COS 126 Lecture 3: UNIX

User Interface

Programming Environment

Operating System

Layers of abstraction in Unix

- bare hardware
- machine language
- kernel
- User level (C programming)
- Command level (shell)
- Window system

standard "programmer's" interface

File System

"Everything in UNIX is a file"

Abstract mechanism for storage

file:

    sequence of bytes

directory:

    "/"

    sequence of files (and directories)

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 *"

Shell

Command interface to UNIX

Just another programming language

    sequences of instructions

        mv file1 tmp; mv file2 file1; mv tmp file2

    variables

        printenv

    arguments, flags

        ls -lt *.c

    conditional

    looping

    ...

"EXTENSIBLE"

    add a new command with

        cc mycommand.c

        mv a.out mycommand

    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

Some Unix commands

lpr	output to printer
man, apropos	online documentation
grep, awk, sed	pattern search (stay tuned)
sort	sort the lines
diff	show differences
cal, date, time	time utilities
mail, news, pine	communication
bc, dc	calculators
cc, lcc, gcc	C compile
ed, vi, troff	text in and out (old)
history	past commands typed

Over 2500 "standard" commands
Thousands more "available" programs

emacs, tex, latex	text in and out
netscape	access web

Filters and pipes

Standard Input, Standard Output

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
gamblerall | avg
```

Don't confuse redirection and piping

```
plotprog > lpr
```

Multiprocessing

Abstraction provided by operating system
- multiple "virtual" machines for your use
- outgrowth of 1960s "time-sharing"
- not found on 1st-generation PC OS's

Multiple windows "active"??

Ex:

emacs hello.c &

ampersand indicates "do this in the background"
- alternatively, could use ctrl-z

```
% emacs hello.c &
[1] 18439
% netscape &
[2] 18434
% jobs
[1]  + Running                emacs hello.c
[2]  - Running                netscape
%
```

For COS126
- one window for editor
- one window for UNIX commands
  - **lcc, a.out, ls, cp**
- [one window for output]

Operating systems

Multics
- timesharing
- file system, protection
- virtual machines

OS/360

Macintosh
- windows, mouse

DOS
- PC standard OS

NextStep
- graphical interface
- (programmers use UNIX)
- future Mac OS?

UNIX
- C language, bootstrapped implementation
- integrated command structure
- simplified, integrated file system
- used by most programmers

Windows 95