System Programming

Introduction to Unix

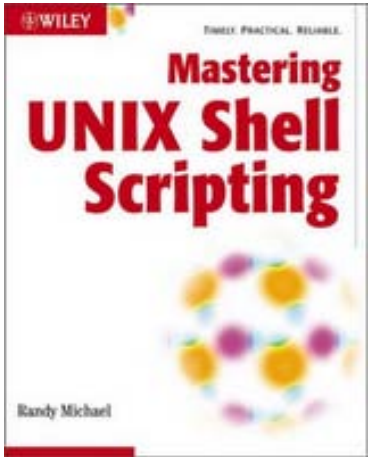
Recommended Course Textbooks

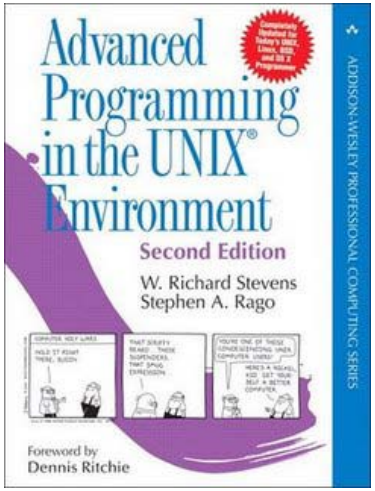
- S. G. Kochan, P. Wood (2003)
  Unix Shell Programming, 3rd Edition, Sams, 460 p.

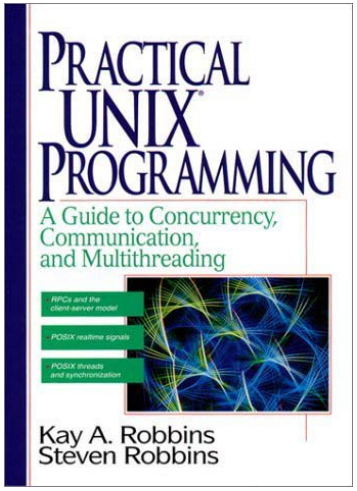
- R. K. Michael (2003)
  Mastering UNIX Shell Scripting, Wiley, 680 p.

- W. R. Stevens, S. A. Rago (2005)
  Advanced Programming in the Unix environment, 2nd Edition, Addison Wesley, 960 p.

- K. A. Robbins, S. Robbins (1996)
  Practical Unix Programming, Prentice Hall, 658 p.







What is Unix?

- A modern computer operating system
- Operating system
  - “a program that acts as an intermediary between a user of the computer and the computer hardware”
  - Software that manages your computer’s resources (files, programs, disks, network, ...)
  - Examples: Windows, MacOS, Solaris, BSD, Linux (e.g. Mandrake, Red Hat, Slackware)
- Modern
  - Stable, flexible, configurable, allows multiple users and programs

Why Unix?

- Used in many scientific and industrial settings
- Huge number of free and well-written software programs
- Open-source operating system (OS)
- Excellent programming environment
- Largely hardware-independent
- Based on standards
- Internet servers and services run on Unix
  - Roughly 65% of the world's web servers are Linux/Unix machines running Apache

Brief History of Unix

- Ken Thompson & Dennis Richie originally developed the earliest versions of Unix at Bell Labs for internal use in 1970s
  - Simple and elegant
  - Borrowed best ideas from other OSs
  - Meant for programmers and computer experts
  - Meant to run on “mini computers”

Early Unix History

- Thompson also rewrote the operating system in high level language of his own design which he called B.
- The B language lacked many features and Ritchie decided to design a successor to B which he called C.
- They then rewrote Unix in the C programming language to aid in portability.
  - Small portion written in assembly language (kernel)
  - Remaining code written in C on top of the kernel

Unix History

- Multics 1965
- First Edition 1971 (AT&T) (CACM 1974, 365-375)
- 1BSD 1977 (Berkeley)
- Sixth Edition 1975 (AT&T)
- 4BSD 1980 (Berkeley)
- SunOS 1985 (Sun)
- System V 1985 (AT&T)
- Tenth Edition 1989 (AT&T)
- 4.3BSD Net/2 1991 (Berkeley)
- First Linux kernel 1992 (Linus)
- Solaris 1993 (Sun)
- FreeBSD-1.0 1993
- NetBSD-1.0 1994
- OpenBSD-2.0 1996
- Max OS X 10.1 2001 (Apple)

