# PHYSICS 2G03
# Scientific Computing

## Using Unix
## (or Linux)

## Unix Operating System

- Unix was designed in the 1970’s
- Goal: A genuine multi-user operating system
- Design: Associate everything with a file
- Design Philosophy: Lots of small utility programs than can be used together

# The Unix Command prompt

When you log in, Unix defaults to a text interface

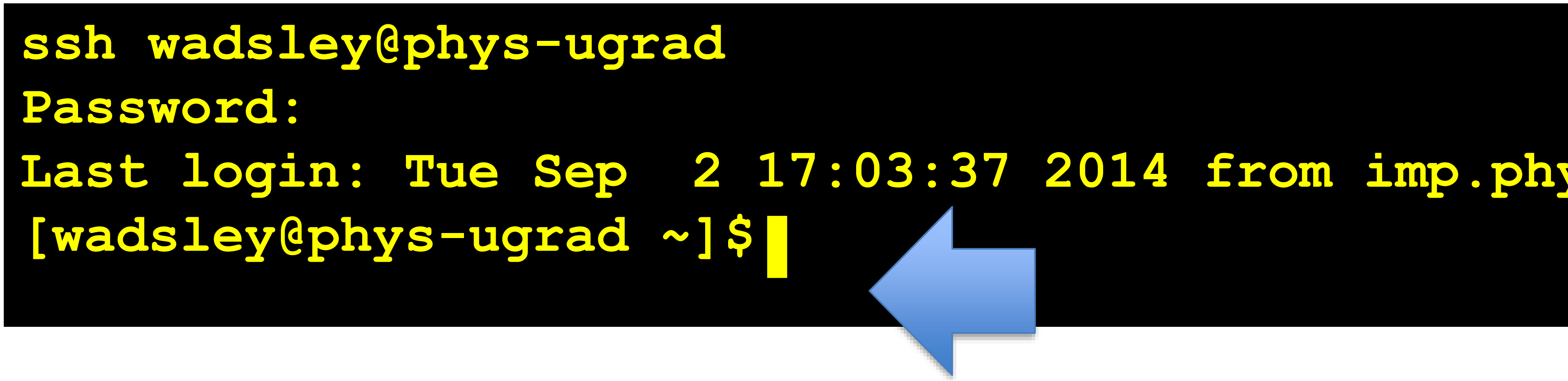
You type commands follow the prompt (after the $ symbol below) and hit enter to execute them

```
ssh wadsley@phys-ugrad
Password:
Last login: Tue Sep  2 17:03:37 2014 from imp.phy
[wadsley@phys-ugrad ~]$
```

## Please log into phys-ugrad NOW!

Log into phys-ugrad with mobaterm/ssh and try out the unix commands we are discussing.

You will need to be familiar with them to do you work for the course.



```
ssh wadsley@phys-ugrad
Password:
Last login: Tue Sep  2 17:03:37 2014 from imp.phy
[wadsley@phys-ugrad ~]$
```

## New Passwords!

- When you are connected to phys-ugrad change your password if it is *2g03*

  use the passwd command
- It must not be a simple English word
- The system administrator will disable accounts without a new password

```
ssh wadsley@phys-ugrad
Password:
Last login: Tue Sep  2 17:03:37 2014 from imp.phy
[wadsley@phys-ugrad ~]$ passwd
```

# Unix Files
# (look at them with ls)

**`ls`** <enter> by default shows your files

**`ls /home/2G03`** <enter> shows files in the directory /home subdirectory 2G03

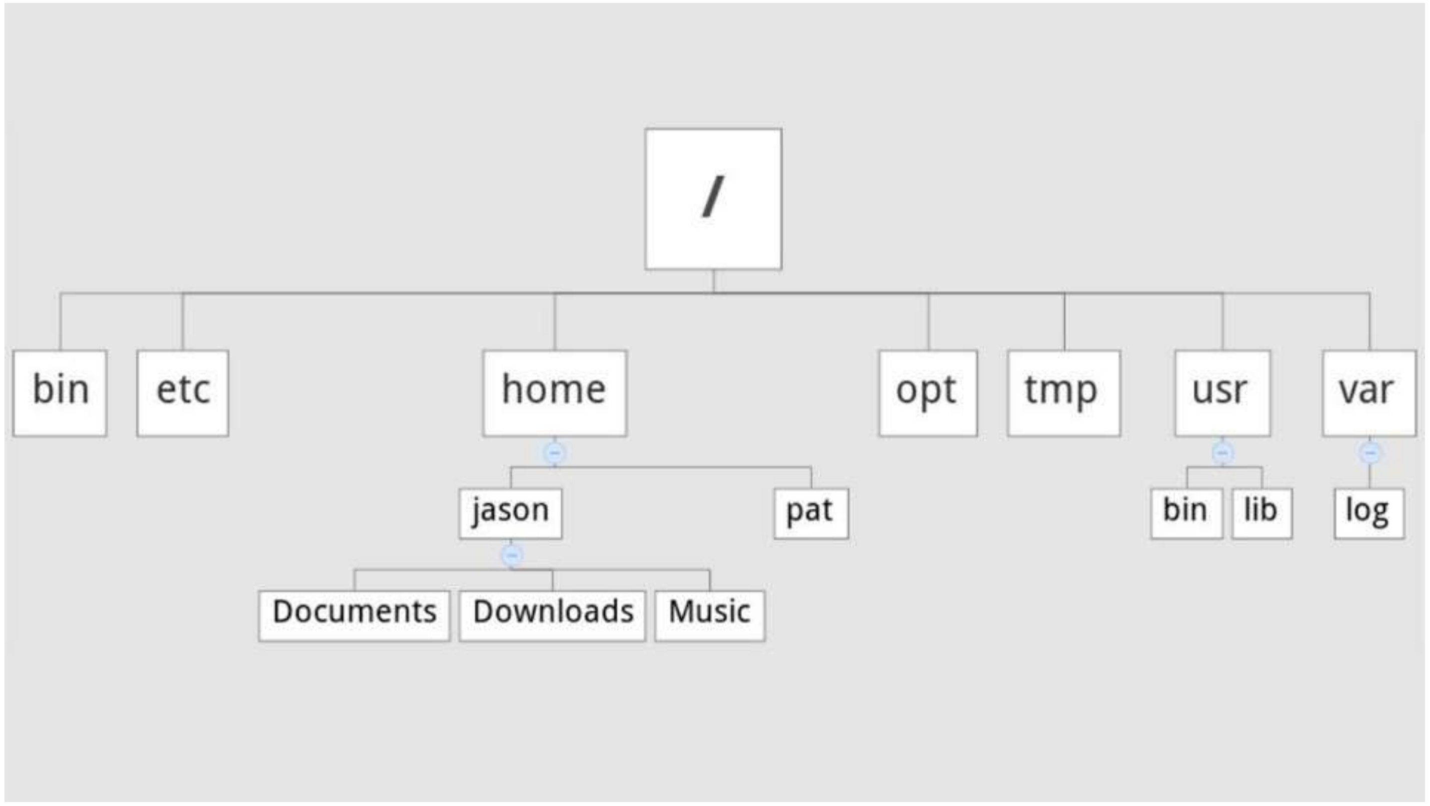
Directories /home are like Windows folders

