Introduction to Unix Shell Programming

Erik Spence

SciNet HPC Consortium

9 June 2014

The Truth about interfaces

- Nobody. Nobody. Nobody, uses a Graphical User Interface (GUI) for HPC. Nobody.
- Why? Because HPC is Unix/Linux based, without a GUI.
- Why?
  - Because the earliest mainframe computers were Unix based, and it's always been that way.
  - You can't have hundreds of people logged into a node, and run GUIs for all of them (but you can run a command line interface).
  - GUIs are slow over networks.
- Who cares? Well, if you're going to do real HPC then you're going to need to interface with these computers, and that means learning how to use the command line.
- This is not to suggest that Linux machines don't have GUIs. They do. It's just the HPC machines that don't.

SciNet compute • calcul CANADA

GUIs versus the command line

- Graphical User Interfaces (GUIs) have many strengths.
  - Very good at operating an existing system.
  - Very good at using existing functionality, existing controls.
  - Programs tend to have lots of functionality built into them, but can only do what they've been programmed to do.
  - Can't save a series of commands to replicate functionality.
  - Easy to learn. Hard to use for big tasks.
- The Command Line Interface (CLI) has a different approach.
  - A blank canvas; you get to program what you want to do.
  - Good at creating new things.
  - Commands that do already exist are very good at doing *one* thing.
  - Commands that you create can be saved and re-used.
  - Hard to learn. Easy to use for big tasks.

SciNet compute • calcul CANADA

"The" shell

Open a Terminal:

- Windows: start up MobaXterm.
- Mac: Applications/Utilities/Terminal (drag this to the dock).
- Linux: xterm, eterm, ...
- The terminal launches a shell. The shell is what you are actually interacting with when you type commands.
- The shell provides access to files, the network, and other programs.
  - You type in commands.
  - The shell interprets them.
  - Performs actions on its own, or launches other programs.
- The most commonly used shell in Linux is bash.
- There are others; mostly the same but some syntax is different.
- Those of you using MobaXterm: go to Settings > Configuration and change your "persistent HOME directory" to a permanent location.

SciNet compute • calcul CANADA

The command line prompt

Now that we've got a terminal open, what do we see? We see the command line prompt!

On MobaXterm, the prompt looks something like this:

```
[ejspence.mycomp]
```

Where 'ejspence' is my username, and 'mycomp' is the name of my computer. On a Mac my prompt might look like this:

```
mycomp:~ ejspence$
```

On a Linux machine, my prompt might look like this:

```
[ejspence@mycomp ~]$
```

All of these are customizable, which we won't be covering today. It doesn't matter what it looks like, so long as you're comfortable with the prompt.

Our first shell script

We will be using the 'bash' shell for this class. It is the most commonly used on Linux systems, is widely available, and is the default on SciNet.

```
[ejspence.mycomp]
[ejspence.mycomp] hello="world"
[ejspence.mycomp] echo Hello, world
Hello, world
[ejspence.mycomp] echo Hello, $hello
Hello, world
```

Don't forget to hit 'Enter' at the end of each line.

The '=' sign tells the shell to create a variable called 'hello' and assign it the value "world". The value of the variable is accessed using the $.

The 'echo' command prints out whatever the shell gives it.

If you get an error message, it's likely you're running a different shell (csh, tcsh, zsh). Type 'bash' to start a bash shell, and try again.

