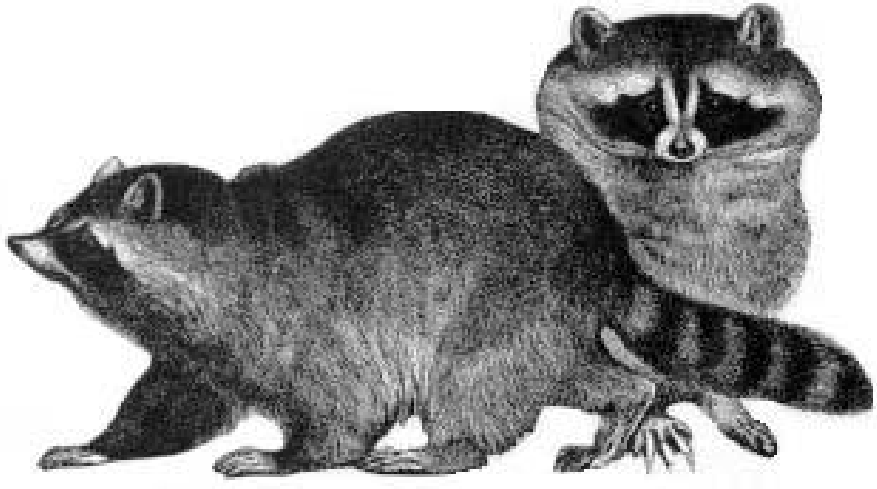
RCS
Revision Control System

Peter Arbenz

arbenz@inf.ethz.ch

Institute of Computational Science

ETH Zürich



What is RCS good for?

- RCS manages multiple revisions of files.

What is RCS good for?

- RCS manages multiple revisions of files.
- RCS is useful for text that is revised frequently, including source code, programs, documentation, graphics, papers, and form letters.

What is RCS good for?

- RCS manages multiple revisions of files.
- RCS is useful for text that is revised frequently, including source code, programs, documentation, graphics, papers, and form letters.
- RCS automates the storing, retrieval, logging, identification, and merging of revisions.

What is RCS good for?

- RCS manages multiple revisions of files.
- RCS is useful for text that is revised frequently, including source code, programs, documentation, graphics, papers, and form letters.
- RCS automates the storing, retrieval, logging, identification, and merging of revisions.
- RCS was first developed by Walter Tichy at Purdue University in the early 1980s. RCS design was an improvement from its predecessor Source Code Control System (SCCS). RCS is the basis of the more recent Concurrent Version System (CVS).

Why use RCS:

- Besides keeping track of what changes were made to a file, RCS tracks who made the change, and when.

Why use RCS:

- Besides keeping track of what changes were made to a file, RCS tracks who made the change, and when.
- If more than one person is working on the same file, RCS allows you to lock a file, so that only one person will be allowed to make changes. Other people can still use the file, e.g., for compiling.

Why use RCS:

- Besides keeping track of what changes were made to a file, RCS tracks who made the change, and when.
- If more than one person is working on the same file, RCS allows you to lock a file, so that only one person will be allowed to make changes. Other people can still use the file, e.g., for compiling.
- RCS provides a safety net for the software developer. When developing, fixing, and improving a program, changes are inevitable. By saving a stable version of your file in RCS, you can later return to a known state if a set of changes does not work out.

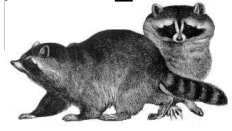
How RCS works:

```
[166] % mkdir rcs_demo
[167] % cd rcs_demo
[168] % mkdir RCS
[169] % cp ~/src/C/stuff/hello.c .
[170] % cat hello.c
#include <stdio.h>

int main()
{
    printf("Hello World\n");
}
```

Checking Files In

```
[171] % ls -o hello.c
-rw-r--r--   1 arbenz   64 Sep 1 09:55 hello.c
[172] % ci hello.c
RCS/hello.c,v  <--  hello.c
enter description, terminated with single '.' o
NOTE: This is NOT the log message!
>> C program that prints a friendly message.
>> .
initial revision: 1.1
done
[173] % ls
RCS/
[174] %
```



