Learning - What's important – is what you need..

- In education and in life you need the skills
  - to prune to bare essentials
  - & know where to get more info if and when needed.
  - The Basics – bare essentials – enough to get you around.
    - Overview – why, what & how…
    - Basic commands and facilities
  - The details – of where to get more info. When needed.
    - A more detailed overview without
      - overload & overwhelm & overstress
  - The details of where to get more info. / an answer
    - System : man, apropos
    - Online : tutorials, examples, blogs,
    - Books : basic – advanced.

Learning - What's important – is what you need..

Give a man a fish, and you feed him for a day,

Teach him how to fish, and he feeds himself for life!

This course will not teach you all about Unix,

but it should give you all you need to know,

To find out what you will need to know about Unix!

You should be able to read and understand Unix

- installation and configuration instructions
- most scripts

What is Unix?

- A multi-user networked operating system
  - "Operating System"
    - Handles files, running other programs, input/output
    - Just like DOS or Windows
  - "Networked"
    - Designed for server use
    - Networking is an intrinsic part of the system
  - "Multi-user"
    - Every user has different settings and permissions
    - Multiple users can be logged in simultaneously
- Tons of fun!!! ☺ (Get a life!)
  - But it might get you a job or make it easier!

Basic Course Overview – enough to get going!

- Basic commands
  - Files
  - Directories
  - Access management - modes
  - Disk space management
  - Processes & process management
- Editors
  - Basic ubiquitous :
    - Ed /Sed / Vi(m) / pico/nano
  - GUI
  - IDE
- Tools & Utilities
  - Grep – basically to find strings, either names, contents, but flexible & programmable with regular expressions : a way of specifying string patterns (grep:- global regular expression print)
  - AWK (open source gawk) -
- Extras
  - bash scripting
  - ssh/VPN
- System Administration
  - System configuration
  - User administration
  - Software installation / compiles etc.

No point in reinventing the wheel,
So rather than developing a script,
Check commands and options,
And tools and options
Before attempting to write a script

Then write a script if unavoidable
Simplifying it by Using the powerful
Commands & options, and tools
covered above!

Handy shortcuts – to minimise typing!?

- Autocompletion…
  - When entering a filename, give it a start, with a few initial characters and press TAB, and the system will autocomplete as far as possible:
    - If the start is unique, then it will complete the filename
    - If the start is not unique, then it will go as far as it can
      - Eg if in /users and type cd cs TAB
        then it will autocomplete to csdipact201
        and expect you to finish off with 2 or 3
- History – saves retyping previous commands, especially handy to :-
  - Fix an error, just run the cursor over (arrow keys not mouse) retype corrections
  - Re-run a complex command where
    - you are liable to make another error,
    - or can't be bothered rethinking and/or retyping it
  - Check out the history to see how you ended up with such wonderful results!?

Beauty of Unix - philanthropic files!?

Cuts out user having to manage (make & clear) temporary middle files!

- All data are handled as files, which are simply a sequence of bytes
  - Including devices : screen, keyboard, printer, and of course…disk,
- Files can be
  - Redirected…using redirection operators '<', '>' …cmd <inputfile >outputfile
    - So screen output can be sent to a file instead
    - And input taken from a file instead of keyboard & vice-versa too
  - Appended…>>append_outputfile
    - Added to an existing file, rather than overwriting the existing one.
  - Piped from program to program : command1 | command2 | command3 …
    - Instead of the user having to create temporary files, with all the time, typing (errors), naming & reclaiming, the system does it all, by redirecting the output of one command into a temporary file buffer which is then used as input to the input of another, system reclaims space afterwards
    - Incredibly handy and powerful 'one-liner' commands composes from simpler commands.
    - Unix philosophy : Keep it simple, make it fast, do it well.
  - Teed – 'tee'
    - Outputs from a program can be split : as in a T-junction

Neat cat Tips – quick & handy for text display/input – edit later

- ***cat filename1 filename2 >filename3***
  - Concatenates or joins
    - the input files files *filename1 filename2*
      - *No designator need for the input file*
    - Into the output file *filename2*
      - *Designated by the redirection operator >*
- ***Cat file >filecopy***
  - Copies file to filecopy, but best to use copy command ***cp file filecopy***
- ***cat filename1***
  - displays the contents of filename1 on std. out – screen
    - No output file is designated, so standard output is used; the screen
- ***cat >filename1***
  - creates a new file named filename1 whose contents are whatever you type on the keyboard
    - No input file is designated, so standard input is used; the keyboard

~$cat >my_file
The cat in the hat
smiled back at me.
^d (end-of-file character CTRL+D pressed together, on new line)
~$

Re-direction operators

<file read standard input from file
>file write standard output to file
(file will be created if not there,
otherwise overwritten!)
>>file append (add to end) standard output
to end of file – will not overwrite file

HERE documents - rarely used
Take data in for the preceding command in a script directly from the following lines within the script, delineated by one or more characters after the angle brackets, in this case a '?'
<< ?
Dataline_1
Dataline_2
…
Dataline-n
?

Pipe to page More or less!

- The output scrolls forever quickly unless you put it through a basic formatter… …simplest is to use a pipe command… …where the output of one cmd / process… …is fed directly into the input to another
  - (…well via temporary files which the system creates & clears up afterwards automatically)
- man ls | more
- man ls | less
- man ls | pg

Pipes for …

Pipes
- 'pipe' output from one command or process
  - Through intermediate files
    - Which the system automatically
      - Creates so they can be used, and saves user doing it
      - Clears up after use to save space
- To another command or process
- And so on…
- *cat myfile | grep key | sort | lpr*
  - From the end:-

    print a sorted list of lines containing key in myfile;

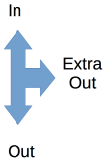
    grep does global regular expression pattern matching

Typical

- ls -alR | grep searchterm
  - ls - alR will
    - List all files in long format Recursively from cwd
  - Piping the output to grep which
    - Will only output lines matching the searchterm
- e.g. ls -al | grep music
  - Will output all filenames from ls within the current directory which have music in their title
  - Saves waiting on scrolling and searching through huge output on screen
- **grep** can use regular expressions to specify text patterns
- **find** is a much more powerful prewritten file find cmd for some uses, but for others ls | grep is more flexible…DIY!

Tee (join – like plumbing T-piece) e.g.to save & see

- Tee command - as in T-juntion - 2 ways
  - Copies std input to std output, and to named file(s)
  - lets you see the output on screen
  - And can pipe a copy to the next command
  - Functions like pipe, but placed differently
- Tee eeteemessagefile | mailx eeteehome
  - Takes standard input until ^d - just like cat
  - saves a copy in eeteemessagefile
  - and mails the message to eeteehome
- Ls -alR | grep music tee musicinfilename
  - Lists all files long format recursively,
  - Searching for filenames with 'music' in their name
  - Writing the output to the screen and to a file musicinfilename, creating it if it is not there, and overwriting it if it is
  - Use tee >>musicinfilename to append output to end of file



Beauty of Unix : do it your way!