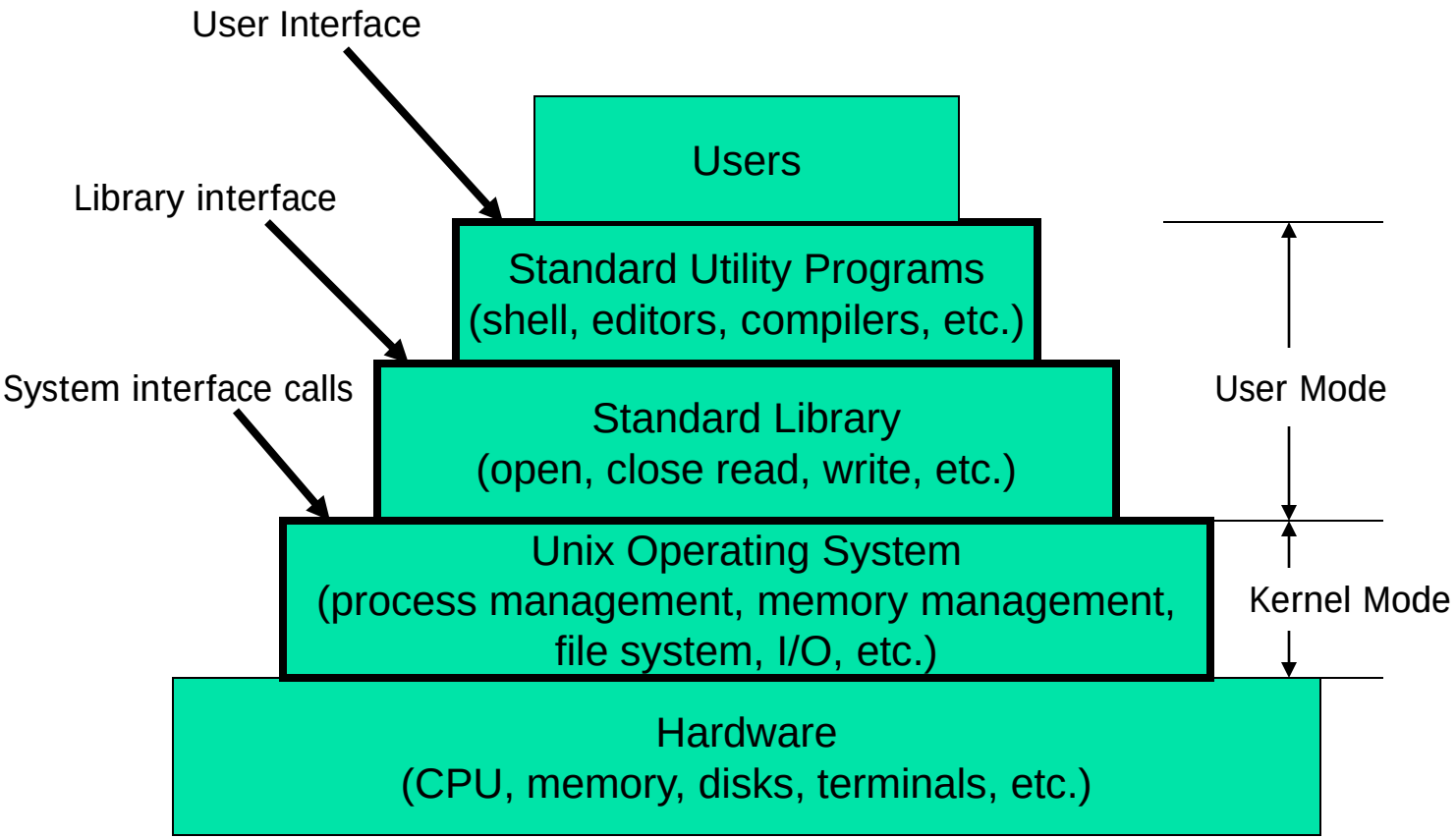
Unix versions

- Two main threads of development:
  - Berkeley software distribution (BSD) (http://www.bsd.org)
  - Unix System Laboratories (http://www.unix.org)
- BSD
  - SunOS 4, Ultrix, BSDI, OS X, NetBSD, FreeBSD, OpenBSD, Linux (GNU)
- SYS V
  - System V (AT&T -> Novell -> SCO), Solaris (SunOS 5), HP-UX (Hewlett-Packard), AIX

Brief History of Linux

- Andrew Tanenbaum, a Dutch professor developed MINIX to teach the inner workings of operating systems to his students
- In 1991 at the University of Helsinki, Linus Torvalds, inspired by Richard Stallman's GNU free software project and the knowledge presented in Tanenbaum's operating system, created Linux, an open-source, Unix-based operating system
- Over the last decade, the effort of thousands of open-source developers has resulted in the establishment of Linux as a stable, functional operating system

Layers in a Unix-based System



Unix Structure

- The *kernel* is the core of the Unix operating system, controlling the system hardware and performing various low-level functions. Other parts of a Unix system (including user programs) call on the kernel to perform services for them.
- The *shell* accepts user commands and is responsible for seeing that they are carried out.

Unix Structure (cont.)

- Over four hundred *utility* programs or *tools* are supplied with the Unix system. These utilities (or commands) support a variety of tasks such as copying files, editing text, performing calculations, and developing software.
- This course will introduce a limited number of these utilities or tools, focusing on those that aid in software development.

