Introduction to Unix

- History
- Definitions
- Types of Shells
- Basic Commands
- Files
- Online Unix Documentation

History of Unix

- Initially invented by Dennis Ritchie and Ken Thompson at AT&T Bell Labs in the early 1970s.
- Can find out more at:
  - http://www.bell-labs.com/history/unix
- Ritchie also developed the C programming language for the purpose of implementing Unix.

Unique Features of Unix

- Multitasking
- Multiuser
- Portability
- Application Libraries and Tools

Windows vs. Unix

- Windows (GUIs) are easier for novice to use.
- Unix allows a user to be more productive.
  - Unix is faster.
    - Typing a few characters is faster than dragging a mouse.
    - GUIs require overhead.
  - A sequence of tasks in Unix can be easily automated.
    - Parameterized scripts can contain a sequence of commands.
  - Tools in unix can be invoked so that they work together.
    - Most Unix tools read from standard input and write to standard output and can also be connected through pipes.

Unix Organization

- Kernel
  - operating system
  - schedules tasks
  - manages resources
- Shell
  - interprets user commands
  - executes programs
- Tools
  - invoked by the shell
  - hundreds available contributed from many sources

Varieties of Unix

- See http://www.levenez.com/unix/history.html
- System 5 (Solaris)
  - developed by AT&T and Sun
  - shell, diablo
- BSD (Berkeley Software Distribution)
  - developed at UC Berkeley
  - later evolved into Linux
  - linprog, quake

Definitions

- Executable
  - A program in a form that can be executed by the operating system.
- Process
  - The activation of an executable.
- Daemons
  - Processes spawned by the kernel (OS) to perform tasks to manage the resources of the computer system.

Definitions (cont.)

- Shell
  - Interprets the commands you type and runs the programs that you specify in your command line.
- Built-in commands
  - Performed directly by the shell without creating a new process.
- Utilities
  - Invoked by the shell by creating a new process.

Filters

- General-purpose utilities that:
  - read from standard input and write to standard output when no arguments are given
  - all information processed by the utility is contained in the input stream or command line arguments
  - the output of any utility should be usable as the input to another utility

Shells

- sh
  - $, Bourne shell, invented by Steve Bourne, commonly used for shell scripts.
- csh
  - %, C shell, closer to C syntax, commonly used for the command line interpreter.
- ksh
  - $, Korn shell, invented by David Korn, like csh but with history editing.
- tcsh
  - % or >, T shell, has all of the features of csh and others like command line editing.
- bash
  - $, Bourne-again shell, built on sh but has more advanced features.

Files

- file – a stream of bytes
- filenames – made up of any character except a slash (/)
  - case sensitive
  - Periods used for file extensions (often to indicate the type of file).
  - Filenames beginning with a dot (.) are treated a little differently.
  - Unix does not automatically make backups of files.

File Extensions (1.12)

- Unix has no specific rules regarding filename extensions.
- It depends on the utilities that access the files.

*.c – a C file

*.cpp – a C++ file

*.s – an assembly file

*.o – an object file

*.a – library archive

*.gz – gzipped file

File Extensions (cont.)

- Some file extensions may be for the user's benefit and are just conventions.
  - *.ps – a postscript file
  - *.pdf – a pdf file
  - *.tar – a tar archive file
  - *.sh – a shell script file
  - *.txt – a file containing regular text
  - *.dat – a file containing data

Wildcards (1.13)

- Can be used as a shorthand for specifying filenames.
- * – matches any character

  ***.sh prog* chap*.ps**
- ? – matches a single character

  prog.? chap?.tex
- [character-list] – matches a single character in the list

  d[1-9].db tmp[A-Za-z].txt

Filesystem (1.14)

- A filesystem is a group of files organized into a tree structure called directories.
- A directory is just a file that contains pointers to other files and other directories.
- / is the root of a file system.
- current working directory – the directory at which you have indicated to the shell that you are currently residing

Paths (1.16)

- Files are accessed by specifying the path of directories.
- absolute paths
  - Use a pathname where you start at / and you list the directories you go through, separated by /'s to reach the file.
- relative paths
  - Start with the location that is the current working directory and the path is relative to that location.

Path Abbreviations (1.16)

- . – indicates the current directory

  ./run.sh
- .. – indicates the parent directory

  ../list.txt
- ~/ – indicates your home directory

  ~/.login
- ~username – indicates a user's home directory

  ~whalley/cop4342exec

Basic Commands

- Some commands are built-in or internal to the shell. These commands do not cause the creation of a new process.
  - cd, alias, setenv
- Many commands are external to the shell. These commands to cause a new process to be created.
  - ls, cat, grep, awk, sed

Basic Commands for Navigating within a Filesystem (31.2 - 31.4)

- cd – change directory

  cd asg1 cd .. cd ~/classes/cop4342 cd
- pwd – print working directory
- mkdir – make a directory

  mkdir asg1
- rmdir – remove a directory

  rmdir asg2

Listing Files (8.2, 8.3, 8.5, 8.9)

- ls
  - list all files in alphabetical order in current working directory, except those that start with '.'
- ls <names>
  - list all files matching names, if directory matches one of the names, then list files in that directory
- ls -l
  - long listing, list files with more information
    - access mode, owner, group, size in bytes, date and time of last modification, and name
- ls -a
  - list all files including those starting with '.'
- ls -t
  - list all files in order of last modification time

Listing Files (cont.)

