Systems Programming/ C and UNIX

Alice E. Fischer

April 2, 2012

1. Preliminaries
   - Syllabus
   - Course Mechanics
2. Multics $\rightarrow$ UNIX
   - What did Unix Offer?
   - UNIX $\rightarrow$ Linux and Darwin
3. Unix System Structure
4. Basic Unix Tools
5. Overview of Unix Commands
6. Summary and Homework

Course Goals

- To increase expertise in C or C++ language programming.
- To do programming projects in a Unix environment.
- To review or learn advanced C/C++ topics, including macros, conditional compilation, dynamic memory, command-line arguments, bit operations, string parsing, const and pointers.
- To interface to standard Unix system functions that manipulate files and directory information.
- To learn UNIX processes and threads and the commands for manipulating them (fork, exec, signal, wait, etc.).
- To learn about interprocess communication through shared memory using semaphores and file locking.
- To learn about interprocess communication and client/server systems (pipes, dup, dup2, sockets).

Prerequisites

- Ability to program fluently in C or C++ at the intermediate level. (CS 212 or CS 610)
- The time and discipline to do the work on time.
- Ability to buy the textbook and use it as a reference manual.
- Willingness and ability to plan before you code and use a modular, structured coding style. Note: You will never get the programs debugged if you try to write everything in the main function.

Procedures

See the syllabus, on the website for details.

- The website: http://eliza.newhaven.edu/sysprog/
- To turn in work: email zipped files to alice@we2skate.net
  Include source code, output, and a makefile.
- Missed exams: contact me within 48 hours!
- Academic integrity: Absolutely no sharing of any sort will be tolerated in this course. All work is to be done independently.
- Timeliness: Do the work when it is assigned. There will be no INC grades except for special problems.

Software Needs

This is a UNIX course!

- You need access to a computer running a UNIX system.
  - If you have a Mac, use it. OS-X is a Unix system.
  - For non-Mac owners, install Linux + VirtualBox on your computer and run Windows inside it.
  - Or, just install a Linux system on your own computer.
  - I do not recommend installing a dual-boot system. Use VirtualBox.
- Use the latest release of Fedora Linux.
- Fedora is free. You can borrow an install disk from Dr. Eggert.
- All the programming work must be done using tools supported by Linux or OS-X.
- Use one of the compilers from the Free Software Foundation: gcc or g++ – not using Visual Studio.

The Evolution of Unix

Image: Elephant Evolution, from dkimages.com

Left to right: Linux, Michigan System, Multics, 1965 CTSS, 1971 UNIX

The World of 1965: An ambitious new system

- Machines were physically huge. They were slow and had very small main memories and small-capacity disks.
- MIT + GE + Bell Labs collaborated to design and build a computer (the GE 645) that supported virtual memory with paging and segmentation.
- Memory size was 256,000 words of 36 bits each: the equivalent of 1,152,000 8-bit bytes or 1 megabyte.
- The intent of Multics was to support multiple simultaneous users, but keep each one user's data and programs secure from the others. To do this, there would be three copies of the page table and segment table for each process. This allowed three levels of access (kernel level, device level, process level) to the user's memory, while fully separating users from the hardware.
- Too slow, too small, too complicated, too ambitious. A costly system with a short life span.

The Original C and UNIX

- In 1969, Dennis Ritchie, Ken Thompson, Brian Kernighan, Douglas McIlroy, and Joe Ossanna created Unix (a simple, sleek, time-sharing operating system) at Bell Labs, in their spare time.
- The C language was developed (replacing FORTRAN and PL/1) for the purpose of writing the Unix system.
- In the next two decades, several variants were developed, including Berkeley Software Distribution (BSD) Unix, from which Linux evolved.
- Both BSD Unix and C were distributed to government and academic institutions, which led to both being ported to a wider variety of machine families than any other operating system.
- As a result, Unix became closely associated with the open software movement.