Getting started

pehlivan@ktuce: /home/pehlivan

```
Using username "pehlivan".
pehlivan@10.0.0.225's password:
Last login: Fri Mar  7 14:58:15 2008 from 192.168.19.212
Sun Microsystems Inc.   SunOS 5.9       Generic May 2002
You have mail.
pehlivan@ktuce:~$
```

The Unix Account

- Logging in to a Unix machine requires an account on that system.
- A user account is associated with login and password.
- “login” is your user name (usually some variant of your real name)
- Your password will not echo as you type
- Remember good password practices

Logging into a UNIX system

init (Process ID 1 created by the kernel at bootstrap)

spawns getty for every terminal device

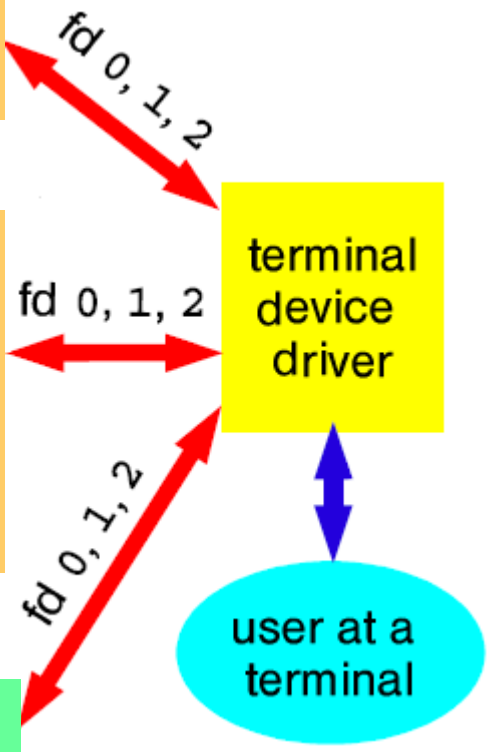
getty opens terminal device, sets file descriptors 0, 1, 2 to it, waits for a user name, usually sets some environment variable (TERM)

invokes login when user name entered

login reads password entry (getpwnam()), asks for user's password (getpass()) and validates it; changes ownership of our terminal device, changes to our UID and changes to our home directory. Sets additional environment variables (HOME, SHEL, USER, LOGNAME, PATH)

invokes our login shell

Login shell (bash)



How Logins were processed

- `init` (using /etc/ttys or /etc/inittab) -> forks and execs `getty` programs on each terminal
- `getty` gets user name -> execs `login`
- `login` verifies password -> execs login shell
- User uses login shell

```
init - spawns -> getty -- starts -> login - starts --> shell
  \                                                      /
   \---------------<<  returns control to <<------------/
```

- Login methods: Using an X display
  - User logins in via getty/login, then runs startx
  - xdm -- reads username & passwd, starts X as that user
    - Somewhat like a startx without the login shell
    - Can start a "terminal" (or shell) window

Secure Login Tools

- Terminal connection
  - PuTTy (on Windows)
  - MindTerm (Java applet)
- Desktop connection
  - X-Win32
    - CDE, KDE, GNOME
  - WeirdX (Java application)
- File transfer
  - WinSCP3
  - SmartFTP

What is a Shell?

- Just a Unix program executed when you log in
- A command interpreter
  - provides the basic user interface to UNIX utilities
- A programming language
  - program consisting of shell commands is called a **shell script**
  - you can put commands in a file and execute it:
    - First, make the file executable (**chmod u+x** ***script-file***)
    - Lines starting with **#** are comments
    - Make use of **interpreter files** (kernel feature!): the first line of your script file must begin with a line:

    #!***pathname optional-arguments***

    where ***pathname*** is an absolute pathname (typically **/bin/sh**, or **/bin/bash**) of the interpreter.

The Shell Prompt

- After logging in, some information about the system will be displayed, followed by a shell prompt, where commands may be entered
  - $
  - %
  - #
  - username@hostname>
  - hostname %

The Shell

- The shell is the program you use to send commands to the Unix system
- Some commands are a single word
  - who
  - date
  - ls
- Others use additional information
  - cat textfile
  - ls -l

Command Syntax

- Commands must be entered exactly. If you make a mistake before entering, delete/backspace to fix it. Be careful!
  - **command** *options argument(s)*
- *Options* modify a command's execution
- *Arguments* indicate upon what a command should act (often filenames)

Example Commands: ls (list)

- ls –l
- ls –a
- ls –la
- ls –a; ls –l
- ls -F
- ls –al textfile1
- ls –al textfile1 textfile2
- ls –al directory

Command Execution

- The current shell (bash)
  - executes built-in commands (echo, kill, pwd, ...) or shell scripts invoked by the . (dot) command: . *shell-script*
  - calls fork() to create a new shell process
    - sub-shell (bash)
- The sub-shell
  - executes a shell script or
  - calls exec() to execute a command or program
  - terminates after script or command execution
- During command execution,
  - the parent either waits, or continues if command is executed in the background

No Shell Prompt

- If you don't get a prompt
  - A program is probably running
  - If you see a special program prompt, try to quit the program (quit, bye, exit)
- If you see nothing, you can
  - Stop the program with CTRL-Z (program will wait until started again)
  - Interrupt the program with CTRL-C (program will usually die)

Logging Out

- **Always** log out when you are done
- Use the exit command to log out of a shell (sometimes logout or CTRL-D)
- Note: if you are running in a windowing environment, logging out of the shell only ends that shell. You must also log out of the windowing, typically selecting an option from a menu.