System Programming

Updated : 1-sept-2019

Overview

System Programming Class CSCM603127

Homework 8x	35 %
Mid-Test	30 %
Final Project	35 %
Bonus	5 %
	105 %

Course Plan

Schedule: Relate to BRP

Mid Term: Week 8

Final Term: Week 16

Reference & Books

1. The Linux Programming Interface , Michael Kerrisk
2. Getting Started with Raspberry Pi, Matt Richardson and Shawn Wallace
3. Linux Manuals
4. Signals Introduction
5. GNU Coding Standards
6. Kernel Compilations
7. Linux Device Driver, 2nd edition, Alessandro Rubini, Jonathan Corbet
8. The Linux Kernel Module Programming Guide, Chapter 2 & 5
9. Raspberry Pi Kernel Compilation (http://elinux.org/RPi_Kernel_Compilation)

System Programming / CSCM603127

Lecturers:

- Ari Wibisono (Reg A & C)
- Hafiyyan Sayyid Fadhlillah (Reg B)
- Rahmat Mustafa Samik Ibrahim (KI)

Overview

Readings for today

- Text Book [1] Chapter 1 and 2

Prologue

- A Brief History - Systems programming and Unix (Chap 1)

Epilogue

- fundamental Unix/systems programming (Chap 2)
- Kernel, Shell, Users/groups, Directory Hierarchy
- File I/O model
- processes

What is Systems Programming?

- Systems Software
- John J. Donovon Diagram

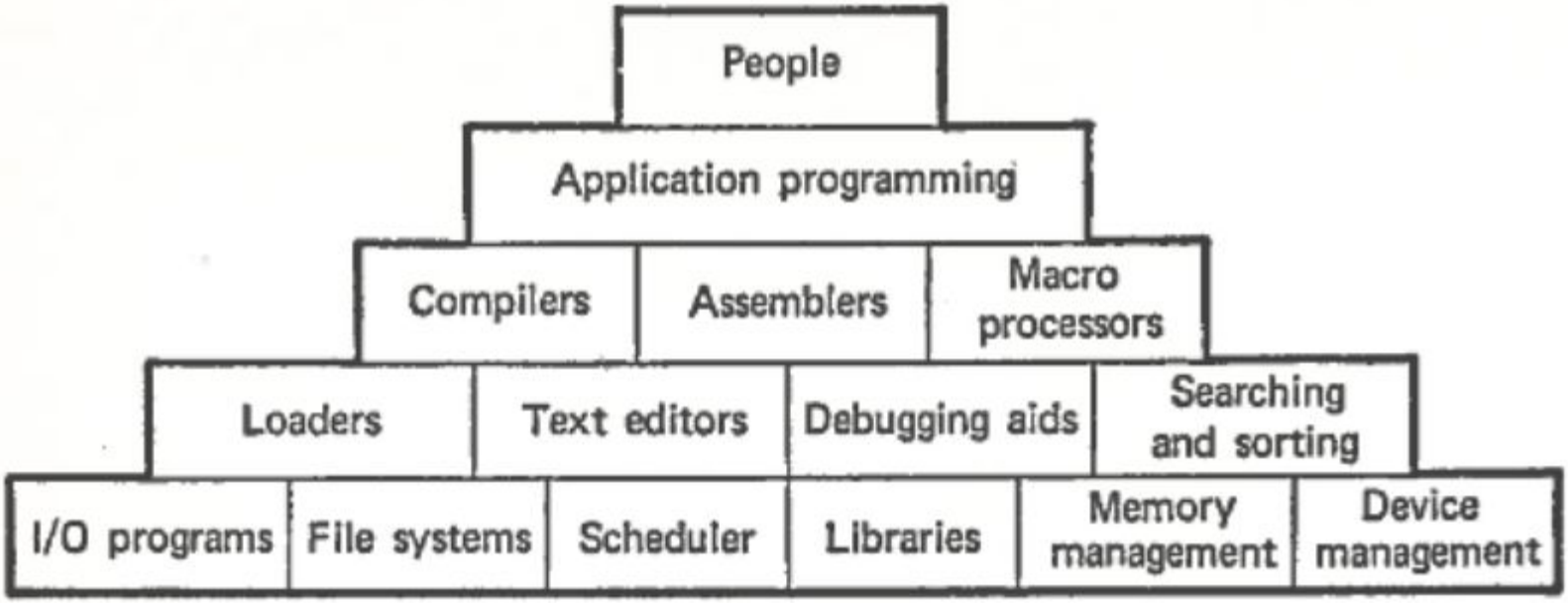


FIGURE 1.1 Foundations of systems programming

Topics in Systems Programming

- The nature of data and computation
- What a computer can do — and can do *efficiently*
- The organization and components of a computer system
- How data is encoded and decoded
- The C Programming Language (and others like it)
- Assembly and Machine Languages
- Basic operating system functionality (file, process, and memory management)
- Interfacing with the O.S.
- Compiling, assembling, linking, loading
- Exploiting systems, Defending systems from attack
- Advanced operating system functionality (virtual memory, interrupts, bootstrapping)
- Device Drivers
- Concurrency basics: threads and events
- Networking basics

