

COMP 2021

Unix and Script Programming



Course Information



Course Information

▶ Lecture

- ▶ Thur 13:00-14:50, Room 2503 (Lift 25/26)

▶ Instructor

- ▶ Dr. Cindy Li, lixin@cse.ust.hk, Room 3535 (Lift 25/26)

▶ Course website

- ▶ <http://course.cse.ust.hk/comp2021/>

▶ Labs

- ▶ Lab 1: Tue 09:00 – 10:50, Room 4214 (Lift 19),
- ▶ Lab 2: Mon 13:00 – 14:50, Room 4214 (Lift 19),

▶ TAs

- ▶ Mr. Zhang Hong and Mr. Hu Shuihai

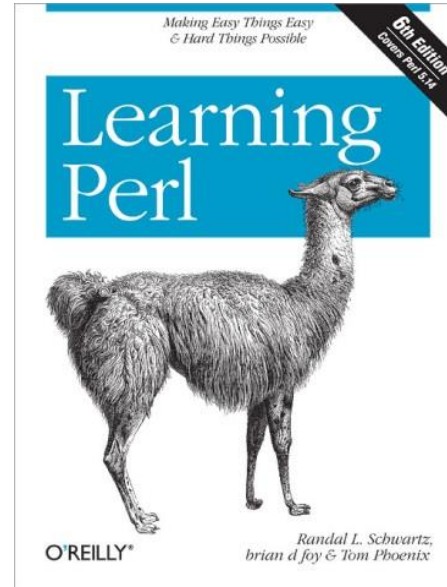
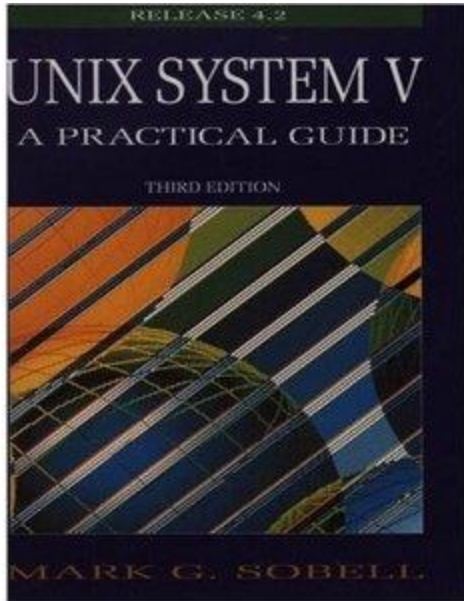


Course Objectives

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



Reference Textbook & Grading Scheme



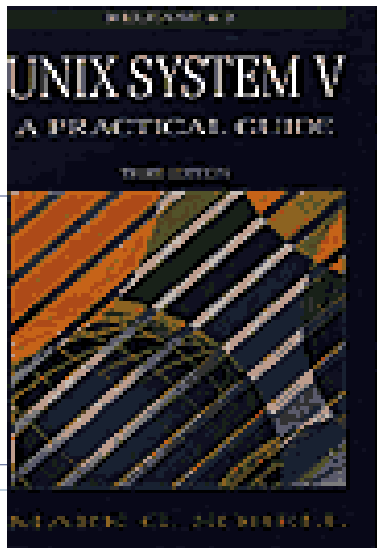
- ▶ Lab assignments (20%)
- ▶ “Midterm” exam (50%) (early May)
- ▶ Project and presentation (30%)



Comp2021 Project & Presentation

- ▶ Propose, implement, and document your own custom application.
- ▶ Choose your own topic that includes Unix, Shell scripting, or Perl
- ▶ Work in groups of normally 2 people.
- ▶ Presentations will be in the last few classes 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 notes
 - ▶ a softcopy of a short paper (4 pages) summarizing your project
 - ▶ source code (Perl, shellsript)