- Can do it your way
  - Scripts : as a sequence of existing commands
  - If you can't find a command then
    - Write a script (a program of existing commands) * to have the same effect
    - And if no script can be configured to do it, then **
      - write a program in C, Java whatever,
      - And make that a command for yourself to use
      - And if it's useful, others may use it too…
- And that's how Unix grew…
  - Co-operative co-operation!
- Basis of Open Source

* Covered in this course

** Too advanced for this course

cs??? (Computer Science Server)

Accessing the servers :

- From the labs
  - From Linux
    - Booting up Kubuntu in the lab & use your account
  - From Windows
    - Terminal Programs:
      - telnet (insecure; cannot be used)
      - PuTTY from Windows
      - ssh Start -> Program Files -> Unix Connectivity -> Ssh -> SSH-x attu
- From outside the labs
  - Ssh in any of PuTTY etc., in Windows, or Terminal/Konsole in OSX / Linux
  - File Transfer Programs
    - Command line (or cli-based GUI clients)
      - ftp (insercure; should not be used)
      - Sftp Secure file transfer (from C&C) (may not have privilege)
  - Apps
    - Filezilla

The Command Prompt

- Commands are the way to "do things" in Unix
- A command consists of a command name and options called "flags"
- Commands are typed at the *command prompt*
- In Unix, *everything* (including commands) is case-sensitive

```
[prompt]$ <command> <flags> <args>
```

```
Cs1>:/abc123/~$ ls -l -a directory1
```

Command Prompt — Command — (Optional) flags — (Optional) arguments

Note: In Unix, you're expected to know what you're doing. Many commands will print a message only if something went wrong.

Two Essential Info Commands

- The most useful commands you'll ever learn:
  - `man` (short for "*man*ual")
  - `info`
- They help you find information about other commands
  - `man <cmd>` or `info <cmd>` retrieves detailed information about `<cmd>`
  - `man -k <keyword>` searches the man page summaries (faster, and will probably give better results)
  - `man -K <keyword>` searches the full text of the man pages

```
cs1:/abc123/dir1$ man -k password
passwd   (5)   - password file
xlock    (1)   - Locks the local X display
          until a password is entered
cs1:/abc123/dir1$ passwd
```

Two Essential Info Commands

- Info, as opposed to man, is category based
- Documents are hyperlinked
  - `info <cmd>` retrieves detailed information about `<cmd>`
  - `info` by itself will give instructions on its usage
  - Type `q` to quit.
  - Type `h` for help.

Scrolling around

Output from man etc. is sent through a page formatter with these commands

- space or f or - forward to next page
- b - back a page
- p - to first page
  - (not to be confused with -P option to specify the default pager which is to be used)
- Also use of the arrow keys will control page scrolling

Stopping a scroll roll & other things

Most modern systems are configured to use a pager to output a page at a time, so it doesn't all scroll by but it is good to know the old handy standbys

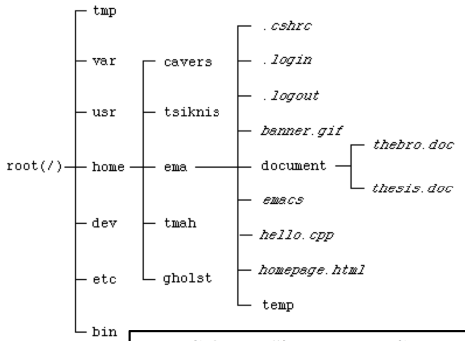
man

- Output
  - ^s - stop
  - ^w - write
- Input
  - ^d - end of file marker

Filesystem : Complexity management!

- Chaos -> order : classify & separate
- Or… 'Divide and conquer"
  - E.g. fast sorting algorithm : quicksort
- Structure data in Files & filesystems according to
  - Type & function
  - Which implies
    - Filesystems : function type : system, users, utilities
    - Files : data type : formats, headers
      - Text : admin, programs, configurations, user credentials,
      - Binary : executables.
      - Special formats : databases, logs
      - Proprietary : optimised for a specific function,
        - (if only to lock a client into a product line!)

Unix Hiearchichal Tree File Structure



Note different file types : endings are mostly for our convenience, all are just bytestreams to Unix!

File systems…

- What do we want?
  - To store data
  - in a structured manner (in a file, in a filing system)
  - so we can
    - find it, … structured organisation; index; search tools
    - use it, .. In programs, and other command sequences
    - and change it if required…
- How do we do it? … a step at a time… this course being the first, of course!

Files in filesystems…

- in a structured manner
  - This implies some sort of structured filing system
    - Files : with a structure so data can be located within the file
    - Filesystem : with a structure so files can be located
    - Code format : Unicode etc.
    - Header records : name, data type, file structure
      - Just know they exist and not be intimidated,
      - don't need to know details,
    - File / filesystem structure
      - Sequential … e.g. text.
      - Indexed … e.g. database

File systems…

- so we can
  - find it, … structured organisation; index; search tools
  - use it, … using application
    - commands
    - Command sequences
      - Scripts : generally imply interpreted & may be interactive
      - Programs : generally compiled & background
  - and change it if required…
    - By editors :
      - Manually and interactively
      - Some of which can be programmed
    - Or programs : such as
      - Dedicated applications
      - Programmed sequences of editing commands : Unix strongpoint!

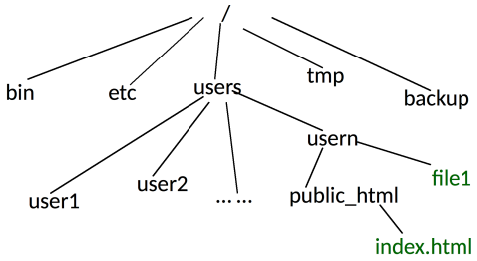
File systems…

- To store data within a file
  - Requires some sort of encoding :
    - Usually just as text characters:
      - Text : ASCII – Roman alphabet,
      - Unicode : all alphabets
    - But can be in other formats
      - Binary : machine executables
      - Numeric format : for number heavy applications
      - Proprietary format : for efficiency for a given application
        - » (or the 'deadly embrace'…lock you in to their product!)
      - Encrypted for privacy
      - With error-corrrection coding …for error correction
      - Distributed for
        - » Speed : parallel access
        - » Fault tolerance : not having all your eggs in one basket

What is directory?

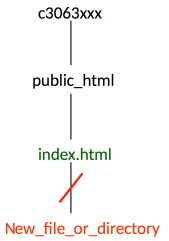
Directories can hold files and other directories

Impose an organised filing structure on files!?



What is a Directory?

A file cannot hold a directory or a file!



What is a Directory?

A collection of files grouped together, either by the user or system, for ease of management.

