COMP 2021

Unix and Script Programming

Course Information

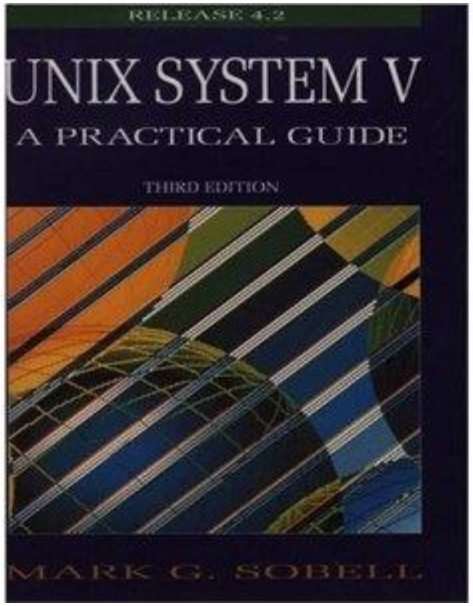
Course Information

- Lecture
  - Tue 11:00-12:50, G009A, CYT Building
- Instructor
  - Dr. Cindy LI, lixin@cse.ust.hk, Room 3535 (Lift 25/26)
- Course website
  - http://course.cse.ust.hk/comp2021/
- Labs
  - Lab 1: Thur 09:00 – 10:50, Room 4214 (Lift 19),
- TAs
  - Mr. Chang Zhang Yu

Course Objectives

- Have a general appreciation of the Unix operating system and its environment
  - Get familiar with shell basics, file structure, everyday commands
  - Be able to write simple shell programs for text/data manipulation and process control
  - Understand regular expressions and use them in Unix utilities and file manipulation
- Script Programming Skills
  - Understand the basics of script programming languages such as PHP and JavaScript, including variable and array, control flow, I/O, and functions
- Web Programming Skills
  - Have a working knowledge of the common HTML commands and CSS
  - Understand how to build web programs using CGI programming in languages such as PHP and JavaScript

Reference & Grading Scheme



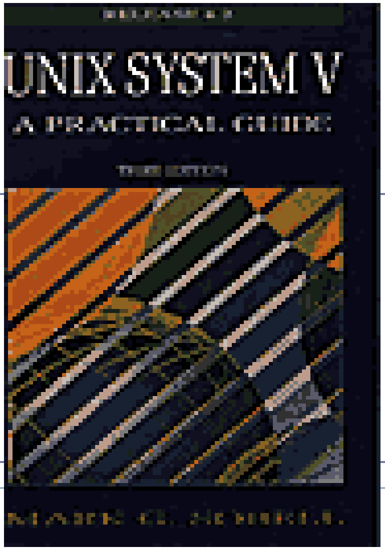
codecademy

w3schools

- Lab attendance (5%)
- Homework assignment (20%)
- Project and presentation (35%)
- Final exam (40%)

Comp2021 Project & Presentation

- Propose, implement, and document your own custom application.
- Choose your own topic that includes Unix, Shell scripting, or PHP/JavaScript
- Work in groups of normally 2 people.
- Presentations will be in the last few lectures of the semester.
- The tentative format for the project is the following:
  - 10-minute presentation (like short conference presentation)
  - 5-minutes for Q&A (while the next group sets up)
- Upload final submission to CASS last Day of Spring term
  - a softcopy of your PowerPoint slides
  - a softcopy of a short paper (4 pages) summarizing your project
  - source code



Introduction to Unix

*nix Systems

What is UNIX?

- UNIX is an *Operating System* (OS).
- An operating system is a control program that helps the user communicate with the computer hardware.

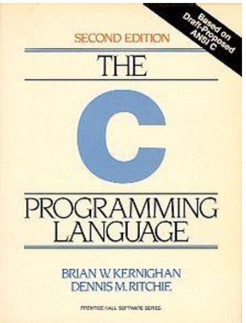
- One of the first widely-used operating systems
- Basis for many modern OSes
- Helped set the standard for multi-tasking, multi-user, interactive systems
- Strictly a teaching tool (in its original form)

Unix Features

- UNIX is an operating system for experts, used on high-end workstations, database servers, and web servers.
- UNIX provides some powerful features:
  - security - private and shared files
  - multi-user support
  - data sent to display, files, or printers in same way
  - interprocess communication
- Microsoft keeps trying to upgrade Windows to replace UNIX as the "OS for experts".