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

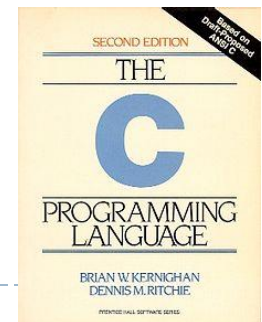
- ▶ UNIX is an operating system for experts, used on high-end workstations, database servers, and web servers.
- ▶ UNIX provides some powerful features:
 - ▶ security - private and shared files
 - ▶ multi-user support
 - ▶ data sent to display, files, or printers in same way
 - ▶ interprocess communication
- ▶ 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:



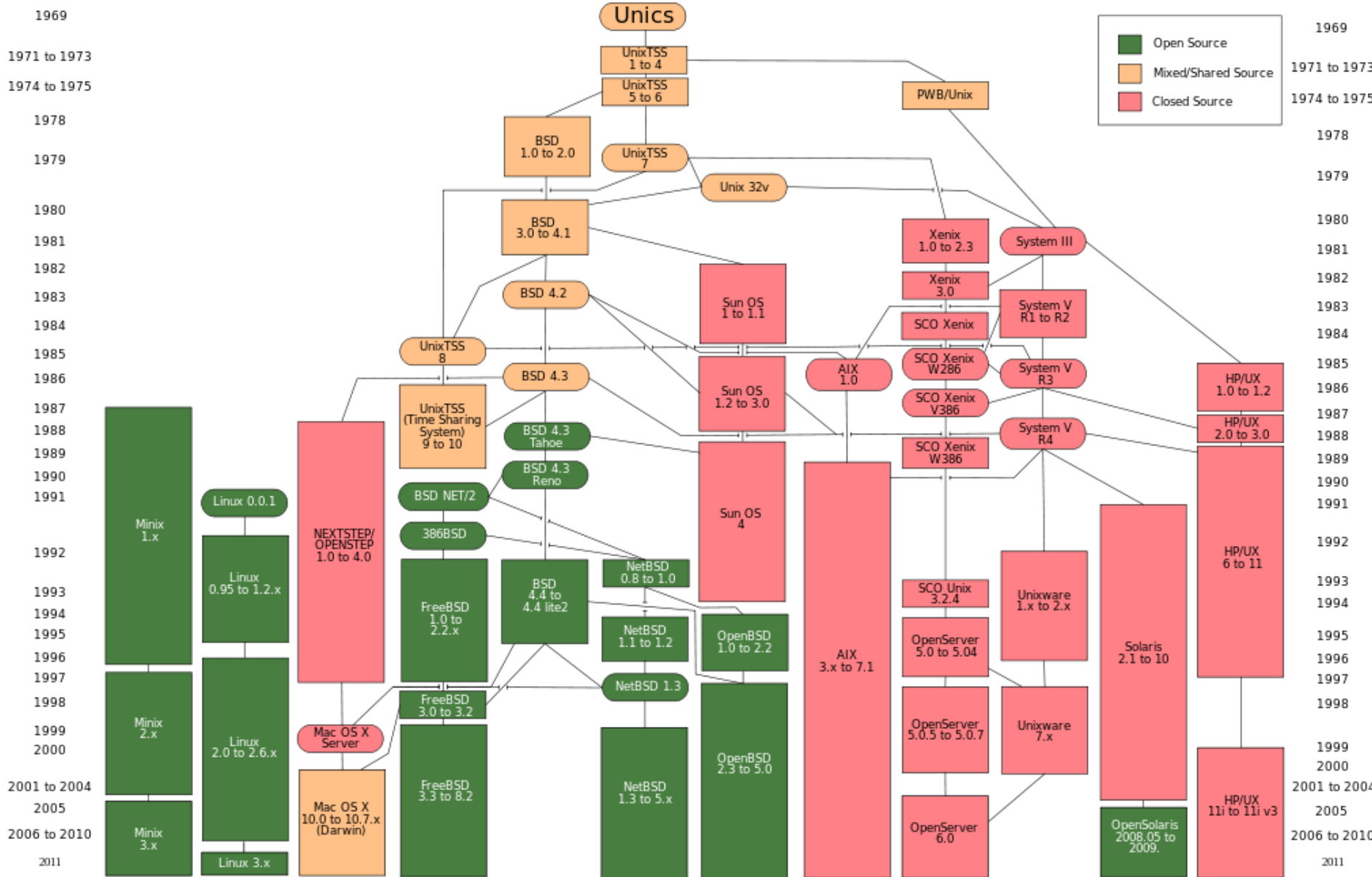
- ▶ Sun Microsystem's *Solaris*
- ▶ Mac OS/X
- ▶ Hewlett-Packard's *HP-UX*
- ▶ IBM's *AIX*
- ▶ SGI's *IRIX*



