

Unix 101

Jenny Wong and Marc Burns

Computer Science Club

September 29, 2011

The Shell

- The **shell** is an application through which you will interact with the Unix system.
- It relies on text input and output rather than buttons, text fields, etc.

- The **shell** is an application through which you will interact with the Unix system.
- It relies on text input and output rather than buttons, text fields, etc.
- On a Windows or Mac machine, workflow follows these steps:
 - 1 find a widget or button within a jumble of widgets and buttons
 - 2 click the widget or button
 - 3 be presented with a new jumble of widgets and buttons
 - 4 goto 1

- The **shell** is an application through which you will interact with the Unix system.
- It relies on text input and output rather than buttons, text fields, etc.
- On a Windows or Mac machine, workflow follows these steps:
 - 1 find a widget or button within a jumble of widgets and buttons
 - 2 click the widget or button
 - 3 be presented with a new jumble of widgets and buttons
 - 4 goto 1
- Interacting with the shell is nice and simple:
 - 1 type in a command (possibly followed by arguments)
 - 2 receive feedback from the shell
 - 3 get prompted to enter another command
 - 4 goto 1

Getting to a Terminal

- In **Linux**, `Alt` + `F2` and type in *gnome-terminal*
- On a **Mac**, `Cmd` + `Space` and type in *terminal*

Some useful window manipulation shortcuts

	Mac	Linux
New Window	<code>Cmd</code> + <code>n</code>	<code>Ctrl</code> + <code>Shift</code> + <code>n</code>
New Tab	<code>Cmd</code> + <code>t</code>	<code>Ctrl</code> + <code>Shift</code> + <code>t</code>
Close Tab	<code>Cmd</code> + <code>w</code>	<code>Ctrl</code> + <code>Shift</code> + <code>w</code>

Commands and Arguments

Input can be split up into different parts:

```
> ls -l -h /u/jsmith/
```

Commands and Arguments

Input can be split up into different parts:

```
> ls -l -h /u/jsmith/
```

Commands are the names of the programs or scripts you wish to run.

Commands and Arguments

Input can be split up into different parts:

```
> ls -l -h /u/jsmith/
```

Commands are the names of the programs or scripts you wish to run.

Arguments are extra space-delimited “words” after the command for specifying details or options. Each word is a single argument.

Commands and Arguments

Input can be split up into different parts:

```
> ls -l -h /u/jsmith/
```

Commands are the names of the programs or scripts you wish to run.

Arguments are extra space-delimited “words” after the command for specifying details or options. Each word is a single argument.

Flags are used to turn on/off some features or indicate what kind of information is in the next argument. They usually begin with a - or --, and single dash flags can often be combined. `ls -lh /u/jsmith/` is equivalent.

- `> cmd --help`

Appending the `--help` flag to a command usually prints out a small blurb on how to use the command in question.

- `> man cmd`

Manpages provide more detailed documentation.

- Search engines are your friend.
- Don't forget you can also ask people!

The Manpages

"An interface to the on-line reference manuals"

The manpages document all possible flags and how the arguments (if any) should be formatted.

> `man man` to read the manpage on how to use the manual.

Quick guide

`↑` or `k` or `PgUp`

Scroll up the document

`↓` or `j` or `PgDn`

Scroll down the document

`/foo, ?foo`

Looks for foo after/before the current position. *Not* case-sensitive.

`n` / `N`

Goes to the next/previous match

`q`

Quit

`cd` change directory

`> cd ..` brings you up one directory,

`> cd ../..` brings you up two, and so forth

`pwd` print working directory

tells you where you are in the filesystem

`ls` list [directory contents]

`> ls` with no arguments tells you what's in the current directory

`> ls ~/archives/` tells you what's in the archives/ folder in your home directory (homedir), ~

`> ls -d */` lists all folders in the current directory

Navigation Pitfalls

Unlike Windows, file and folder names are case sensitive.
KITTY.JPG is different from kitty.jpg.

Navigation Pitfalls

Unlike Windows, file and folder names are case sensitive.
KITTY.JPG is different from kitty.jpg.
Spaces in file paths need to be **escaped**:

- `> cd ~/Cat Pictures` will be interpreted incorrectly
- try `> cd ~/Cat\ Pictures` or `> cd ~/'Cat Pictures'`

Navigation Pitfalls

Unlike Windows, file and folder names are case sensitive.
KITTY.JPG is different from kitty.jpg.
Spaces in file paths need to be **escaped**:

- `> cd ~/Cat Pictures` will be interpreted incorrectly
- try `> cd ~/Cat\ Pictures` or `> cd ~/'Cat Pictures'`

Files with names beginning with a `.` are **hidden**. Passing the `-a` flag (the `a` stands for all) will reveal them.

Permissions

- read, write, and execute
- Binary files, scripts and folders need to be executable to be run/open.
- The `chmod` command is used to edit permissions.

`chmod mode file_or_dir`

`mode` is of the form `[ugo][+-][rwx]`

`u`, `g`, `o` designate who the changes apply to (the file's owner (user), group, and all others, respectively)

`+`, `-` tell `chmod` whether you intend to give or remove the following permissions

Exercise: What does `> chmod go-rwx somefile` do?

Manipulating Files

Tip: Note that each name comes from the first two consonants of a real English word that matches what the command does. Many Unix commands are (shortened) English words with removed vowels.

`cp` *old_file* *new_file* c*opies* *old_file* (source) to *new_file* (destination)

`mv` *old_file* *new_file* m*oves* the *old_file* to *new_file*

`rm` *file* r*emoves* file. Beware! File will be permanently gone on most unix systems.

CSCF Snapshots

In general, there is no undo tool for deleted files. However, in the CSCF computing environment, there are hourly backups in the directory `~/ .snapshot`. Use `cd` and `ls` to look around, and `cp` to “restore” the file. See www.cs.uwaterloo.ca/cscf/howto/snap for more details.

Manipulating Folders

- use `-r` when copying a folder (`mv` does not need it to work)
- the `-r` flag causes `cp` to run recursively on the contents of a folder

Tricky Details

If the destination (last argument) is a folder that *already exists*, `cp` makes a new folder with the original name inside the existing folder. If the destination folder doesn't exist, `cp` creates the dest. folder and copies the contents of the source folder into it.

```
> cp -r ~/foo/ ~/www/foo/
```

e.g. After running the command above, there will be a folder `~/www/foo/` that contains the same things as `~/foo/`.

Making and Editing Files

- **nano** is a command-line text editor.
- To create a file, open up nano, type something and exit (`Ctrl` + `x`). You will be asked whether you'd like to save your work.
- **vim** and **emacs** are two other popular command-line text editors.
- **mkdir** and **rmdir** create/delete empty folders (see manpages).

Connecting to Other Machines Over the Network

- SSH stands for Secure SHell. `scp` is secure copy.
- To connect to another machine, use the `ssh` command. Enter your password for that machine if asked.

```
> ssh jsmith@linux.student.cs
```

- To copy a file from another machine to yours, try:

```
> scp jsmith@linux.student.cs:~/file .
```

- Alternately, to copy a file from your machine to another:

```
> scp file jsmith@linux.student.cs:~
```

- In general, arguments are of the form `user@host:location` (the blue part is optional).

More Advanced Searching

- **grep** searches for lines inside files that match criteria that you specify (i.e. lines that contain the word 'cat'). Wikipedia's Grep page explains usage very nicely.
- **find** searches for file/folder names that fit criteria you specify in arguments. Similarly, the Wikipedia Find page is a good reference.