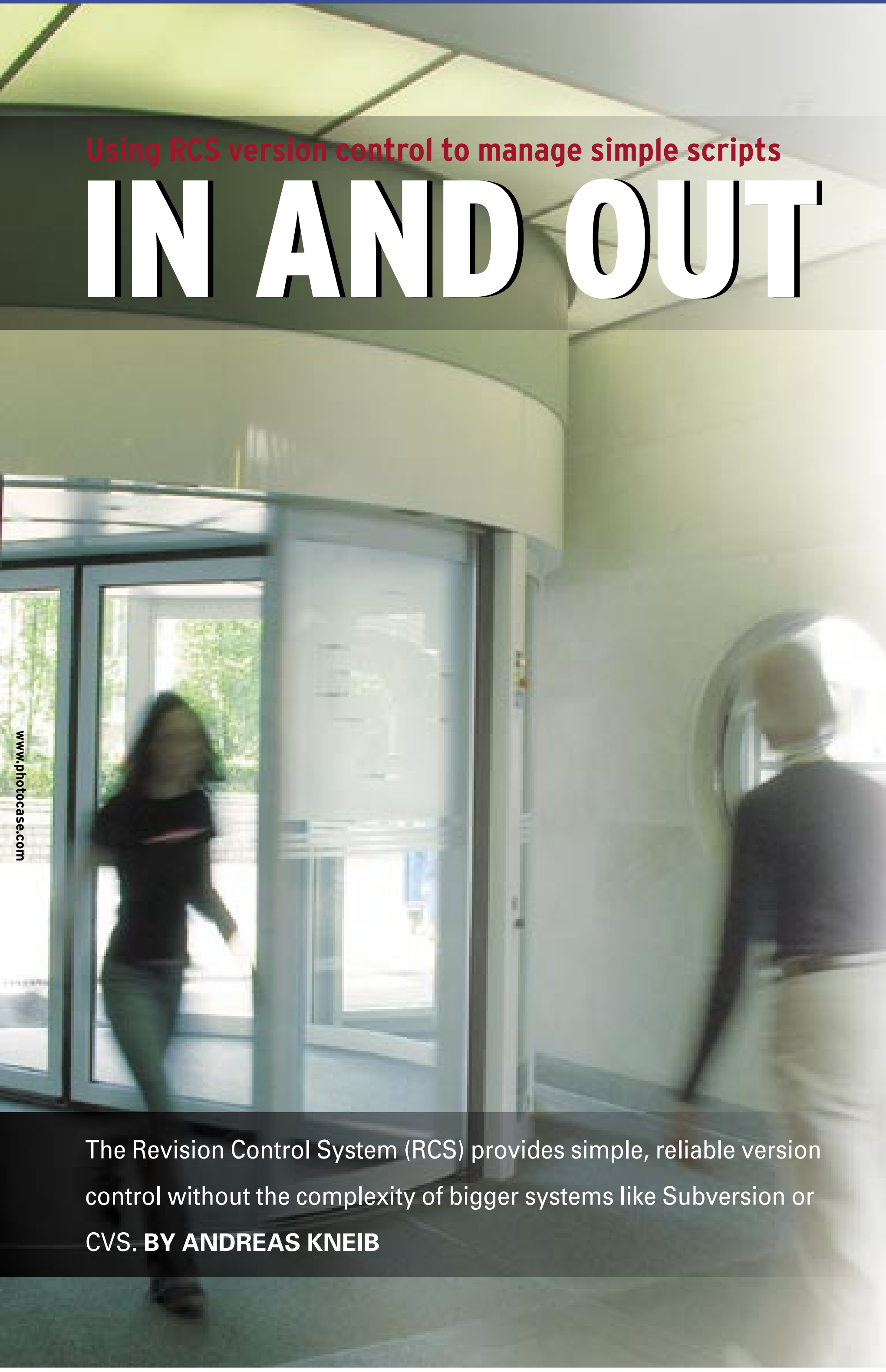

Using RCS version control to manage simple scripts

IN AND OUT

The Revision Control System (RCS) provides simple, reliable version control without the complexity of bigger systems like Subversion or CVS. **BY ANDREAS KNEIB**

If you are the kind of Linux user with a collection of homegrown scripts, such as Perl snippets for downloading web comics or Shell scripts for backing up files, you may have considered adopting some form of version management tracking. For a single user, it doesn't always make sense to use one of the popular, but complex, tools, such as CVS [2] or Subversion [3]. If you're looking for a simpler level of version control, you may want to try the Revision Control System (RCS).