Unix separates directories with a /

# Unix Files (look at them with ls)

The top directory is /

**`ls /`** &lt;enter&gt; shows the full file system



File Tree

# Unix Files
(look at them with ls)

The top directory is /

**`ls /`** <enter> shows the full file system

| | |
|---|---|
| /bin | core programs |
| /sbin | system programs |
| /usr | user programs |
| /home | user home directories |

# Unix Files

System directories

/dev -- devices (e.g. mouse, disk)

/etc -- system config files

/var -- logs, emails, printing

/proc -- a list of all processes

## Unix for the user

- On a Unix system you can run many programs simultaneously
- Normally this is done by entering commands at the prompt
- There are commands to look at files, users, processes, etc…

## Info Commands

- Some commands/programs are just for looking at the machine and who’s there
- e.g. whoami, hostname, who

# A few unix Commands to try (hit enter after each one)

```
$ whoami
wadsley
$ hostname
phys-ugrad
$ who
$ more /proc/cpuinfo
$ gedit &
$ xemacs &
$ xterm &
$ xeyes &
$ env | grep SHELL
$ top   (q to exit)
```

## Unix Shell

- The shell is the program that looks at your commands and works out what you want to run
- We use **tcsh** for the shell
- Tcsh manages the command prompt that appears in an xterm

- Other shells: sh, ksh, bash

## Unix Shell: Tcsh

- You can configure how tcsh operates
- The .cshrc file in your home directory is your personal configuration

- **`more ~/.cshrc`**

(~ is a shortcut for your home directory)

# Unix Shell: Tcsh configuration

Configuration of the shell

- type: set   (and hit enter)

Alias command: Sometimes its convenient to have shortcuts

e.g. **`alias x xterm`**

Typing x now is the same as xterm

**`unalias x`**        get rid of the alias!

# Multiple terminals on phys-ugrad

- Sometime more than one terminal is useful:
  Type **`xterm &`** to make a new one
- For programming I like to have an xterm for compiling and running and an emacs window for editing the program

## Unix Commands

- Unix commands have a generic structure:

Command [options] [arguments]

| | |
|---|---|
| **`ls`** | ls command (to look at files) |
| **`ls –a`** | ls command with –a option |
| **`ls –a /home/2G03`** | ls command with –a option and a single argument |

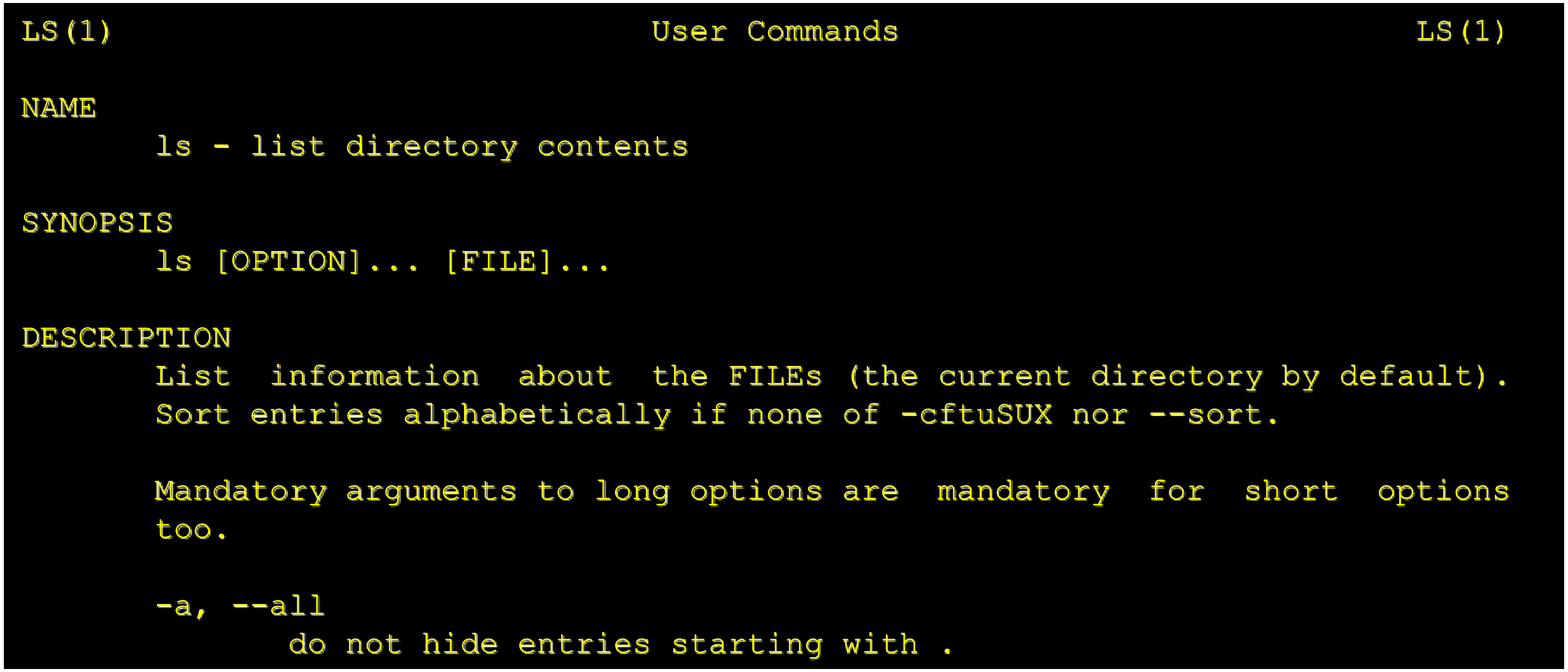
## Finding out more about commands

- Manual pages provide a way to discover what commands do and what options are available
- There is always google of course.  There are also reference books.