Short History of Unix

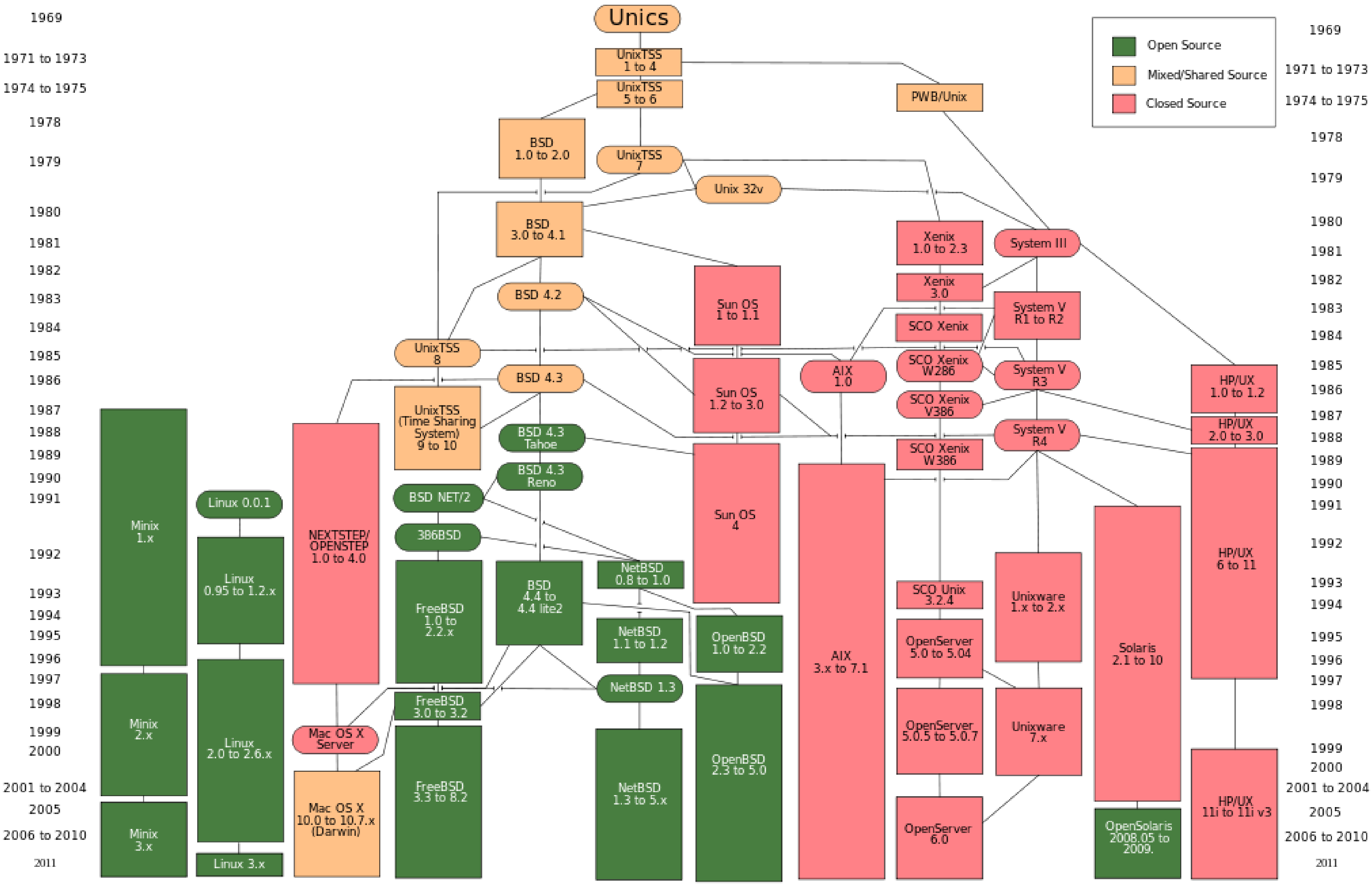
- 60s The ambitious project MULTICS (Multiplexed Information and Computing System) fails, but a number of seminal ideas (like pipes and shells) are proposed
- 69 Ken Thompson, Dennis Ritchie (et al.) at Bell Labs start working on a file system UNICS, which is later changed to UNIX.
  - UNIX was "small, simple and clean", and distributed freely to many universities, where it becomes popular
- 73 Thompson and Ritchie rewrote UNIX in C
  - Greatly facilitate its further development and porting to other hardware
- 81 Berkley UNIX 4.1 BSD (Berkeley Software Distribution) : vi, C shell, virtual memory
- 91 Linux, GNU, and others: similar to UNIX, but their source code rewritten, very popular and widespread, free
- Currently, The Open Group is responsible for developing UNIX



