Shells

Shells

- A shell is a command line interpreter that is the interface between the user and the OS.
- The shell:
  - analyzes each command
  - determines what actions are to be performed
  - performs the actions
- Example:
  - wc –l file1 > file2

Which shell?

- sh – Bourne shell
  - Most common, other shells are a superset
  - Good for programming
- csh or tcsh – command-line default on CDF
  - C-like syntax
  - Best for interactive use. Not good for programming.
- bash – default on Linux (Bourne again shell)
  - Based on sh, with some csh features.
- zsh - newer shell
  - In many ways an extension of bash
  - programmable command-line completion

Changing your shell

- I recommend changing your working shell on CDF to bash
  - It will make it easier to test your shell programs.
  - You will only need to learn one set of syntax.
- What to do:
  - chsh <userid> bash
  - Logout and log back in.
  - .profile is executed every time you log in, so put your environment variables there

Important Shell Concepts

- Input/Output Redirection
  - Make standard input and/or standard output use files instead
- Pipes
  - Send the standard output of one program to the standard input of another
- Job control
  - foreground, background processes
  - killing and suspending processes
- Environment variables
- Wild cards

Redirection

- Input-output redirection
  - prog < infile > outfile
  - ls >& outfile # csh and bash stdout and stderr
  - ls > outfile 2>&1 # sh stdout and stderr

Pipes

- send the output from one command to the input of the next.
  - ls -l | wc
  - ps –aux | grep reid | sort

Job Control

- A job is a program whose execution has been initiated by the user.
- At any moment, a job can be running or suspended.
- Foreground job:
  - a program which has control of the terminal
- Background job:
  - runs concurrently with the parent shell and does not take control of the keyboard.
- Start a job in the background by appending &
- Commands: ^Z, jobs, fg, bg, kill

File Name Expansion

- `ls *.c`
- `rm file[1-6].?`
- `cd ~/bin`
- `ls ~reid`
- `ls *.[^oa]` - ^ in csh, ! in sh
- * stands in for 0 or more characters
- ? stands in for exactly one character
- [1-6] stands in for one of 1, 2, 3, 4, 5, 6
- [^oa] stands in for any char except o or a
- ~/ stands in for your home directory
- ~reid stands in for reid's home directory

Extra Material

- We likely will not cover much of the following material in class, but you should look at what some of the commands do, and you may find it useful.

Shell Programming (Bourne shell)

- Commands run from a file in a subshell
- A great way to automate a repeated sequence of commands.
- File starts with #!/bin/sh
  - absolute path to the shell program
  - not the same on every machine.
- Can also write programs interactively by starting a new shell at the command line.
  - Tip: this is a good way to test your shell programs

Example

- In a file:

```
#!/bin/sh
echo "Hello World!"
```

- At the command line:

```
skywolf% sh
sh-2.05b$ echo "Hello World"
Hello World
sh-2.05b$ exit
exit
skywolf%
```

Useful commands

- cut
- grep
- find
- ps
- sort
- uniq
- pwd

```
NAME
       cut - remove sections from each line of
files
SYNOPSIS
       cut [OPTION]... [FILE]...
DESCRIPTION
       Print selected parts of lines from each
FILE to standard output.

       -c, --characters=LIST
output only these characters

       -d, --delimiter=DELIM                    use
DELIM instead of TAB for field delimiter

       -f, --fields=LIST                   output
only these fields
```

find [path…] [expression]

- Expression
  - Options:
    - -maxdepth level
  - Tests:
    - -name pattern
      - Base of file name matches shell pattern pattern
    - -newer file
      - File was modified more recently the file.
  - Actions
    - -print
    - -exec

find and xargs

- find . -name "*.java" -print
  - Displays the names of all the Java files in directories in and below the current working directory.
- xargs
  - Build and execute command lines from standard input.

```
find . -name "*.java" -print | xargs grep
"import junit"
```

The power of pipelines

- How many people with cdf accounts are using the bash shell as their default shell?
- First we need to know that the default shell is stored in /etc/passwd

```
g4wang:x:10461:1009:Wang Guoyu:/h/u3/g4/00/
g4wang:/var/shell/bash
g4ali:x:10462:1009:Ali Muhammad:/h/u3/g4/00/
g4ali:/var/shell/tcsh
g4lily:x:10463:1009:Hu Lily:/h/u3/g4/00/
g4lily:/var/shell/tcsh
```

37

The power of pipelines

- Solution: `(almost)`

```
grep bash /etc/passwd | wc
```

- Answer:

- How many CDF accounts are there?
  - wc /etc/passwd
- Answer:

Another problem

- If I am logged into seawolf, how can I find out how many people are running bash or tcsh right now?
- Step 1: Display active processes using ps.
  - man ps
  - ps normally shows processes associated with your terminal use the options aux to display all processes.

More on grep and pipes

- Step 2: Extract the processes running bash.
- Solution:

```
ps aux | grep bash
```

- Step 3: Weed out the grep process (man grep)
- Solution :

```
ps aux | grep bash | grep -v grep
```

More on grep and pipes

- Step 4: Get rid of duplicate names
  - Strip out only the name
  - Use cut to break each line into fields.
  - Two ways to do it:
    - cut -d " " -f 1
      - Set the delimiter to be a space and select the first field.
    - cut -c -8
      - Select characters from beginning to the 8th one

More on grep and pipes

- Now get rid of duplicates

```
ps aux | grep bash |grep -v grep | cut -d
" " -f 1 | sort | uniq
```

- And finally, count them…

```
ps aux | grep bash |grep -v grep | cut -d
" " -f 1 | sort | uniq | wc -l
```