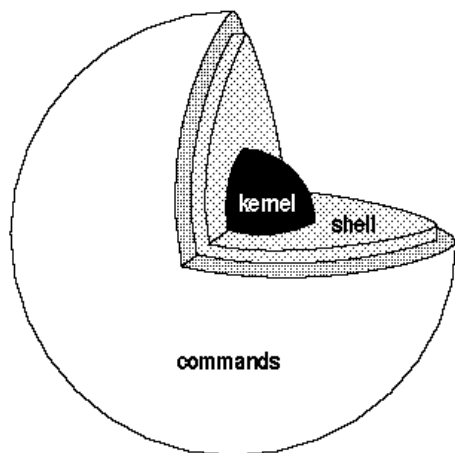


# UNIX COMMANDS AND SHELLS

UNIX Programming 2014 Fall by Euseong Seo

# What is a Shell?

- A system program that allows a user to execute
  - ▣ Shell functions (internal commands)
  - ▣ Other programs (external commands)
  - ▣ Shell scripts
- A psychic between kernel and users



```
2. bash
Last login: Mon Aug 11 10:43:56 on ttys000
MacPro:~ euisseong$
```

# Various Shell Breeds

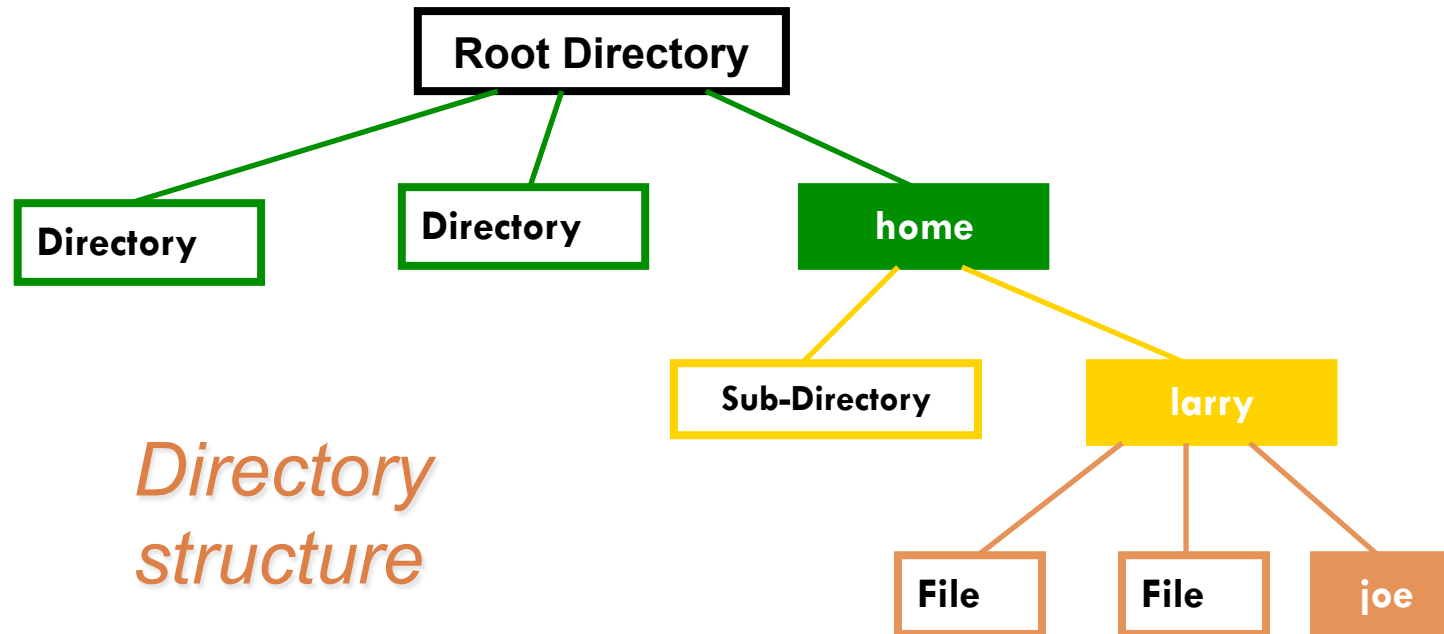
- Bourne shell (*sh*)
  - ▣ Written by [Stephen Bourne](#) at AT&T
  - ▣ Proposed the concept of pipe, redirection, command substitution, variables, control structures and so on
- C shell (*csh*)
  - ▣ Written by [Bill Joy](#) at UCB
  - ▣ Incorporated many new features including C-style control structure and expression grammar, history, editing, aliases, directory stack, tilde notation and so on
- TENEX C shell (*tcsh*)
- Bourne again shell (*bash*)
  - ▣ Being used by most Linux distros and Mac OS X