For example:

```
man ls
```

# ls Manual Page

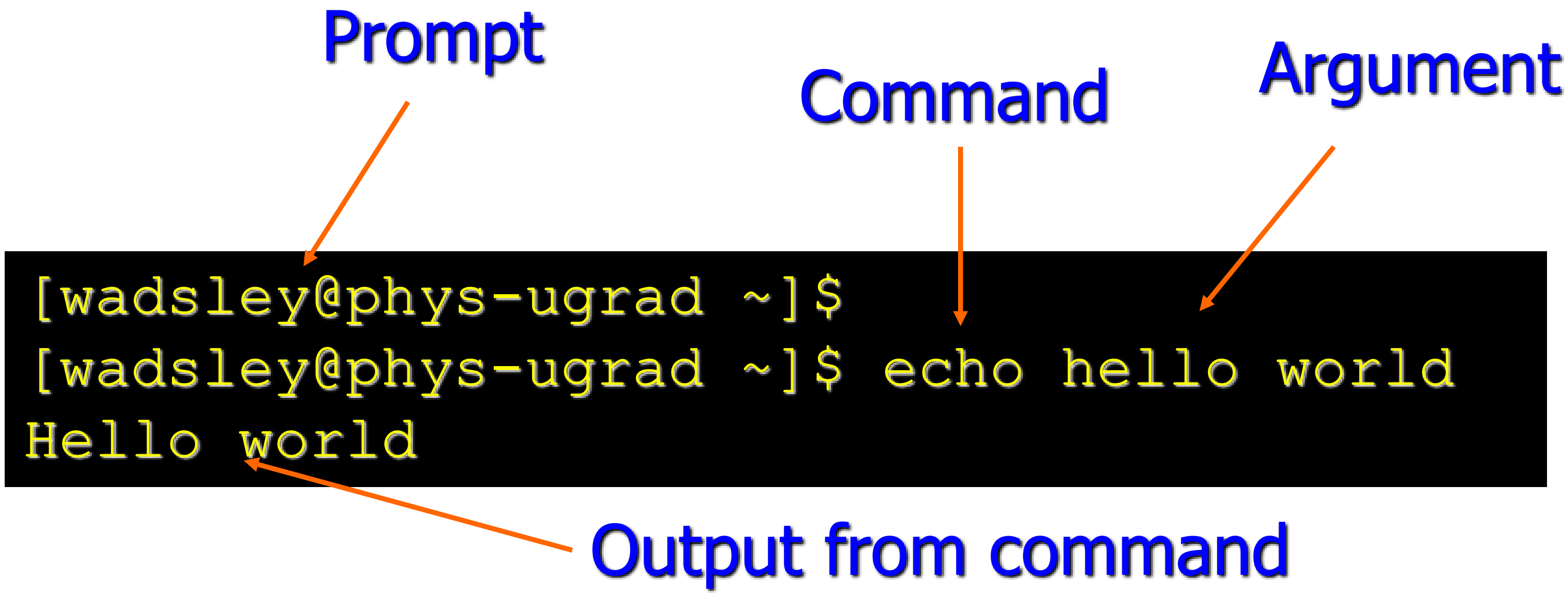


```
LS(1)                        User Commands                       LS(1)

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 -cftuSUX nor --sort.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a, --all
              do not hide entries starting with .
```

**q quit, space page down,  /string search for text**

## Unix Shells

- When you log in using ssh, phys-ugrad starts a tcsh shell program to interpret what you type at the command prompt

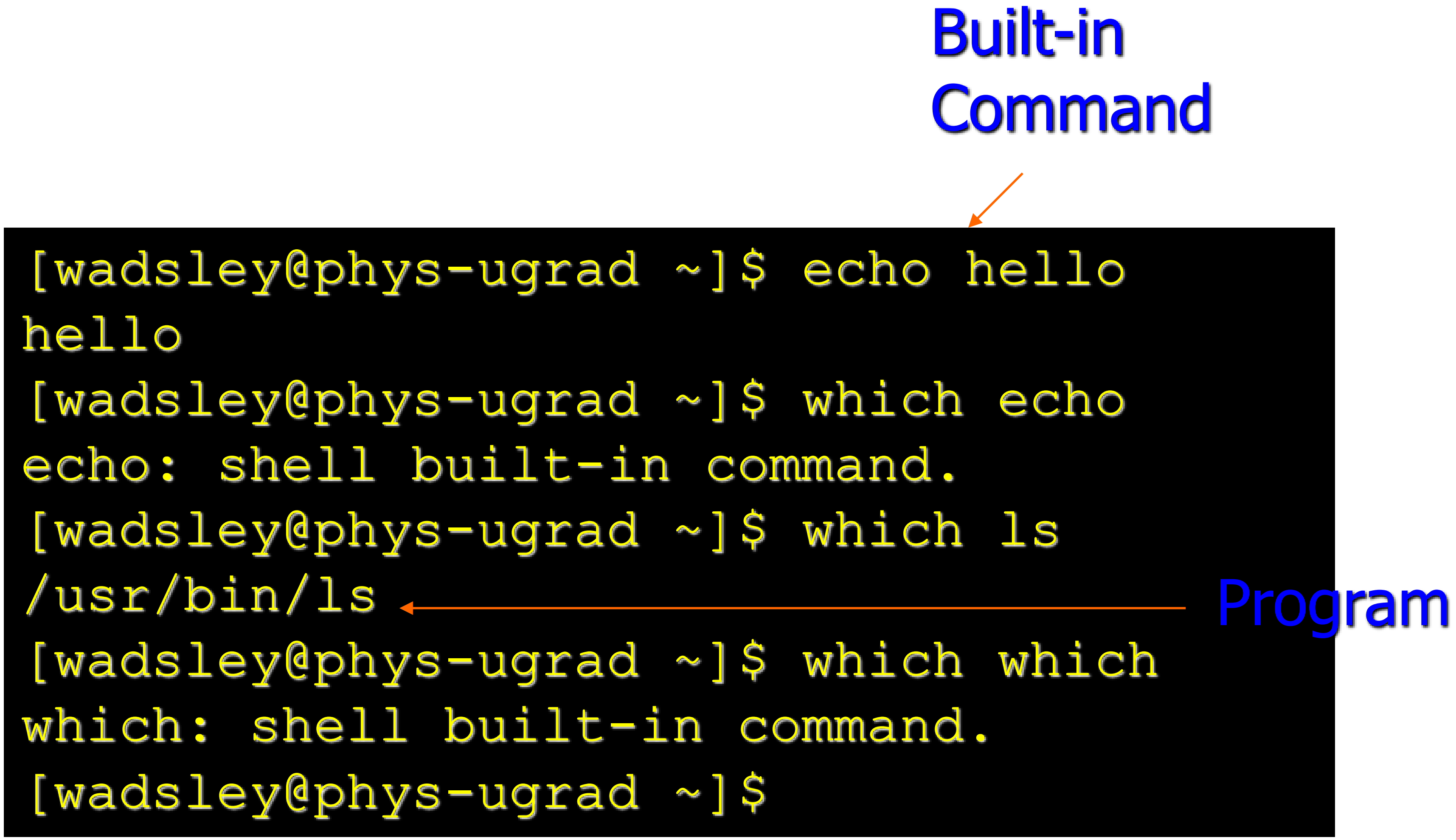


```
[wadsley@phys-ugrad ~]$
[wadsley@phys-ugrad ~]$ echo hello world
Hello world
```