etc

Systems Programming as Application Programming ?

- Application programming aims to produce software which **provides services to the user** (e.g. word processor)

- Sytems programming aims to produce software which **provides services to the computer hardware & other software** (e.g. disk defragmenter). **It requires a greater degree of hardware awareness.**

Systems Programming Languages ?

- **First version – assembly language,**
  - macro processors,
  - linkers …
- **Unix/C version circa 1987 ➔ 2002 version**
  - Make
  - M4/cpp macro processors
  - Shell1: variables, regexps, filename completion, history
  - Shell2: I/O redirection, signals, …
  - webserver
- **2008 Summer version – done in Ruby, Python, etc**

Which Language?

- Python: 31%
- C: 20%
- C++: 14%
- Other: 9%
- Java: 8%
- Perl: 7%
- JavaScript: 4%
- PHP: 3%
- Ruby: 3%

Best Scripting Language

- Python 37.1%
- Bash/Shell scripts 27%
- Perl 11.8%
- PHP 8.4%
- JavaScript 6.7%
- Ruby 4.9%
- Other 2.1%
- Lua 2%

Readers' Choice Awards 2014

Scripting vs Programming ?

What is the differences between Scripting vs Programming Language ?

Anyone ?

More Linux Journal Best of 2012

- **Best IDE** –Eclipse 27%, vim 26%, Netbeans 5%
- **BEST REVISION CONTROL SYSTEM -** Git 63%,Subversion 18.6%,
- **BEST OPEN-SOURCE CONFIGURATION MANAGEMENT TOOL -** Puppet 42%
- **BEST PLATFORM FOR DEVELOPING RICH INTERNET APPS -** HTML5 87%
- **BEST PACKAGE MANAGEMENT APPLICATION -** apt 34%
- **BEST CONTENT MANAGEMENT SYSTEM -** WordPress 35%, Drupal 28%, Joomla 15%
- **BEST LINUX BOOK**
  - Tie between Linux in a Nutshell and Debian Administrator's Handbook
  - Second place goes to The Linux Programming Interface: A Linux and UNIX System Programming Handbook, by Michael Kerrisk.
- **BEST OTHER LINUX-BASED GADGET** – Raspberry Pi

Linux Journal December 2012

Systems Programming Course (2014)

- **The Usual Suspects: programming assignments**
  - ls, ar, cpp/m4, shell1, shell2, SSH
- **New possibilities**
  - Windows – hmm ?
    It's hard to start learning system programming on windows
- **Environments: Linux Debian 7.0 (or 8/9), Posix Threads, ARM processors, VirtualBox, Cloud Computing, Raspberry Pi**
- **Unix Software: Shells. Gcc/g++, make, script/pseudo-terminals**
- **Languages: Shell, C, C++, Python, PHP**
- **Editors/IDEs: Nano/pico, Vi/Vim, Emacs**
- **Software Engineering: ~~Subversion~~ Git, gitlab.cs.ui.ac.id**

A Brief History of Time (UNIX and C)

- **1969 – First Unix Ken Thompson at AT&T Bell Labs**
  - Unix was a “pun” on MULTICS
  - Ideas from Multics:
    - Tree structured file system
    - Program for interpreting commands (shell)
    - Files – unstructured streams of bytes
- **1970 Unix rewritten in assembly for DEC PDP-11 (previously for PDP-7)**
- **C – Dennis Ritchie – a systems programming language**
  - BCPL → B (Thompson) → C
- **1973 Kernel rewritten in C – eases porting to other machines**
- **1984 Turing Award Lecture – C compiler learned backdoor**