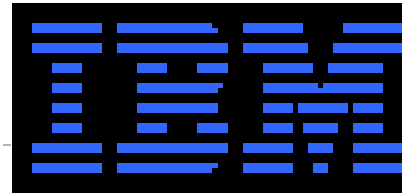
- ▶ Free Unix and Unix-like Operating system

- ▶ GNU project
- ▶ Linux
 - ▶ Pieced together by a Finnish guy named Linus Torvalds
 - ▶ Redhat, Fedora, Debian, Ubuntu, etc.





Who Uses UNIX?



- ▶ Big companies. They especially use UNIX servers, preferring its stability. They can afford to hire employees with UNIX experience.



HEWLETT
PACKARD



- ▶ 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

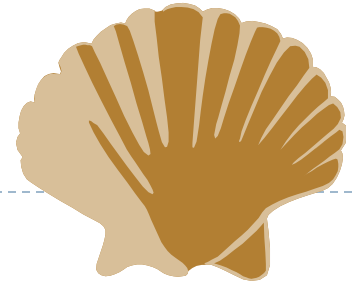


Most Important Feature of UNIX

- ▶ 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!



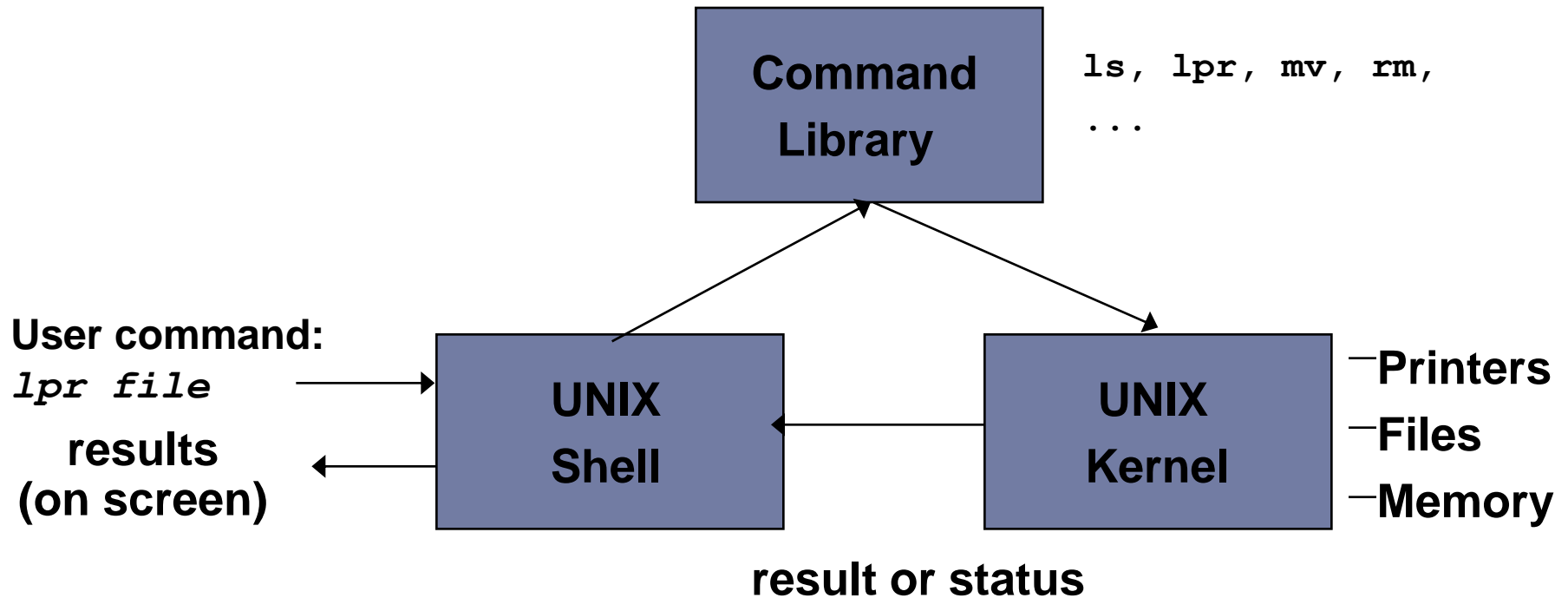
UNIX Shells



- ▶ 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.)



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 `cs12wk01` through `cs12wk41`.
- ▶ `cs12wk01` means “CSLab2, workstation#1”
- ▶ The full machine name for `cs12wk01` is:
`cs12wk01.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 `~`



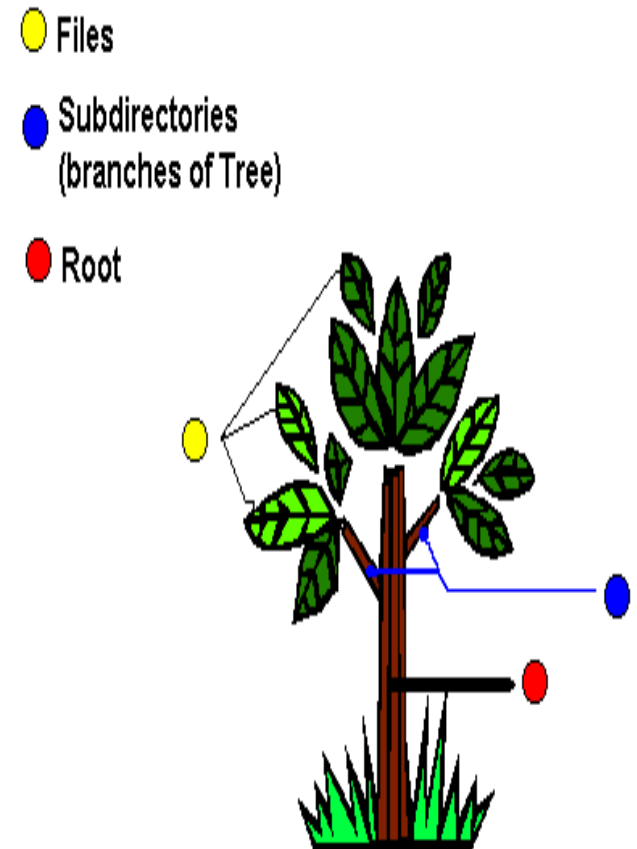


Unix Utilities

We roll our sleeves and get our hands dirty

Unix File System

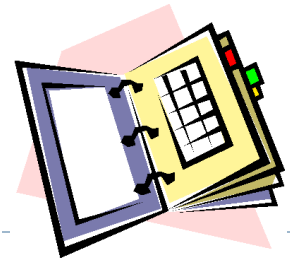
- ▶ Unlike windows, UNIX has a single global “root” directory /
 - ▶ Instead of a root directory for each disk/volume
- ▶ All files and directories are case sensitive
 - ▶ `hello.txt != hEllo.tXt`
- ▶ Directories are separated by / instead of \ in windows
 - ▶ UNIX: `/homes/lixin/comp2021`
 - ▶ Windows: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



What's Where

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

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



UNIX File Utilities

- ▶ `ls` **list** directory contents
- ▶ `cd` **change** **directory**
- ▶ `pwd` **print** **w**orking **d**irectory
- ▶ `cat` **display** file
- ▶ `more` **display** one screen of file
- ▶ `rm` **remove** (delete) a file
- ▶ `rmdir` **remove** (delete) **directory**
- ▶ `cp` **copy** source file to target file
- ▶ `mv` rename or **move** 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