## Built-in vs. program

- Some commands are built into tcsh and other are separate programs
- Use **`which`** to find out if a command is built-in to tcsh or if there is a program that does it

# Built-in commands vs. Programs



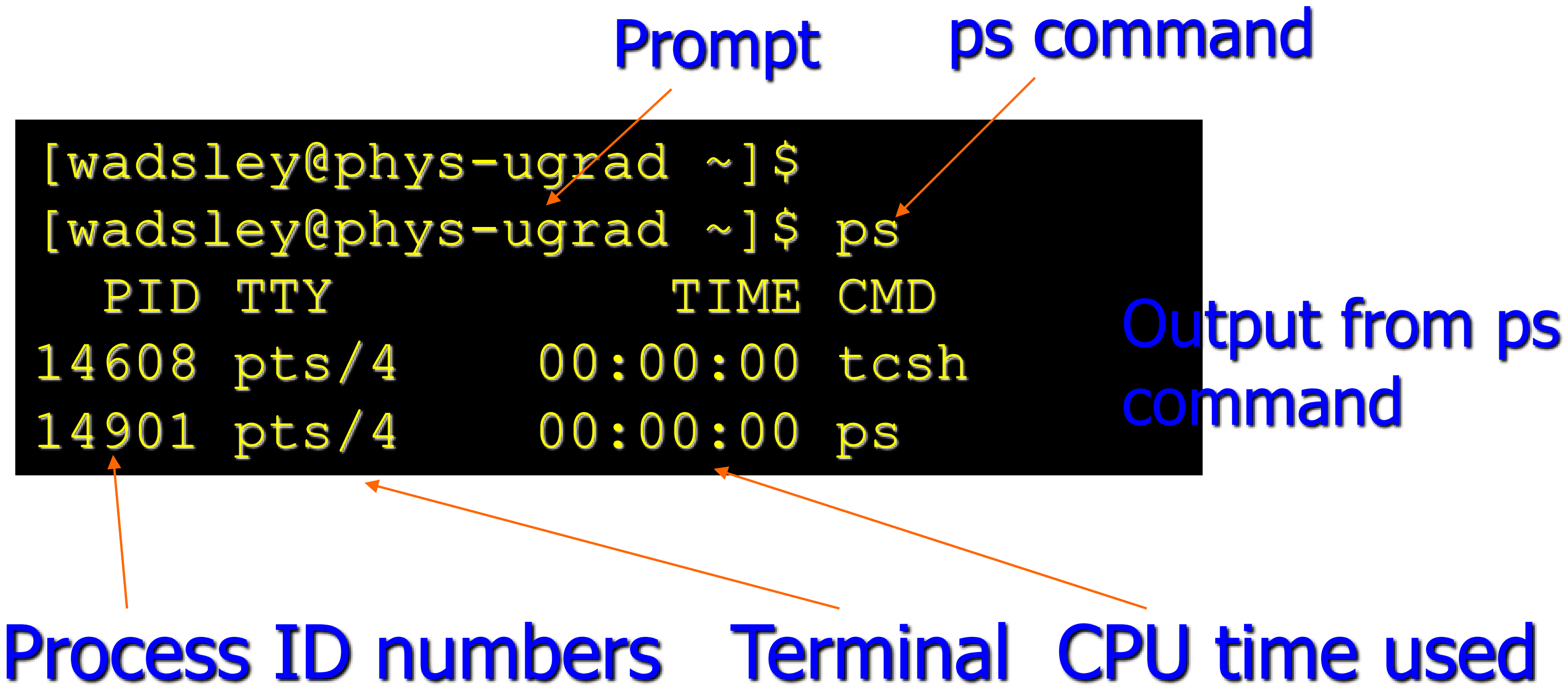
```
[wadsley@phys-ugrad ~]$ echo hello
hello
[wadsley@phys-ugrad ~]$ which echo
echo: shell built-in command.
[wadsley@phys-ugrad ~]$ which ls
/usr/bin/ls
[wadsley@phys-ugrad ~]$ which which
which: shell built-in command.
[wadsley@phys-ugrad ~]$
```

## Programs and processes

- If a new program is started you can see it listed as a process
- **ps** lists processes

  By default it lists the ones associated with your current terminal

# Unix Shells



```
[wadsley@phys-ugrad ~]$
[wadsley@phys-ugrad ~]$ ps
  PID TTY          TIME CMD
14608 pts/4    00:00:00 tcsh
14901 pts/4    00:00:00 ps
```

# Unix Shells

- When you start an xterm, it creates an xterm program to draw the terminal window AND another tcsh program to interpret what you type there…

- Note: Windows makes lots of processes too – you can see them with the task manager (ctrl-alt-delete). If a program doesn't work properly you can kill it in the task manager.

Unix gives you more direct control over this.