Your home directory typically contains a public_html directory.

Your public_html directory typically contains an "index.html" file.

Structured Filesystem

- Filesystem / Directory tree
  - Moving about : to get or put what & where
  - Adding/Removing your own (sub)directories
- Files …
  - Cannot store other files or directories, just data
  - Create, edit and remove.
  - Edit
- Directories are 'special' files
  - Which merely list other files, including directories 'within' them,
    - but in reality just point to other files
      - which are logically gathered internal to the directory,
      - but physically external to the directory on other disk blocks..

Directories

- In Unix, files are grouped together in other files called *directories*, which are analogous to *folders* in Windows
- Directory paths are separated by a forward slash: /
  - Example: `/cs1/abc123/classes/cs1235`
- The hierarchical structure of directories (the directory tree) begins at a special directory called the *root*, or /
  - *Absolute paths* start at /
    - Example: `/cs1/abc123/classes/cs1235`
  - *Relative paths* start in the current directory
    - Example: `classes/cs1235` (if you're currently in `/cs1/abc123`)
- Your home directory is where your personal files are located, and where you start when you log in.
  - Example: `/cs1/abc123`

What's a directory?

- Files are grouped in the directory structure.
- The file-system is arranged like hierarchical tree (inverted)structure.
- The top of the tree is called "root" which usually contains several sub-directories.
- In UNIX "/"(forward slash) is used to present the "root".
- Some variants have a directory called 'root' at that level, which is not to be confused with the root of the filesystem tree.
- The directory 'root' at the root of the tree, merely is the root account home directory.

Directions...from a directory ...

Remember, this was before GUI's, could not see the big picture!
Think or text output only, needed to know location, and options at each level.

- Directories are like
  - Fingerposts at each node in the directory tree
  - Tables of contents (abbrev. toc – common in CS)
  - Index
- Which show
  - Options and contents at each level
  - Effectively a path to a file.
- Filenames are effectively pathnames to a file!
  /users/my_group/my_homedir/my_subdir1/
  /users/csdipact2013/
- NB : backslash '\' in Microsoft (MS-DOS, Powershell)
- But going forward in Unix '/' (or forwardslash)

Directory navigation

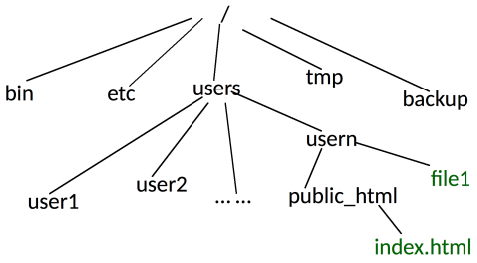
- **Where** am I? Print (present) working directory : **pwd**
  - System will display your current directory,
    that is : where you are in the filesystem tree
  - Note some command prompts (the bit at the start of a line, inviting a new command when finished the previous one) give partial information on this already.
- **What**'s here? List directory contents : **ls**
  - This will give a list of files in the current directory
    - Desktop Documents Downloads ...etc

    Common qualifiers are listed in the next slide
- **Where** can I go? **How** to change directory? : **cd**
  - cd destination_directory

  the destination_directory is specified the same way as any other filename, as outlined on the previous slide, including full absolute and relative names: /users/csdipact20nn/urid/Desktop/snap ... ~/Desktop/snap... etc.

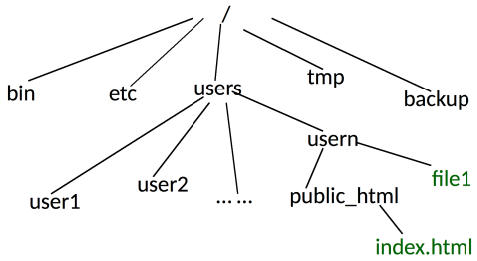
Specifying Paths

What is the **absolute path** to index.html?



Specifying Paths

What is the **relative path** to index.html (assuming that usern is your pwd)?



Pathnames

- Absolute Pathnames
  - In the previous tree /users/usern/file1 is an absolute pathname.
- Relative pathnames
  - If you are already in the users directory, the relative pathname for file1 is usern/file1.
  - If you were already in another user directory, the relative pathname could be written as
    ../usern/file1

Where are we (going!)?

- Where are we?
  - pwd - present/print working directory …
    /users/csdipact2012/jsad1

    by default you login to your home directory,
    although almost everything can be configured in Unix

  - Returns a pathname from root of directory tree
    - Although system configuration settings can be changed…to display a relative pathname (e.g. within user's home directory)

- What's there when we get there?
  - List the directory contents using - ls

How do we get there?

- cd – change directory…
  - Absolute - starting from the root …
    - cd /users/my_group/my_homedir/my_subdir1/
  - Relative
    - To home directory
      - If I'm at home : cd my_subdir1 … or … cd ./my_subdir1
        . Denotes the current directory, needs ./ or omit entirely
      - Or if I'm not : cd ~/my_subdir1
        ~ is a shortcut for your home directory!
      - Or if I'm in a directory with a common parent
        e.g. in csdipact2013 & want to go to csdipact2012
        cd ../csdipact2012
        .. Denotes a parent directory, in this case /users…

What's there in a directory – list directory - ls

ls *[names]* – list files contained in a directory *name* or that match a file *name*. If no *name* is given list those files in current directory.

- *ls –a* list all files including hidden files
- *ls –l* list in long format (including details like permissions, owner, size, etc.), works very much like *dir*
- *ls –al* list all files (including hidden files) in long format
- *ls –dl dir_name* lists information about the directory, "dir_name".

What's there in a directory – list directory - ls

Common qualifiers:-

ls just gives a tab separated list of filenames
Desktop Documents …etc

ls -l gives them 1 per line…**with a header line** with total blocksize for dir.

```
cs1> ls -l
total 96
drwxr-xr-x  3 jsad1 csdipact2012 4096 2012-09-29 12:15 CS5005_demo
drwx------  2 jsad1 csdipact2012 4096 2011-09-29 09:45 Desktop
drwx------  9 jsad1 csdipact2012 4096 2012-09-29 13:53 Documents
```

ls -1 gives them 1 per line…**without the header line** indicating dir. size…
handy for counting files, by piping to **word count** with a **linecount** flag **–l**

ls -1 | wc –l (using ls –l would give wrong filecount, as it includes 'total 96'

More ls qualifiers..

ls -a - gives **all** files, including some hidden system ones beginning with a ' . '

```
.      ..     .adobe     .bash_history
Desktop   Documents            sum
```

ls -l - gives a **long** detailed listing, showing lots of file attributes,

```
total 92
drwx------      2 jsad1 csdipact2012 4096     2011-09-29 09:45 Desktop
drwx------      7 jsad1 csdipact2012 4096     2011-11-28 15:14 Documents
-rw-------      1 jsad1 csdipact2012    72     2011-11-07 14:31 sum
```

ls –al - gives long detailed listing, including hidden system ones beginning with a ' . '

```
total 376
drwx--x--- 43   jsad1 www-data      8192 2011-12-07 13:06 .
drwxr-xr-x 38   root  csdipact2012   4096 2011-10-24 14:46 ..
drwx------  3   jsad1 csdipact2012   4096 2010-10-14 15:35 .adobe
-rw-------  1   jsad1 csdipact2012   8241 2011-12-19 22:23 .bash_history
```

Starting letters: b, c, d, l, s, p
indicate : **b**lock, **c**haracter (special), **d**irec:ory, **l**ink(soft symbolic), **s**ocket, **p**ipe (static named FIFO, as opposed to temporary ones generated during ls -1 | wc –l
etc..