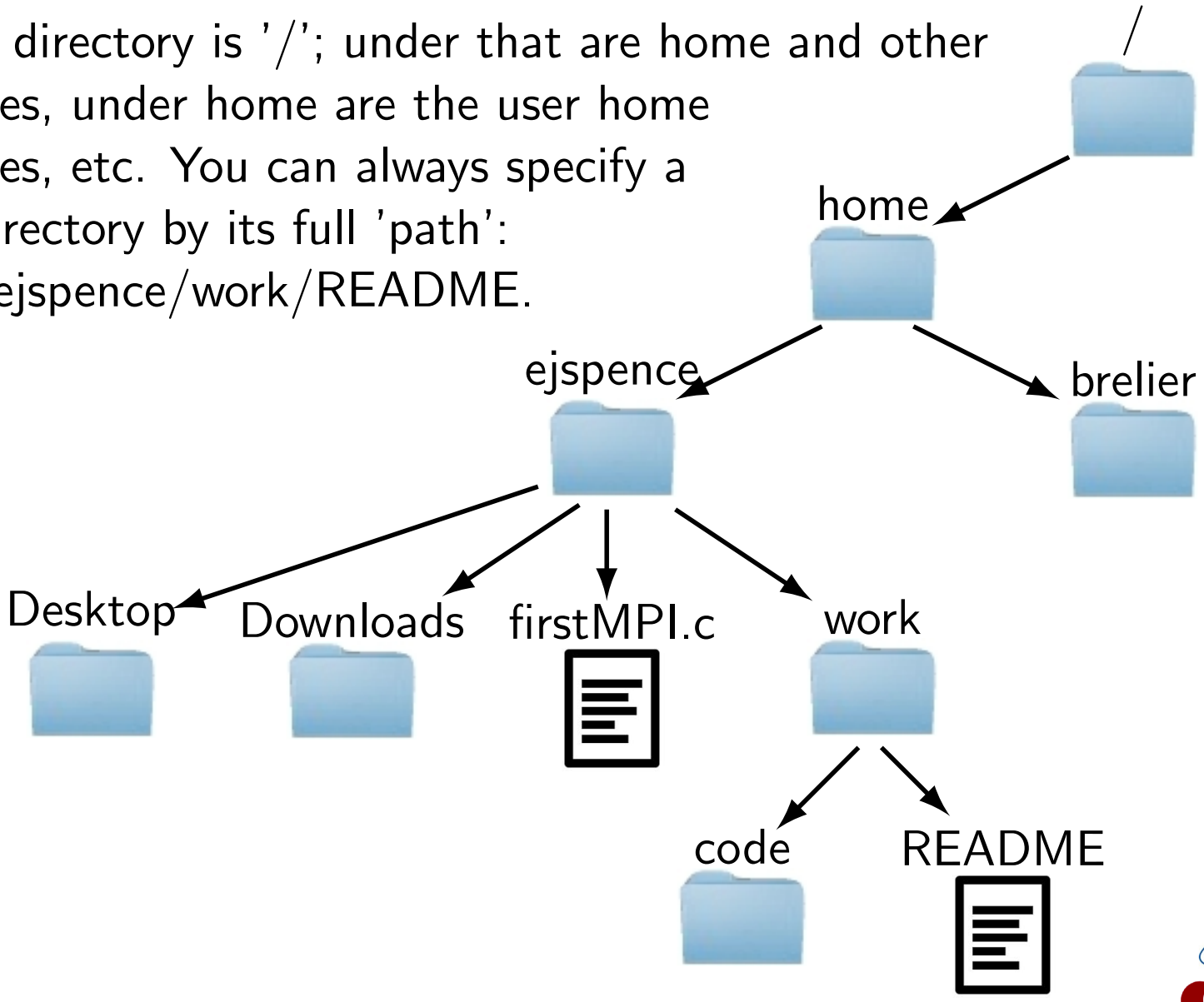
Basics: home sweet home

- When you launch a shell, you start in your home directory, this is the top directory of all of your stuff.
- The home directory is /home/mobaxterm for MobaXterm, /Users/username on Macs, /home/username on Unix/Linux systems.
- The home directory is universally represented by the ~ symbol.
- Directories are sometimes called folders because of how they are represented in GUIs. We will call them directories.
- On Unix systems directories are listings of files, including other directories.
- If you are using MobaXterm your home directory will be put in your "persistent HOME directory" location, as set in Settings > Configuration.

SciNet compute • calcul CANADA

A typical Linux directory tree

The top directory is '/'; under that are home and other directories, under home are the user home directories, etc. You can always specify a file or directory by its full 'path': /home/ejspence/work/README.