# Unix Shells

-u <user> option
Show all processes belonging to <user>

```
[wadsley@phys-ugrad ~]$
[wadsley@phys-ugrad ~]$ ps -u wadsley
  PID TTY          TIME CMD
14608 pts/0    00:00:00 tcsh
14901 pts/0    00:00:00 ps
[wadsley@phys-ugrad ~]$ xterm &
[wadsley@phys-ugrad ~]$ ps -u wadsley
  PID TTY          TIME CMD
14608 pts/0    00:00:00 tcsh
15600 pts/0    00:00:00 xterm
15690 pts/3    00:00:00 tcsh
15701 pts/0    00:00:00 ps
```

# Tcsh: Don’t want to wait?

- Some commands start a program which takes a long time to finish (or never does)
- If you don’t want to wait for it to stop, put it in the background with &
- e.g. **`xterm &`**

# File commands

| | |
|---|---|
| **`ls`** | List files |
| **`mv`** | Move or rename files |
| **`cp`** | Copy files |
| **`rm`** | Delete files |

Note: Two styles for cp, mv

**`cp file1 file2`** Copy single file

**`cp file1 file2 … fileN directory`**
Copy many files to directory

# File Commands

Examples. Here I am working in a directory called test that had no files to start with

```
[wadsley@phys-ugrad ~/test]$ ls
[wadsley@phys-ugrad ~/test]$ touch testfile1
[wadsley@phys-ugrad ~/test]$ ls
testfile1
[wadsley@phys-ugrad ~/test]$ mv testfile1 newnamefile
[wadsley@phys-ugrad ~/test]$ ls
newnamefile
[wadsley@phys-ugrad ~/test]$ cp newnamefile acopyofit
[wadsley@phys-ugrad ~/test]$ ls
acopyofit  newnamefile
[wadsley@phys-ugrad ~/test]$ ls *file
newnamefile
[wadsley@phys-ugrad ~/test]$ rm newnamefile
rm: remove regular empty file 'newnamefile'? y
[wadsley@phys-ugrad ~/test]$ ls
acopyofit
[wadsley@phys-ugrad ~/test]$
```

# Files

Tcsh has a simple set of *regular expressions* for matching files

| | |
|---|---|
| `ls /home/2G03` | everything |
| `ls /home/2G03/*.pdf` | everything ending in .pdf |
| `ls /home/?G03` | everything starting with any one character and ending in G03 |
| `ls HW[123]` | Matches HW1,HW2,HW3 |
| `ls [a-z]*` | Anything starting with a lowercase letter |

# Files

Wildcard summary:

|  |  |
|---|---|
| ? | Any one character |
| * | Zero or more characters |
| [abc] | Any one of the characters listed |
| [A-D] | Any one character in the range |
| {.cpp,.o} | One of the comma separated sets |

- e.g **`ls *.{cpp,o}`**

  List all files ending in .cpp or .o

# Special Directories

| | |
|---|---|
| ~/ | My home directory |
| ~bob/ | Bob’s home directory |
| ./ | The current directory |
| ../ | The directory above this one |

e.g. /home is the directory above /home/bob

# Absolute vs. relative path

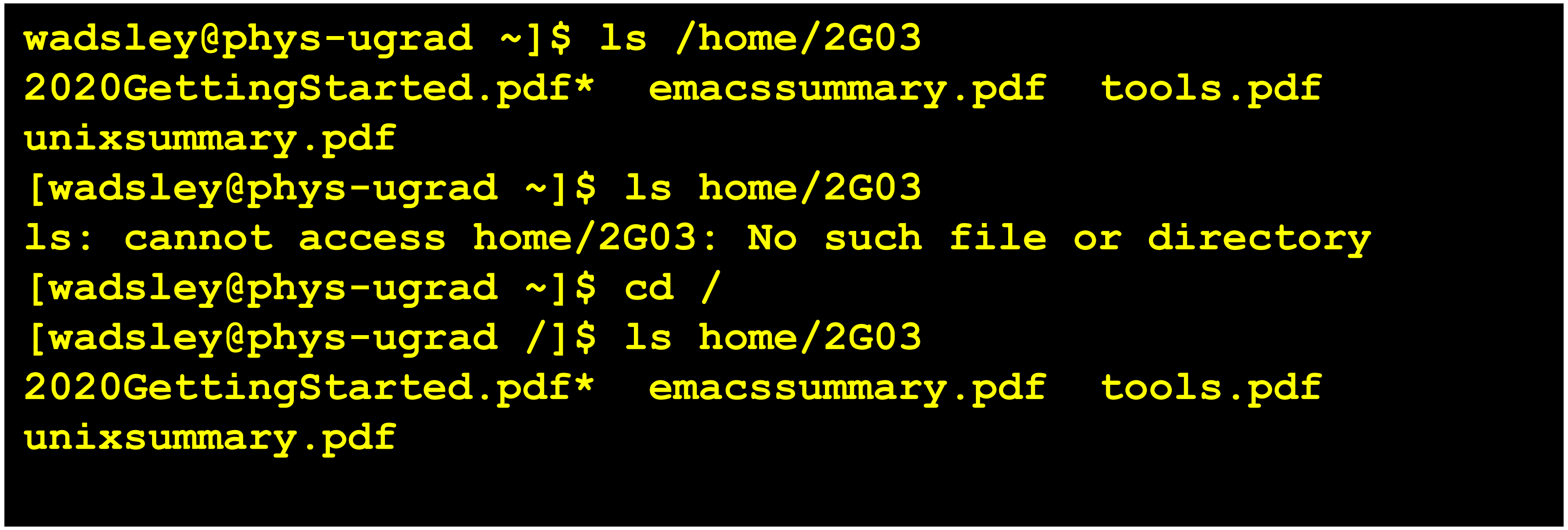
If you start with / it starts at the top of the file tree

e.g. ls /home/2G03 (absolute path)

If you don't it looks for the file or directory in your current directory

e.g. ls home (relative path)

This only works if you are in a directory with something called home there



```
wadsley@phys-ugrad ~]$ ls /home/2G03
2020GettingStarted.pdf*  emacssummary.pdf  tools.pdf
unixsummary.pdf
[wadsley@phys-ugrad ~]$ ls home/2G03
ls: cannot access home/2G03: No such file or directory
[wadsley@phys-ugrad ~]$ cd /
[wadsley@phys-ugrad /]$ ls home/2G03
2020GettingStarted.pdf*  emacssummary.pdf  tools.pdf
unixsummary.pdf
```