Recursive…appilcation to all subdirectories

- ls –R
  - Will recursively list all files within
    - The current directory
    - And any directories within the current one
      - recursively

Metacharacters - use based on Regular Expressions

Characters with special meaning to the shell
*, ?, [...],
* matches any grouping of zero or more characters
- but in RegEx zero or more of preceding char only
? matches any single character
[...] allows matching a range of characters
[0-9] matches any digit
[A-Z] matches any capital letter
[a-d] matches a, b, c, or d
[aeiou] matches any vowel

Metacharacter Examples

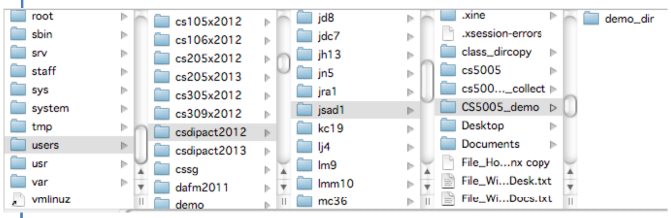
- Suppose a directory listing of your files shows the following files:

urmac%ls
Boy toy coy cow soy1 soy2 soy.1 soy1.1
bow mow1 mow2 mow3 say1.1 say1.2 say1.3
hay shay tray fray flay chow slay bay buy

- How do we select groups of these files using metacharacters?
  - Will see more later when we do regular espressions but basic idea is a few characters followed by o/a and then w/y; or in RegExp expressed as… *[oa][wy]

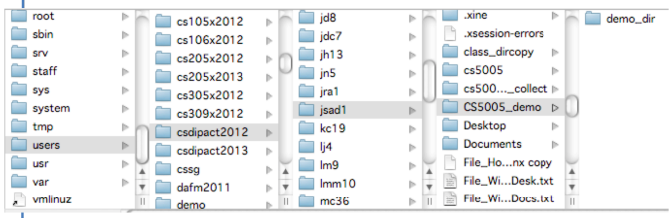
- e.g. ls *[oa][wy]

Directory – holds files, incl. other directories
Files – cannot hold files, or directories



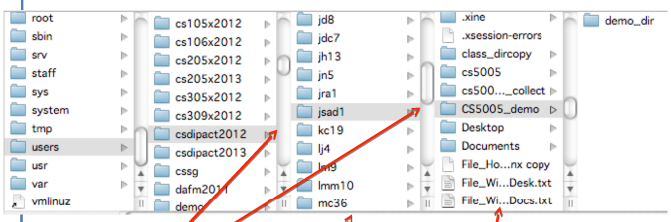
- Views
  - Treeview : as above, or something similar
  - Grid – icons on your desktop, size may vary
  - Coverflow – iTunes like flypast
  - List : the old reliable...and most flexible ... Unix use (although indentation can convey tree structure in text!)
  - Hybrid : some mix of those above...

Directory – holds files, incl. other directories
Files – cannot hold files, or directories



NB
- The root directory of the filesystem is in the leftmost column in this tree view.
- The directory named root, within that column is:
  - the home directory of the root user account
  - And not the root of the filesystem tree.
- Most other directories at the root level are system related
  - Can ignore for now, and in most cases, forever!

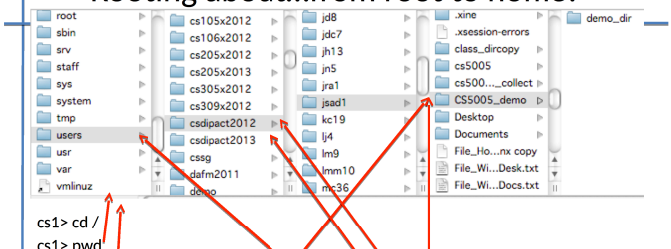
Check out the neighbourhood..!
GUI File Manager vs Terminal / Konsole




```
cs1> pwd
/users/csdipact2012/jsad1
cs1> cd ..
cs1> pwd
/users/csdipact2012
cs1> ls
am29  ... Jd8 ...  jsad1 ... mc3 ... Sx1
cs1> cd jsad1
```

```
cs1> pwd
/users/csdipact2012/jsad1
cs1> ls
class_dircopy   Music          Shopping
Cs5005          Pictures       sum
Cs5005...._collect  Public     Templates
CS5005_demo     public_html    Videos
Desktop         rubb           Windows
```

Rooting about...from root to home!




```
cs1> cd /
cs1> pwd
/
cs1> cd users/csdipact2012/jsad1
cs1> pwd
/users/csdipact2012/jsad1
cs1> cd /
cs1> pwd
/
cs1> cd ~
cs1> pwd
```

Unorthodox via grandparent!
cs1> cd ~
cs1> cd ../../csdipact2013
cs1> pwd
/users/csdipact2013

Basic Directory commands.

- To create a new directory:
  - mkdir *dir_name*
- To remove an empty directory:
  - rmdir *dir_name*
- To remove a directory that has files and subfolders:
  - rm –R *dir_name*
- *Hazard…delete all your files from your home directory*
  - cd ~ go to home directory
  - rm –R * remove recursively all files

    * is wildcard matching all text strings

For safety :
use rmdir rather than rm
As it blocks deletion of
a non-empty directory

NEVER RISK : rm – R , If you are Root user or have sysadmin privileges, as you can wipe virtually the entire filesystem, if you are in the root directory

Wildcard ' * ' : with risk of 'wild' behaviour!

- ' * ' is powerful and dangerous…always
  - Even for experienced (& absent minded) users.
- It has 2 context dependent meanings

1. And In the shell ' * ' can mean any string, basically any number of any characters
2. But in Regular Expressions, used elsewhere, editors, grep (a kind of find), prog. Languages, it means any number of the previous character!
3. And to complicate things Regular expressions can be used within some shell commands!