# Command Format

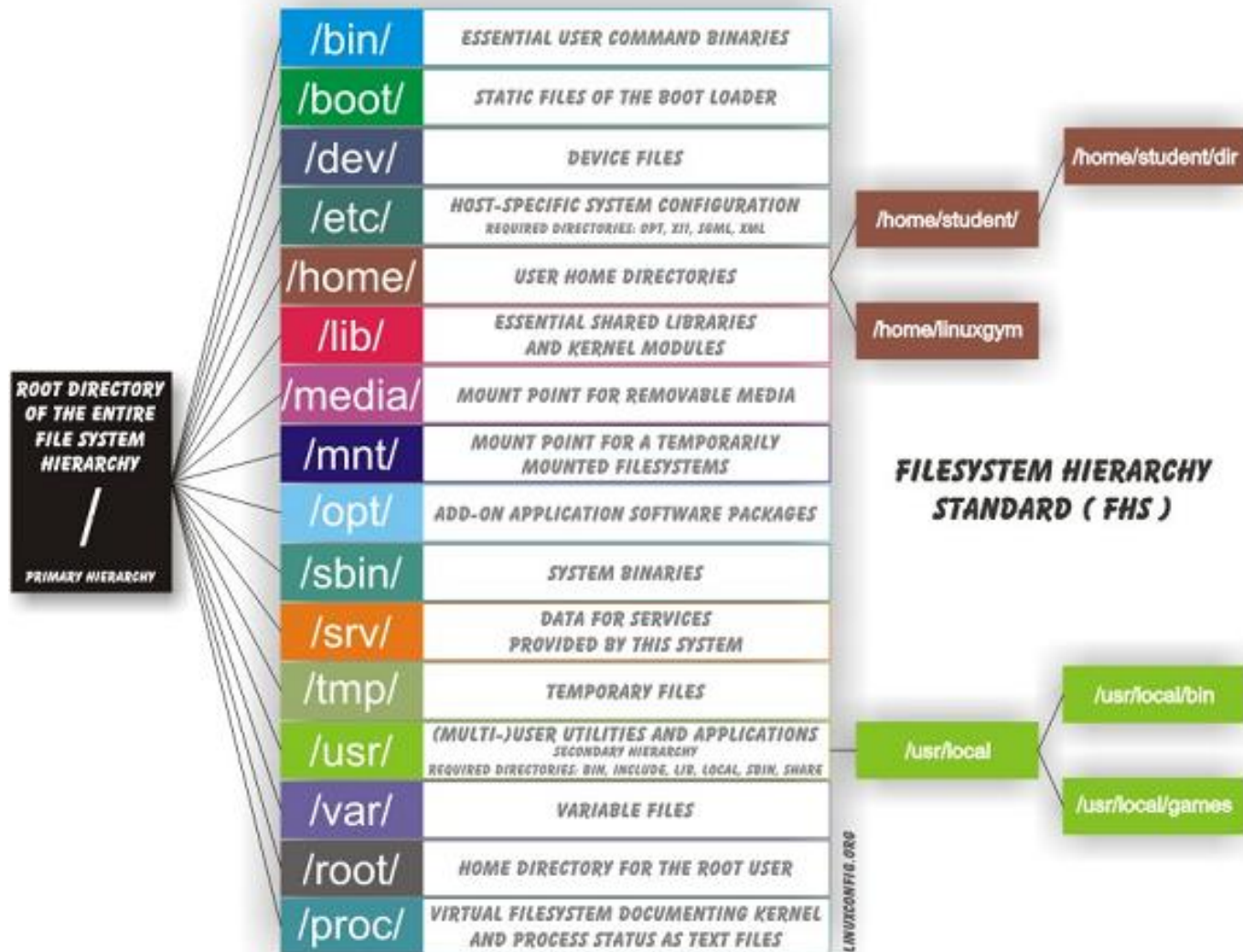
---

- Format principle
  - ▣ % commandname [*arg1*] ... [*argN*]
  - ▣ % means prompt
- Types of arguments
  - ▣ Options
    - Indication of modes of operations
    - Prefixed with - or -- (GNU style)
  - ▣ Operands
    - Data to work with
    - Actual data or file names