# Directory commands
# Directories are Windows Folders

| | |
|---|---|
| **`pwd`** | My current directory |
| **`cd`** | Change to new directory |
| **`mkdir`** | Make a new directory |
| **`rmdir`** | Remove empty directory |

Note: Your current directory is probably part of your prompt:

```
[wadsley@phys-ugrad ~]$ cd tmp
[wadsley@phys-ugrad ~/tmp] pwd
/1/home/wadsley/tmp
```

## Directory Commands

```
wadsley@phys-ugrad ~]$ mkdir test
[wadsley@phys-ugrad ~]$ cd test
[wadsley@phys-ugrad ~/test]$ ls
[wadsley@phys-ugrad ~/test]$ touch file1 file2 file3
[wadsley@phys-ugrad ~/test]$ ls
file1  file2  file3
[wadsley@phys-ugrad ~/test]$ mkdir junk
[wadsley@phys-ugrad ~/test]$ ls
file1  file2  file3  junk/
[wadsley@phys-ugrad ~/test]$ mv file2 junk
[wadsley@phys-ugrad ~/test]$ ls
file1  file3  junk/
[wadsley@phys-ugrad ~/test]$ ls junk
file2
[wadsley@phys-ugrad ~/test]$ rm junk
rm: cannot remove 'junk': Is a directory
[wadsley@phys-ugrad ~/test]$ rmdir junk
rmdir: failed to remove 'junk': Directory not empty
[wadsley@phys-ugrad ~/test]$ rm junk/file2
rm: remove regular empty file 'junk/file2'? y
[wadsley@phys-ugrad ~/test]$ rmdir junk
[wadsley@phys-ugrad ~/test]$ ls
file1  file3
[wadsley@phys-ugrad ~/test]$
```

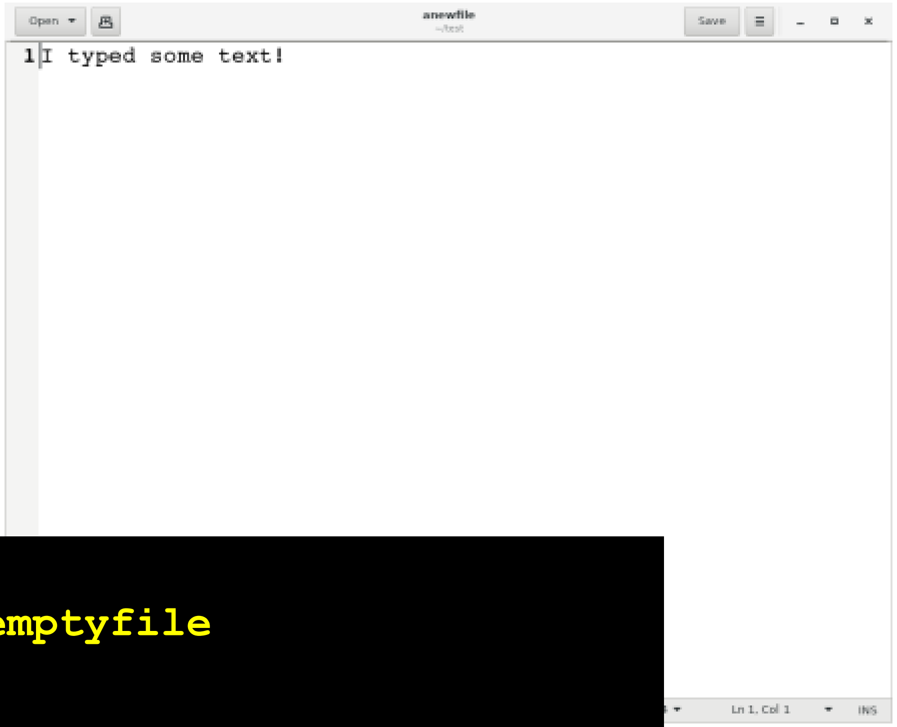
## Text file utilities

Utilities to look at files that are text:

| | |
|---|---|
| `gedit` | like windows notepad editor |
| `micro` | in terminal editor |
| `xemacs` | pop-up emacs editor |
| `more file` | Look at file one page at a time |
| `less file` | Like more but smarter |
| `head file` | Look at the top 10 lines |
| `tail –5 file` | Look at the last 5 lines |

# Making files

touch vs. gedit editor



```
I typed some text!
```

```
[wadsley@phys-ugrad ~/test]$ ls
[wadsley@phys-ugrad ~/test]$ touch emptyfile
[wadsley@phys-ugrad ~/test]$ ls -l
total 0
-rw-rw-r-- 1 wadsley wadsley 0 Sep  9 13:32 emptyfile
[wadsley@phys-ugrad ~/test]$ gedit anewfile &
[1] 22588
[wadsley@phys-ugrad ~/test]$ ls -l
total 12
-rw-rw-r--  1 wadsley wadsley   19 Sep  9 13:32 anewfile
-rw-rw-r--  1 wadsley wadsley    0 Sep  9 13:32 emptyfile
[wadsley@phys-ugrad ~/test]$ more anewfile
I typed some text!
[wadsley@phys-ugrad ~/test]$ more emptyfile
[wadsley@phys-ugrad ~/test]
```

# Searching text files

grep is a really useful command to look for specific words, variable names, etc… in one or many files

e.g.

```
grep main *.cpp
```

Search all files given for lines with the word main in them

```
Grep main /home/2G03/hello/*
```

# Regular Expressions

- Grep has a very powerful set of *regular expressions* for matching text
- It is much bigger than the simple use of *, ? for files with tcsh
- We will revisit this later…

## Tcsh: jobs

- Jobs is a built in tcsh command to look at commands you entered that are still going
- Jobs is a bit more user friendly than dealing with process ID numbers