Basics: the file system

I will be assuming I am on a MobaXterm terminal. Your output will likely differ somewhat if you are on a different system.

```
[ejspence.mycomp]
[ejspence.mycomp] pwd
/home/mobaxterm
[ejspence.mycomp] ls
Desktop LauncherFolder MyDocuments
[ejspence.mycomp] ls /home
ejspence mobaxterm
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`arg`	mandatory argument
`[arg]`	optional argument

- 'pwd' stands for 'present working directory'. It will print the directory you are currently in. As mentioned on the last slide, you begin in your home directory.
- 'ls' stands for 'list'. If no argument is given it lists the contents of the current directory, otherwise it lists the contents of the argument. Some implementations of ls include colour.

Creating directories

```
[ejspence.mycomp] pwd
/home/mobaxterm
```

```
[ejspence.mycomp] ls
Desktop LauncherFolder MyDocuments
```

```
[ejspence.mycomp] mkdir firstdir
```

```
[ejspence.mycomp] ls -F
Desktop@ LauncherFolder@ MyDocuments@
firstdir/
```

```
[ejspence.mycomp] mkdir /home/mobaxterm/2ndir
```

```
[ejspence.mycomp] ls -F
2ndir/ Desktop@ LauncherFolder@ MyDocuments@
firstdir/
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`arg`	mandatory argument
`[arg]`	optional argument

- 'mkdir' stands for 'make directory', it creates a new directory. It puts the directory in the current directory, unless a different path is specified.
- 'ls -F' lists the directory, as before, but labels directories with a '/', and links with a '@'.

Moving between directories

```
[ejspence.mycomp] ls
2ndir Desktop LauncherFolder MyDocuments
firstdir
[ejspence.mycomp] mkdir firstdir/temp
[ejspence.mycomp] cd firstdir
[ejspence.mycomp] pwd
/home/mobaxterm/firstdir
[ejspence.mycomp] ls
temp
[ejspence.mycomp] cd temp
[ejspence.mycomp] pwd
/home/mobaxterm/firstdir/temp
[ejspence.mycomp] cd ..
[ejspence.mycomp] pwd
/home/mobaxterm/firstdir
[ejspence.mycomp] cd ~
[ejspence.mycomp] pwd
/home/mobaxterm
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`arg`	mandatory argument
`[arg]`	optional argument

'cd' stands for 'change directory'. It moves you to the directory you specify. With no argument it moves you to the home directory.

Tips for getting around

Some common commands for moving around your directories:

- The directory above is represented by the '..' symbol; the current directory is represented by the '.' symbol:
  - 'cd ..' goes up a directory.
  - 'cd ../..' goes up two directories.
  - 'cd ../otherdir' goes up one directory and then down into 'otherdir'.
  - 'cd firstdir/seconddir/../..' goes nowhere.
  - 'cd ./././.' also goes nowhere.
- You can use absolute paths: 'cd /home/mobaxterm/firstdir/temp'.
- ~ is the symbol for your home directory, on whatever system you are using. 'cd ~/work' goes to my ~/work directory (/home/mobaxterm/work).
- 'cd' without any arguments goes to your home directory (~), from no matter where you are.
- 'cd -' goes back to the directory you were in previously.

Tips for using the command line

Some more helpful tips for using the command line:

- Use the 'tab' key, it will 'auto-complete' the available options based on what you've already typed,
  - start typing your command, and then hit 'tab'
  - the shell will fill in the rest, if there is only one option.
  - if nothing happens, there is either no option or more than one option.
  - hit the tab key twice, this will list all available options
  - continue typing to reduce the number of options, then hit tab again to fill in the rest.
- Use 'Ctrl-a' to go to the beginning of the command line, 'Ctrl-e' to go to the end of the line.
- Use the up arrow. This scrolls through the shell's 'history'.

History

```
[ejspence.mycomp] history
.
.
14 [2014-06-05 11:23:42] mkdir firstdir/temp
15 [2014-06-05 11:23:47] cd firstdir
16 [2014-06-05 11:23:49] pwd
17 [2014-06-05 11:23:50] ls
18 [2014-06-05 11:23:53] cd temp
19 [2014-06-05 11:23:55] pwd
20 [2014-06-05 11:23:58] cd ..
21 [2014-06-05 11:23:59] pwd
22 [2014-06-05 11:24:03] cd
23 [2014-06-05 11:24:05] pwd
24 [2014-06-05 11:24:11] history
[ejspence.mycomp]
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`arg`	mandatory argument
`[arg]`	optional argument

- The history command prints the commands that you've typed at the command line. "history 10" prints the last 10 commands.
- Use the up arrow to access the entries.