# UNIX Directory Structures



# UNIX Directory Structures



# UNIX Directory Structure

- Directory representation
  - ▣ Tree-like structure
  - ▣ /directory1/directory2/directory3/
- Root directory
  - ▣ Root of directory tree
  - ▣ /
- Current working directory (CWD)
  - ▣ Where you are currently working in
  - ▣ ./
- Parent directory
  - ▣ Parent of CWD
  - ▣ ../
- Home directory
  - ▣ For user's personal data
  - ▣ ~

# Traversing Directories

```
2. euseong@accept: /usr (ssh)
euseong@accept:~$ ls
Archive  Documents  Music  Pictures  Temp      Videos
Desktop  GPKI       NPki    Public   Templates Works
euseong@accept:~$ pwd
/home/euseong
euseong@accept:~$ cd Temp
euseong@accept:~/Temp$ cd /
euseong@accept:/$ ls
bin  home      lib32      logs      opt  sbin  usr
boot initrd.img lib64      lost+found proc  srv   var
dev  initrd.img.old libnss3.so media     root  sys   vmlinuz
etc  lib       libx32     mnt       run   tmp   vmlinuz.old
euseong@accept:/$ cd usr/lib
euseong@accept:/usr/lib$ pwd
/usr/lib
euseong@accept:/usr/lib$ cd ..
euseong@accept:/usr$ pwd
/usr
euseong@accept:/usr$ cd .
euseong@accept:/usr$ pwd
/usr
euseong@accept:/usr$
```





# Shell Built-in Features

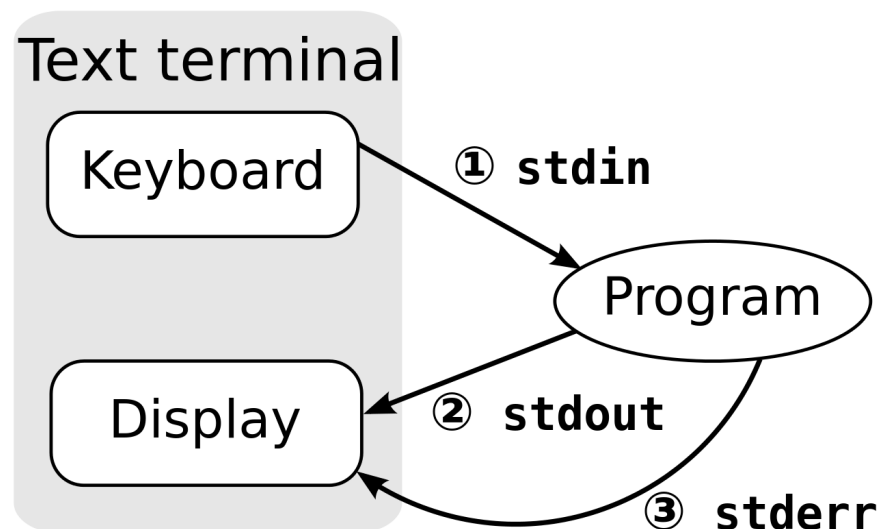
# Help!



- help
  - ▣ Display information about built-in commands
  - ▣ help [-dms] [pattern]
  - ▣ Without options, this will display the list of built-in commands
  - ▣ Let's do help help
- "RTFM!"