UNIX Versions

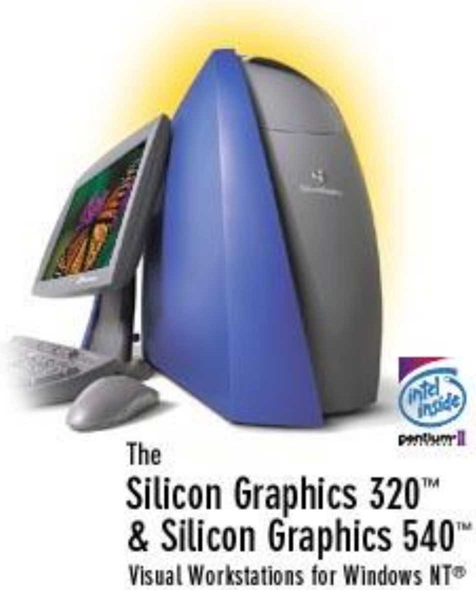
- There are two main types of UNIX:
  - BSD (Berkeley Software Distribution)
  - System V (developed at AT&T)
- Our book covers UNIX System V
- There are many different versions of UNIX for different hardware:
  - Sun Microsystem's *Solaris*
  - Mac OS/X
  - Hewlett-Packard's *HP-UX*
  - IBM's *AIX*
  - SGI's *IRIX*

- Free Unix and Unix-like Operating system
  - GNU project
  - Linux
    - Pieced together by a Finnish guy named Linus Torvalds
    - Redhat, Fedora, Debian, Ubuntu, etc.


Unics
UnixTSS 1 to 4
UnixTSS 5 to 6
PWB/Unix
BSD 1.0 to 2.0
UnixTSS 7
Unix 32v
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
Sun OS 1 to 1.1
BSD 4.2
System V R1 to R2
SCO Xenix
UnixTSS 8
AIX 1.0
SCO Xenix W286
System V R3
HP/UX 1.0 to 1.2
BSD 4.3
Sun OS 1.2 to 3.0
UnixTSS (Time Sharing System) 9 to 10
SCO Xenix V386
HP/UX 2.0 to 3.0
BSD 4.3 Tahoe
System V R4
SCO Xenix W386
BSD 4.3 Reno
Minix 1.x
NEXTSTEP/ OPENSTEP 1.0 to 4.0
Linux 0.0.1
BSD NET/2
Sun OS 4
386BSD
NetBSD 0.8 to 1.0
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
AIX 3.x to 7.1
SCO Unix 3.2.4
Unixware 1.x to 2.x
Solaris 2.1 to 10
HP/UX 6 to 11
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
NetBSD 1.3
FreeBSD 3.0 to 3.2
Minix 2.x
Linux 2.0 to 2.6.x
Mac OS X Server
OpenServer 5.0.5 to 5.0.7
Unixware 7.x
Mac OS X 10.0 to 10.7.x (Darwin)
FreeBSD 3.3 to 8.2
NetBSD 1.3 to 5.x
OpenBSD 2.3 to 5.0
HP/UX 11i to 11i v3
Minix 3.x
OpenServer 6.0
OpenSolaris 2008.05 to 2009.
Linux 3.x
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2010
2011
🟩 Open Source
🟧 Mixed/Shared Source
🟥 Closed Source

Who Uses UNIX?

- Big companies. They especially use UNIX servers, preferring its stability. They can afford to hire employees with UNIX experience.
  - Computer manufacturers such as Sun, SGI, IBM, and HP
  - Computer chip manufacturers like Motorola & Intel
  - Software companies
  - Banks
  - Hong Kong Government
  - Hospital Authority
  - Universities
- Small companies that use Linux
  - OS free