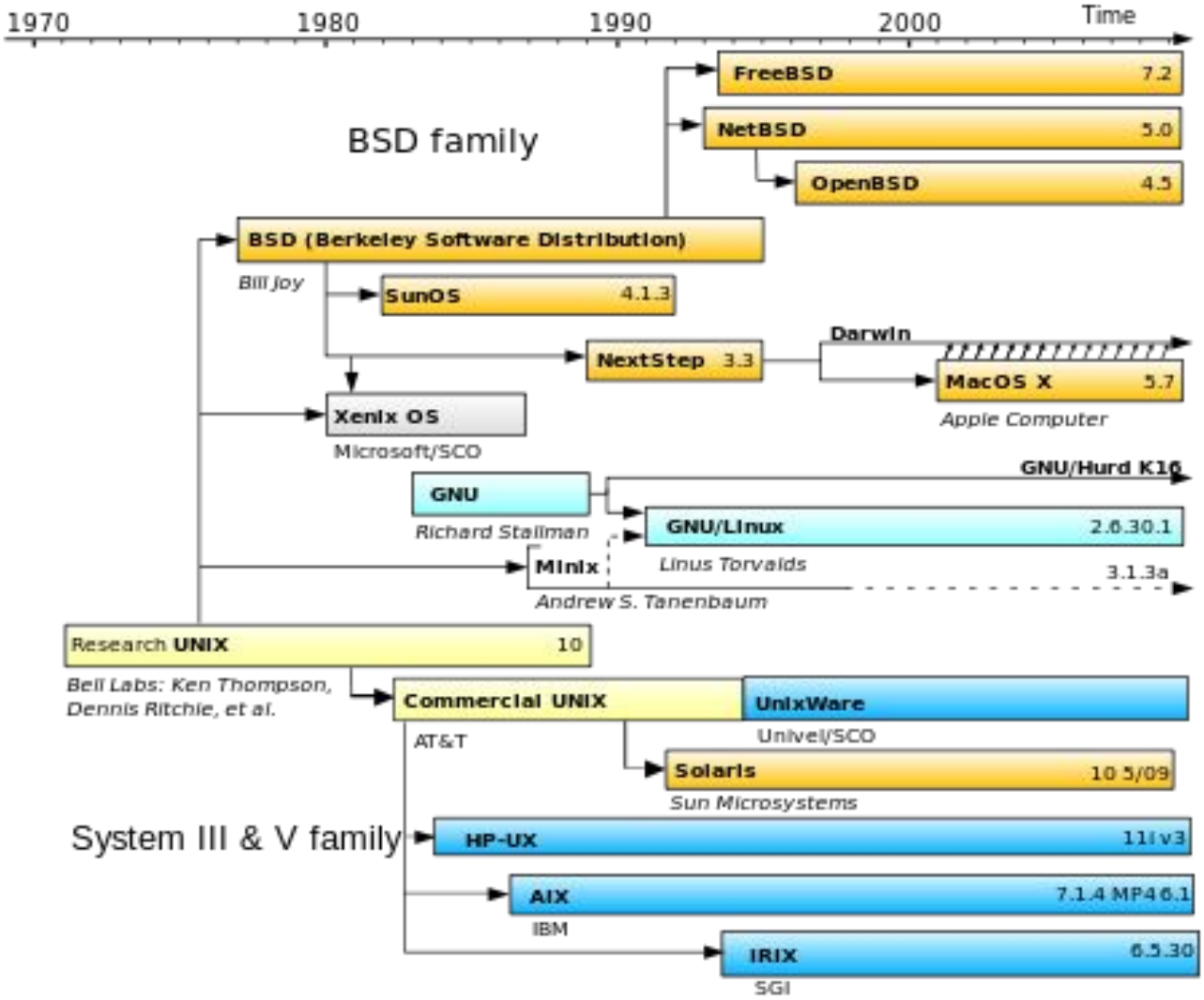
Berkeley Software Division (BSD)

- **(1975) Thompson visiting Prof. at UC-Berkeley**
- **A student Bill Joy added new features**
  - Vi editor
  - C shell
  - First paging virtual memory management (Unix) BSD 4.2
  - Sendmail, Pascal compiler
  - Later co-founded Sun Microsystems
- **BSD 4.2 (1983) - full TCP/IP & sockets API**

Unix after Unix 7th Edition (1979)

- BSD continued at UC-Berkeley
- Bell Labs System III ➔ Systems V
- POSIX standard (1988)
- Other Software
  - X windows
  - Free Software Foundation
  - GNU Public License
- Minix - (1988)Unix like; MINI-uniX; for education; A. Tannenbaum