# Displaying Messages

- `echo`
  - Write arguments to **standard output**
  - `echo [-neE] [arg ...]`
  - Useful for shell scripting
  - `printf` of shell



# Directory Traversing Again



- `pwd`
  - ▣ Print the name of the current working directory
  
- `cd`
  - ▣ Change CWD
  - ▣ `cd [dir]`
  - ▣ Default DIR is the value of HOME **environment variable**

# Environment Variables



- A set of dynamic named values
- Affect the way running processes will behave
- A part of operating system environment
- Examples
  - ▣ HOME=/Users/euiseong
  - ▣ PWD=/Users/euiseong/Documents/Papers/vbuffer
  - ▣ PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/usr/texbin

# Set and Unset Environment Variables

- Set an environment variable
  - ▣ `export [name[=value]...] [-p]`
  - ▣ Example
    - `export TEMP=/tmp`
    - `export EMERGENCY`
- Unset an environment variable
  - ▣ `export -n name`
  - ▣ Example
    - `export -n TEMP`
- Referring to an environment variable
  - ▣ `$variable_name`
  - ▣ Example
    - `echo $PATH`
    - `export PATH=$PATH:./`

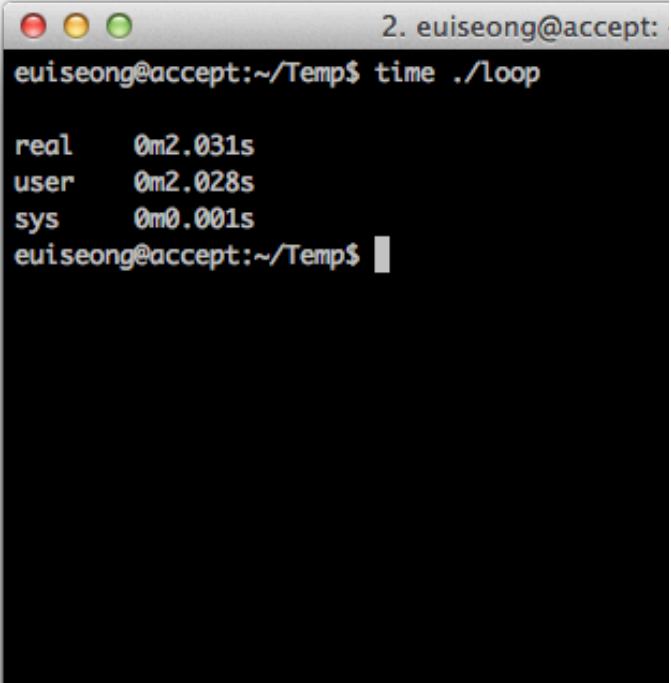
# Log In and Out



- Login shell
  - ▣ A shell obtained from a connection channel such as terminal, ssh or telnet
- logout
  - ▣ Exit a login shell
  - ▣ Returns an error if not executed in a login shell
- exit
  - ▣ Exit the shell
- Can you tell the difference between logout and exit?

# Measuring Time for Processing

- time pipeline
  - Report time consumed by `pipeline`'s execution



```
2. euseong@accept:
euseong@accept:~/Temp$ time ./loop

real    0m2.031s
user    0m2.028s
sys     0m0.001s
euseong@accept:~/Temp$
```

A terminal window with a title bar containing three colored circles (red, yellow, green) and the text "2. euseong@accept:". The terminal shows a command prompt "euseong@accept:~/Temp\$" followed by the command "time ./loop". The output of the command is displayed on the next line: "real 0m2.031s", "user 0m2.028s", and "sys 0m0.001s". The prompt "euseong@accept:~/Temp\$" is shown again on the following line with a cursor.





# External Commands

# Manual Pages

---

- man
  - Show manual pages
  - First command to remember
  - man man
- Layout
  - Name
  - Synopsis
  - Description
  - Example
  - See Also

# Directory and File Management

- ls [-ABCFGHLOPRSTUW@abcdefghijklmnopqrstuvwxyz1] [file ...]
  - ▣ List of files in a directory
  - ▣ -l option shows in the long format
- cp
  - ▣ Copy files or directories
  - ▣ cp -r
    - Recursive operation (for directory copy operations)
- rm
  - ▣ Remove files or directories
  - ▣ rm -f
    - Force operation