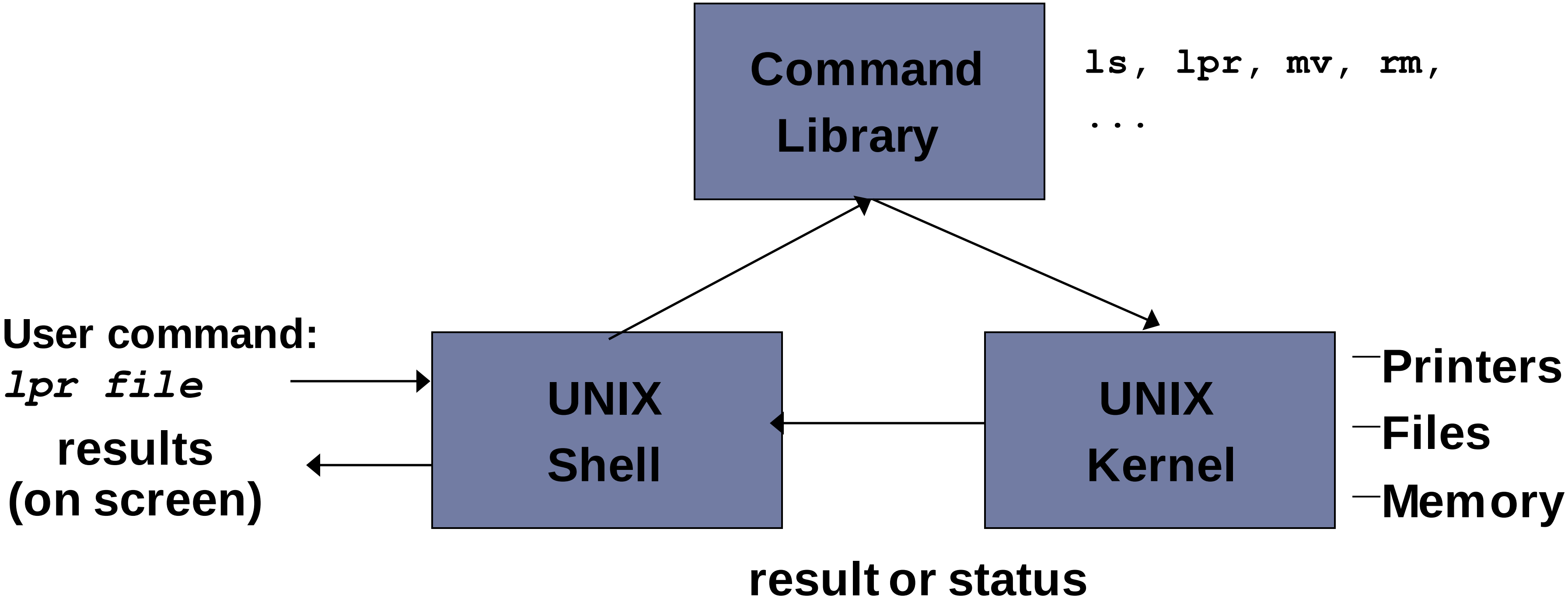
Most Important Feature of UNIX

- Most important feature of UNIX: **STABILITY**
  - Many years to get the bugs out
  - Important in shared environments and critical applications
- Shared Environments Example: University
  - Windows crashes 1-2 times/month in labs
  - UNIX servers crash usually only when hard disk fails
  - UNIX more reliable than Windows
- Critical Applications
  - Bank – Don't want to lose money in ATM transactions!
  - Hospital - Don't want to wait for reboot during operation!
  - Airport - Air traffic control landing planes.
  - PCCW - Don't want phone system going down!

UNIX Shells

- A shell is a program that allows the user to interact with the UNIX system (usually via command line):
  - Read user's input and parses it
  - Evaluates special characters
  - Setup pipes, redirections, and background processing
  - Find and setup programs for execution

Unix Shells (cont.)



Command Library

ls, lpr, mv, rm, ...

User command: *lpr file*

results (on screen)

UNIX Shell

UNIX Kernel

Printers

Files

Memory

result or status

Popular Shells

- sh Bourne shell (the original shell)
  - a popular shell made by Stephen Bourne
- csh C-shell (pronounced as “sea shell”)
  - interactive and close to C, default shell for BSD-based systems
- tcsh Like csh with more functions (Lab2 default)
- bash “Bourne again” shell
  - default shell for the GNU OS, most Linux distributions, and OSX
- ksh Korn shell
- zsh Z-shell