UNIX/BSD Family



http://en.wikipedia.org/wiki/Linux

GNU

- **Recursive acronym "GNU's not UNIX"**
  - http://www.gnu.org/ not http://www.gnu.com/
- **Richard Stallman (1983) Goal a free Unix**
  - Known for Free Software movement, GNU, Emacs, gcc
  - Never really released GNU operating system
- **Free Software Foundation**
  - http://www.fsf.org/

http://en.wikipedia.org/wiki/GNU

Linux

- (1991) Linus Torvalds
- For Intel x86 systems
- Moved to big Iron (Mainframe Machine)
- more than 90% of today's 500 fastest supercomputers run some variant of Linux
- Network routers
- Embedded systems
- Android

http://en.wikipedia.org/wiki/Linux

The kernel

- What is the kernel?
- What distinguishes it from the Operating System?

Kernel Tasks

- Process scheduling
- Memory management
- File System
- Access to devices
  - /dev – device drivers
- Networking
- System call API

Kernel Mode and user mode

- When an application makes a system call

- Time command

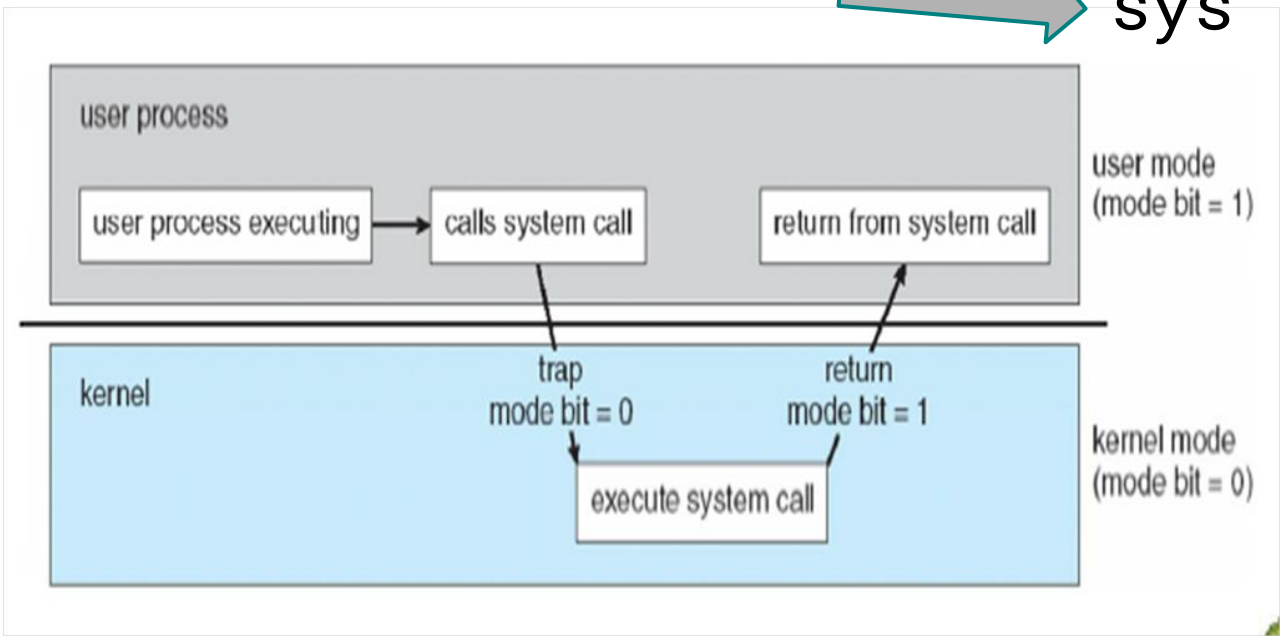
Example:

```
root>time [command]
real    0m33.067s
user    0m13.545s
sys     0m5.596s
```