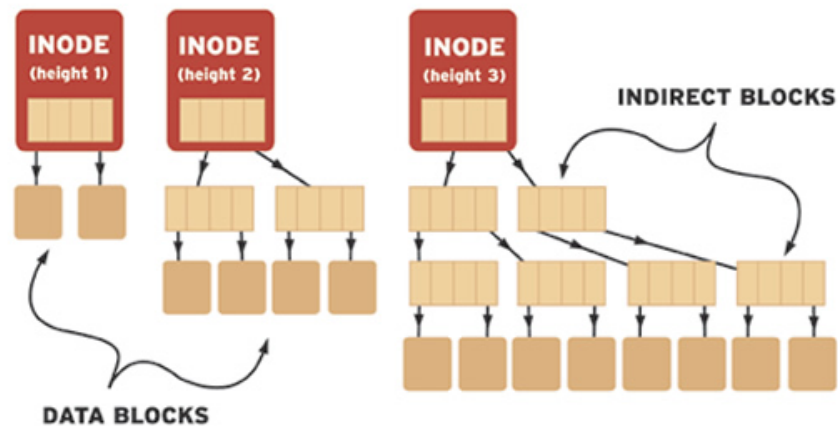
# Directory and File Management



- mkdir
  - ▣ Make a directory
- rmdir
  - ▣ Remove a directory
- file
  - ▣ Determine file type

# Symbolic Links and Hard Links

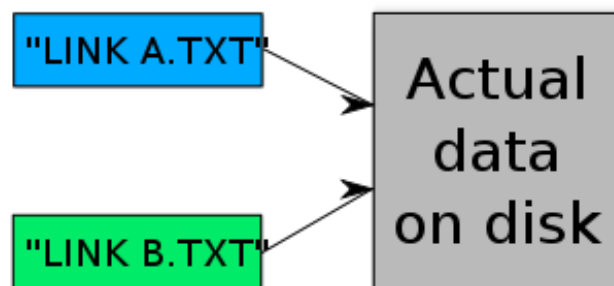
- Data and metadata
  - ▣ Metadata are what users see
  - ▣ In UNIX, an inode entry is a file to a user



# Symbolic Links and Hard Links

- Hard links

- Links between i-nodes and data blocks



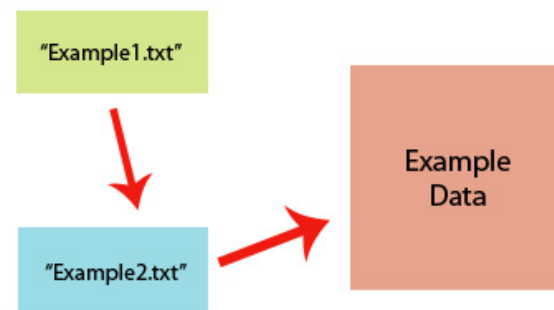
- If you erase a link, the data will not be removed from disks when there is a remaining link to the data
- An inode without a link to data is an erased file

# Symbolic Links and Hard Links

## □ Symbolic links

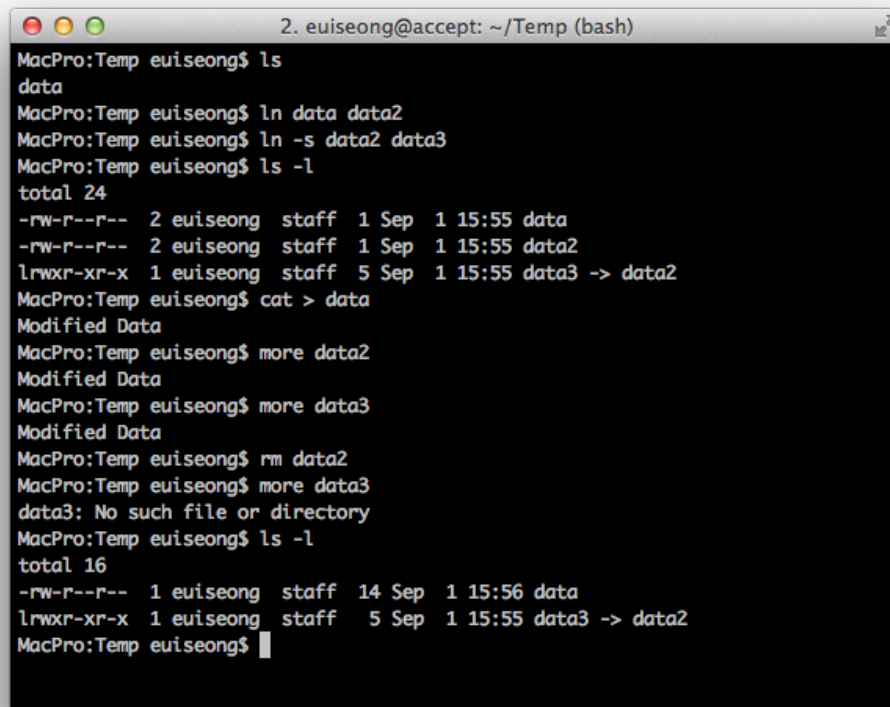
### □ A link between an inode and another inode

- Very much like a shortcut
- Automatically access “Example Data” when you access “Example1.txt”
- Useful but problematic
  - Removal of Example2.txt?
  - Removal of Example1.txt?
  - Can form a cycle!



