Basics of UNIX

June 3, 2015

By UNIX, I mean any UNIX-like operating system, including Linux and Mac OS X. On the Mac you can access a UNIX terminal window with the Terminal application (under /Applications/Utilities from the Finder). Most modern scientific computing is done on UNIX-based machines, often by remotely logging in to a UNIX-based server.

If you're using your own Mac, you should install the Xcode developer tools.

1 Connecting to a UNIX machine from {UNIX, Mac, Windows}

See the file *remoteConnect.pdf* in the howtos folder in the repository, https://github.com/berkeley-stat243/stat243-fall-2014/blob/master/howtos/remoteConnect.pdf. While those instructions are focused on connecting remotely to the SCF [only relevant for those of you with an SCF account], the same general tools can be used for connecting to any remote machine. In addition, this SCF help page has information on logging in to remote machines via ssh without having to type your password every time. This is a nice way to save yourself some time.

2 Version control

We'll be using Git extensively to do version control. Git stores the files for a project in a *repository*.

1. To clone (i.e., copy) a repository (in this case from Github) [*berkeley-stat243* is the project and *stat243-fall-2014* is the repository]:
```
>git clone https://github.com/berkeley-stat243/stat243-fall-2014
```
2. To update a repository to reflect changes made in a remote copy of the repository:
```
>cd /to/a/directory/within/the/local/repository
> git pull
```

We'll see a lot more about Git in section, where you'll learn to set up a repository, make changes to repositories and work with local and remote versons of the repository.

3 Files and directories

1. Files are stored in directories (aka folders) that are in a (inverted) directory tree, with "/" as the *root* of the tree

2. Where am I?

   ```
   > pwd
   ```

3. What's in a directory?

   ```
   > ls

   > ls -a

   > ls -al
   > ls -lrt
   ```

4. Moving around

   ```
   > cd /accounts/gen/vis/paciorek/teaching/243

   > cd ~paciorek/teaching/243

   > cd ~/teaching/243

   > cd stat243-fall14-243 # provided I am in 'teaching'

   > cd ..

   > cd -
   ```

5. Copying and removing

   ```
   > cp

   > cp -r

   > cp -rp # preserves timestamps and other metainfo (VERY handy
   for tracing your workflows if you move files between machines)

   > mkdir

   > rm

   > rm -r
   ```

```
> rm -rf # CAREFUL!
```

To copy between machines, we can use *scp*, which has similar options to *cp*:

```
> scp file.txt paciorek@radagast.berkeley.edu:~/research/.
```

```
> scp paciorek@radagast.berkeley.edu:/data/file.txt
~/research/renamed.txt
```

6. File permissions: **`ls -al`** will show the permissions for the '*user*', '*group*', '*other*' (*ugo*) for **r**eading, **w**riting and e**x**ecuting files (*rwx*):

   - to allow a file to be executed as a program:
     ```
     > chmod ugo+x myProg # myProg should be compiled code or a
     shell script
     ```
   - to allow read and write access to all:
     ```
     > chmod ugo+rw code.R
     ```
   - to prevent write access:
     ```
     > chmod go-w myThesisCode.R
     ```

7. Compressing files

   - the *zip* utility compresses in a format compatible with zip files for Windows:
     ```
     > zip files.zip a.txt b.txt c.txt
     ```
   - *gzip* is the standard in UNIX:
     ```
     > gzip a.txt b.txt c.txt # will create a.txt.gz, b.txt.gz,
     c.txt.gz
     ```
   - *tar* will nicely wrap up entire directories:
     ```
     > tar -cvf files.tar myDirectory
     > tar -cvzf files.tgz myDirectory
     ```
   - To unwrap a tarball
     ```
     > tar -xvf files.tar
     > tar -xvzf files.tgz
     ```
   - To peek into a zipped (text) file:
     ```
     > gzip -cd file.gz | less
     > zcat file.zip | less
     ```

8. Disk space

   - To see how much space is free in the various filesystems available on a machine:
     ```
     > df -h
     ```
   - To see how much space is used in a directory:
     ```
     > du -h
     ```
   - To see how much space you are using out of your overall quota
     ```
     > quota -s
     ```

4 A variety of UNIX tools/capabilities

Many UNIX programs are small programs (tools, utilities) that can be combined to do complicated things.

1. For help on a UNIX program, including command-line utilities like *ls*, *cp*, etc.
   ```
   > man cp
   ```
2. What's the path of an executable (and implicitly, does it exist on the system)?
   ```
   > which R
   ```
3. Tools for remotely mounting the filesystem of a remote UNIX machine/filesystem as a 'local' directory on your machine - see this SCF FAQ

   (a) Cloud storage: Dropbox and other services will mirror directories on multiple machines and on their servers.

4. To do something at the UNIX command line from within R, use the `system()` function in R:
   ```
   > system("ls -al")
   ```
5. Files that provide info about a UNIX machine:

   - */proc/meminfo* [in particular the *MemTotal* value]
   - */proc/cpuinfo* [or use `nproc --all`]
   - */etc/issue*
   - Example: to find out how many processors a machine has:
     ```
     > grep processor /proc/cpuinfo
     ```

6. There are (free) tools in UNIX to convert files between lots of formats (pdf, ps, html, latex, jpg). This is particularly handy when preparing figures for a publication. My computing tips page lists a number of these. On a related note *pandoc* is a program for converting between markup formats, including Markdown, HTML, Latex and MS Word.

5 Editors

For statistical computing, we need an editor, not a word processor, because we're going to be operating on files of code and data files, for which word processing formatting gets in the way.

5.1 Some useful editors

- traditional UNIX: *emacs*, *vim*
- Windows: *WinEdt*
- Mac: *Aquamacs Emacs, TextMate, TextEdit*
- Be careful in Windows - file suffixes are often hidden
- RStudio provides a built-in editor for R code files

5.2 Basic emacs

- *emacs* has special modes for different types of files: R code files, C code files, Latex files – it's worth your time to figure out how to set this up on your machine for the kinds of files you often work on
  - For working with R, ESS (emacs speaks statistics) mode is helpful. This is built into Aquamacs emacs. Alternatively, the Windows and Mac versions of R, as well as RStudio (available for all platforms) provide a GUI with a built-in editor.
- To open emacs in the terminal window rather than as a new window, which is handy when it's too slow (or impossible) to tunnel the graphical emacs window through ssh:

  ```
  > emacs -nw file.txt
  ```

Table 1: Helpful *emacs* control sequences

Sequence	Result
`C-x,C-c`	Close the file
`C-x,C-s`	Save the file
`C-x,C-w`	Save with a new name
`C-s`	Search
`ESC`	Get out of command buffer at bottom of screen
`C-a`	Go to beginning of line
`C-e`	Go to end of line
`C-k`	Delete the rest of the line from cursor forward
`C-space`, then move to end of block	Highlight a block of text
`C-w`	Remove the highlighted block, putting it in the kill buffer
`C-y` *(*after using **`C-k`** or **`C-w`***)*	Paste from kill buffer ('y' is for 'yank')