Our commands so far

There are a couple of things to observe about the commands we've seen so far:

- The commands are designed to be fast and easy to use.
- The commands do, essentially, only one specific thing.
- The commands are pretty cryptic. Either you know them or you don't.
- Commands can take options. These are usually indicated with a '-something' flag (such as 'ls -F').

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`arg`	mandatory argument
`[arg]`	optional argument

As you may have hoped, the purpose of this class is to teach you enough commands that you will be able to survive the Unix command line.

Man pages

Know a command but aren't sure how to use the options? Use the man (manual) page!

- Most programs have a man page describing its use and all available options.
- These pages are good for finding out more about a command you already use, but are less good for learning new commands.
- Many programs have gazillions of options.
- No human being who has ever lived has know all the options for 'ls'.
- Over time you will find a few that you find useful for your favourite commands.
- Unfortunately, MobaXterm dumps all man pages together, so you need to scroll down to find the entry you want.

SciNet compute • calcul CANADA

Man pages: help!

Use the man (manual) page for a list of all flags for a command.

```
[ejspence.mycomp] man ls
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]...  [FILE]...
DESCRIPTION
  List information about the FILEs (the
  current directory by default).  Sort
  entries alphabetically if none of -
  cftuvSUX nor --sort.

Mandatory arguments to long options are
mandatory for short options too.
-a, --all
  do not ignore entries starting with .
-A, --almost-all
  do not list implied .  and ..
...
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`arg`	mandatory argument
`[arg]`	optional argument

Not sure how to use the command? Not sure what options there are? Check the man page!

Where is my USB stick?

```
[ejspence.mycomp] pwd
/home/mobaxterm
```

```
[ejspence.mycomp] df -h
Filesystem      Size      Used   Available    Use%   Mounted on
C:/Users/IBM_AD~1/AppData/Local/Temp/MOBAXT~1.1/bin
                29.9G    16.8G        13.1G     56%   /usr/bin
C:/Users/IBM_AD~1/AppData/Local/Temp/MOBAXT~1.1/lib
                29.9G    16.8G        13.1G     56%   /usr/lib
C:/Users/IBM_AD~1/AppData/Local/Temp/MobaXterm7.1
                29.9G    16.8G        13.1G     56%   /
C:/Users/IBM_AD~1/DOCUME~1/FakeHome
                29.9G    16.8G        13.1G     56%   /home/mobaxterm
C:              29.9G    16.8G        13.1G     56%   /drives/c
E:               7.2G   736.2M         6.5G     10%   /drives/e
```

```
[ejspence.mycomp] cd /drives/e
```

```
[ejspence.mycomp] pwd
/drives/e
```

You'll need to use the 'df -h' command to find it.

Wildcards

Wildcards (*) capture all possible combinations that fit a given description.

```
[ejspence.mycomp] pwd
/drives/e
```

```
[ejspence.mycomp] cd software/mobaxterm
```

```
[ejspence.mycomp] ls
Development.mxt3 Emacs.mxt3 Git.mxt3
NEdit.mxt3 MobaXterm_Personal_7.1.exe
.
```

```
[ejspence.mycomp] ls G*
Git.mxt3 Gvim.mxt3
```

```
[ejspence.mycomp] ls *.mxt3
Development.mxt3 Git.mxt3 NEdit.mxt3
Subversion.mxt3 Emacs.mxt3 Gvim.mxt3
Python.mxt3
```

```
[ejspence.mycomp] cd /drives/e
```

```
[ejspence.mycomp] cd SCMP101_shell/data
```

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`arg`	mandatory argument
`[arg]`	optional argument

The shell expands the wildcard into a list of all possible matches, and passes the list to the command.

Manipulating files: copying

