Shell Characteristics

- Command-line interface between the user and the system
- Automatically starts when you log in, waits for user to type in commands
- A Unix shell is both a command interpreter, which provides the user interface to the rich set of utilities, and a programming language, allowing these utilities to be combined.

Main Shell Features

- Interactivity
  - Aliases
  - File-name completion
- Scripting language
  - Allows programming (shell scripting) within the shell environment
  - Uses variables, loops, conditionals, etc.
  - Next lecture

Various Unix Shells

- sh (Bourne shell, original Unix shell)
- ksh (Korn shell)
- csh (C shell, developed at Berkeley)
- tcsh
- bash (Bourne again SHELL)
- Differences mostly in level of interactivity support and scripting details

Bourne Again SHELL

- We will be using bash as the standard shells for this class
- Superset of the Bourne shell (sh)
- Borrows features from sh, csh, tcsh, and ksh
- Created by the Free Software Foundation

Changing Your Shell

- On most Unix machines (including the lab) . . .
  - which bash
  - chsh
- On some machines . . .
  - Ypchsh
Environment Variables

- A set of variables the shell uses for certain operations
- Variables have a name and a value
- Current list can be displayed with the env command
- A particular variable’s value can be displayed with `echo $<var_name>`

Environment Variable Examples

- Some interesting environment variables:
  - `HOME /home/grads/callgood`
  - `PATH /usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin`
  - `PS1 \u@\h:\w\$`
  - `USER callgood`
  - `HOSTNAME mango.cslab.vt.edu`
  - `PWD /home/grads/callgood/cs2204`

Setting Environment Variables

- Set a variable with `<name>=<value>`
- Examples:
  - `PS1=myprompt>`
  - `PS1=$USER@$HOSTNAME:`
  - `PS1="multiple word prompt> "`
  - `PATH=$PATH:$HOME/bin`
  - `PATH=$PATH:~`
  - `DATE=`date`

Aliases

-Aliases are used as shorthand for frequently-used commands
- Syntax: `alias <shortcut>=<command>`
- Examples:
  - `alias ll="ls -lF"`
  - `alias la="ls -la"`
  - `alias m=more`
  - `alias up="cd .."`
  - `alias prompt="echo $PS1"`

Repeating Commands

- Use `history` command to list previously entered commands
- Use `fc -l <m> <n>` to list previously typed commands from m through n

Editing on the Command Line

- bash provides a number of line editing commands; many are the same as `emacs` editing commands
  - `M-b` Move back one word
  - `M-f` Move forward one word
  - `C-a` Move to beginning of line
  - `C-e` Move to end of line
  - `C-k` Kill text from cursor to end of line
Login Scripts

- You don't want to enter aliases, set environment variables, etc., each time you log in
- All of these things can be done in a script that is run each time the shell is started

Login Scripts (cont)

- For bash, order is ...
  - /etc/profile
  - ~/.bash_profile
  - ~/.bash_login (if no .bash_profile)
  - ~/.profile (if neither are present)
  - ~/.bashrc
- After logout ...
  - ~/.bash_logout

Example .bash_profile (partial)

```
#!/bin/bash

# include .bashrc if it exists
if [-f ~/.bashrc ]; then
  . ~/.bashrc
fi

# Set variables for a warm fuzzy environment
export CVSROOT=/cvsroot
export EDITOR=/usr/local/bin/emacs
export PAGER=/usr/local/bin/less
```

Example .bashrc (partial)

```
#!/bin/bash

# abbreviations for some common commands
alias f=finger
alias b=history
alias j=jobs
alias l="ls -lF"
alias la="ls -alF"
alias lo=logout
alias ls="ls -F"
```

Login Shell

```
/etc/profile
  ~/.bash_profile
  ~/.bashrc
  login shell
  interactive shell
    ~/.bashrc
  interactive shell
    ~/.bashrc
```

Background Processing

- Allows you to run your programs in the background

callgood@mango:~/$ emacs textfile &
callgood@mango:~/$
Each shell (and in fact all programs) automatically open three “files” when they start up:
- Standard input (stdin): Usually from the keyboard
- Standard output (stdout): Usually to the terminal
- Standard error (stderr): Usually to the terminal

Programs use these three files when reading (e.g. cin), writing (e.g. cout), or reporting errors/diagnostics.