Why was Unix Popular

- Unlike its predecessor, Fortran Monitor System, it supported full time-sharing.
- Like Multics, it supported multiple concurrent jobs with scheduling and protection.
- Like Multics, it supported remote login and terminals.
- Like Multics, file and process security were built into the system.
- Unix provided the first modern file system.
- The Unix system and its C language were easily portable to many platforms.

Unix Tools

Extensive libraries of tools were developed at many universities for common system administration tasks.

- Shells (sh (Bourne), Korn, **bash**; csh (C shell), **tcsh**)
- Shell macros.
- Makefiles and the make system.
- emacs and vi text editors.
- The gnu C compiler – still the world's best.
- Typesetting systems, from troff (Kernighan, early 90's) to Tex (Knuth, 1989) and LaTex (Lamport, early 90's).
- diff, sed, awk, grep, uniq, cat, less, sort, etc.

Modern UNIX-like Implementations

The term "Unix" is copyrighted. Technically, all the variants that we use are not Unix, they are Unix-like. Most faithfully implement the basic features prescribed by the Posix standard; differences are in the implementations and in some advanced features. For brevity, in this course, I will use "Unix" instead of "Unix-like".

- Linux runs on Intel-based PC's and similar hardware.
- SunOS and Sun Solaris
- IBM AIX
- Macintosh OS-X is built on top of the Darwin System, derived from BSD Unix and resting on the Mach microkernel (Carnegie-Mellon).

How does UNIX differ from Windows?

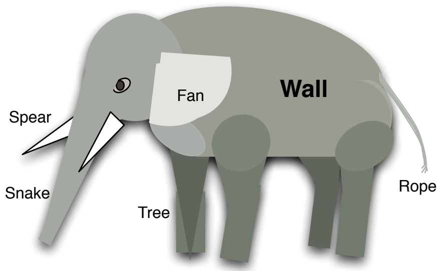
- Implementations are free, and vast amounts of free software exist for any UNIX system.
- UNIX is a layered system with a well-defined command layer that is separate from the GUI interface.
- UNIX applications can be installed and updated without fussing with a "registry". UNIX apps live in self-contained packages that can be moved around and installed anywhere.
- UNIX is more fully debugged (being much older and more stable).
- The file system is built around a combination of privacy and sharing.
- Every UNIX system can function as a server (i.e., for remote login).
- There is far, far less problem with malware, and invasions are more easily corrected. Estimates are that over half the Windows installations in the USA are currently functioning as zombies.

So why do people still use Windows?

- Everyone else uses Windows.
- Windows supports C#, a useful proprietary language.
- Until recently, a Windows system was bundled with the the hardware sold by every major manufacturer.
- It is still difficult to buy a cheap computer without buying Windows.
- Mac's are expensive and only a Mac can run OS-X.
- There are so many versions of Linux that it gets confusing (Ubuntu, Fedora, Debian). Each one tries to assemble a collection of free parts that are intended (and configured) to work together. None is perfect.
- It takes knowledge and judgement to install a full Linux system.
- There is no single desktop system with Linux. Common choices are:
  - KDE: the designers tried to emulate Windows, but did a somewhat better job.
  - Gnome: from Gnu, less Windows-like. Current versions are pretty good.

The Blind Men and the Elephant

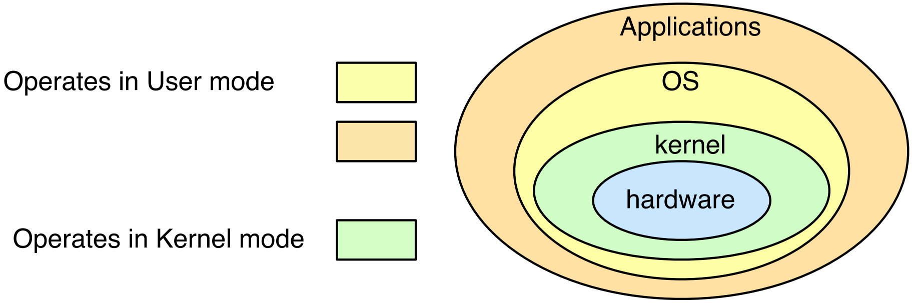
Six blind men came upon an elephant. Each one felt a part of the beast to find out what an elephant looked like.