```
[ejspence.mycomp] ls
Bert          Lawrence alexander jamesm
Frank_Richard THOMAS   gerdal
[ejspence.mycomp] cd gerdal
[ejspence.mycomp] ls
.
Data0413 Data0468 Data0528 Data0558
[ejspence.mycomp] ls *27*
Data0227 Data0279
[ejspence.mycomp] cp Data0227 Data0227-new
[ejspence.mycomp] ls *27*
Data0227 Data0227-new Data0279
[ejspence.mycomp] cp Data0227 ..
[ejspence.mycomp] ls ..
Bert      Frank_Richard THOMAS    gerdal
Data0227  Lawrence      alexander jamesm
```

'cp' stands for 'copy'; it copies a file.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`arg`	mandatory argument
`[arg]`	optional argument

Wildcards can appear anywhere in the variable you are searching for. They don't need to come at the end.

Manipulating files: moving

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data/gerdal
[ejspence.mycomp] ls *27*
Data0227 Data0227-new Data0279
[ejspence.mycomp] mv Data0227-new new.txt
[ejspence.mycomp] ls *27*
Data0227 Data0279
[ejspence.mycomp] ls *txt
new.txt
[ejspence.mycomp] mv new.txt ../Data0227
[ejspence.mycomp] ls *txt
ls: *txt:  No such file or directory
[ejspence.mycomp] cd ..
[ejspence.mycomp] ls *27*
Data0227
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`arg`	mandatory argument
`[arg]`	optional argument

- 'mv' stands for 'move'; it moves a file and/or renames it.
- mv can overwrite a file, so be careful when moving things!

Manipulating files: deleting

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data
```

```
[ejspence.mycomp] cd ..
```

```
[ejspence.mycomp] ls
Bert       Frank_Richard THOMAS    gerdal
Data0227   Lawrence      alexander jamesm
```

```
[ejspence.mycomp] ls *27*
Data0227
```

```
[ejspence.mycomp] rm Data0227
```

```
[ejspence.mycomp] ls *227*
ls: *227*:  No such file or directory
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`arg`	mandatory argument
`[arg]`	optional argument

- 'rm' stands for 'remove'; it deletes a file. It does not delete directories, by default.
- rm does not 'move the file to the Trash'. It deletes it; it's gone; it's not recoverable. Be sure before you use rm.

Copying directories

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data
```

```
[ejspence.mycomp] mkdir temp
```

```
[ejspence.mycomp] cp gerdal/Data0227 temp
```

```
[ejspence.mycomp] ls temp
Data0227
```

```
[ejspence.mycomp] cp temp temp2
cp: omitting directory 'temp'
```

```
[ejspence.mycomp] cp -r temp temp2
```

```
[ejspence.mycomp] ls
Bert          Lawrence  alexander jamesm
Frank_Richard THOMAS    gerdal    temp
temp2
```

```
[ejspence.mycomp] ls temp2
Data0227
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`arg`	mandatory argument
`[arg]`	optional argument

'cp' will only copy files by default. To copy directories, including everything within them, use 'cp -r'.

Deleting directories

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data
[ejspence.mycomp] ls temp
Data0227
[ejspence.mycomp] rm temp
rm: temp: is a directory
[ejspence.mycomp] rmdir temp
rmdir: 'temp': Directory not empty
[ejspence.mycomp] rm temp/*
[ejspence.mycomp] ls temp
[ejspence.mycomp] rmdir temp
[ejspence.mycomp] rm temp2/*
[ejspence.mycomp] rmdir temp2
[ejspence.mycomp]
```

'rmdir' deletes a directory.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`arg`	mandatory argument
`[arg]`	optional argument

Uncharacteristically for Linux, rmdir protects you. You can't delete a directory with files in it, you must delete the files first.

Checking file types

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data
[ejspence.mycomp] cd alexander
[ejspence.mycomp] ls
.
.
.
data_379.DATA data_434.DATA data_530.DATA
data_420.DATA data_502.DATA data_560.DATA
data_297.DATA data_357.DATA data_421.DATA
[ejspence.mycomp] file data_560.DATA
data_560.DATA: ASCII text
[ejspence.mycomp]
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`file file`	type of file
`arg`	mandatory argument
`[arg]`	optional argument

The 'file' command tells you what type of file it is.

Looking inside files

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data/alexander
```

```
[ejspence.mycomp] more data_560.DATA
#
Reported:  Sat May 7 10:50:03 2011
Subject:  georgeSpice437
Year/month of birth:  1997/12
Sex:  M
CI type:  20
Volume:  3
Range:  5
Discrimination:
```

```
[ejspence.mycomp]
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file

`arg`	mandatory argument
`[arg]`	optional argument

'more' lists the contents of the file.