Checking Files Out

```
[192] % co hello.c
RCS/hello.c,v  -->  hello.c
revision 1.1
done
[193] % ls -o hello.c
-r--r--r--   1 arbenz   64 Sep 1 10:46 hello.c
[194] % cc hello.c -O -o hello ; ./hello
Hello World
[195] %
```

Locking Files

```
[206] % co -l hello.c
RCS/hello.c,v  -->  hello.c
revision 1.1 (locked)
done
[207] % ls -o hello.c
-rw-r--r--   1 arbenz   64 Sep 1 10:59 hello.c
[208] % emacs hello.c
[209] % cc hello.c -O -o hello ; ./hello
Hello World!
```

```
[210] % ci hello.c
RCS/hello.c,v  <--  hello.c
new revision: 1.2; previous revision: 1.1
enter log message, terminated with single '.' o
>> Added "!"
>> .
done
[211] % co -l hello.c
[212] % ls -o
total 20
drwxr-xr-x   2 arbenz    80 Sep 1 11:17 RCS/
-rwxr-xr-x   1 arbenz 13472 Sep 1 11:01 hello*
-r--r--r--   1 arbenz    65 Sep 1 11:01 hello.c
```

Comparing Versions of a File

```
[326] % rcsdiff -c -r1.1 hello.c
===================================================================
RCS file: RCS/hello.c,v
retrieving revision 1.1
diff -c -r1.1 hello.c
*** hello.c     2003/09/01 10:16:00     1.1
--- hello.c     2003/09/01 10:23:35
***************
*** 2,6 ****

  int main()
  {
!     printf("Hello World\n");
  }
```

```
--- 2,6 ----

  int main()
  {
!     printf("Hello World!\n");
  }
```

In order to save space RCS uses the observation that most of the text will remain unchanged from one revision to the next. It uses the standard UNIX utility `diff` which is used to generate a comparison of which lines are different between two text files. (In general) only the most recent version of the files are stored.

Revision History

The `rlog` program provides information about the archive file and the logs of each revision stored in it. A command like

```
[123] % rlog hello.c
```

will print the version history of the file, each revision's creation date and userids of author and the person who locked the file.

RCS and Emacs

The Version Control facility of emacs(1) works as a front end to RCS.
When editing a file with emacs(1) which is registered with RCS, the command `vc-toggle-read-only` (bound to `C-x C-q` by default) will check a file in to the emacs's Version Control, and then into RCS. Emacs will open a buffer where you can type a log message to be included in the RCS log.

RCS and CVS

- RCS is useful for simple systems. RCS is very simple to setup, with little administrative work. RCS is used in a centralized area where everyone works. Very strong locking of files - concurrency eliminated.

RCS and CVS

- RCS is useful for simple systems. RCS is very simple to setup, with little administrative work. RCS is used in a centralized area where everyone works. Very strong locking of files - concurrency eliminated.
- Concurrent development by multiple developers is not possible due to file locking and being limited to a single working directory. If you are involved in a large project then use the Concurrent Version System (CVS). CVS is a “state-of-the-art” open source version control system layered on top of RCS, designed to manage entire software projects, see `http://www.cvshome.org/`

RCS and CVS

- RCS is useful for simple systems. RCS is very simple to setup, with little administrative work. RCS is used in a centralized area where everyone works. Very strong locking of files - concurrency eliminated.
- Concurrent development by multiple developers is not possible due to file locking and being limited to a single working directory. If you are involved in a large project then use the Concurrent Version System (CVS). CVS is a “state-of-the-art” open source version control system layered on top of RCS, designed to manage entire software projects, see `http://www.cvshome.org/`

More information

Official RCS Homepage
`http://www.cs.purdue.edu/homes/trinkle/RCS/`

Page at the GNU project
`http://www.gnu.org/software/rcs/`

Tutorial:
`http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/rcs/`