Simple Revision Control with RCS

Ferry Boender

Revision History

Revision 1.0 May 24, 2010 Revised by: FB

1. Preface

I do a lot of stuff in text file on my system. I keep notes and todo's in them, I write drafts using a text editor and even complete user guides and other documentation. I also keep my personal planning in Gnome Planner which stores its data in an XML file, which in essence is also text. Every so often I wish I could revert a draft back to an older version, or perhaps just have little peak at how it was a couple of days ago. I also really wanted to be able to look back at my planning to see where I made mistakes and underestimated the time required for a particular task.

A revision control system would be ideal in these circumstances. However a full-featured revision control system such as Subversion or Git would be a little overkill in this situation. Enter our old and forgotten friend RCS. RCS has been around since at least 1982, and has since been deprecated by CVS, Subversion and the various other centralized and decentralized versioning systems. It is however perfectly suited for keeping revision histories of single or a small collection of files. RCS is easier to understand than just about any other revision control system too.

2. Installing

Let's look at how it works. First, we have to install it. It used to be installed by default on most systems, but that's no longer the case. If you're running a Debian-based distribution (such as Ubuntu), you can simply install it by typing:

```
aptitude install rcs
```

We've now got the RCS revision system installed, and we can start using it.

3. Using RCS

Let's create an empty text file, called `simplerevisioncontrolwithrcs.txt` in the `drafts` directory:

```
~$ cd drafts
```

```
~/drafts$ touch simple_revision_control_with_rcs.txt
```

RCS consists of a couple of simple command-line utilities:

- *ci*: Check In RCS revisions. This allows you to save the changes made to a file that is managed by RCS.
- *co*: Check out RCS working copy. This command retrieves a specific version (the latest by default) of a file in a RCS repository.

So let's convert our empty draft to an RCS repository by checking it in. This means RCS will create a file based on the file you want to check in with the appropriate information such as when it has been been changed, what changed, etc.

```
~/drafts/$ ci simple_revision_control_with_rcs.txt
simple_revision_control_with_rcs.txt,v  <--  simple_revision_control_with_rcs.txt
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> Simple revision control with RCS
>> .
initial revision: 1.1
done
```

There. Our initial empty file is gone, and in its place is a file under RCS revision control: `simplerevisioncontrolwithrcs.txt,v`. It has been given a revision number by RCS: revision 1.1.

We can now make a working copy of this file by doing a checkout. This will retrieve the latest modifications from the text file and create a new file with the original name in the current directory. The first thing we should do though, is to turn off locking. We're working on this file ourselves, and it will never be shared, so we don't care about locking at all. Locking can be turned off using the `-U` option of the `rcs` tool:

```
~/drafts$ rcs -U ./simple_revision_control_with_rcs.txt,v
RCS file: ./simple_revision_control_with_rcs.txt,v
done
```

Now we can do a checkout so we can edit our draft. Let's do this, and add a line to it:

```
~/drafts$ co simple_revision_control_with_rcs.txt,v
simple_revision_control_with_rcs.txt,v  -->  simple_revision_control_with_rcs.txt
revision 1.1
done
~/drafts$ echo "Hello world" > simple_revision_control_with_rcs.txt
```

Now let's commit this change to the repository so RCS knows about it:

```
~/drafts$ ci simple_revision_control_with_rcs.txt
simple_revision_control_with_rcs.txt,v  <--  simple_revision_control_with_rcs.txt
```

```
new revision: 1.2; previous revision: 1.1
enter log message, terminated with single '.' or end of file:
>> Added greeting.
>> .
done
```

RCS keeps an entire history of all changes made to the files it keeps under revision each time you do a check in. Take a look at the history of the file with the command `rlog`:

```
~/drafts$ rlog ./simple_revision_control_with_rcs.txt,v
RCS file: ./simple_revision_control_with_rcs.txt,v
Working file: simple_revision_control_with_rcs.txt
head: 1.2
branch:
locks: non-strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 2; selected revisions: 2
description:
Simple revision control with RCS
----------------------------
revision 1.2
date: 2009/06/09 07:48:50;  author: todsah;  state: Exp;  lines: +1 -0
Added greeting.
----------------------------
revision 1.1
date: 2009/06/09 07:46:24;  author: todsah;  state: Exp;
Initial revision
=============================================================================
```

That lists some basic information on the file, the repository and the two revisions we've made so far. We can now easily check out earlier revisions of the file by specifying a specific revision, or a date:

```
~/drafts$ co -r1.1 simple_revision_control_with_rcs.txt
simple_revision_control_with_rcs.txt,v  -->  simple_revision_control_with_rcs.txt
revision 1.1
done
~/drafts$ cat simple_revision_control_with_rcs.txt
~/drafts$
```

RCS has given us back our file as it existed at revision 1.1, which was the empty file. Let's retrieve the version of the file as it was at 14:00 today:

```
~/drafts$ co -d"14:00" simple_revision_control_with_rcs.txt,v
simple_revision_control_with_rcs.txt,v  -->  simple_revision_control_with_rcs.txt
revision 1.2
writable simple_revision_control_with_rcs.txt exists; remove it? [ny](n): y
done
```

4. Conclusion

In conclusion: RCS may appear to be completely dead, but it still has its uses. I know many people out there will think to themselves "Why not just use Git or Subversion"? I understand the attraction of very powerful tools, but I also really believe in keeping things simple, and using the right tool for the job. I don't know much about Git, so perhaps it is ideal for such a simple requirement, but I don't really feel like learning it just to version control a single file.

5. About this document

5.1. Copyright / License