•. GOOD ADVICE :
  - ALWAYS BEFORE USING : ***rm file****, (and even without the –r recursive flag)check by issuing an ***ls file**** to see what files will be involved …

Directories continued…

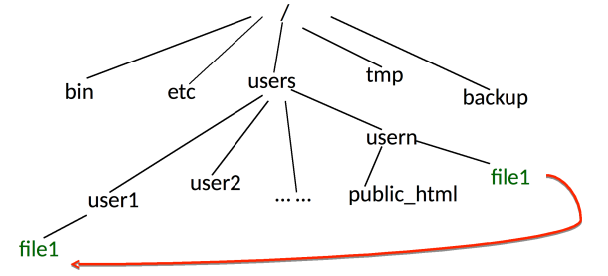
- To delete files
  - rm *file_name*
- To create files within a directory
  - For speed and ease, if it's simple text, use catenate

    which normally concatenates (jams) a filelist into a single file

    but here is used this way: cat >*new_file_name*
    - no input file is specified, it defaults to standard input (the keyboard)
    - This redirects standard input, the keyboard, to the new file ,
    - finish with CTRL+d denoting end of file; end of input.
    - CTRL+D is often used to end things in Unix.
  - Or use an editor…
    - Nano
    - Kate :KDE Advanced Text Editor
    - Vim : Vi improved; vi = visual ;-0
    - Emacs : highly programmable editor, powerful, flexible, complex
  - Or any text editor…

Move (like grafting a branch!) & copy (duplicating within pwd)

cp /users/usern/file1 /users/user1/file1
NB mv removes original
- Both will overwrite (&lose) any existing files with same destination filename
- Interactive flag will check with user… -i (that's an 'I' not an 'l', an eye not an ell
  –Do you want to overwrite file …..?
- mv moves a directory into a target directory (inconsistent with mv on files, but avoids directory being overwritten…)



Moving files about

1. File to file
   - No problem, **except** can overwrite existing destination file
2. File to directory
   - Since a file cannot hold other files like a directory, files are copied **into** destination directories
3. Directory to a new directory or into another
   - Sure, But cannot put a directory within itself!
4. Directory to file – no way!
   - Impossible, since a file cannot act as a directory,
   - However, a directory listing can be stored as text in a file …

     ls > dirlistext or append using ls >>dirlistext

Files: - Re/Move, copy

- **Copy** – to copy a file **retaining** the source
  - **cp sourcefilename destinationfilename**

  Where source and destination are filenames…
- **Move** – to move a file to a new destination, **removing** the source : effectively renaming by copying source to new location,

  with removal of old 'sourcefilename'
  - **mv sourcefilename destinationfilename**
- **ReMove/delete** – to remove a file
  - **rm sourcefilename**

Files: Accidental deletion Hazard

- In both cases (cp & mv)…
  - destination file will be created if it does not already exist,
  - If the destination file already exists, it will be overwritten!
- **Hazard**
  - If a file already exists with the same destinationfilename,
    it will be overwritten and original contents lost

    The –i (interactive flag) will check with user before overwriting, unless suppressed with the –f (force) flag.

    (Seems odd to choose to ignore an alert, except when sure, and to avoid interruptions, when dealing with many files!)

Files – move / copy **files to a directory**

- Clearly a data file is not a directory,
  - **Can't move or copy a data file to become a directory,**
    - Instead Files are moved into the target directory
  - **Definitely can't move or copy a directory to a file!**
    - (but can redirect a directory listing into a file as text!)
- Absolute pathnames can be used
  - Long but no confusion
  - cp /users/urgroup/urid/urfile /users/urgroup/urid/urdirectory
- Relative pathnames are handy – same effect as above
  - E.g. if you are in your home directory, containing both :
  - cp urfile urdirectory
- cp copies the source to the destination.
- mv just moves it to the destination, removing the source

Directory – Re/Move, copy

- Create
  - Make directory:- **mkdir**
- Remove – to remove an empty directory: basically deleting
  - **rmdir sourcefilename**
- Remove – to remove a non-empty directory: basically deleting
  - **rm -r sourcefilename**
  - the recursive flag(-r –R) means that the entire filesystem tree within the directory is affected,
  - Recursive :
    - vital for filesystem work, all branches from location
    - runs-again within every subdirectory encountered,
  - Dangerous :
    - eg rm –r * will delete everything in the current directory
    - With root access, the entire filesystem can be deleted!
    - *** DON'T RISK IT *** (is being obstructed/removed on many systems)

Directory – Re/Move, copy

- Move and copy work identically, except
  - copy retains the source directory original copy
  - Move removes the source directory original copy on successful copying.
  - Both must use the recursive option to work for source directories, empty or not
    - all directories include two hidden entries to the current and parent directories . & ..
      - ls –a shows these hidden ones
- If the desination directory name
  - Does not exist :
    - it will be created & contents of the source directory placed there
  - Exists
    - A copy of the source directory will be placed within it

But the real deal is more complex

- Most filesystems use links…to save
  - Space : only one copy, but can pop up anywhere
  - And time : only one copy, so only one edit!
- BUT ALL OF THESE copy, (re)move commands have peculiar exceptions & rules with respect to links.
- This is further complicated, because there are different types of links with different behaviour.
- And complicated even further on our system, which has a central file server:
  - Which has update delays especially when loaded, so the central filesystem will not reflect local updates for a while,
  - Which does not handle some of the more obtuse aspects of Unix links properly.

Bigger picture – other file types

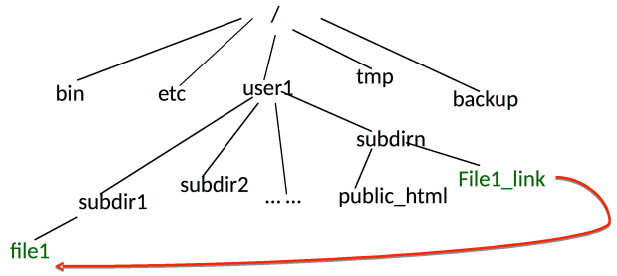
- **Linked files** .. Like alias in Mac, shortcut in Windows / Android
  - not a copy, just a reference…saves space & confusion : edits to only one copy
  - Hard… links to inode, file stays until all gone (inode is file index node on disk)
  - Soft … links to name, link lost if name gone
- **Directories** – files which store other files

Except for unamed pipes, the following are generally not seen in basic Unix :-

- **Pipes** are powerful extensions for stringing simple commands to create powerful one-line programs, with a few versions.
  - Unamed pipes:
    - temporary files created by the system to 'pipe' output from one process into another; the process can be a program or command.
  - Named pipes
    - Extension to unamed pipes with the same purpose and behaviour, but exist as static non-volatile byte-stream
- **Sockets** are similar to pipes in function, but are data types are extended from byte-streams to datagram sequences used in networking protocols, offering compatibility with networked Inter Process Communication (IPC)

Links – don't replicate, just reference!

- Insert a link (alias / pointer) in any directory to point to the file
- Links can be
  - Soft – point to the original source filename entry in it's own directory –
    - Issues – if the source filename (& file) is deleted, then a link is broken …messy
    - Some filesystem maintenance commands do not manage soft links correctly
  - Hard – effectively point to the file on disk, file remains until all links deleted!