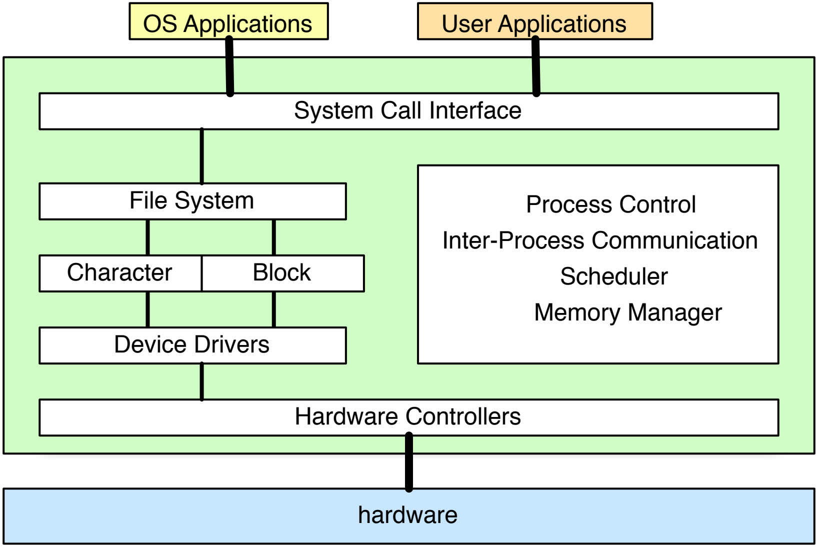
- The man who touched the elephant's side decided it was like a wall.
- The man with the tail thought it was a rope.
- A third man found two trees.
- A fourth found a powerful snake.
- The tallest two men found a spear and a fan.

So they argued all day and all night about the nature of an elephant!

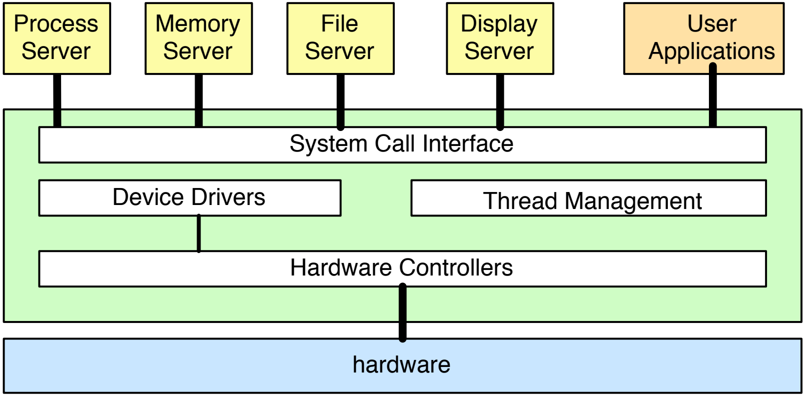
Kernel Mode and User Mode



UNIX Kernel Structure

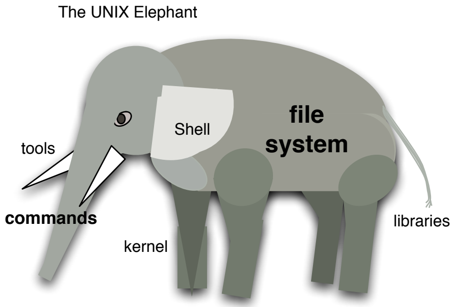


The Mach Micro-kernel Structure



Unix User Interface

The Unix manual (man pages) is divided into sections, according to the nature of the facilities it documents.