▶▶ It's all Relative... except when it's not

Relative Path Shortcuts

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

Example	If we start in <code>/homes/lixin/comp2021/lab</code>
<code>cd</code>	<code>/homes/lixin</code>
<code>cd .</code>	<code>/homes/lixin/comp2021/lab</code>
<code>Cd ..</code>	<code>/homes/lixin/comp2021</code>



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 args (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`
 - ▶ `finger`
 - ▶ `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
```

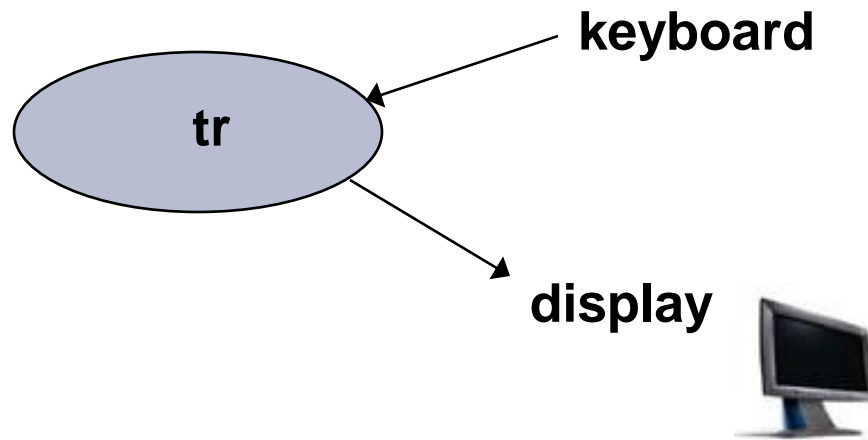
```
tr -d t  deletes t
```



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.



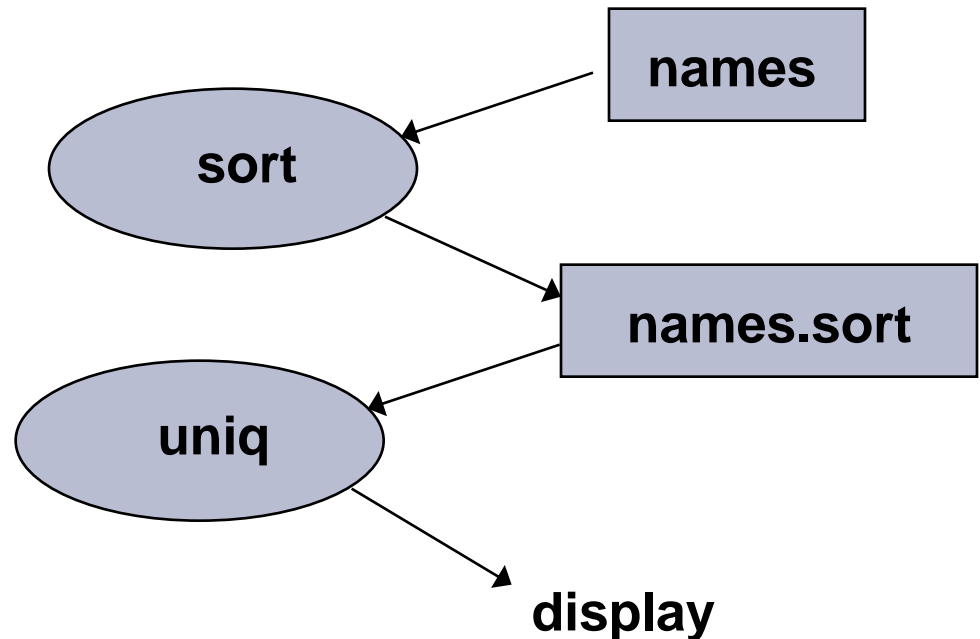
- ▶ What if we want to operate on files?
 - ▶ Piping: `cat somefile | tr 'AEIOU' 'aeiou'`
 - ▶ Input redirection: `tr 'AEIOU' 'aeiou' < somefile`



Input/Output Redirection

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

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





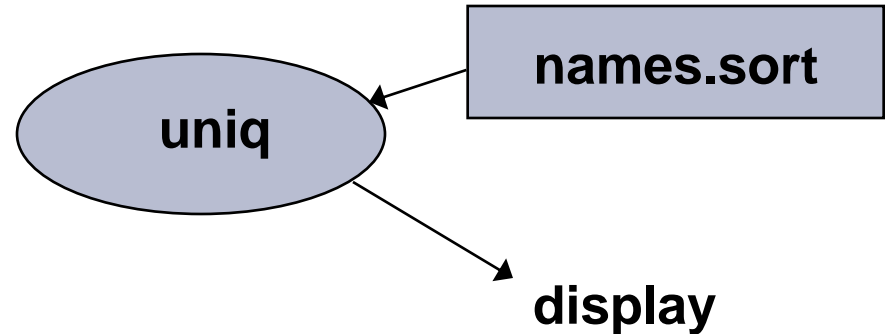
Input/Output Redirection

- ▶ Using the “<” character after a command to redirect standard input:

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

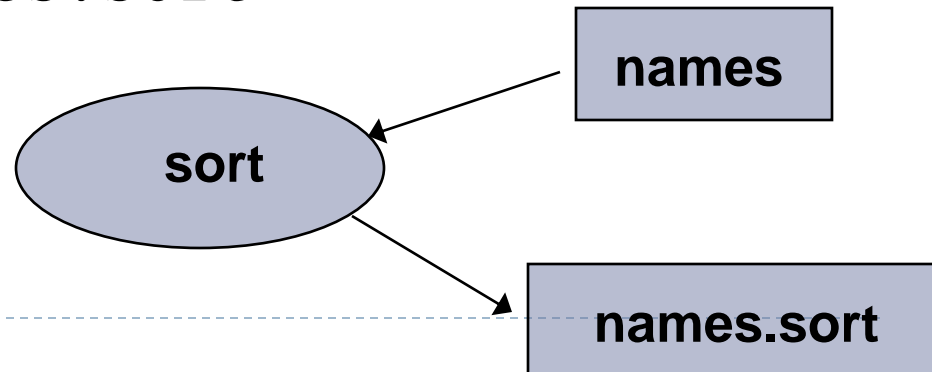
This is the same as:

```
$ 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

Piping

```
<command 1> | <command2>
```

Passes the output from command1 to input of command2

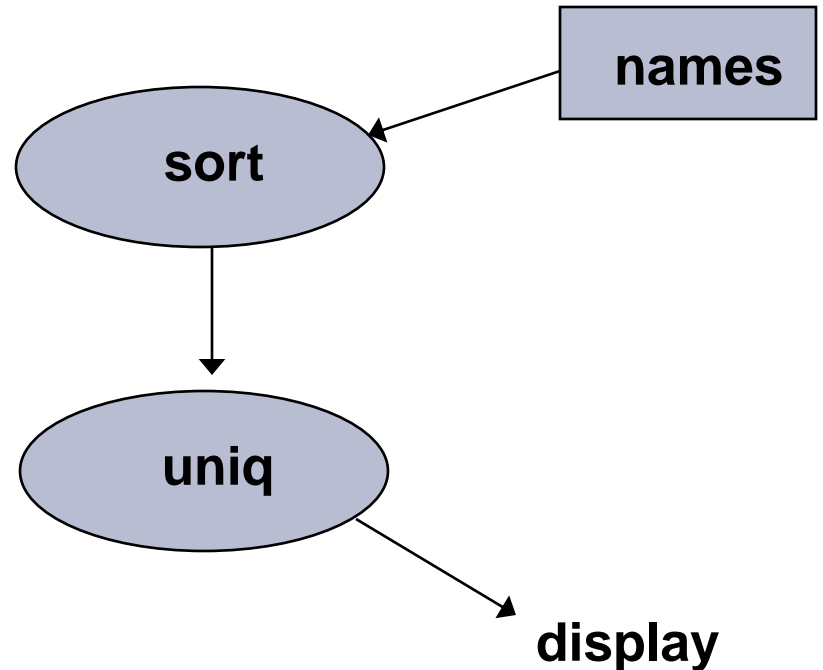
Works for lots of programs that take input and provide output to the terminal



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
```

