control systems, is definitely an advantage to programmers working individually as well as in teams. Yet, as with just about anything else, things get more complicated the more people are involved.

Initially SCCS requires that programmers import their source code into the system. To bring these files in, the user must create a directory named SCCS. Each source file becomes an `sfile` in the SCCS directory (i.e. `s.source.c`). After that, working within the system one interacts with these sfiles – and not the source files themselves – when performing maintenance or development. From then on, the program and the programmer(s) use the system keep track of all changes.

To edit a file in the system it needs to be 'checked out.' This places a lock on the file so that no one else can work on the file at the same time. The programmer would then make edits and once they are satisfied with the changes (and the program runs correctly*) the would then `delta`, or merge, their changes back into the sfile. At each delta the programmer writes (or ought to write) a comment on the changes made. (Note that only changes are written to the sfile, a full copy is not made). The deltas are and normalized and serialized. With the deltas, and their accompanying programmer comments, the project is managed.

TLDR;

sfile

The sfile is like a database file. The original source code is imported with the `admin` program and transformed into the sfile format. These files are then stored in a directory named `SCCS`. Changes are recorder at each `merge` with programmer comments and other useful meta data.

editing

To edit a file the programmer uses the `get` program with the `-e` flag, or with the `edit` command. A copy of the requested version of the file is then returned to the programmer. The sfile for the file being edited is then locked for any user other than the programmer who checked the file out. After the desired modifications are made (and the program compiles and runs correctly), the changes must be merged back into the sfile. If one wishes to discard their changes they can use the `unedit` command to back out of any changes made.

merging

To write the changes made to the source code, one must merge the changes with the `delta` program. This writes the changes to the sfile. Upon issue the `sccs delta` command, the programmer will be prompted to comment on the changes; this comment will be appended to meta data containing an `SID` (SCCS ID), or version number which is of the form `r.l` where `r` is the 'release' number and `l` is the 'level' – which I think of as major and minor version number.

compiling

According to Eric Allman of the University of California at Berkeley's "An Introduction to the Source Code Control System" (available at: http://sccs.sourceforge.net/man/sccs.me.html):

> "A good technique is to edit the files you need, make all necessary changes and tests, compiling and editing as often as necessary without making deltas. When you are satisfied that you have a working version, delta everything being edited, re-get them, and recompile everything."

To `get` a file for compilation use the `sccs get [file]` command. This returns a read only copy of the current version of the `file` that can be used for compilation. To grab all the source files use the `sccs get SCCS/` command in the directory containing your SCCS directory.

Examples

```
$> ls -F
addrBookUI*  bld/         inc/         makefile     README       src/
```

Example project: address book program.

`bld`	`object files`
`inc`	`headers`
`src`	`source files`

Initializing the SCCS

To initialize the SCCS system (using bash):

1. Create an SCCS directory.
2. Import source files into sfiles.

```
$> cd src
$> ls
actions.c     addrBookUI.c  helpers.c
$> mkdir SCCS save
$> for x in $(ls);do
> sccs admin -i$x $x
> mv $x save/
> done
No id keywords (cm7)
No id keywords (cm7)
No id keywords (cm7)
$>
```

The output, `No id keywords (cm7)` is related to the consent of ID keywords, which I won't cover, but this is just a warning (generated for each source file imported into the system). After issuing the above set of commands our source code has been transformed into sfiles (i.e. `s.helpers.c`) and can be found in the `SCCS` directory. The original `.c` files were moved to save for archiving or disposal.

```
$> ls
save  SCCS
$> ls SCCS
s.actions.c     s.addrBookUI.c  s.helpers.c
```

Editing

To work on a source file, it must be checked out of the system with the `get` program using the `-e` option, or using the `edit` command.

```
$> sccs get -e helpers.c
1.1
new delta 1.2
88 lines
$> sccs edit actions.c
1.1
new delta 1.2
396 lines
$> ls
actions.c  helpers.c  save       SCCS
```

As you can see the two source files have been made available for editing. To back out of editing use the `sccs unedit [file]`.

Merging a file

After edits have been made, they need to be merged into the project (recorded in the sfile).

```
$> sccs delta actions.c
comments? Moved a comment
No id keywords (cm7)
1.2
2 inserted
1 deleted
395 unchanged
```

Compilation

Make sure all edits have been make and recoreded in the sfile.

For individual files

```
$> sccs get helpers.c
1.2
88 lines
No id keywords (cm7)
$> gcc -c helpers.c
$> sccs clean
```

For the entire source code

```
$> sccs get SCCS

SCCS/s.actions.c:
1.2
397 lines
No id keywords (cm7)

SCCS/s.helpers.c:
1.2
88 lines
No id keywords (cm7)

SCCS/s.addrBookUI.c:
1.1
263 lines
No id keywords (cm7)
$> make addrBookUI clean
---output omitted--
$> sccs clean
```

File status

The follow command can be use to tell which files are being edited and by which users.

```
$> sccs info
Nothing being edited
$> sccs info -ujason
Nothing being edited by jason
$> sccs get -e helpers.c
$> sccs check
   helpers.c: being edited: 1.2 1.3 jason 14/09/27 19:10:47
```

The `prt` program is use to get information on what changes have been made to a particular file.

```
$> sccs prt helpers.c

SCCS/s.helpers.c:

D 1.2   14/09/27 20:40:13 jason        2 1     00000/00000/00088


D 1.1   14/09/27 19:50:38 jaosn        1 0     00088/00000/00000
date and time created 14/09/27 19:50:38 by jason
```

`get` with the `-m` and -p options to get a detailed history of when changes were made.

```
$> sccs get -m -p helpers.c
1.2
1.1
/*********************************************************************
1.1     //
1.1     // AUTHOR:      Jason Favrod
1.1     //
1.1     // DATE:        17 Sept 2014
1.1     //
1.1     // FILE:        helpers.c
1.1     //
1.1     // DESCRIPTION: A collection of helper functions.
1.1     //
1.2     // New Comment
1.1
**********************************************************************/
1.1     #include "helpers.h"
1.1
1.1
1.1
/********************************************************************
---output omitted---
```