

AMath 483/583 — Lecture 2 — March 30, 2011

Today:

- Unix
- Version control — main ideas
- Client-server version control, e.g., CVS, svn
- Distributed version control, e.g., Mercurial, GIT

Friday:

- Mercurial examples

Notes:

Outline of quarter

- Unix
- Version control (Mercurial)
- Compiled vs. interpreted languages
- Fortran 90
- Makefiles
- Parallel computing
- OpenMP
- MPI (message passing interface)
- Python scripting
- Graphics / visualization

Notes:

TA and Office Hours

TA: Grady Lemoine

See the [Class Catalyst Page](#) for contact info, updated hours.

RJL's office hours in Guggenheim 415C:

Monday, Wednesday, Friday 4:30 – 5:30

Grady's office hours in Guggenheim 406:

Monday, Tuesday, Friday 1:30 – 2:30

There is also a Discussion Board on the [Class Catalyst Page](#), feel free to post (**and answer!**) questions about getting things to work.

Notes:

Other references and sources

- Links in notes and [bibliography](#) (more to come...)
- Wikipedia often has good intros and summaries.
- [Software Carpentry](#) course of Greg Wilson, Toronto.
- Other courses at universities or supercomputer centers. See [bibliography](#).
- Textbooks. See [bibliography](#).

Notes:

Unix (and Linux, Mac OS X, etc.)

See the [class notes Unix page](#) for a brief intro and many links.

Unix commands will be introduced as needed and mostly discussed in the context of other things.

Notes:

Version control systems

Originally developed for large software projects with many developers.

Also useful for single user, e.g. to:

- Keep track of history and changes to files,
- Be able to revert to previous versions,
- Keep many different versions of code well organized,
- Easily archive exactly the version used for results in publications,
- Keep work in sync on multiple computers.

Notes:

Server-client model:

Original style, still widely used (e.g. CVS, Subversion)

One **central repository** on server.

Developers' workflow (simplified!):

- Check out a **working copy**,
- Make changes, test and debug,
- Check in (**commit**) changes to repository (with comments). This creates new **version number**.
- Run an **update** on working copy to bring in others' changes.

The system keeps track of **diffs** from one version to the next (and info on who made the changes, when, etc.)

A **changeset** is a collection of **diffs** from one commit.

Notes:

Server-client model:

Only the server has the full history.

The working copy has:

- Latest version from repository (from last **checkout**, **commit**, or **update**)
- Your local changes that are not yet committed.

Note:

- You can retrieve older versions from the server.
- Can only *commit* or *update* when connected to server.
- When you *commit*, it will be seen by anyone else who does an *update* from the repository.

Often there are **trunk** and **branches** subdirectories.

Notes:

Distributed version control

Mercurial (hg) uses a distributed model:

When you **clone** a repository you get **all** the history too, All stored in **.hg** subdirectory of top directory.

Usually don't want to mess with this!

Ex: (backslash is continuation character in shell)

```
$ hg clone \  
  http://bitbucket.org/rjleveque/uwamath583s11 \  
  mydirname
```

will make a complete copy of the class repository and call it **mydirname**. If **mydirname** is omitted, it will be called **uwamath583s11**.

This directory has a subdirectory **.hg** with complete history.

Notes:

Distributed version control

Mercurial (hg) uses a distributed model:

- [hg commit](#) commits to your clone's .hg directory.
- [hg push](#) sends your recent changesets to another clone by default: the one you cloned from (e.g. bitbucket), but you can push to any other clone (with write permission).
- [hg pull](#) pulls changesets from another clone by default: the one you cloned from (e.g. bitbucket)
- [hg update](#) applies changesets to your working copy

Note: pushing, pulling, updating only needed if there are multiple clones.

Friday: simpler example of using hg in a single directory.

Notes:

Distributed version control

Advantages of distributed model:

- You can commit changes, revert to earlier versions, examine history, etc. without being connected to server.
- Also without affecting anyone else's version if you're working collaboratively. Can commit often while debugging.
- No problem if server dies, every clone has full history.

For collaboration will still need to push or pull changes eventually and may need [hg merge](#).

Note: If you use two different clones of your bitbucket repos. (e.g. on different machines), always do [hg pull -u](#) before making local changes or you will have to learn how to merge.

Notes:

Bitbucket

You can examine class repository at:

<http://bitbucket.org/rjleveque/uwamath583s11>

Demo of Source, Changeset tabs...

See also <http://mercurial.selenic.com/wiki/UnderstandingMercurial>

<http://mercurial.selenic.com/wiki/UnderstandingMercurial>

Notes:

Aside on Unix diff

Often very useful (beyond version control).

Displays the differences between two files.

Ex: Go into `$CLASSHG/codes/fortran` and you will see files `demo1.f90` and `demo1.f90`, which has been slightly changed.

```
$ diff demo1.f90 demo2.f90
6c6
< ! Changed one variable name to illustrate 'hg diff'
---
> ! This version has a bug!
11c11
< integer :: m
---
> integer :: n
```

The lines marked `<` are from the first file, those marked `>` are from the second.

The other lines mean:

Lines 6 and 11 were changed as indicated.

Notes:

Aside on Unix diff, xxdiff

For files with many changes, you may want to do:

```
$ diff demo1.f90 demo2.f90 | more
```

The vertical bar means [pipe](#) the output of the first command to the second command. The [more](#) command displays 1 screenfull at a time.

Or try [xxdiff](#), which opens a window displaying the files side by side with changes highlighted.

```
$ xxdiff demo1.f90 demo2.f90
```

Notes:

hg diff command

Now try:

```
$ cd $CLASSHG/codes/fortran
$ hg log demo1.f90 | more
```

Lists all the hg changesets in which file `demo1.f90` was changed.

Note changeset 10:54971910d50a has the log message "Fixed a bug: forgot to change n to m in declaration".

(Number 10: is clone-dependent!)

To see the changes from previous version:

```
$ hg diff -r9 -r10 demo1.f90 | more
```

To see if any changes were made since then:

```
$ hg diff -r10 tip demo1.f90 | more
```

[tip](#) means most recent committed version.

Notes:

hg diff command

To see if any changes were made in working copy compared to tip:

```
$ hg diff demo1.f90 | more
$ hg diff | more      # shows diffs in all files
```

To check status of files in working version:

```
$ hg status           # for entire clone
$ hg status .         # for this directory
$ hg status -amr      # added, modified, removed
$ hg status *.f90     # only for .f90 files

$ hg help status      # for more options
```

Notes:

Using xxdiff in hg

Modify the file `.hg/hgrc`, to add:

```
[extensions]
hgext.extdiff =
```

(Put in `$HOME/.hgrc` to apply in all directories.)

Then you can do:

```
$ hg extdiff -p xxdiff -r9 -r10 demo1.f90
```

Might want to add to `.bashrc`:

```
alias hgd = "hg extdiff -p xxdiff"
```

Then you can do:

```
$ hgd -r9 -r10 demo1.f90
```

Notes: