

CSCI 553: Networking III

Unix Network Programming

Spring 2008



Shell Basics



Today's Agenda

- Sign NISL account forms, Determine assignment 0.1 completion
- Demo of ssh & x windows access
 - Windows ssh access, if possible
 - ssh and X access from a Linux
 - Ubuntu virtual machine in Computer Lab
- Command Line Editing (Handout)
- Shell
- More Shell
 - Special Characters (CTRL-C, CTRL-D, CTRL-S/Q, CTRL-Z)
- Editors
 - vi (Handout)
 - kate



Assignment 0.1

- Sign NISL account forms
- Determine assignment 0.1 completion



SSH & X Window Access

- Demo of SSH and X Window access
 - MS Windows ssh access, if possible
 - ssh and X access from a Linux
 - Ubuntu virtual machine in Computer Lab (new nisl account access handout)



Command Line Editing

- Command Line Editing (Handout)

An Aside: Command Line Editing



- up-arrow: previous command
- down-arrow: next command
- ctrl-a: go to beginning of line
- ctrl-e: go to end of line
- tab: complete current command or file name



Shell Basics

Introduction to the Unix Command Line



An Aside: A Note on Linux and Unix Philosophy (The Big Picture)

- Most of the tools used in this course are freely available under various open source licenses
- Makes Unix/Linux development attractive for this reason along
- However, Unix philosophy of software development goes even beyond this, and is embodied in many of the tools we will look at.
 - We will learn more about this as the course progresses and you get more familiar with Unix/Linux environment.



An Aside: A Note on Linux and Unix Philosophy (The Big Picture)

— UPU Ch #1 Basics of Unix Philosophy

- Rule of Modularity: Write simple parts connected by clean interfaces.
- Rule of Clarity: Clarity is better than cleverness.
- Rule of Composition: Design programs to be connected to other programs.
- Rule of Separation: Separate policy from mechanism; separate interfaces from engines.
- Rule of Simplicity: Design for simplicity; add complexity only where you must.
- Rule of Parsimony: Write a big program only when it is clear by demonstration that nothing else will do.
- Rule of Transparency: Design for visibility to make inspection and debugging easier.
- Rule of Robustness: Robustness is the child of transparency and simplicity.
- Rule of Representation: Fold knowledge into data so program logic can be stupid and robust.
- Rule of Least Surprise: In interface design, always do the least surprising thing.
- Rule of Silence: When a program has nothing surprising to say, it should say nothing.
- Rule of Repair: When you must fail, fail noisily and as soon as possible.
- Rule of Economy: Programmer time is expensive; conserve it in preference to machine time.
- Rule of Generation: Avoid hand-hacking; write programs to write programs when you can.
- Rule of Optimization: Prototype before polishing. Get it working before you optimize it.
- Rule of Diversity: Distrust all claims for "one true way".
- Rule of Extensibility: Design for the future, because it will be here sooner than you think.



Introduction

- Most modern tools have a graphical user interface (GUI)
 - They're easier to use
 - A picture is worth a thousand words
- But command-line user interfaces (CLUIs) still have their place
 - Easier to build a simple CLUI than a simple GUI
 - Higher action-to-keystroke ratio
 - Once you're over the learning curve
 - Easier to see and understand what the computer is doing on your behalf
 - Which is part of what this course is about
 - Most important: it's easier to combine CLUI tools than GUI tools
 - Small tools, combined in many ways, can be very powerful

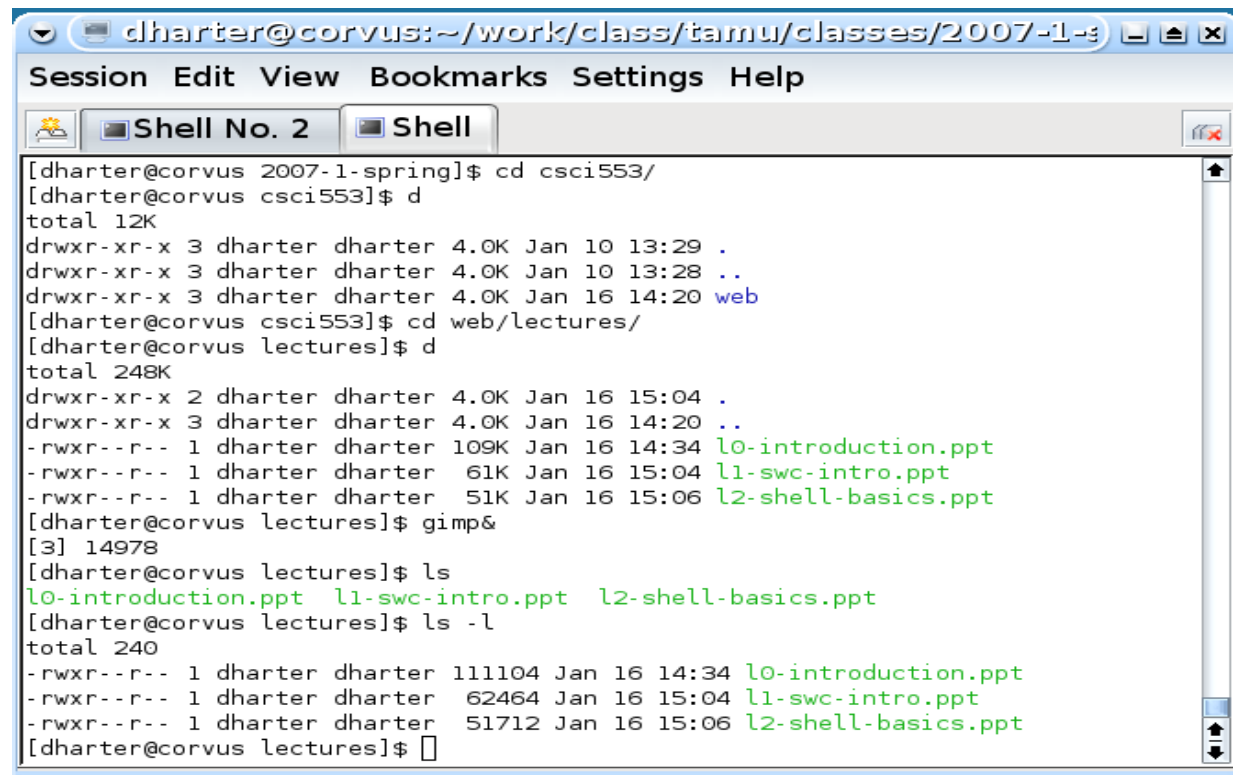


Introduction

- You are encouraged to follow along and try out things as they are presented
 - In this and future lectures.
- Please switch drivers, occasionally, if you are sharing a computer.
- K-Menu->System->Terminal Program (Konsole)

The Shell

- The most important command-line tool is the *command shell*
 - Usually just called “the shell”
 - Looks (and works) like an interactive terminal circa 1980



```
dharter@corvus:~/work/class/tamu/classes/2007-1-1-3
Session Edit View Bookmarks Settings Help
Shell No. 2 Shell
[dharter@corvus 2007-1-spring]$ cd csci553/
[dharter@corvus csci553]$ d
total 12K
drwxr-xr-x 3 dharter dharter 4.0K Jan 10 13:29 .
drwxr-xr-x 3 dharter dharter 4.0K Jan 10 13:28 ..
drwxr-xr-x 3 dharter dharter 4.0K Jan 16 14:20 web
[dharter@corvus csci553]$ cd web/lectures/
[dharter@corvus lectures]$ d
total 248K
drwxr-xr-x 2 dharter dharter 4.0K Jan 16 15:04 .
drwxr-xr-x 3 dharter dharter 4.0K Jan 16 14:20 ..
-rwxr--r-- 1 dharter dharter 109K Jan 16 14:34 l0-introduction.ppt
-rwxr--r-- 1 dharter dharter 61K Jan 16 15:04 l1-swc-intro.ppt
-rwxr--r-- 1 dharter dharter 51K Jan 16 15:06 l2-shell-basics.ppt
[dharter@corvus lectures]$ gimp&
[3] 14978
[dharter@corvus lectures]$ ls
l0-introduction.ppt l1-swc-intro.ppt l2-shell-basics.ppt
[dharter@corvus lectures]$ ls -l
total 240
-rwxr--r-- 1 dharter dharter 111104 Jan 16 14:34 l0-introduction.ppt
-rwxr--r-- 1 dharter dharter 62464 Jan 16 15:04 l1-swc-intro.ppt
-rwxr--r-- 1 dharter dharter 51712 Jan 16 15:06 l2-shell-basics.ppt
[dharter@corvus lectures]$
```

Fig L2.1: A Shell in Action



The Shell

- Manages a user's interactions with the operating system by:
 - Reading commands from the keyboard
 - Executing those commands...
 - ...or running another program
 - Displaying the output
- The shell is just one program among many
 - Many different shells have been written
 - The Bourne shell, called `sh`, is an ancestor of many of them
 - It's still a lowest common denominator that you can always rely on
- We'll use `bash` (the Bourne Again Shell) in this course

The Shell is Not the Operating System

- The *operating system* is *not* just another program
 - Automatically loaded when the computer boots up
 - Runs everything else (including shells)
- The OS manages the computer's hardware
 - Provides a common interface to different chips, disks, network cards, etc.
 - So that user-level applications can run anywhere
- The OS also keeps track of what programs are running, what privileges they have, etc.
 - Which makes it crucial to security

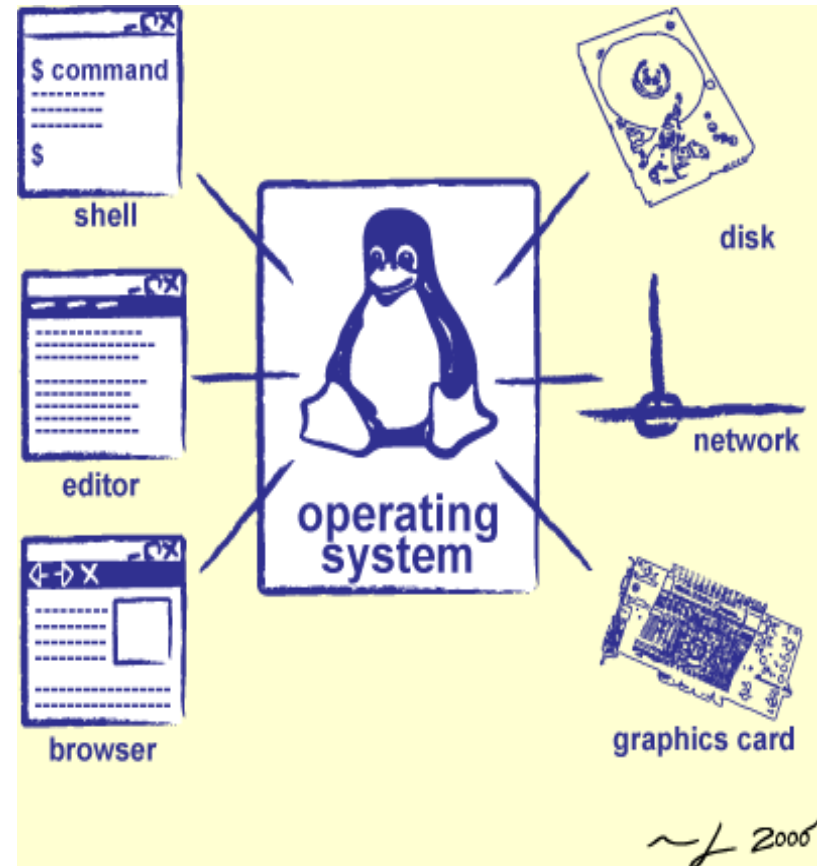


Figure L2.2: Operating System

The File System

- The *file system* is the set of files and directories the computer can access
 - “Everything that doesn't go away when you reboot”
- Data is stored in files
 - By convention, files have two part names, like notes.txt or home.html
 - Most operating systems allow you to associate a *filename extension* with an application
 - E.g., .txt is associated with an editor, and .html with a web browser
 - But this is all just convention: you can call files (almost) anything you want
- Files are stored in directories (often called folders)
 - Directories can contain other directories, too
 - Results in the familiar *directory tree*
 - Everything in a particular directory must have a unique name
 - Otherwise, how would you identify it?
 - But items in different directories can have the same name
- On Unix, the file system has a unique *root directory* called /
 - Every other directory is a child of it, or a child of a child, etc.
- On Windows, every *drive* has its own root directory
 - So C:\home\gwwilson\notes.txt is different from J:\home\gwwilson\notes.txt
- Note: file and directory names are case-sensitive on Unix, but case-*ins*sensitive on Windows
 - So don't ever rely on case differences when naming things

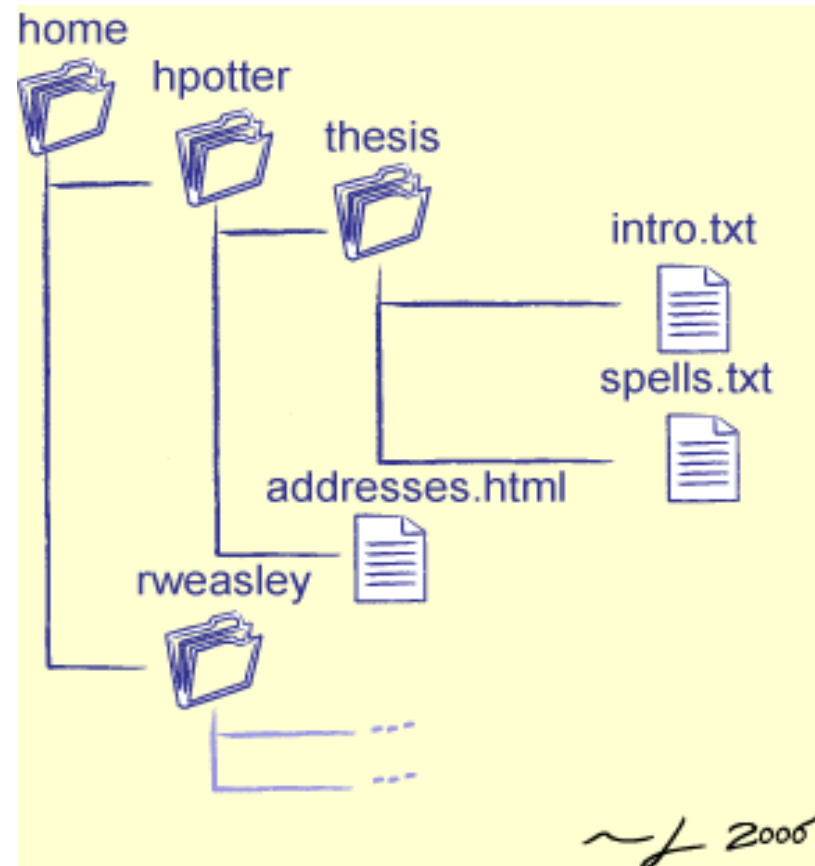


Figure L2.3: A Directory Tree

Paths

A *path* is a description of how to find something in a file system

- An *absolute path* describes a location from the root directory down
 - Equivalent to a street address
 - Always starts with "/"
 - /home/hpotter is Harry Potter's home directory
 - /courses/swc/web/lec/shell.html is this file
- A *relative path* describes how to find something from some other location
 - Equivalent to saying, "Four blocks north, and seven east"
 - From /courses/swc, the relative path to this file is web/lec/shell.html

Every program (including the shell) has a *current working directory*

- "Where am I?"
- Relative paths are deciphered relative to this location
- It can change while a program is running

Two special directory names:

- "." (pronounced "dot") is the current directory
- ".." (pronounced "dot dot") is the directory one level up
 - Also called the *parent directory*
 - In /courses/swc/data, .. is /courses/swc
 - In /courses/swc/data/elements, .. is /courses/swc/data

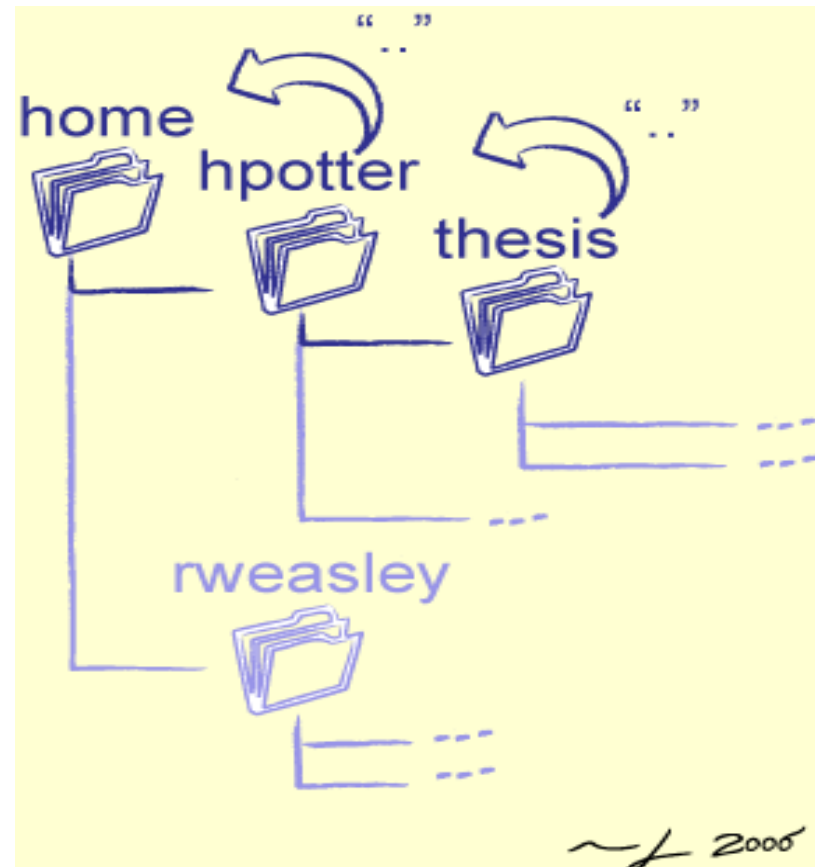


Figure L2.4: Parent Directories



Navigating the File System

- Easiest way to learn basic Unix commands is to see them in action
- Type `pwd` (short for "print working directory") to find out where you are
 - Unfortunately, most Unix commands have equally cryptic names

```
$ cd swc
$ pwd
/home/csci553/hpotter/swc
```
- Then type `ls` (for "listing") to see what's in the current directory

```
$ ls
LICENSE.txt conf data docs index.swc license.swc print.css swc.css tests Makefile config.mk depend.mk img lec press
sites swc.dtd util
```
- To see what's in the data directory, type `ls data`

```
$ ls data
bio elements haiku.txt morse.txt pdb solarsystem.txt
```
- Or:
 - Type `cd data` to "go into" data
 - I.e., change the current working directory to data
 - Type `ls` on its own
 - Type `cd ..` to go back to where you started

```
$ cd data
$ pwd
/home/hpotter/swc/data
$ ls
bio elements haiku.txt morse.txt pdb solarsystem.txt
$ cd ..
$ pwd /home/csci553/hpotter/swc
```

Execution Cycle

When you type a command like ls, the OS:

- Reads characters from the keyboard
- Passes them to the shell (because it's the currently active window)

The shell:

- Breaks the line of text it receives into words
- Looks for a program with the same name as the first word
 - See in a moment how the shell knows where to look
- Runs that program

That program's output goes back to the shell...

...which gives it to the OS...

...which displays it on the screen

- (Actually, the OS hands it to the window manager, which takes care of the display)

All well-designed software systems work this way

- Break the task down into pieces
- Write a tool that solves each sub-problem
- Hook 'em up

Allows you to:

- Develop and test components independently
- Replace or re-use components incrementally
- Add new components as you go along