Looking inside files, continued

```
[ejspence.mycomp] cat data_560.DATA
#
Reported:  Sat May 7 10:50:03 2011
Subject:  georgeSpice437
.
.
```

```
[ejspence.mycomp] less data_560.DATA
#
Reported:  Sat May 7 10:50:03 2011
Subject:  georgeSpice437
.
.
```

'more', 'cat', and 'less' all output the contents of the file, but in different ways. Can you tell the differences? Type 'q' to get out of 'more' or 'less'.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`arg`	mandatory argument
`[arg]`	optional argument

SciNet compute • calcul CANADA

cat'ing files together

```
[ejspence.mycomp] ls *DATA
.
.
data_379.DATA data_434.DATA data_530.DATA
data_420.DATA data_502.DATA data_560.DATA
data_297.DATA data_357.DATA data_421.DATA
[ejspence.mycomp] cat *DATA > all-DATA
[ejspence.mycomp] ls *DATA
all-DATA data_297.DATA data_357.DATA
.
.
data_346.DATA data_415.DATA data_498.DATA
data_550.DATA data_292.DATA data_347.DATA
data_420.DATA data_502.DATA data_560.DATA
[ejspence.mycomp]
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`arg`	mandatory argument
`[arg]`	optional argument

- 'cat' dumps the input (whatever it is) to the screen.
- '>' redirects the input to a file, instead of the screen.

cat'ing files together, continued

```
[ejspence.mycomp] less all-DATA
#
Reported:  Wed Aug 17 13:56:38 2011
Subject:  madonnaStarr178
Year/month of birth:  1995/02
Sex:  N
CI type:  8
Volume:  7
Range:  3
Discrimination:  5
#
Reported:  Thu May 19 09:08:14 2011
Subject:  paulSpice199
Year/month of birth:  1994/01
Sex:  M
CI type:  24
Volume:  4
Range:  9
.
.
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`arg`	mandatory argument
`[arg]`	optional argument

Redirection fun

- cmd > file takes the output that would have gone to the screeen, creates a new file called file, and redirects (dumps) the output to the file. If the file already exists the previous content of the file is overwritten.
- cmd >> file takes the output that would have gone to the screen, and *appends* it to file. If the file doesn't already exist then it is created.
- cmd < file takes file and uses it as input to cmd.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd

`arg`	mandatory argument
`[arg]`	optional argument

More redirection fun

```
[ejspence.mycomp] cat < all-DATA
#
Reported:  Wed Aug 17 13:56:38 2011
Subject:  madonnaStarr178
Year/month of birth:  1995/02
Sex:  N
CI type:  8
Volume:  7
Range:  3
Discrimination:  5
#
Reported:  Thu May 19 09:08:14 2011
Subject:  paulSpice199
Year/month of birth:  1994/01
Sex:  M
CI type:  24
Volume:  4
Range:  9
.
.
```

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd

`arg`	mandatory argument
`[arg]`	optional argument

Head/Tail

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data/alexander
[ejspence.mycomp] head -4 all-DATA
#
Reported:  Wed Aug 17 13:56:38 2011
Subject:  madonnaStarr178
Year/month of birth:  1995/02
[ejspence.mycomp] echo "nice" >> all-DATA
[ejspence.mycomp] tail -5 all-DATA
CI type:  20
Volume:  3
Range:  5
Discrimination:
nice
[ejspence.mycomp]
```

'head'/'tail' prints the first/last 10 lines of the input.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file

Word count

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data/alexander
```

```
[ejspence.mycomp] wc all-DATA
441 1173 7184 all-DATA
```

```
[ejspence.mycomp] wc -l all-DATA
441 all-DATA
```

```
[ejspence.mycomp] wc -w all-DATA
1173 all-DATA
```

```
[ejspence.mycomp] wc -c all-DATA
7184 all-DATA
```

```
[ejspence.mycomp] wc -w *DATA
.
.
24 data_550.DATA
23 data_560.DATA
2346 total
```

'wc' stands for 'word count'. It counts the number of words/lines/characters in the input.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`mkdir dir`	create a directory
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file
`wc file`	word count data of file

SciNet compute • calcul CANADA

find

- Wildcards are very powerful.
- from the data directory, type 'ls */*00*'.