Files.

- Create
  - Use an editor :
    - Trad ubiquitous unix : ed / sed / Vi(m) / pico / nano /
    - Newer GUI text editors & IDE's
  - or the cat to concatenate a few files together
  - Or output from a program.
- Names, paths and Access modes
- Directories / subdirectories etc.
- Copy
- Move
- Remove
- Disk space management

Bigger picture – other file types

- **Linked files**
  - not a copy, just a reference…saves space
  - Hard… links to inode, file stays until all hard links gone (inode is
  - Soft … links to name, link lost if name gone
- **Directories** – files which appear to store other files, but only store names & links

The following ones are generally not seen in basic use of Unix:-

- **Pipes** are powerful extensions for stringing simple commands to create powerful one-line programs, with a few versions.
  - Unamed pipes:
    - temporary files created by the system to 'pipe' output from one process into another; the process can be a program or command.
  - Named pipes
    - Extension to unamed pipes with the same purpose and behaviour, but exist as static non-volatile byte-stream
- **Sockets** are similar to pipes in function, but are data types are extended from byte-streams to datagram sequences used in networking protocols, offering compatibility with networked Inter Process Communication (IPC)

Basic File commands

- **Create**
  - Using editors, program output, or the cat to concatenate a few files together
  - If no input file is specified and output redirected to an output file, then that output file is created within the current directory
  - **cat >newfilename**
- **Copy** – to copy a file
  retaining the source
  - **cp sourcefilename destinationfilename**

  Where source and destination are filenames…
- **Move** – to move a file to a new destination,
  removing the source : effectively renaming
  usually achieved by copying source to new location, with removal of old.
  - **mv sourcefilename destinationfilename**
- **Remove** – to remove a file : basically deleting it
  - **rm sourcefilename**
  - NB **rmdir** is recommended for directory removal, as it will only delete empty ones; as it cannot remove files within the directory.
  - But **rm** with
    - -d will attempt to delete directories;
    - –R options will attempt to delete everything Recursively, including directories.

(sub)directory directions : Names, paths and Access modes

- Files are stored in topically or logically related groups which are
- specified by a **pathname** or **name**
  - the path of directory names to access the file, separated by '/' .
  …e.g. **/dir1/dir2/dir3/localfile /users/csdipact2012/jsad1/sum**
- In one of two ways
  - Either **full** or **absolute** to the root of the system directory tree
    - Signified and starting with:-
      /full_pathname_from_root
  - Or **relative** to
    - The current directory
      - Signified and starting with:- **localfilename**
    - The parent directory
      - Signified and starting with:- **../sibling_directory/nepotism**
    - The user's home directory
      - Signified and starting with:- **~/pathname_within_home_dir**

Directory navigation

- **Where** am I? Print (present) working directory : **pwd**
  - System will display your current directory,
    that is : where you are in the filesystem tree
  - Note some command prompts (the bit at the start of a line, inviting a new command when finished the previous one) give partial information on this already.
- **What**'s here? List directory contents : **ls**
  - This will give a list of files in the current directory
    - Desktop Documents Downloads …etc

    Common qualifiers are listed in the next slide
- **How** to change directory? : **cd**
  - cd destination_directory

  the destination_directory is specified the same way as any other filename, as outlined on the previous slide, including full absolute and relative names: /users/csdipact20nn/urid/Desktop/snap … ~/Desktop/snap… etc.

What's in a directory – list directory - ls

Common qualifiers:-

ls just gives a tab separated list of filenames
Desktop Documents ...etc

ls -l gives them 1 per line...with a header giving total blocksize used for dir.

```
cs1> ls -l
total 96
drwxr-xr-x  3 jsad1 csdipact2012 4096 2012-09-29 12:15 CS5005_demo
drwx------  2 jsad1 csdipact2012 4096 2011-09-29 09:45 Desktop
drwx------  9 jsad1 csdipact2012 4096 2012-09-29 13:53 Documents
```

ls -1 gives them 1 per line...without the header indicating size...
handy for counting files..
Pipe the output into a word count with line option : **ls -1 | wc –l**

ls -a - gives all files, including some hidden system ones beginning with a ' . '

.	..	.adobe	.bash history
Desktop	Documents	sum	

More ls qualifiers..

ls -l - gives a long detailed listing, showing lots of file attributes,

```
total 92
drwx------      2 jsad1 csdipact2012 4096     2011-09-29 09:45 Desktop
drwx------      7 jsad1 csdipact2012 4096     2011-11-28 15:14 Documents
 -rw-------     1 jsad1 csdipact2012   72     2011-11-07 14:31 sum
```

ls –al - gives long detailed listing, including hidden system ones beginning with a ' . '

```
total 376
drwx--x--- 43  jsad1 www-data      8192 2011-12-07 13:06 .
drwxr-xr-x 38  root  csdipact2012  4096 2011-10-24 14:46 ..
drwx------  3  jsad1 csdipact2012  4096 2010-10-14 15:35 .adobe
-rw-------  1  jsad1 csdipact2012  8241 2011-12-19 22:23 .bash_history
```

Starting letters: b, c, d, l, s, p
indicate : **b**lock, **c**haracter (special), **d**irectory, **l**ink(soft symbolic), **s**ocket, **p**ipe (static named FIFO, as opposed to temporary ones generated during ls -1 | wc –l
etc..

File access modes / permissions

chmod – change modes /access permissions, else file is useless
permits the file owner or root

- to set or clear : access permissions (or modes)
- for groups : using alphabetic (letter) or binary (bit) codes

Executable bit on a directory allows it to be searched

Groups
a – all groups below
u – user
g – group
o – others

\+ set
\- Clear
= sets specified permissions within specified user groups and clears all others

Permissions
r – read
w – write
x - execute
Lots others

chmod ug+x myscript	d ? ? x ? ? x ? ? ?
chmod 751 myscripts	d r w x r – x - - x
chmod a-x badscripts	d ? ? - ? ? - ? ? -
chmod 600 badscript	- r w - r - - r - -
chmod a=r badscript	- r - - r - - r - -

? – unchanged from whatever it was

Directory manipulation

- Create
  - Make directory:- **mkdir**
- Copy – to copy a dirctory
  retaining the source
  - As for files :- **cp sourcedirname destinationdirname**
  - And if desired, **recursively**
    - Use the –R option with the command : **cp –R source dest**
    - (all files, including hard, but not soft links (unless –L is also used), within are also copied)
- Move – to move a file to a new destination directory,
  (the destination directory must exist and not be a simple file, or it will fail!)
  removing the source : effectively renaming
  usually achieved by copying source to new location, with removal of old.
  - mv sourcefilename destinationfilename
  - Again if desired **recursively** (all files within are also copied)
    - Use the –R option with the command : **cp –R source dest**
- Remove – to remove / delete a directory, but directory must be empty : safeguard
  - **rmdir sourcefilename**
- NB **rm** with **–d** or **–R** options will attempt to remove all files, including directories.