# Symbolic Links and Hard Links

- In `source_file [target_file]`
  - ▣ Make links
  - ▣ `-s` for symbolic links



```
2. euseiong@accept: ~/Temp (bash)
MacPro:Temp euseiong$ ls
data
MacPro:Temp euseiong$ ln data data2
MacPro:Temp euseiong$ ln -s data2 data3
MacPro:Temp euseiong$ ls -l
total 24
-rw-r--r--  2 euseiong  staff  1 Sep  1 15:55 data
-rw-r--r--  2 euseiong  staff  1 Sep  1 15:55 data2
lrwxr-xr-x  1 euseiong  staff  5 Sep  1 15:55 data3 -> data2
MacPro:Temp euseiong$ cat > data
Modified Data
MacPro:Temp euseiong$ more data2
Modified Data
MacPro:Temp euseiong$ more data3
Modified Data
MacPro:Temp euseiong$ rm data2
MacPro:Temp euseiong$ more data3
data3: No such file or directory
MacPro:Temp euseiong$ ls -l
total 16
-rw-r--r--  1 euseiong  staff  14 Sep  1 15:56 data
lrwxr-xr-x  1 euseiong  staff  5 Sep  1 15:55 data3 -> data2
MacPro:Temp euseiong$
```



# Where to Find



- which program
  - ▣ Show the location of a program
  - ▣ Search for user-specific \$PATH
- whereis program
  - ▣ Show the location of a program
  - ▣ Search for hard-coded directory list

# Find Files

---

- `find [options] [-f path] path ... [expression]`
  - ▣ Walk a file hierarchy
  - ▣ Example
    - `find . -name "*hello*"`
    - `find . -newer "main.c"`

# User Profile Management



- chsh
  - ▣ Change user's login shell
- passwd
  - ▣ Change user's login password

# Date and Calendar



- `cal`
  - ▣ Display a calendar and the date of easter
- `date`
  - ▣ Display or set date and time

# Screen Clear



- clear
  - Clear screen
  - CTRL+L is an **alias** for clear

# Alias



- `alias [name[=value]]`
  - ▣ A built-in command
  - ▣ Make an alias of a command
  - ▣ `alias rm="rm -rf"`
- `unalias`
  - ▣ Erase an alias

# String Operations

- **wc**
  - ▣ Word (or line) count of a file
- **grep**
  - ▣ Generalized **regular expression** parser
  - ▣ Search a word in a text file
  - ▣ `grep -r "word" ./*`
- **cmp**
  - ▣ Compare two files
  - ▣ Merely tells you whether two files differ
- **diff**
  - ▣ Compare files line by line
  - ▣ `diff -rupN original/ new/ > original.patch` for Linux patch creation

# String Operations

---

- patch
  - ▣ Apply a diff file to an original
  - ▣ `patch < patchfile`
- `cat [filename]`
  - ▣ concatenate and print files
- `more (or less) [filename]`
  - ▣ Read a file page by page
- head and tail
  - ▣ Display first and last lines of a file, respectively





# Making Things Work Together

# Redirection

- Redirect standard output/error to a file or a file stream to standard input
- `>`
  - ▣ Redirect output to a file
  - ▣ `ls > result`
- `>>` Append redirected output to a file
  - ▣ `ls >> result`
- `<`
  - ▣ Redirect a file to a standard input
  - ▣ `cat < /etc/passwd`
- `&>`
  - ▣ Redirect both input and standard error to a file
  - ▣ `ls >& /dev/null`

# Pipeline



- A set of processes
  - ▣ Chained by their standard streams
  - ▣ Output of a process feeds directly as input to next one
  - ▣ Denoted by symbol ‘|’
  - ▣ Output of a program serves as input of another
- Combination of simple utilities can do more complex tasks
  - ▣ `ps A | grep 'usr/local/bin' | wc -l`