Getting Started on UNIX

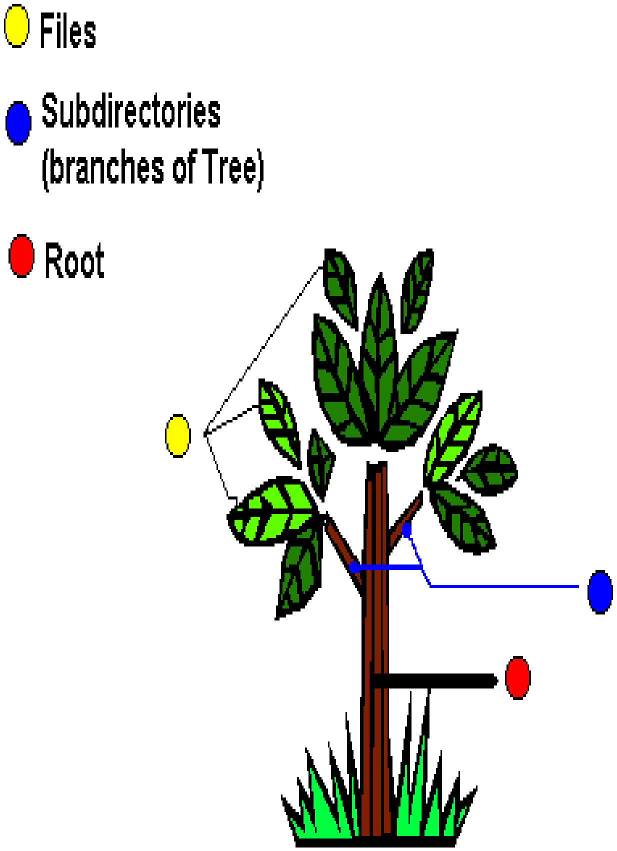
- The machines in CS Lab2 are named `csl2wk01` through `csl2wk41`.
- `csl2wk01` means "CSLab2, workstation#1"
- The full machine name for `csl2wk01` is: `csl2wk01.cse.ust.hk`
- Where are my stuff?
  - Your files can be found in your home directory, usually located at `/homes/username`
  - Home directory can also be accessed using ~

Unix Utilities

We roll our sleeves and get our hands dirty

Unix File System

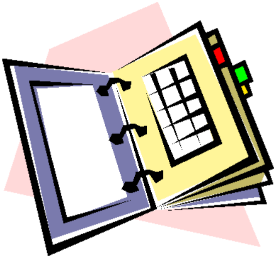
- Unlike windows, UNIX has a single global “root” directory /
  - Instead of a root directory for each disk/volume
- All files and directories are case sensitive
  - `hello.txt != hEllO.tXt`
- Directories are separated by / instead of \ in windows
  - `UNIX: /homes/lixin/comp2021`
  - `Windows: D:\Documents\comp2021`
- “Hidden” files begin with “.”: `.gimp`



What’s Where

Folder	Content
`/dev`	Hardware devices can be accessed here - usually you don’t mess with this stuff.
`/mnt`	Frequently used to mount disk drives
`/usr`	Mostly user-installed programs and their related files
`/etc`	System-wide settings
`/bin`	System programs
`/usr/bin`	Most user programs
`/usr/local/bin`	A few other user programs

- Programs are usually installed in one of the "binaries" directories



UNIX File Utilities

`ls`	**lis**t directory contents
`cd`	**c**hange **d**irectory
`pwd`	**p**rint **w**orking **d**irectory
`cat`	display file
`more`	display one screen of file
`rm`	**rem**ove (delete) a file
`rmdir`	**rem**ove (delete) **dir**ectory
`cp`	**c**o**p**y source file to target file
`mv`	rename or **m**o**v**e a file

Let’s Move Around and Do Stuff

- Where am I now?

Print Working Directory
`pwd`
• Prints the full path of the current directory
• Handy on minimalist systems when you get lost

- What’s here?

The list command
`ls [flags][file]`
• Lists directory contents (including subdirectories)
• Works like the dir command from DOS
• The `-l` flag lists detailed file/directory information (more later)

Relative and Absolute

- How to move around?