1. User commands
2. System calls
3. Libraries
4. Devices
5. Configuration
8. System administration
9. System kernel interfaces

Section 6 is no longer used and Section 7 contains summaries and miscellaneous information.

The Command Shell

- Unix is based on a security model.
- Files have owners; processes have owners.
- The shell is a process
- When you open a shell it loads the owner's `.cshrc` file, which lives in the owner's home directory.
- It consists of a sequence of shell commands and constant declarations that set up the shell preferences the way the user likes it.
- Shells can execute both shell script and executable programs.

Shell Services

The shell is a system program that provides an interface for users.

- A shell is opened for the user when he logs in remotely or double-clicks on the shell icon.
- It expects to receive command lines from a terminal or a terminal emulator. (How ancient!)
- It parses the command line, consisting of a command name followed by switches and other arguments.
- Commands can contain clauses that redirect the `stdin`, `stdout` or `stderr` streams to files.
- The shell parses and partially interprets the command line, expands wild-card patterns, substitutes variables, redirects streams if appropriate, and starts up a process executing the desired code.
- The resulting parameters are passed along to the new process.

Command Lines

We use a shell by typing command lines into the window. Elements are:

- The name of the command to run
- Switches
  - Short switches start with – followed by a single letter.
  - Long switches start with – – followed by a word.
  - A switch can be followed by an argument string.
  - Several short switches can be given after a single –
- Arguments: any string that does not start with -
- Stream redirection clauses:
  - `<filename` redirects stdin to the named file.
  - `>filename` redirects stdout, discarding former file contents.
  - `>>filename` redirects stdout to append to the named file.

A file-name argument that contains a * is a glob pattern, to be expanded by the shell.

Sample Command Lines

Identify the command names, short switches, long switches, arguments, glob patterns, and redirection:

```
bash-3.2$ pwd
bash-3.2$ slogin eliza.newhaven.edu
bash-3.2$ cd w/java
bash-3.2$ ls -l
bash-3.2$ gcc -o myCommand myMain.c myClass.c
bash-3.2$ rsync -avu w/java/ eliza.newhaven.edu:w/java/
bash-3.2$ rsync -au --verbose w/java/ kira:w/java/
bash-3.2$ grep -i "CS 657" *.html > mytemp.txt
bash-3.2$ rm *.bak *.log
```

Basic UNIX Commands You Need to Know

Command	Meaning
man commandName	Show manual page for commandName.
man 3 functionName	Show manual page for this function.
pwd	Print path name of current working directory.
whoami	Print username of person currently logged in.
passwd	Permits you to change your password
cat myfile.txt	Copies myfile.txt to stdout.
more myfile.c	Display contents of file on screen. Enter key moves forward a line. Space moves a page.
less myfile.txt	Like more, but type b to move backward.
diff file1.txt file2.txt	Display differences in two versions of same file.
gzip myfile.txt	Compress a file.

Search Paths

When you execute a command from the shell, the shell-process must find the definition of that command on your machine.

- To do so, it uses a search path that was loaded when you logged on.
- The search path tells where your commands are on your machine. There is no “special” place where they must be, but built-in commands are typically found in these places:
  /bin: the most fundamental commands.
  /usr/bin: most commands and utility programs.
  /sbin: commands for system administrators.
  Some commands are built into the shell.
- By default, your current working directory is NOT on the search path. To execute a command you just compiled, you need to change the search path or specify the current working directory as part of the command name:
  ```
  ./alice w/java
  ```

A Beginner's .cshrc File

- Your .cshrc file sets you preferences when you open a shell window.
- There are a few things I need to work in a civilized manner.
- Please put a .cshrc file in your home directory with these contents:

```
# Set my preferences
path = ($path .)
alias c99 gcc -std=c99
alias la ls -a
alias ll ls -l
set prompt = "%m:[%n] %~~> "
umask 13
set noclobber
```

- Note: In a shell, execute cd to go to your home directory.
- If you have trouble giving the correct name to this file, create a file with any name, then use the shell to change the name.

