Function and Use of an SCCS Interface Program

L. E. Bonanni
A. Guyton (4/1/80 revision)

Bell Laboratories
Piscataway, New Jersey 08854

ABSTRACT

This memorandum discusses the use of a Source Code Control System Interface Program to allow more than one user to use SCCS commands upon the same set of files.

1. INTRODUCTION

In order to permit UNIX† users with different user identification numbers (user IDs) to use SCCS commands upon the same files, an SCCS interface program is provided to temporarily grant the necessary file access permissions to these users. This memorandum discusses the creation and use of such an interface program. This memorandum replaces an earlier version dated March 1, 1978.

2. FUNCTION

When only one user uses SCCS, the real and effective user IDs are the same, and that user ID owns the directories containing SCCS files. However, there are situations (for example, in large software development projects) in which it is practical to allow more than one user to make changes to the same set of SCCS files. In these cases, one user must be chosen as the **owner** of the SCCS files and be the one who will **administer** them (e.g., by using the *admin* command). This user is termed the *SCCS administrator* for that project. Since other users of SCCS do not have the same privileges and permissions as the SCCS administrator, they are not able to execute directly those commands that require write permission in the directory containing the SCCS files. Therefore, a project-dependent program is required to provide an interface to the *get*, *delta*, and, if desired, *rmdel*, *cdc*, and *unget* commands.[1]

The interface program must be owned by the SCCS administrator, must be executable by non-owners, and must have the *set user ID on execution* bit on (see *chmod*(1)[2]), so that, when executed, the *effective* user ID is the user ID of the administrator. This program's function is to invoke the desired SCCS command and to cause it to *inherit* the privileges of the SCCS administrator for the duration of that command's execution. In this manner, the owner of an SCCS file (the administrator) can modify it at will. Other users whose *login* names are in the *user list*[3] for that file (but who are *not* its owners) are given the necessary permissions only for the duration of the execution of the interface program, and are thus able to modify the SCCS files only through the use of *delta* and, possibly, *rmdel* and *cdc*.

† UNIX is a trademark of Bell Laboratories.

[1] Other SCCS commands either do not require write permission in the directory containing SCCS files or are (generally) reserved for use only by the administrator.

[2] All references of the form *name*(*N*) refer to item *name* in section *N* of the *UNIX User's Manual*.

[3] This is the list of login names of users who are allowed to modify an SCCS file by adding or removing deltas. The login names are specified using the *admin*(1) command.

3. A BASIC PROGRAM

When a UNIX program is executed it is passed (as argument 0) the *name* by which it is invoked, followed by any additional user-supplied arguments. Thus, if a program is given a number of *links* (names), it may alter its processing depending upon which link is used to invoke it. This mechanism is used by an SCCS interface program to determine which SCCS command it should subsequently invoke (see *exec*(2)).

A generic interface program (**inter.c**, written in C) is shown in *Attachment I*. Note the reference to the (unsupplied) function **filearg**. This is intended to demonstrate that the interface program may also be used as a pre-processor to SCCS commands. For example, function **filearg** could be used to modify file arguments to be passed to the SCCS command by supplying the *full* path name of a file, thus avoiding extraneous typing by the user. Also, the program could supply any additional (default) keyletter arguments desired.

4. LINKING AND USE

In general, the following demonstrates the steps to be performed by the SCCS administrator to create the SCCS interface program. It is assumed, for the purposes of the discussion, that the interface program **inter.c** resides in directory **/x1/xyz/sccs**. Thus, the command sequence:

```
cd /x1/xyz/sccs
cc ... inter.c −o inter ...
```

compiles **inter.c** to produce the executable module **inter** (**...** represents arguments that may also be required). The proper mode and the *set user ID on execution* bit are set by executing:

```
chmod 4755 inter
```

Finally, new links are created, by (for example):[4]

```
ln inter get
ln inter delta
ln inter rmdel
```

Subsequently, *any* user whose shell parameter PATH (see *sh*(1)) specifies that directory **/x1/xyz/sccs** is to be searched first for executable commands, may execute, for example:

```
get −e /x1/xyz/sccs/s.abc
```

from any directory to invoke the interface program (via its link *get*). The interface program then executes **/usr/bin/get** (the actual SCCS *get* command) upon the named file. As previously mentioned, the interface program could be used to supply the pathname **/x1/xyz/sccs**, so that the user would only have to specify:

```
get −e s.abc
```

to achieve the same results.

5. CONCLUSION

An SCCS interface program is used to permit users having different user IDs to use SCCS commands upon the same files. Although this is its primary purpose, such a program may also be used as a pre-processor to SCCS commands since it can perform operations upon its arguments.

[4] The names of the links may be arbitrary, provided the interface program is able to determine from them the names of SCCS commands to be invoked.

Attachment I

SCCS Interface Program **inter.c**

```
main(argc, argv)
int argc;
char *argv[];
{
        register int i;
        char cmdstr[LENGTH]

        /*
        Process file arguments (those that don't begin with "-").
        */
        for (i = 1; i < argc; i++)
                if (argv[i][0] != '-')
                        argv[i] = filearg(argv[i]);

        /*
        Get "simple name" of name used to invoke this program
        (i.e., strip off directory-name prefix, if any).
        */
        argv[0] = sname(argv[0]);

        /*
        Invoke actual SCCS command, passing arguments.
        */
        sprintf(cmdstr, "/usr/bin/%s", argv[0]);
        execv(cmdstr, argv);
}
```