Change directory
`cd [directory name]`
• Changes directory to `[directory name]`
• If not given a destination defaults to the user’s home directory
• Takes both absolute (`cd /homes/lixin/comp2021`) and relative (`cd comp2021`) paths

- Absolute path
  - Location of a file or folder starting at /
- Relative path
  - Location of a file or folder beginning at the current directory

Relative Path Shortcuts

Shortcuts	
~	Current user’s home directory
.	Current directory
..	The parent directory of the current directory

Example	If we start in `/homes/lixin/comp2021/lab`
`cd`	`/homes/lixin`
`cd .`	`/homes/lixin/comp2021/lab`
`Cd ..`	`/homes/lixin/comp2021`

Create File or Directory

- Create a new file

Using `touch`
`touch [flags] <filename>`
• Adjusts the timestamp of the specified file. With no flags uses the current date/time
• If the file does not exist, `touch` creates it

  - File extensions (.exe, .txt) often don't matter in UNIX. `touch` create a blank plan-text file.

- Create a new directory

Make directory
`mkdir [flags] <directory>`
• Makes a new directory with the specified names
• Can use relative/absolute paths to make directories outside the current directory

Delete File or Directory

Delete file

Remove file

```
rm [flags] <file>
```

- Using wildcards (more later) you can remove multiple files
- `rm *` removes every files in the current directory
- `rm *.jpg` removes every .jpg file in the current directory

- `rm –i filename` prompts before deletion

  - By default, `rm` can't remove directories

Delete directory

Remove directory

```
rmdir [flags] <directory>
```

- Removes an **empty** directory. Throws an error if the directory is not empty

- `–r` flag delete a directory and all its subdirectories
- `rm –r /homes/lixin/oldstuff`

Copy and move

Copy

```
cp [flags] <file> <destination>
```

- Copy a file from one location to another
- To copy multiple files you can use wildcards (such as *)

- `cp -r <src> <dest>` copies a complete directory
- `cp *.mp3 /music/` copies all .mp3 files from the current directory to `/homes/<username>/music/`

- Unlike cp, the move command automatically recurses for directories

move

```
mv [flags] <source> <destination>
```

- Moves a file or directory from one place to another
- Also used for renaming, just move from `<oldname>` to `<newname>`

More about Flags/Options

- Most commands take flags (also called options).
- These usually come before any targets and begin with a -
  - One Option `ls -l`
  - Two Options `ls -l -Q` or `ls -lQ`

Get Help?

The manual command
`man <command_name>`
• Brings up the manual page (manpage) for the selected command
• manpages are **system-specific** (unlike google results)
• Pretty comprehensive list of all possible options/parameters

- Be aware, there are subtle differences with options on different systems. Always refer to `man` for the most precise answer.

Class Activity

- Let's try to play with the following commands
  - `who`
  - `whoami`
  - `write`
- Task: find out a friend who logged on the same Unix machine and send a message to him/her
- More commands to try
  - `echo`
  - `date`

More Utilities

`wc`

- How many lines of code are in my program?
- How many words are in this document?
- Good for bragging rights

Word, Character, Line, and Byte count with `wc`
• `wc -l:` count the number of lines
• `wc -w:` count the number of words
• `wc -m:` count the number of characters
• `wc -c:` count the number of bytes

More Utilities

`sort`

- Sort the lines of a text file alphabetically
  - `sort -ru file`
    - Sorts the file in reverse order and deletes duplicate lines
  - `sort -k 2 file`
    - Sorts the file using the second column

`uniq`

- `uniq file`
  - Discards all but one of the successive identical lines
- `uniq -c file`
  - Prints the number of successive identical lines next to each line

More Utilities

- head   Display first n lines of file

```
$ head -n file
```

- tail   Display last n lines of file

```
$ tail -n file
```

- grep   Find a pattern in a file

```
$ grep ”pattern” file
```

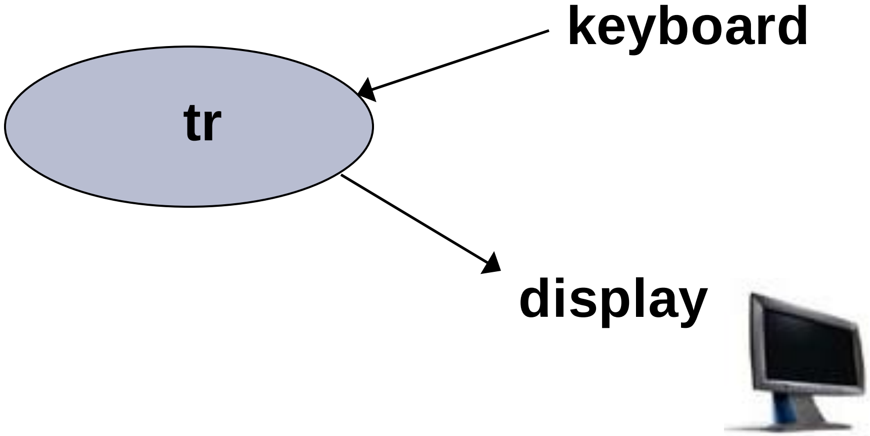
Character Manipulation

The Translate Command
`tr [options] <char_list1> [char_list2]`
• translate or delete characters
• char_lists are strings of characters
• By default, searches for characters in char_list1 and replaces them with the ones that occupy the same position in char_list2