Time at User Level + Kernel Level + I/O wait + etc

Time consumed at User Level

Time consumed at Kernel Level



Directory Hierarchy - fig 2.1

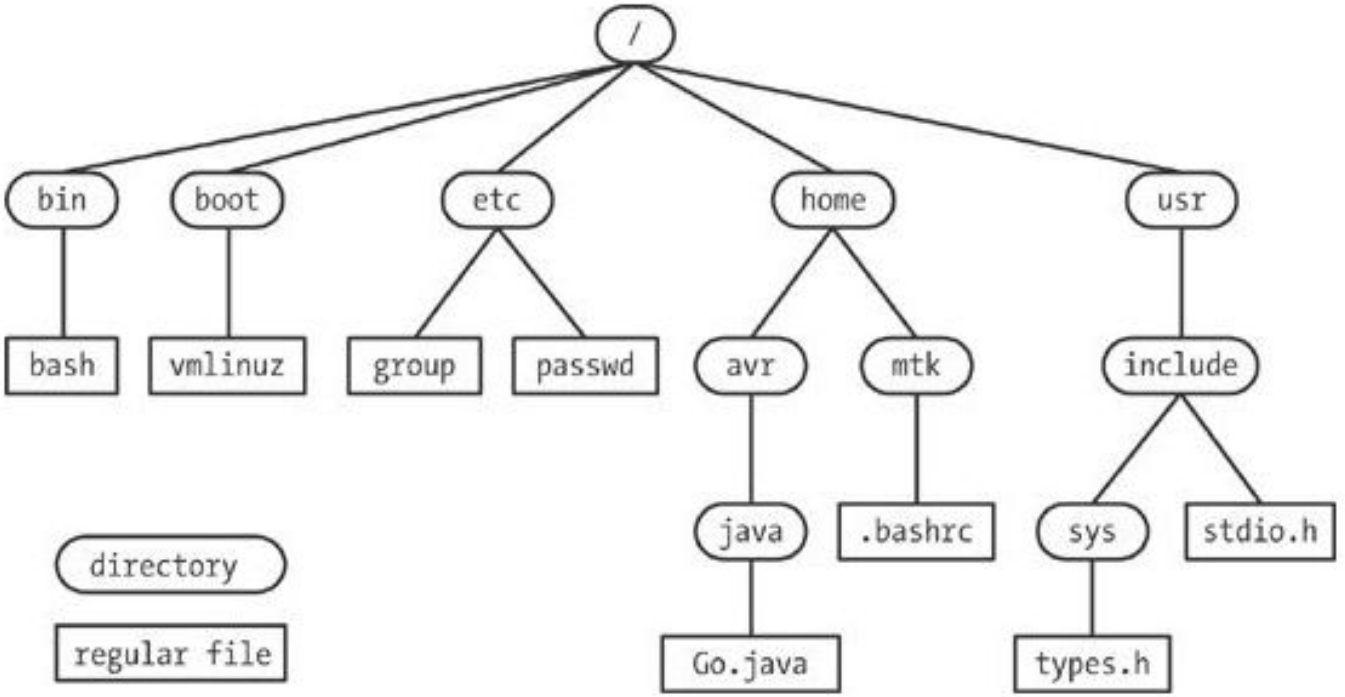


Figure 2-1. Subset of the Linux single directory hierarchy

File Types

- **Filenames**
- **Regular files**
- **Special Files**
  - Directory
    - . (this directory),
    - ..(my parent)
  - Symbolic link

```
ardhi> ln -s README.md something.md
ardhi> ls -l something.md
lrwxrwxrwx 1 ardhi ardhi 9 Aug  8 11:15 something.md ->
README.md
```