RCS is a well tested and stable tool with excellent support. If your distribution does not come with RCS, you can either download it from the GNU project homepage [1], or get it direct from the official RCS homepage [4]. The *INSTALL* and *INSTALL.RCS* files from the unpacked tarball, *rcs-5.7.tar.Z*, provide notes for installing from the source files.

Preparation

Let's assume you need to manage your do-it-yourself scripts in the *~/bin* directory; in this case, you would start by running *mkdir ~/bin/RCS* to create a working directory for RCS. This is the repository that RCS uses to store a copy of the files it is managing, along with the *,v* extension.

The version in the RCS repository is the original file, with a few add-ons. When you make changes to the file, you will actually be working on a copy.

To show you how RCS works, I'll start by creating a sample script. First I'll launch the cat tool and create a script as a so-called **here document**.

```
[akneib]~ > cat > ⮐
~/bin/world.sh <<__EOF__⮐
#!/bin/sh echo "Hello World!" ⮐
__EOF__
```

Checking In and Out

To allow the Revision Control System to accept the sample file, you need to pass the file to the program. The *ci* (for check in) command handles this. After giving the command, the version management tool prompts you to describe the file or add some notes. Type a dot in a blank line to complete your description. This completes the check-in process, and RCS assigns version number 1.1 to your sample script:

```
[akneib]~/bin > ci world.sh
RCS/world.sh,v  <--  world.sh
enter description, ⮐
terminated with single ⮐
'.' or end of file:
NOTE: This is NOT ⮐
the log message!
>> Description of file
>> .
initial revision: 1.1
done
```

Running the *ls ~/bin* command reveals that the *world.sh* file has disappeared from the directory. The file has now been converted and is stored as a revision file below *~/bin/RCS/world.sh,v*(Listing 1).

This behavior may not be what you expected, as the script has been removed

GLOSSARY

Here document: A here document uses a special kind of redirection to extract a longer text block from a code block.

from its previous path and no longer works. You can either check the file out or use the *-u* (for unlocked) option when checking the file in. The *ci -u world.sh* command creates a non-editable copy below *~/bin* when you check in a file.

To avoid being prompted to supply a comment, you can pass the quoted comment by setting the *-m* flag when checking in a file. For example, *co -u -m"Description" world.sh* would check in a file, automatically add a comment, and then check out a read-only copy.

The *-u* option tells RCS not to lock the file. This option is quite useful if multiple users are working on a single file. A version control system not only supports change tracking for the file, it also remove the danger of different users overwriting each other's changes.

To open the script in your editor and make some changes to it, set the *-l* (for locked) option when checking the script out; this option prevents access by other users, even though the other users might actually have access privileges for the file. The following command handles this:

```
[akneib]~/bin > ↩
co -l world.sh
RCS/world.sh,v  --> ↩
 world.sh
revision 1.1 (locked)
done
```

This gives you exclusive write access to *world.sh*, as the *ls -l world.sh* command then reveals.

You can then go on to open the script in your favorite editor and add a comment line. I will be using the interactive ed editor in these examples. ed is available as a variant of the more popular vim editor on many systems, and although using ed may seem a bit strange at first, ed is a great tool for quick changes to text files.

```
[akneib]~/bin > ↩
ed world.sh
28
.
echo "Hello World"
i
# This is a comment line
.
wq
59
```

The tool outputs the number of lines in the file at the start and at the end. The dot tells ed to output the current line. At the same time, ed quits the input, which you start by pressing the [i] key.