The Command Prompt jargon

- Commands are the way to "do things" in Unix
- A command consists of a command name and options called "flags"
- Commands are typed at the *command prompt*
- In Unix, *everything* (including commands) is case-sensitive

```
[prompt]$ <command> <flags> <args>
```

```
Cs1>:/abc123/~$ ls –l –a directory1
```

Command Prompt
Command
(Optional) flags
(Optional) arguments

Note: In Unix, you're expected to know what you're doing. Many commands will print a message only if something went wrong.

Two Basic Commands

- The most useful commands you'll ever learn:
  - `man` (short for "*man*ual")
  - `info`
- They help you find information about other commands
  - `man <cmd>` or `info <cmd>` retrieves detailed information about `<cmd>`
  - `man –k <keyword>` searches the man page summaries (faster, and will probably give better results)
  - `man –K <keyword>` searches the full text of the man pages

```
cs1:/abc123/dir1$ man –k password
passwd   (5)   - password file
xlock    (1)   - Locks the local X display
                 until a password is entered
cs1:/abc123/dir1$ passwd
```

Two Basic Commands (info)

- Info, as opposed to man, is category based
- Documents are hyperlinked
  - `info <cmd>` retrieves detailed information about `<cmd>`
  - `info` by itself will give instructions on its usage
  - Type `q` to quit.
  - Type `h` for help.

Directories

- In Unix, files are grouped together in other files called *directories*, which are analogous to *folders* in Windows
- Directory paths are separated by a forward slash: /
  - Example: `/cs1/abc123/classes/cs1235`
- The hierarchical structure of directories (the directory tree) begins at a special directory called the *root*, or /
  - *Absolute paths* start at /
    - Example: `/cs1/abc123/classes/cs1235`
  - *Relative paths* start in the current directory
    - Example: `classes/cs1235` (if you're currently in `/cs1/abc123`)
- Your home directory is where your personal files are located, and where you start when you log in.
  - Example: `/cs1/abc123`

Directories (continued)

- Handy directories to know
  - ~ Your home directory
  - .. The parent directory
  - . The current directory
- `ls`
  - *LiSt*s the contents of the specified directories (or the current directory if no files are specified)
  - Syntax: `ls [<file> … ]`
  - Example: `ls backups`
- `pwd`
  - *P*rint *W*orking *D*irectory

Directories (continued further)

- `cd`
  - *C*hange *D*irectory (or your home directory if unspecified)
  - Syntax: `cd <directory>`
  - Examples:
    - `cd backups/unix-tutorial`
    - `cd ../class-notes`
- `mkdir`
  - *MaKe DIR*ectory
  - Syntax: `mkdir <directories>`
  - Example: `mkdir backups class-notes`
- `rmdir`
  - *ReMove DIR*ectory, which *must be empty*
  - Syntax: `rmdir <directories>`
  - Example: `rmdir backups class-notes`

Files

- Unlike Windows, in Unix file types (e.g. "executable files, " "data files," "text files") are *not* determined by file extension (e.g. "foo.exe", "foo.dat", "foo.txt")
- Thus, the file-manipulation commands are few and simple …
- Many commands only use 2 letters
- `rm`
  - *ReMove*s a file, ***without a possibility of "undelete!"***
  - Syntax: `rm [options] <file(s)>`
  - Example: `rm tutorial.txt backups/old.txt`
  - -r option: `recursive (delete directories)`
  - -f option: `force. Do no matter what`

Files (continued)

- `cp`
  - *CoP*ies a file, preserving the original
  - Syntax: `cp [options] <sources> <destination>`
  - Example: `cp tutorial.txt tutorial.txt.bak`
  - -r option: `recursive (copies directories)`
- `mv`
  - *MoVe*s (renames) a file or directory, destroying the original
  - Syntax: `mv [options] <sources> <destination>`
  - Examples:
    - `mv tutorial.txt tutorial.txt.bak`
    - `mv tutorial.txt tutorial-slides.ppt backups/`

Note: Both of these commands will over-write existing files without warning you!

Permissions for files

- Files are owned by both a user and a group
- You will either belong to ugrad_cs or year_name
- Each file has 3 sets of permissions
  - Permissions for the user who owns it (user permissions)
  - Permissions for the group that owns it (group permissions)
  - Permissions for everyone else ('other' or 'world' permissions)
- There are 3 types of permissions
  - Read -- controls ability to read a file
  - Write -- controls ability to write or change a file
  - Executable -- controls whether or not a file can be executed as a program

Permissions for files (example)

To see the permissions on a file, do a '`ls -l`'

```
attu4:/abc123/dir1$ ls -l
-r--r--r--  1 dir1  year_name  17375 Apr 26 2000  rgb.txt
-rw-r--r--  1 dir1  year_name  17375 Apr  5 02:57 set10.csv
drwxr-xr--  1 dir1  year_name   1024 Jan 19 19:39 tests
```

Changing Permissions

`chmod [ugo]+[rxw] <filename>`

e.g. `chmod u+w rgb.txt` ← gives me permission to change rgb.txt

Changing Ownership

`chown <user>.<group> <filename>`

e.g. `chmod abc123.year_name rgb.txt`

← year_name group now owns rgb.txt

Note: You cannot change which user owns file, unless you are an administrator, for security reasons.

Permissions on directories

- Directories can also be thought of as a file that lists all the files it contains.
- They also belong to a user and a group
- They have the same 3 sets of permissions
- For directories, this is what the permissions mean
  - Read -- You can read the list of files (eg. You can use "ls" and get the directory contents)
  - Write -- You can change the list of files (eg. You can add or remove files from the directory)
  - Executable -- You can use the directory list to let the operating system "find" the file. This means, you have access to the file. All directories generally have execute permission.

Shell Shortcuts

- Tab completion
  - Type part of a file/directory name, hit `<tab>`, and the shell will finish as much of the name as it can
  - Works if you're running `tcsh` or `bash`
- Command history
  - Don't re-type previous commands – use the up-arrow to access them
- Wildcards
  - Special character(s) which can be expanded to match other file/directory names
    - `*` Zero or more characters
    - `?` Zero or one character
  - Examples:
    - `ls *.txt`
    - `rm may-?-notes.txt`

Review : basic file commands

- Where am I in the filesystem $pwd
- What's here $ls
- How can I move around the filesystem $cd
- Make my own pad / directory $mkdir
- And fill with a few files $nano, vim
  - Even $cat >newfile,
  - List the file $cat newfile
  - Or just a bit of it $head newfile, $tail newfile
- Rename or move them around $mv
- Even duplicate them $cp
- Or dump them $rm, rmdir
- Where can I find out more about
  - a command $man
  - Any topic $apropos $man -k

Disk space administration

These commands can be used to determine or identify files/directories with substantial disk use

- either intrinsically with options etc.
  (which vary with bash Linux/BSD versions)
- Or additionally with pipes using sort and grep,

- du – displays disk block usage statistics
- df – displays info on disk free..

