



Development: Revision control

CS 2204

Class meeting 8



Overview of software development process

- Creation of source files (.c, .h, .cpp)
 - RCS, this week
- Compilation (e.g. *.c → *.o) and linking
 - gcc and make, last week
- Running and testing programs
 - gdb debugger, next week



What is revision control?

- A way to ensure that edits on files are
 - Logged / archived
 - Consistent
- Especially applicable when projects include
 - Multiple files
 - Multiple editors (developers)



Why do you need revision control?

- Software development projects are normally done in teams
- Multiple people may be responsible for code in a single file
- You may want to freeze a working version and create a new revision branch
- You may want to roll back development to a previous point in time



Overview of RCS

- Maintains complete revision information for a set of files (not limited to source code)
- Uses major and minor revision numbers (e.g. 1.1, 2.3)
- Supports locking files so that only one user can edit at a time
- Supports merging two edits of the same file
- Can automatically place revision information within the file itself



Key RCS commands

- `rcs` (administer project)
- `ci` (check in file)
- `co` (check out file)
- `rlog` (view the log)
- `rcsdiff` (see changes since last revision)
- For more, see UIAN ch. 19



Basic RCS usage

- Create a subdirectory `RCS` of the directory containing the files in question
- Check in the files to create an initial revision 1.1
- Check out and lock files to edit them
- Check files back in to create a new revision and allow others to edit them



Checking out files

- Assume we start with file `main.c` checked in (`RCS/main.c,v`)
- `co main.c`
 - Creates read-only file `main.c` in `pwd`
- `co -l main.c`
 - Creates writable file `main.c` in `pwd`
 - Locks file so others can't check it out



Checking in files

- Assume we've locked and edited main.c
- `ci main.c`
 - Creates a new revision (e.g. 1.2)
 - Removes main.c from pwd
- `ci -u main.c`
 - Creates a new revision
 - Leaves read-only main.c in pwd
- `ci -l main.c`
 - Creates a new revision (checkpoint)
 - Allows you to keep editing



Setting the revision number

- Perhaps we've done major revisions to revision 1.4 of main.c and we want to start a new branch (2) on the revision tree
- `ci -r2 main.c`




Using keyword substitution

- Each time you check in a new revision, RCS keeps information about the author, date/time, a log message (which you provide), etc.
- You can show this information in files by placing special tags within the files when setting up RCS



Keyword example

```
/*  
* Last revision:  
  $Revision$ by  
  $Author$ on $Date$  
* $Log$  
*/  
  
main()  
{  
    ...  
}
```



```
/*  
* Last revision: $Revision:  
  1.1 $ by $Author: bowman $  
  on $Date: 2001/10/10  
  20:29:27 $  
* $Log: main.c,v $  
* Revision 1.1  2001/10/10  
  20:29:27  bowman  
* Initial revision  
*  
*/  
  
main()  
{  
    ...  
}
```



RCS hints

- Usually use `ci -u file` to keep a read-only copy of *file* for compilation, etc.
- Use `co -l -rR file` to retrieve revision number *R* of *file* instead of most recent revision