After making these changes to the file, check the file in once more. RCS will ask you to supply a note to describe the changes you made. You can quit writing the note by typing a single dot in a line.

```
[akneib]~/bin > ↩
ci -u world.sh
RCS/world.sh,v ↩
 <--  world.sh
new revision: 1.2; ↩
previous revision: 1.1
enter log message, ↩
terminated with ↩
single '.' ↩
or end of file:
>> Added comment line
>> .
done
```

This whole process ups the version number for *world.sh* from 1.1 to 1.2. But what happens if you do not like the new version and would prefer to revert to revision number 1.1 of your script? If you decide you want to revert to an earlier version, you need the check out command's *-r* option to specify the revision you would like to check out:

```
[akneib]~/bin > ↩
co -r1.1 world.sh
RCS/world.sh,v  --> ↩
 world.sh
revision 1.1
done
```

The following cat output demonstrates that you again have the original version of the script without the addition of the comment line:

```
[akneib]~/bin > cat world.sh
#!/bin/sh
echo "Hello World!"
```

However, typing *co -r1.2 world.sh* would check out version 1.2 from the repository.

Versions

All of this checking in and out may be fine, but you may be missing the control aspect of version control. The rlog and rcsdiff commands give you control over the revisions. Typing *rlog world.sh* outputs details of the program, such as the description and the authors of the various revisions, along with the various annotations (Listing 2).

Rcsdiff tells you the differences between versions 1.1 and 1.2 of your script. Just like with the *co* command, the *-r* option defines the revision numbers that rcsdiff will compare, as in the following:

```
[akneib]~/bin > ↩
rcsdiff -r1.1 -r 1.2 world.sh
===================
RCS file: RCS/world.sh,v
retrieving revision 1.1
retrieving revision 1.2
diff -r1.1 -r1.2
1a2
> # This is a comment line
```

In addition to this, RCS supports a number of variables; one of the most common variables adds author and status data to a file. Start by checking out the file in question using the command we looked at earlier: *co -l world.sh*. Then change the line that says *# This is a comment line* by inserting the RCS variable *$Id$* after the hash. To keep things simple, I will be using the ed editor for this again:

```
[akneib]~/bin > ed world.sh
59
2
# This is a comment line
c
# $Id$
.
wq
35
```

After saving the file, give the *ci -u world.sh* to check it back in. A quick glance at the script tells us that the version management tool has expanded the *$Id$* variable:

```
[akneib]~/bin > cat world.sh
#!/bin/sh
# $Id: world.sh,v 1.3 2006/02/06
10:05:59 akneib Exp $
echo "Hello World!"
```

The line gives you details such as the filename, the revision number, the date and time, and the author of the file. The check out manpage, which you can open by running the *man co* command, provides additional information on the various options available with this command.

Conclusions

The Revision Control System is a great tool for managing short scripts or configuration files. RCS provides safety and security for complex system management tasks, yet the RCS command set includes only a few short and easy-to-use commands. ■

Listing 1: Revision file

```
head    1.1;
access;
symbols;
locks; strict;
comment @# @;

1.1
date    2006.02.01.14.16.10;
   author akneib;  state Exp;
branches;
next    ;

desc
@Description of file
@

1.1
log
@Initial revision
@
text
@#!/bin/sh
echo "Hello World!"
@
```

Listing 2: Rlog output

```
RCS file: RCS/world.sh,v
Working file: world.sh
head: 1.2
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 2;
   selected revisions: 2
description:
Description of file
----------------------------
revision 1.2
date: 2006/02/06 07:38:50;
   author: akneib;  state: Exp;
   lines: +1 -0
Added comment line
----------------------------
revision 1.1
date: 2006/02/01 14:16:10;
   author: akneib;  state: Exp;
Initial revision
==============================
==============================
```

INFO

[1] GNU RCS project homepage: *http://www.gnu.org/software/rcs/rcs.html*

[2] Concurrent Versions System (CVS): *http://www.nongnu.org/cvs/*

[3] Subversion project homepage: *http://subversion.tigris.org*

[4] RCS homepage: *http://www.cs.purdue.edu/homes/trinkle/RCS/*