Unix for Newbies

Content borrowed and updated (with permission) from Duane A. Bailey's guidelines from 2007.

Getting help on unix:

man <command-name>	Get full description of command
man -k <keyword>	List command mentioning keyword in title

Logging in and out of a system:

logout	Terminate session
exit	Terminate current "shell"
ssh <username>@<remote host>	Login securely as username to remote host

File Manipulation:

emacs <file>	Edit a text file (See the emacs cheatsheet)
mv <old> <new>	Move or rename <old> file as <new> file
rm <file(s)>	Delete file(s) from filesystem
cp <orig> <duplicate>	Copy <orig> to file named <duplicate>
sftp <remote host>	Secure batch file transfers between machines
scp host:<orig> host:<dub>	Securely transfer files between machines
cat <file>	Display or catenate file contents to screen
more <file>	Display file page by page (but: use less)
less <file>	Display file page by page (use this one)
head <file>	Display the first few lines of a file
tail <file>	Display the last few lines of a file
grep <pattern> <file>	Search for pattern within file
source <file>	reach commands from file

Directory manipulation:

cd <directory>	Goto Directory (change focus to files in <dir>)
ls	List files in current directory
pwd	Print working directory (where am I?)
mkdir <name>	Make a new subdirectory (in this one) with Name
rmdir <name>	Remove an (empty) subdirectory

Printing and Mail:

firefox <url>	Open browser (under X11) at url
acroread <acrobat file>	Open a ".pdf" file in acrobat reader
gs <postscript file>	Open a ".ps" file with the ghostscript reader
lynx	Text-only (fast!) browser

C Compilation and Debugging:

gcc <file.c>	Compile C program into "a.out" executable
gcc -o <executable> <file.c>	Compile C program into executable
gcc -g -c <file.c>	Compile C program into debuggable file
gcc -o <exec> <f.o> <g.o>	Link object files (f and g) into single exec
gcc -Wall -o <exec> <file.c>	Compile and link file with all warnings enabled
gcc -std-gnu99 -o <exec> <file.c>	Compile and link file under 1999 standards
gdb <executable>	Run executable under debugging control
gdb <executable> <core>	Resurrect coredump with executable as model

Information about Users and Systems:

w	Who's on the system
top	What the top cpu processes
ps	List processes on this system
whoami	Who is logged in at this window
finger <user>	Get details on user
last <user>	List last time users used this machine
uptime	Print stats on machine

Key Websites:

http://www.cs.washington.edu	CS homepage
http://homes.cs.washington.edu/~<username>	Your homepage
/cse/web/homes/<user>	Where your web files live

Some Tips To Make Your UNIX Life More Reasonable

0. Walk away from the machine.
If you're not making progress, don't waste time banging your head (literally or figuratively) against the machine. Save your files, print the buggy output, or create a backup and walk away. Find someone to talk to about something else. Your mind will work on the problem while you go get a snack and take a break.

1. Read man pages.
Realize that you don't know everything. Take the time to learn how man pages are structured and what they can teach you. You find answers to many questions on the internet, but you learn even more by finding and reading the man pages. Better, you learn elegant solutions to questions you didn't think to ask.

2. Learn the emacs keystrokes.
It will save you time when you work on a machine whose mouse or arrow keys aren't working, and the keystrokes often work in other editors. Many cursor manipulations from emacs are based on history from the bash shell:

```
^P = previous command
^N = next command
^R = search for command from the past by typing a few letters
^A = go to the beginning of the command line
^E = go to the end of the command line
^B = go back one character
^F = go forward one character
^D = delete this character
<del> = delete previous character
```

3. Learn about your environment.
Shells have survived the test of time by helping their users do powerful and complex tasks. You want to learn more about 'aliases' and 'shell scripts'. Brownie points for learning about: find, tar, gawk, perl, rsync

4. Stay organized.
Create directories to organize your files within the home directory. Organize by some sane method and delete temporary files that pile up and confuse matters.

5. Make backups, use version control
We all screw up sometime, usually at 3am the night before the deadline. Use version control systems (cvs, svn, git, mercurial...) to keep a revision history you can roll back to. In a pinch, just copy the file and save it when you're making large changes. Your sleeping habits will thank you.

6. Work with others.
Working with others (usually in a lab) lets you get help when you need it and learn about how others are doing the work. When the whole lab is suffering through the same project, it makes your night that much easier. And it makes home a more pleasant place to return to.

7. Practice.
Practice more than you think you need. You know why.

8. Write.
Good writing is hard, and many computer scientists don't write enough to write well. Start with comments on your code, work on documentation for large projects, and learn how to write prose with ease.

9. Tell others what you're working on.
You learn how much you know (and how much you don't) when you try and explain your work to others. Working on computer science is only good so far as you can share it with others.