

Lecture 12: Version Control & RCS

Kenneth M. Anderson
Software Methods and Tools
CSCI 3308 - Fall Semester, 2003

Version Control

- Things change...
 - new requirements lead to
 - new or changed designs that lead to
 - new or changed source code, etc.
 - or bugs are discovered and must be fixed
- ...software engineers need to keep track of all these changes

October 6, 2003

© University of Colorado, 2003

2

Version Control, continued

- Versioning
 - Keeping track of the changes to a file from one editing session to the next
 - Computer Science has developed algorithms that can automatically detect the changes to a file
 - especially text files
 - One such algorithm is encapsulated in a Unix tool called `diff`

October 6, 2003

© University of Colorado, 2003

3

DIFF Example

- world1.cc

```
% diff world1.cc world2.cc
2c2
< printf("hello world");
---
> printf("Hello World!");
```
- world2.cc

```
% diff world2.cc world1.cc
2c2
< printf("Hello World!");
---
> printf("hello world");
```

October 6, 2003

© University of Colorado, 2003

4

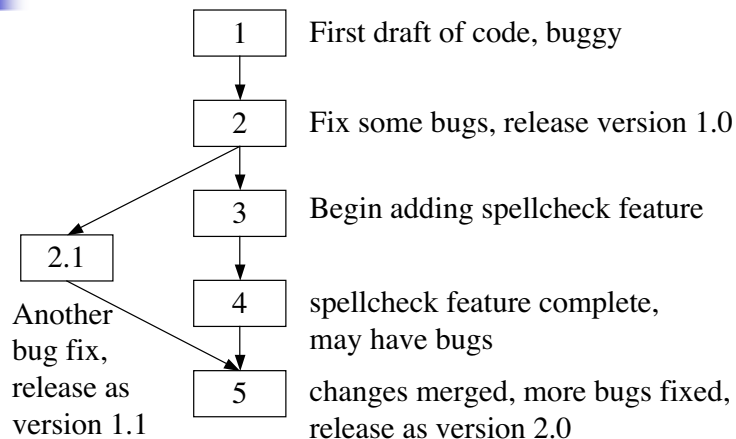
More on DIFF

- Command Line
 - diff file1 file2
 - “show me how to change file1 into file2”
- Output
 - Displays the “ed” commands needed to change file1 into file2
 - “2c2” in the previous example meant change line 2 in file1 into line 2 of file2
- See the diff and ed man pages for complete details

Version Control

- Tracks all changes associated with a file
- Why?
 - *Rollback*: Changes are not always an improvement. Often, you need to return to a prior version
 - *Experimentation*: Lets software engineers explore “what if” scenarios
 - *Internationalization*: Different versions for different languages
 - *Historical Record*: A customer reports a bug on version 1.1, but you are at version 2.3. You need a way to recreate version 1.1 to check out the bug

Tracking Changes



Version Graphs

- All changes to a single file are tracked via a graph
 - nodes are distinct versions
 - edges denote that the target node was derived from the source node
- There are three possible configurations
 - *Extension*: a single version derived from a single version
 - *Split*: multiple versions derived from a single one
 - *Merge*: a single version derived from multiple versions; merge typically done “by hand”



Version Control Files

- Version graphs are stored in version control files
- Space saving techniques are used
 - Forward deltas
 - The original version is stored, all subsequent versions are stored as sets of changes or “deltas”
 - Backward deltas
 - The most recent version is stored, all previous versions are stored as sets of changes



Version Control Systems

- Version Control Systems let you check versions out of a version control file
 - If I ask for version 2.2, and we are using a forward delta system
 - then it starts with the original file and applies all of the deltas that lead to version 2.2 in the version graph



RCS: Revision Control System

- The version control system for our class is RCS.
- RCS is a backward delta system
 - It stores the complete text for the most recent version of the file
 - And derives old versions by applying deltas to the recent version



Check-In and Check-Out

- Commonly used RCS commands
 - ci - check in
 - Used to check changes into a version control file; can also create an initial version control file
 - co - check out
 - Used to check out a specific version of a file



RCS Locks

- RCS has a notion of locks
 - If you check out a file, with a lock, no other person can check out that file for editing (they can still read it, however)
 - This prevents multiple people from changing the same file at once
 - This is a different strategy than one used by another popular version control system, CVS
 - CVS allows a file to be edited by multiple people, but then potentially requires a “merge” step when checking a file into the repository