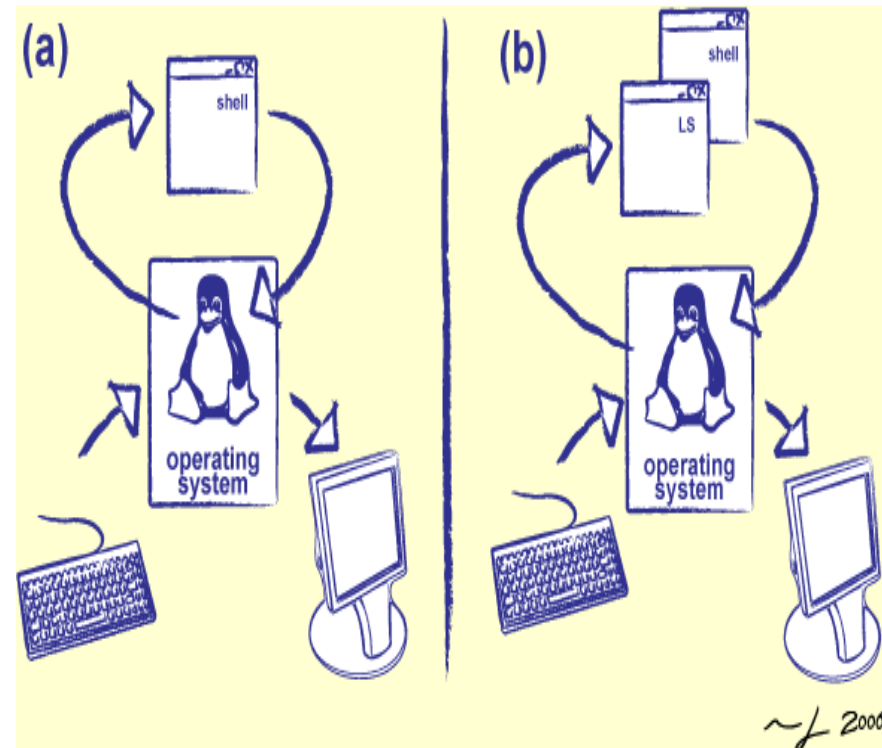


Figure L2.5: Running a Program



Providing Options

- Can make ls produce more informative output by giving it some *flags*
 - By convention, flags for Unix tools start with "-", as in "-c" or "-l"
 - Some flags take arguments (such as filenames)
- Show directories with trailing slash

```
$ ls -F
```

```
LICENSE.txt conf/ data/ docs/ index.swc license.swc print.css swc.css tests/ Makefile  
config.mk depend.mk img/ lec/ press/ sites/ swc.dtd util/
```

- Show all files and directories, including those whose names begin with .
 - By default, ls doesn't show things whose names begin with .
 - So that . and .. don't always show up

```
$ ls -a
```

```
.. .svn Makefile config.mk depend.mk img lec press sites swc.dtd util  
.. LICENSE.txt conf data docs index.swc license.swc print.css swc.css tests
```

- We'll see what the .svn directory is for later

- Flags can be combined

```
$ ls -a -F
```

```
.. .svn/ Makefile config.mk depend.mk img/ lec/ press/ sites/ swc.dtd util/  
.. LICENSE.txt conf/ data/ docs/ index.swc license.swc print.css swc.css tests/
```



Providing Options

- man is your FRIEND
 - The basic set of Unix commands use man pages to document options
 - Options are a well supported paradigm in Unix, we will look again at option parsers in c & python.



Creating Files and Directories

- Rather than messing with the course files, let's create a temporary directory and play around in there

```
$ mkdir tmp
```

- Note: no output (but -v ("verbose") would tell mkdir to print a confirmation message)

- Go into that directory: no files there yet

```
$ cd tmp
```

```
$ ls
```

- Use the editor of your choice to create a file called earth.txt with the following contents:

```
Name: Earth
```

```
Period: 365.26 days
```

```
Inclination: 0.00
```

```
Eccentricity: 0.02
```

- Notepad (on Windows) runs in a window of its own
- Pico (on Unix) takes over the shell window temporarily
- I would suggest for now learning basic vi, and using kate when you get to larger programming projects
- We'll look at more advanced programming editors later

- Easiest way to create a similar file venus.txt is to copy earth.txt and edit it

```
$ cp earth.txt venus.txt
```

```
$ vi venus.txt
```

```
$ ls -t venus.txt earth.txt
```

- -t tells ls to list by modification time, instead of alphabetically



Looking at Files

- Check the contents of the file using `cat` (short for “concatenate”)
 - Prints the contents of a file to the screen

```
$ cat venus.txt
Name: Venus
Period: 224.70 days
Inclination: 3.39
Eccentricity: 0.01
```
- Compare the sizes of the two files using:
 - `ls -l` (“-l” meaning “long form”)

```
$ ls -l
total 2
-rwxr-xr-x 1 gvwilson None 73 Jan 4 15:58 earth.txt
-rwxr-xr-x 1 gvwilson None 73 Jan 4 15:58 venus.txt
```

 - Fifth column is size in bytes
- `wc` (for “word count”)

```
$ wc earth.txt venus.txt
4 9 73 earth.txt
4 9 73 venus.txt
8 18 146 total
```

 - Columns show lines, words, and characters

Basic Tools



* Especially these

man **	Documentation and help for commans
cat	Concatenate and display text files
cd *	Change working directory
clear	Clear the screen.
cp *	Copy files and directories.
date	Display the current date and time.
diff	Show differences between two text files.
echo	Print arguments.
head	Display the first few lines of a file.
ls *	List files and directories.
mkdir *	Make directories.
more / less	Page through a text file.
mv	Move (rename) files and directories
od	Display the bytes in a file (octal dump)
passwd	Change your password.
pwd	Print current working directory.
rm *	Remove files.
rmdir	Remove directories.
sort	Sort lines.
tail	Display the last few lines of a file.
uniq	Remove adjacent duplicate lines.
wc	Count lines, words, and characters in file.



Summary

- Command-line tools will be with us for a long time
 - Easiest way to do many simple tasks
 - Easiest way to see what the computer is actually doing
- Often the only thing you can rely on having on a new machine
 - A handful of basic commands will get you a long way