Example
`tr ‘AEIOU’ ‘aeiou’` changes all capital vowels to lower case vowels
`tr a-z A-Z` converts lower to upper case
`tr -d t` deletes t

Pipes and Redirection

- On UNIX, the *standard input* (stdin) is the keyboard; the *standard output* (stdout) is the display screen. `tr` waits for you to type in the data from the keyboard and displays the sorted data on the screen.

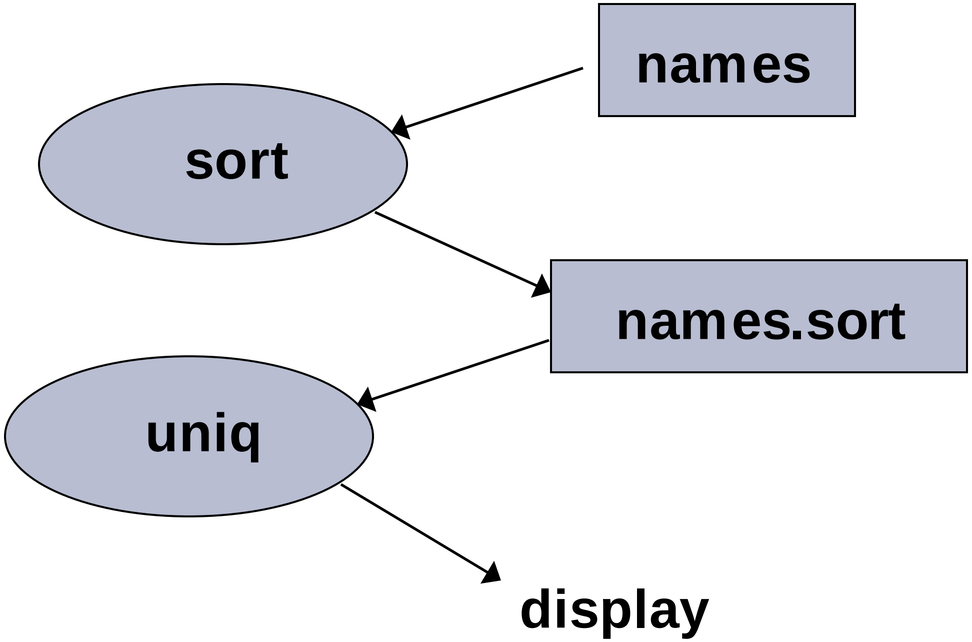


- What if we want to operate on files?
  - Piping: `cat somefile | tr 'AEIOU' 'aeiou'`
  - Input redirection: `tr 'AEIOU' 'aeiou' < somefile`

Input/Output Redirection

- Using the “>” character after a command to redirect standard output:

```
$ sort names > names.sort
$ uniq names.sort
```



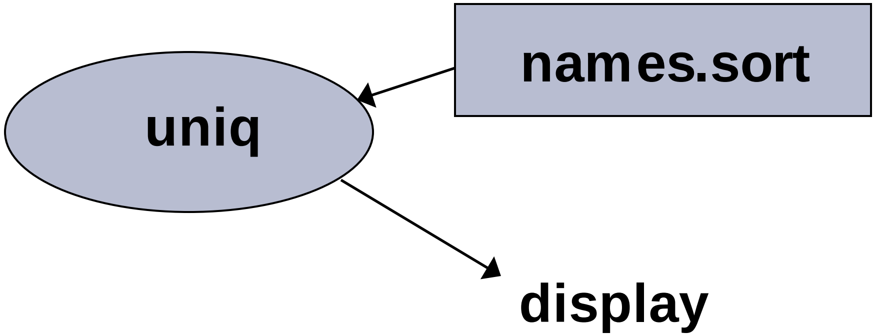
Input/Output Redirection

- Using the "<" character after a command to redirect standard input:

  ```
      $ uniq < names.sort
  ```