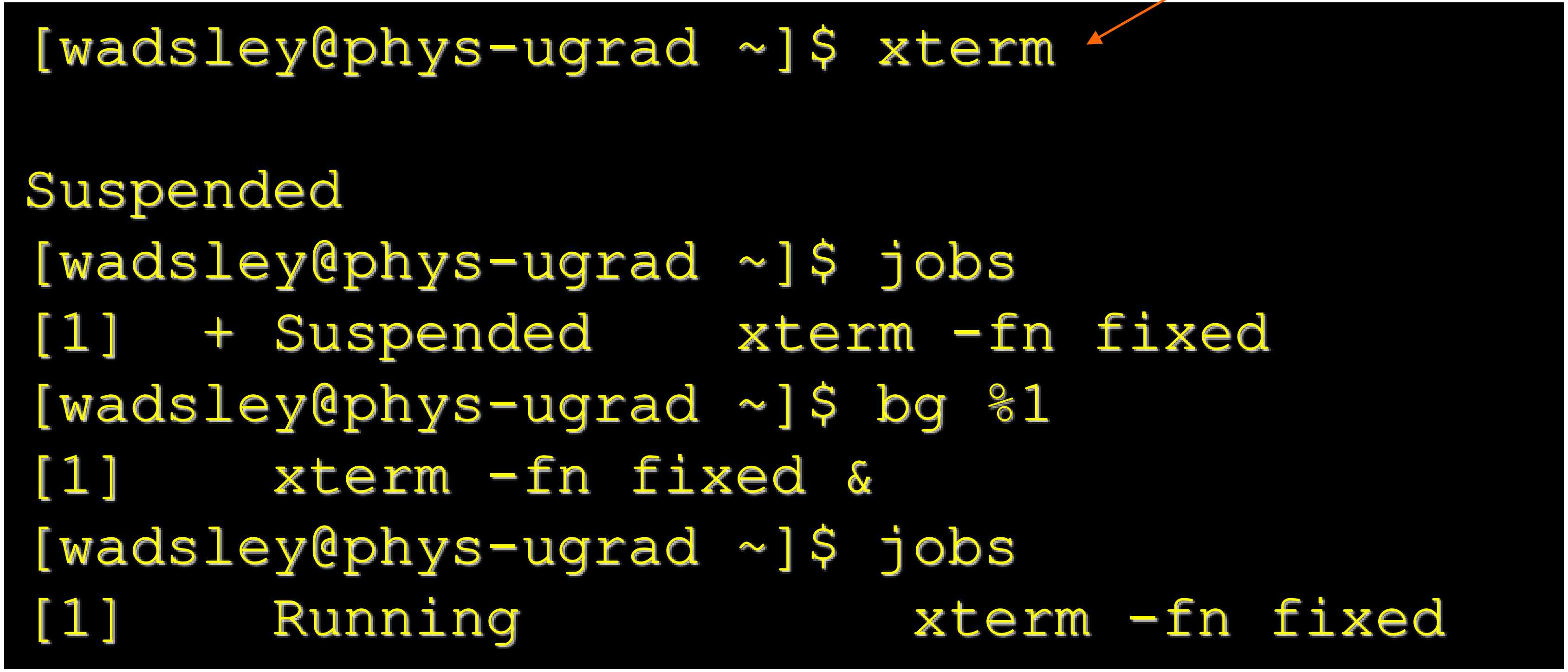
# jobs

```
[wadsley@phys-ugrad ~]$ xterm &
[wadsley@phys-ugrad ~]$ jobs
[1]  + Running      xterm -fn fixed
[wadsley@phys-ugrad ~]$ kill %1
[wadsley@phys-ugrad ~]$ jobs
[1]  Exit 15      xterm -fn fixed
```

But what if you forget to use the & to put something in the background?

## Switching foreground to background

Control-Z here



```
[wadsley@phys-ugrad ~]$ xterm

Suspended
[wadsley@phys-ugrad ~]$ jobs
[1]  + Suspended    xterm -fn fixed
[wadsley@phys-ugrad ~]$ bg %1
[1]    xterm -fn fixed &
[wadsley@phys-ugrad ~]$ jobs
[1]    Running             xterm -fn fixed
```

## Tcsh: job management

Make an xterm and then put it in the background:

Try these steps (C-z is control z)

```
xterm
  C-z
jobs
bg
```

# Tcsh: job management

| | |
|---|---|
| jobs | List jobs associated with this terminal |
| fg %2 | put job 2 in the foreground |
| bg %1 | put job 1 in the background |
| kill %1 | kill job 1 |
| C-c | Kill a foreground job |

- C-z suspend, C-s pause, C-q continue

C- means hold down the control key first

*Important:* ***Control C*** *will get you out of most programs (it forces a quit)*

## Tcsh: Command Line

- Unix potentially involves a lot of typing
- To avoid this, lots of short cuts have been developed

e.g.

- Up and down arrow – reuse previous commands

## Tcsh: Command Line history

See old commands Enter: history

- Enter: **`!3`** run 3rd command in history again
- Enter: **`!hi`** run last command starting with hi

# Tcsh: TAB
# Command completion

<TAB>: Hit the **tab** button to attempt to finish the command or filename. Very useful to avoid typing out long filenames

- **`xter<TAB>`** $\rightarrow$ **`xterm`**
- **`ls /home/2G<TAB>`** $\rightarrow$ **`ls /home/2G03/`**

## Tcsh Command Line

- emacs editor shortcuts work on the tcsh command line too
- E.g. Ctrl-A start of line, Ctrl-E end of line

(There is an emacs short cut summary handout on avenue)

# Tcsh: Mouse

- Under X windows, xterm, emacs, you can use the mouse to cut and paste text
- button 1 (left) – mark text (try multiple clicks)
- button 3 (right) – cut text (menu)
- button 2 (middle) – paste text (macbook: option click) (Windows: SHIFT insert works if no middle button)

## Redirecting output to files >

- Unix was designed to make files and programs work together well
- Redirection > put output from a program into a file