RCS Version Control File

- The version control file for RCS has a suffix “,v”
- You can think of this file as a tar file
 - A single file with multiple files stored within
 - This time the contained files are just different versions of the same file
 - And all but one are stored as “deltas”



RCS Example

- Start with a file

```
rw- CommentBook
%ci CommentBook
initial revision: 1.1
r-- CommentBook,v
```
- To Retrieve original file

```
%co CommentBook
r-- CommentBook
r-- CommentBook,v
```

Note: the version control file is created in the same directory as the original file UNLESS there is a directory called RCS

If RCS exists, the version control file is created in the RCS directory



RCS Example, continued

- The checked out file is called “the working file”
 - You can always retrieve a working file from the version control file

```
r-- CommentBook
r-- CommentBook,v

%rm CommentBook
r-- CommentBook,v

%co CommentBook
r-- CommentBook
r-- CommentBook,v
```

RCS Example, continued

- A normal co command produces a read-only working file; to get a writeable version, you must lock the working file

```
%co -l CommentBook
rw- CommentBook
r-- CommentBook,v
```
- After you have made changes, you can check them in to make a new version

```
%ci CommentBook
new revision: 1.2
r-- CommentBook,v
```

Flags for CO

- `-r#` - Retrieve revision number #
- `-l` - Retrieve a locked working file
- `-u` - Retrieve an unlocked working file
- `-p` - Print contents of file to stdout, do not retrieve a working file
- `-f` - force overwrite of working file
- `-q` - Quite mode; diagnostics not printed
- `-I` - Interactive Mode
- See man page for more info

Flags for CI

- `-r#` - Save as revision number #
- `-l` - Save and retrieve a new locked working file
- `-u` - Save and retrieve an unlocked working file
- `-i` - Initial check in, report error if a ,v file already exists
- `-f` - Force creation of new version
- `-q` - Quite mode
- `-I` - Interactive Mode
- See man page for more info

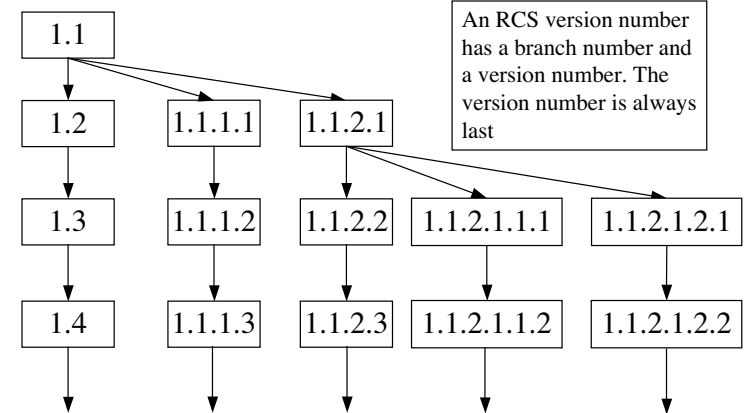
RCS Keywords

- RCS will look for certain keywords in a file and will substitute values for them during check-in and check-out
 - Think of these keywords as Make Macros, where RCS provides the definition for the Macro
 - Keywords are delimited with two dollar signs
 - Example:if `$Author$` appears in a text file stored in RCS
 - then when checked out, RCS replaces the above string with:
 - `$Author: kena $`
 - or more generically with `$Author: <username> $`

Example Keywords

- `$Author$` - name of user who performed ci
- `$Date$` - The date and time of a ci
- `$Locker$` - name of the user who locked a revision
- `Log` - log message supplied during ci
- `$Revision$` - revision number assigned during ci
- `$Name$` - The name of a revision (if named)
- `$RCSfile$` - The name of the RCS file without path info
- `$Source$` - The full pathname of the RCS file
- `$Header$` - See co man page for details
- `Id` - See co man page for details

RCS Version Numbering



Other RCS Tools

- `rccs` - an admin tool; can perform tasks such as breaking locks, change log messages, erase versions, etc. See man page for details.
- `rlog` - display log messages for a particular file (each time you check in a file, you are asked to enter a log message describing the changes)
- `rcsdiff` - a diff command for rcs versions
 - e.g. `rcsdiff -r1.1 -r1.2 CommentBook` will show the differences between the specified versions without checking those versions out