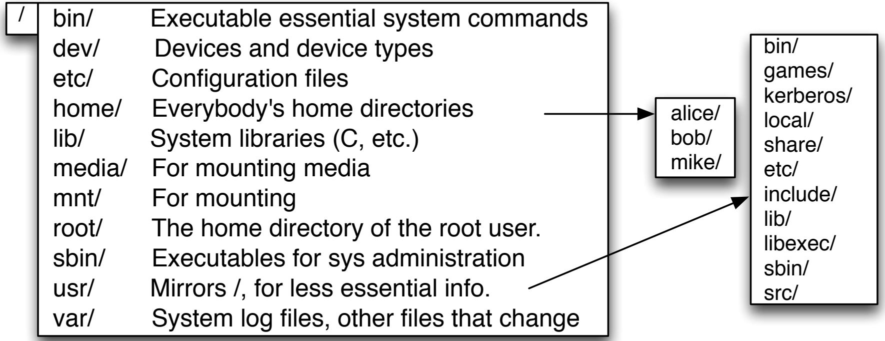
A Fully Mounted File System

Windows pathnames all start with a device code, such as `C:` . Unix pathnames don't.

- All Unix pathnames start at the root directory, which is written as: /
- When a removable device (stick memory, CD) is inserted into the computer, a modern Unix system will automatically `mount` it. (Many years ago, the user wrote the mount command himself.)
- For this purpose, several `mount points` are built into the Unix file system. For example, `/bin/media` and `/bin/mnt`.
- The root directory of the removable device is grafted onto the mount point.
- Thereafter, the files can be reached from anywhere else in the file system.

The Unix Root directory

The Organization of a Linux File System:



/		
	bin/	Executable essential system commands
	dev/	Devices and device types
	etc/	Configuration files
	home/	Everybody's home directories
	lib/	System libraries (C, etc.)
	media/	For mounting media
	mnt/	For mounting
	root/	The home directory of the root user.
	sbin/	Executables for sys administration
	usr/	Mirrors /, for less essential info.
	var/	System log files, other files that change

What is in the bin directory?

The `bin` directory stores the executable code and scripts for commands that are necessary for basic system administration. Here are the most familiar and useful:

bash	cat	df	ps
tcsh	echo	ln	pwd
date	chmod	ls	rm
hostname	chown	mkdir	rmdir
kill	cp	mv	unlink

Some things are in /bin in any Unix-like system; the things listed here are the same in OS-X and Linux. However, Linux has important commands in /bin that are found in /usr/bin in OS-X.

In /bin or in /usr/bin: More Basic Commands

Languages:
emacs
ed
vi
awk
sed
c99
gcc
g++
java
python
ruby

Directories:
cd
find
mount
stat
tar

Files:
cat
diff
less
more
touch
umask

Utilities:
ftp
grep
gzip
gunzip
make
man
svn
rdiff-backup
rsync
sort
uniq

Shell:
alias
chsh
echo
logout
ping
passwd
path
rehash
slogin
ssh
su
sudo
which
whoami

Summary

Tonight's topics included:

- A brief history of Unix.
- The shell and the `.cshrc` file.
- The structure and features of a command line.
- Basic, essential shell commands.
- Mounted file systems.
- The Unix reference manual, or man pages,
  1. Section 1 Shell commands
  2. Section 3 System libraries

Homework

- Install a Unix system on your own computer if you do not already have one. If you need both Windows and Linux, install Virtual Box and put Windows inside the box (or vice versa).
- In the man pages, look up four commands from the list on lecture notes page 30. Choose an unfamiliar command from each category (Directories, Files, Utilities, Shell). Turn in a 1-sentence description of what each command is used for.
- Write a program to display and analyze a command line. See P1 instructions. Use your IDE to create the program, but compile and run it using a command shell. This program is under 20 lines in either C or C++.