CS 126 Lecture P2: Introduction to Unix

Outline

• Background
• Files
• Processes
• Interactions
• Conclusion
What does an OS do?
- Make lives easy: hides low level details of bare machine
- Make lives fair: arbitrate competing resource demands

What we learn here: the interfaces by OS to upper layer
- User interface
- Programmer’s interface
- Command line vs. graphical user interface (more later)
A Brief History

• Multics (65-70)
  • Ambitious OS project at MIT
  • Pioneered most of the innovations in modern OS
  • A little ahead of its time

• Unix
  • Thompson and Ritchie (69): simplicity and elegance
  • AT&T (70-80s): continued development and “shepherding” it out of AT&T
  • Berkeley (“BSD”) (78-93): maturation (e.g. TCP/IP)
  • Various flavors of commercial Unix (80-90s): convergence and fragmentation
  • Linux (91-): new life

Outline

• Background

• Files
  • A simple and powerful abstraction for storage (disks)
  • Extended for things beyond disks

• Processes

• Interactions

• Conclusion
A Hierarchical Name Space: Same as folders and files on Windows or MacOS

File System

* "Everything in UNIX is a file"
* Abstract mechanism for storage

file: sequence of bytes
directory: sequence of files (and directories)

/bin /lib /etc /sys /tmp /u ...
/win/ ... /usr ... /dev ...

filename: sequence of directory names on the path from "/" to the file

File manipulation commands

- `cat, more`: show the contents
- `cp`: copy
- `rm`: remove (delete)
- `mv`: move (rename)
- `ls`: list file names
- `mkdir, rmdir`: create, delete directory
- `pwd`: name of current directory
- `cd`: change directory

```
- "..": current directory
- "...": parent directory
- "~": my home directory
- "xx": xx's home directory
- `chmod`: change permissions mode

- "*": any sequence of characters

DON'T TYPE "rm *"
```

why?
Outline

• Background
• Files
• Processes
  - An abstraction for the processor (CPU)
  - “Everything” (almost every command) is a process
• Interactions
• Conclusion

- A Unix “command” is the same as a Windows “program”
- Instead of clicking its icon under Windows, we simply type its name to invoke it on a command line.
Outline

• Background
• Files
• Processes
• **Interactions**
  - (between files and processes)
• Conclusion
I/O Redirection and Pipes

1: “Standard I/O”, 2: default attachment, 3: redirect output
4: redirect both input and output, 5: pipes

Filters and pipes

- Standard Input, Standard Output

stdin → command → stdout

abstract files for command interfaces

Redirection:
- standard input from file
- standard output to file

a.out > saveanswer
sort < myfile > myfilesorted

Piping:
- connect standard output of one command to standard input of the next

ls | wc -l | plotprog | lpr
plotprog | lpr

Don’t confuse redirection and piping

plotprog > lpr

outputfile
C Shell (/bin/csh)

```csh
#! /bin/csh
printf "Hello world! Give me a number: \n"
set n = $<
printf "Thanks! I've always been fond of %d \n" $n
```

Don’t worry about the details here.

- The program that’s running inside your terminal window
- Much more than just manipulating files and launching commands
- It’s an “interpreter”, with its own powerful programming language!
- Try your first “csh script”?

---

Command interface to UNIX

- Just another programming language
  - sequences of instructions
    - mv file1 tmp; mv file2 file1; mv tmp file
  - variables
    - printenv
  - arguments, flags
    - ls -lt *.c
  - conditional
  - looping
...

"EXTENSIBLE"

- add a new command with
  - cc avg.c
  - mv a.out avg
- also can add new commands with
  - chmod 755 doit
  - doit
where "doit" is a file with shell commands

Primary use
- low-overhead “programming” to manipulate files
- invoke commands
Outline

• Background
• Files
• Processes
• Interactions
• Conclusion

Choose Your Weapons Wisely

• C or Csh? “System programming” or “scripting”?
• Abstractions:
  - System programming
    - Compiled, rich types
    - Good for creating components which demand high performance or involve complex algorithms
  - Scripting
    - Interpreted, manipulates strings, less efficient
    - Good for gluing together existing components
    - Rapid development for gluing and GUI