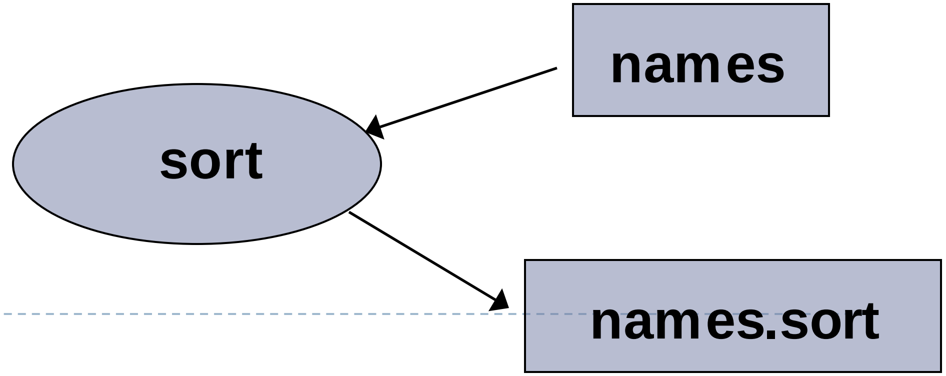
  This is the same as:

  ```
      $ uniq names.sort
  ```



- Using input and output redirection together:

  ```
  $ sort < names  > names.sort
  ```



Piping

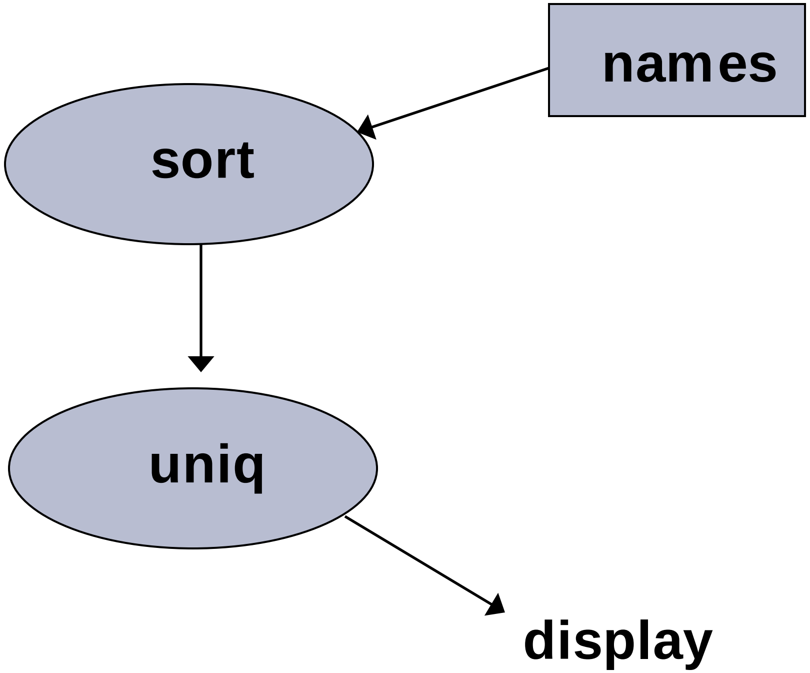
- Combining simple commands together to do more powerful things. This is accomplished using the "pipe“ character

<table>
<tr><th>Piping</th></tr>
<tr><td><command 1> | <command2></td></tr>
<tr><td>Passes the output from command1 to input of command2</td></tr>
<tr><td>Works for lots of programs that take input and provide output to the terminal</td></tr>
</table>

Pipes

- The standard output of a program can be "piped" into the standard input of another program:

```
$ sort names | uniq
```



Pipes

- Several pipes can be connected:

```
$ cat names | sort | uniq
Barak Obama
Bill Clinton
Bill Gates
George W. Bush
```

- Pipes and I/O redirection can be used together:

```
$ sort -r names | uniq > names.rev
$ cat names.rev
George W. Bush
Bill Gates
Bill Clinton
Barak Obama
```

Putting things together

An Example

We can put some of these together commands together now to do interesting things.

```
tr 'A-Z ' 'a-z\n' < file | sort | uniq -c | sort
-rn | head -n 10
```

tee

- What if you want to redirect your output to a file and still see it on the stdout?

tee Example

```
ls -l | tee output.txt
```