Using grep and sort, simply with list directory (ls) can select files of a specific type and sort them by size / modification date, so you can decide which ones to archive or delete.

Check manual (man) for more information and usage examples.

ls -1 | grep 'whatever' | sort –(field no e.g. date or size)

Or variations with intrinsic command options.

Backups

Various Options

- Entire filesystems / partitions
  - dump/restore
  - dd – disk duplicate (bit by bit)
- Selective:
  - tar - tape archive, simple, stable, ubiquitous & updated : widely used
  - cpio – great but modifies file (create & access) times
  - rsync – (remote sync of files) incremental copy across the network

Tar backup – assuming previous setup

Archiving entire filesystem from root

$ **cd /**

$ **sudo tar –cf /dev/st0**

- The tar command then
  - creates an archive(**c**) (x for extraction)
    - actually appends it to existing one if present, creates otherwise
  - on the argument to file **f /dev/st0** (**f**) which could be any file, on the net.
- To (de)compress the archive, use **J** to call bzip2:

  $ **sudo tar –cjf /dev/st0 .**
- To create a backup of a directory mydir

  $ tar cvf /dev/st0 mydir
- To make a verbose archive to dev of everything in the current directory beginning with an a

  tar cvf dev a*
- To **extract** just replace the c (create) with x (extract); otherwise identical.

Selective restoring with tar …

- Usually, the entire archive is restored to the original state; all files etc.
- But, any selected file can be restored alone, by giving it's original pathname as used in creating the archive.
- A toc (table of contents) can be recreated from the archive to help locate the original pathname

  tar tf /dev/st0 > tocfile
- Which you can browse
- Or grep to find the selected file's pathname.

  grep myfile tocfile
  - Or be really concise… (will only work on archival tape with a rewind option)

    tar xvf /dev/st0 `tar tf /dev/st0 | grep 'myfile'`
  - And give it time to locate the file, before proceeding with restoration

    tar xvf /dev/st0 `tar tf /dev/st0 | grep 'myfile' ; sleep 60`

Processes – ps

Several *nix flavour variants BSD etc… but all basically the same, use man ps | less for further info on specifics

- ps aux
  - (**a**ll, **u**ser, **x** (without) being tied to a terminal)
- ps axu | sort –rk 3 | head
  - can pipe to sort and sort by a key field
    - in this case k = 3, which for axu flags is the cpu use, so the list is sorted by cpu use
    - r flag with sort does a reverse sort, so it is sorted in descending cpu use, so that the cpu hogs are at the top of the list
  - then piped to head which outputs the 'TOP 10'

Alternative :

- top
  - dynamically updated every few secs
  - displays processes in order of decreasing cpu use
  - terminated by ctrl+c (often indicated by ^C or occasionally ^+C)

Note that the stats on virtual systems are virtually misleading!

Some super admin commands mostly for superuser.

Great for system monitoring for analysis & resolution of problems & bottlenecks – again, check for more info using man, books etc.

May not generally be installed or available…

General

- vmstat
  - although it seems specific to virtual memory, (not machine) it is a generally useful command which gives a general widely useful overview

```
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 0  0     84 2737840 344060 4699420    0    0     3     3    3    0  1  1 98  0
```

- mpstat – multiprocessor version of vmstat

Virtual Memory :- swapon, swapinfo

Disk I/O :- iostat

Disk Performance :-

- xdd (examine disk devices)
- sar (system activity recorder/monitor) – similar to vmstat, but logs over time (default is 10 minute intervals since midnight, so providing historical data)

How to get More info

- Man

  | more

  pipes it to more which outputs a page at a time on each press of the space bar etc. but cannot scroll backwards

  | less

  pipes it to more which outputs a page at a time on each press of the space bar etc and can scroll either direction.

  -k

  -K
- Apropos
- Online
- Books

Tutorials

- Unix
  - Commonly referenced (you only need the first two!)
    - http://www.ee.surrey.ac.uk/Teaching/Unix/
- VIM
  - Vimtutor :with Unix
  - http://www.openvim.com/tutorial.html
  - http://vim-adventures.com/

Review : basic file commands

- Where am I in the filesystem $pwd
- What's here $ls
- How can I move around the filesystem $cd
- Make my own pad / directory $mkdir
- And fill with a few files $nano, vim
  - Even $cat >newfile,
  - List the file $cat newfile
  - Or just a bit of it $head newfile, $tail newfile
- Rename or move them around $mv
- Even duplicate them $cp
- Or dump them $rm, rmdir
- Where can I find out more about
  - a command $man
  - Any topic $apropos $man -k

More UNIX commands

- pwd –let you know the absolute pathname of your current working directory (Print Working Directory)
- cd *[dir]* – change directory
  - *cd ..* –go back to parent directory. "**..**" is the relative pathname to the parent directory.
  - "**.**" -stands for current (working) directory.
  - "**~**" – the tilde ~ character can refer your home directory

NB Although directories cannot be directly edited by a user, but changed only by making and (re)moving files (incl. directories), directories are otherwise treated as data files by the system.

Directory Navigation - review

- Directory Paths
  - unlike MS-DOS, UNIX file systems use forward slashes
  - Eg. /user/csdipact2013/your_id/ (UNIX)
    \mydocuments\myword\ (MS)
- Change Directory: cd *directory_path*
  - Go up to parent directory: cd ..
  - Go to home directory: cd ~
- Useful Commands
  - pwd: returns "present/print working directory"
  - ls: "list contents of directory"
    - ls : tab delimited list
    - ls –l : one entry per line, long format with file attributes
    - ls –a : list all files, including hidden (system) ones..
    - ls -la : lists all files incl. hidden files + shows file attributes

Directories (continued)

- Handy directories to know
  - ~ Your home directory
  - .. The parent directory
  - . The current directory
- `ls`
  - *LiSts* the contents of the specified directories (or the current directory if no files are specified)
  - Syntax: `ls [<file> … ]`
  - Example: `ls backups`
- `pwd`
  - *P*rint *W*orking *D*irectory

Directories (continued further)

- `cd`
  - *C*hange *D*irectory (or your home directory if unspecified)
  - Syntax: `cd <directory>`
  - Examples:
    - `cd backups/unix-tutorial`
    - `cd ../class-notes`
- `mkdir`
  - *MaKe DIR*ectory
  - Syntax: `mkdir <directories>`
  - Example: `mkdir backups class-notes`
- `rmdir`
  - *ReMove DIR*ectory, which *must be empty*
  - Syntax: `rmdir <directories>`
  - Example: `rmdir backups class-notes`