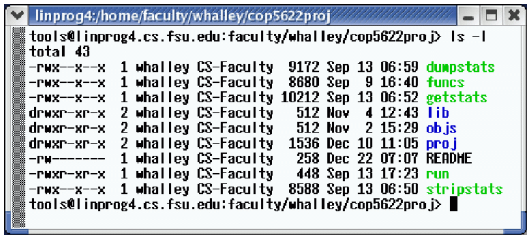
- ls -d
  - list name and not the contents of the directory
- ls -F
  - identifies the type of file with a trailing character
    - * for files with execute access
    - / for files that are directories
    - @ for files that are symbolic links

File Permissions (50.2)

- 3 types of processes that can access a file
  - user or owner: spawned by the user who created the file
  - group: spawned by members of the same group
  - others: spawned by anyone else
- 3 types of permissions
  - read: the ability to use the file as input
  - write: the ability to replace or update the file
  - execute : the ability use the file to create a process

File Permissions (cont.)

- Can be viewed by using the ls -l command.



Changing Permissions (50.5)

- Use the chmod command.
  - numeric using octal
    - "chmod 754 misc.txt" sets misc.txt to rwxr-xr--
  - symbolic
    - "chmod u=rwx,g=rx,o=r misc.txt" is same as above
    - "chmod og+r misc.txt" means add read permission for others and processes in the same group
    - "chmod o-rw misc.txt" means take away read and write permissions for others

Removing Files (14.3)

- "rm <filename>" to remove a file
  rm misc.txt
- "rm -r <dirname>" removes a directory and all of the files within it
  rm -r tmp
- Cannot recover a file after removing it short of doing a backup.

Copying and Renaming Files (10.9, 10.12)

- "cp <filename1> <filename2>" to copy a file.

  cp misc.txt misc.txt.old

  cp -r mydir otherdir

  cp figure1.ps figure2.ps tmp/
- "mv <filename1> <filename2>" to rename a file

  mv misc.txt misc.txt.old

  mv *.cpp ../.

Standard Input, Output, and Error (36.15)

- standard input (0: stdin)
  - the default place where a process reads its input
- standard output (1: stdout)
  - the default place where a process writes its output
- standard error (2: stderr)
  - the default place where a process can send its error messages

Redirecting Standard I/O (43.1)

- The ability to redirect standard input and output provides great flexibility.
  - The same tool can be used in many different ways.
  - Different tools can interact with one another.
- Default input is from the keyboard. Can redirect standard input from a file.
  a.out < input.txt
- Default output is to the terminal. Can redirect standard output to a file.
  a.out > output.txt

Redirecting Standard Error (43.1)

- Default error is to the terminal. Can redirect both standard output and standard error to a file.
  - tcsh

    a.out >& alloutput.txt
  - sh

    a.out 2>&1 alloutput.txt
- Can redirect standard error to a separate file.
  - tcsh

    (a.out > output.txt) >& error.txt

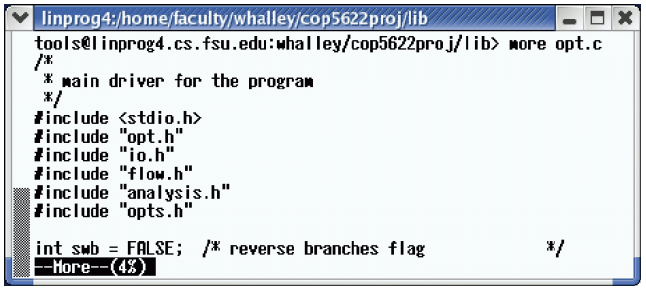
    (a.out > /dev/null) >& error.txt
  - sh

    a.out > output.txt 2> error.txt

Displaying Files (12.2,12.3)

- cat – print a file to standard output

  cat grades.txt
- more or less – print a file to the screen one page at a time to standard output



More or Less

- Both have similar commands.
  - space (' '): display another screen of lines
  - ^D: go forward one half screen of lines
  - newline ('\n'): display one more line
  - ^B: go back one screen of lines
  - ^U: go back one half screen of lines
  - q: exit from more or less
  - /pattern: searches forward for the first line containing the specified pattern
  - v: go into the vi editor at the current displayed line
  - =: display the current line number
- Less does not read the entire input file before starting, so it is more efficient with large files.

Appending to Files (43.1)

- Use the >> operator to append to a file.

  cp prologue_info.txt > tmp.out

  prog1.exe >> tmp.out

  prog2.exe >> tmp.out

  cat epilogue_info.txt >> tmp.out

Pipes (43.1)

- Pipes allow the standard output of one program to be used as the standard input of another program without specifying a temporary file.
- '|' operator means take the standard output from the command on the left and feed it as standard input to the command on the right.

```
prog1.exe < input.dat | prog2.exe | prog3.exe > output.dat
```

- Advantages
  - Fewer characters to type.
  - More efficient.

Splitting Standard Output (43.9)

- The tee utility copies its standard input to one or more files as well as standard output.

```
prog1.exe | tee out1.txt out2.txt | prog2.exe > out3.txt
prog.exe | tee tmp.out > prog.out
```

Accessing Documentation for Commands and Utilities (2.1)

- All Unix systems come with online documentation.
- To find out about the options for a specific command or utility type:

```
man <command_name>
    man ls
    man strcpy
```

- To find commands to perform a specific task:

```
man -k <topic_name>
    man -k directory
```