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data/alexander
```

```
[ejspence.mycomp] cd ..
```

```
[ejspence.mycomp] ls */*00*
.
.
Bert/audioresult-00330.txt Bert/audioresult-00460.txt Frank_Richard/data_500
Bert/audioresult-00332.txt Bert/audioresult-00466.txt Lawrence/Data0300
Bert/audioresult-00350.txt Bert/audioresult-00470.txt Lawrence/Data0400
```

- This finds files which contain '00' in the name, in any subdirectory one level below this one.
- Similarly for 'echo */*00*'.
- But it can only match the specified levels of directories.
- 'find' is a tool which lets you find files anywhere below a given directory, based on arbitrary criteria.

find, continued

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data
```

```
[ejspence.mycomp] find . -print
.
.
./jamesm/data_553.txt
./jamesm/NOTES
./jamesm/data_374.txt
./jamesm/data_280.txt
./jamesm/data_375.txt
./jamesm/data_476.txt
./jamesm/data_264.txt
```

```
[ejspence.mycomp]
```

'find . -print' tells find to look for files starting in the directory '.', and to print the results.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`mkdir dir`	create a directory
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file
`wc file`	word count data of file
`find dir`	find files

Feeding find commands

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data
```

```
[ejspence.mycomp] find . -exec echo {} \;
.
.
./jamesm/data_553.txt
./jamesm/NOTES
./jamesm/data_374.txt
./jamesm/data_280.txt
./jamesm/data_375.txt
./jamesm/data_476.txt
./jamesm/data_264.txt
```

'find . -exec echo {} \;' tells find to execute the 'echo' command on everything which gets put in {}, which are the filenames. The command ends with '\;'.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`mkdir dir`	create a directory
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file
`wc file`	word count data of file
`find dir`	find files

More find options

```
[ejspence.mycomp] pwd
/drives/e/SCMP101_shell/data
```

```
[ejspence.mycomp] find . -type d
.
./Lawrence
./Frank_Richard
./gerdal
./Bert
./alexander
./THOMAS
./jamesm
```

```
[ejspence.mycomp]
```

The '-type' argument specifies the type of file that you're looking for. 'd' and 'f', directories and regular files, are the two most commonly used options.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`mkdir dir`	create a directory
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file
`wc file`	word count data of file
`find dir`	find files

More find options, continued

```
[ejspence.mycomp] find . -type f -name "*09*"
./gerdal/Data0409
./alexander/data_309.DATA
./jamesm/data_509.txt
[ejspence.mycomp]
```

The '-name' argument specifies the characteristics of the name of the file to be found.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`mkdir dir`	create a directory
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file
`wc file`	word count data of file
`find dir`	find files

Logout

```
[ejspence.mycomp] logout
```

What to do when you're finished? Use the 'logout' command to exit the terminal session cleanly (you don't need to do this now). Ctrl-d also works.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`mkdir dir`	create a directory
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file
`wc file`	word count data of file
`logout`	close the terminal session

Enough to get started

- These commands, and a few more we'll learn after the break, are enough to get started with using the command line.
- As you have seen, Unix commands are simple, and are designed to do one specific thing.
- By combining these commands together we will be able to do more interesting things.
- If there is functionality that you think ought to exist, it probably does. Ask someone what the command is, or google it.

Our commands

`echo arg`	echo the argument
`pwd`	present working directory
`ls [dir]`	list the directory contents
`cd [dir]`	change directory
`history [num]`	print the shell history
`man cmd`	command's man page
`cp file1 file2`	copy a file
`mv file1 file2`	move/rename a file
`rm file`	delete a file
`mkdir dir`	create a directory
`rmdir dir`	delete a directory
`file file`	type of file
`more file`	scroll through file
`less file`	scroll through file
`cat file`	print the file contents
`cmd > file`	redirect output to file
`cmd >> file`	append output to file
`cmd < file`	use file as input to cmd
`head file`	print first 10 lines of file
`tail file`	print last 10 lines of file
`wc file`	word count data of file
`logout`	close the terminal session