Try:
```
ls /home/2G03 > junk
more junk
```

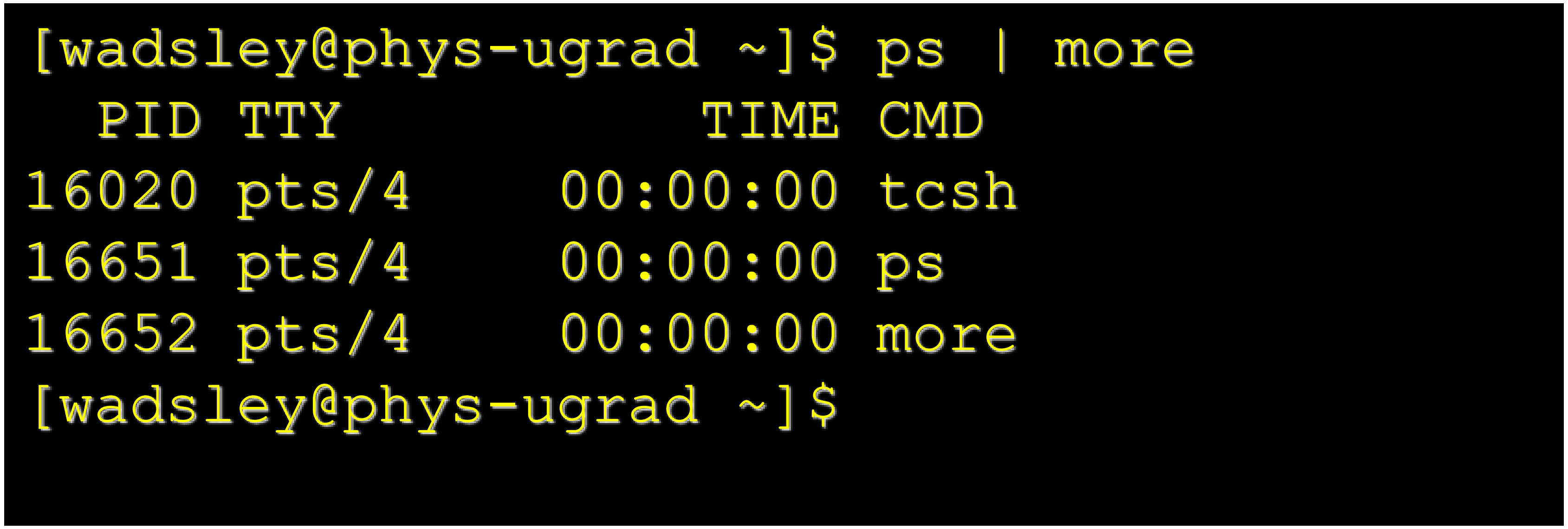
More is a program to look at text one page at a time

## Pipes |

- Programs can work together too
- Try: `ls /bin/* | more`

The output from ls is put into the more program. All of the programs run at the same time.

# Pipe | Example



```
[wadsley@phys-ugrad ~]$ ps | more
  PID TTY          TIME CMD
16020 pts/4    00:00:00 tcsh
16651 pts/4    00:00:00 ps
16652 pts/4    00:00:00 more
[wadsley@phys-ugrad ~]$
```

more starts just after ps and waits for the output from ps

## Using Text file utilities with | (pipe)

```
See files in order of most recent first
[wadsley@phys-ugrad ~]$ ls -lt /bin/
...lots of files...
[wadsley@phys-ugrad ~]$ ls -lt /bin/ | head
total 350784
-rwxr-xr-x    1 root root       28104 Jul  3 09:52 gencat*
-rwxr-xr-x    1 root root       35680 Jul  3 09:52 getent*
-rwxr-xr-x    1 root root       69688 Jul  3 09:52 iconv*
-rwxr-xr-x    1 root root       46616 Jul  3 09:52 locale*
-rwxr-xr-x    1 root root      339632 Jul  3 09:52 localedef*
-rwxr-xr-x    1 root root       19264 Jul  3 09:52 pldd*
-rwxr-xr-x    1 root root       28824 Jul  3 09:52 sprof*
-rwxr-xr-x    1 root root       26544 Jul  3 09:52 getconf*
-rwxr-xr-x    1 root root       24976 Jul  3 09:52 makedb*
```

# Redirection operators

- command < file

  Take input from file

- command > file

  Put output into file (overwrite file)

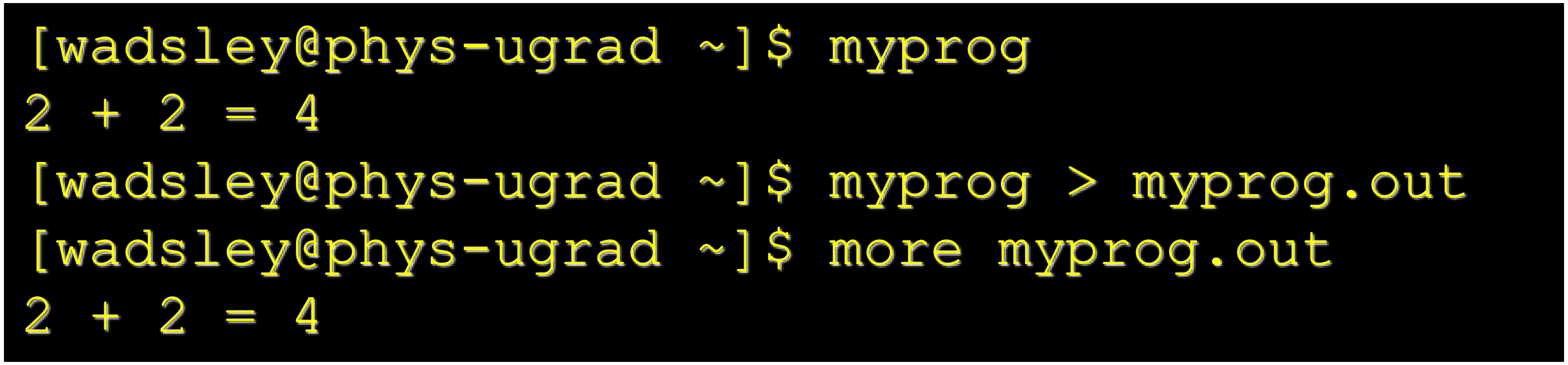
- command >> file

  Put output at end of file

- command1 | command2

  Pipe output from command1 into command 2

# Redirection and Homework



```
[wadsley@phys-ugrad ~]$ myprog
2 + 2 = 4
[wadsley@phys-ugrad ~]$ myprog > myprog.out
[wadsley@phys-ugrad ~]$ more myprog.out
2 + 2 = 4
```

Redirection with > provides an easy way to capture the output of HW programs you write to hand in the results

# Experiment!

People learn Unix by doing things

I encourage you to try things

Remember that man pages exist to give you help on commands, also refer to the handout and internet resources:  google!

You don’t need to be an expert on all of unix/linux – just a few basic commands such as the ones here are enough.