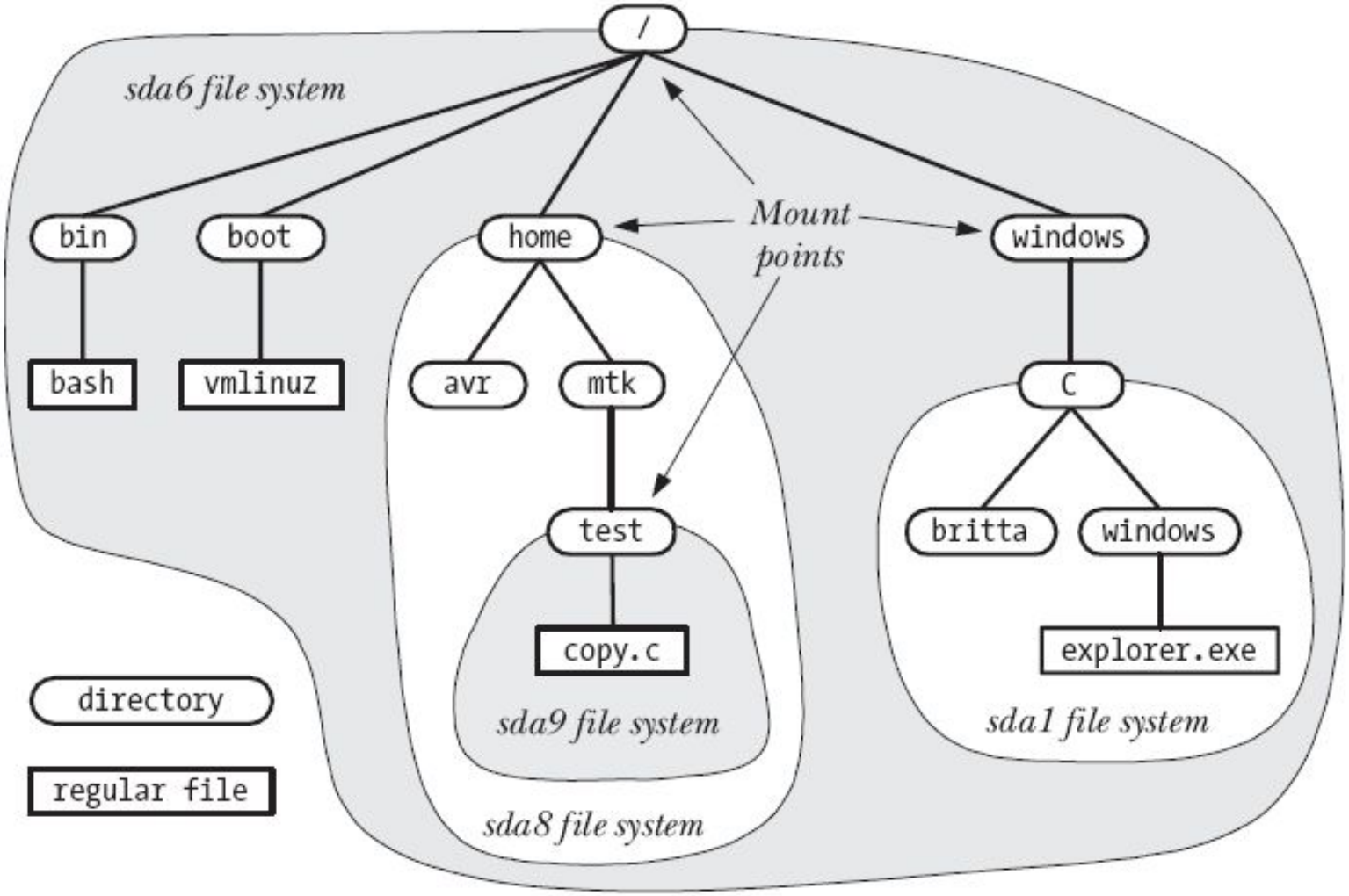
Pathnames

- **Unix basic commands with hierarchy**
  - Current working directory – each running process has a “current working directory” assigned to it
  - cd dir - (change directory to dir)
  - pwd
- **Relative paths**
  - Example: When we are at /home/adin/ which we need to access file.txt, then the relative path of that file is ./file.txt
- **Absolute Paths - start with /**
  - / = root of the directory hierarchy
  - If we are using the sample before, the absolute path would be /home/adin/file.txt

The / class subtree

- **/ (can be read as “slash”) class subtree**
  - On Linux, as on other UNIX systems, all files from all file systems reside under a single directory tree.
  - At the base of this tree is the root directory, / (slash). Other file systems are mounted under the root directory and appear as subtrees within the overall hierarchy. The superuser uses a command of the following form to mount a file system:
    - *$ mount device directory*

The / class subtree



The Shell

- **Command interpreter in original Unix**
  - Read command
  - Perhaps pre-process command
  - Fork/execute
  - Return exit status of command
- **A little history revisited**
  - Bourne Shell (sh)
  - C chell
  - Korn shell
  - Bourne Again shell (bash)
- **The Current and Future**
  - Fish

Users, Groups and Permissions

- **User ID (UID)**
  - Based on /etc/passwd
- **Group (GID)**
  - Based on /etc/groups

- **Permissions**

```
ardhi> ls -l
-rw-rw-r--  1 ardhi ardhi     8 Aug  8 11:15 README.md
lrwxrwxrwx  1 ardhi ardhi     9 Aug  8 11:15 something.md ->
README.md
-rwxrwxr-x  1 ardhi ardhi    13 Aug  8 11:20 ascript.sh
```

- **Super user = root; sudo; su**

Man - the Online Manual

- The sections of the manual

1. Commands, ex: man 1 chmod
2. Library functions, ex: man 2 chmod
3. System Calls, ex: man 3 fopen
4. Special files (usually found in /dev), ex: man 4 urandom
5. File formats and conventions eg /etc/passwd
6. Games
7. Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8. System administration commands (usually only for root)
9. Kernel routines [Non standard]

Man - the command

root> man chmod

CHMOD(1) User Commands
CHMOD(1)

NAME

chmod - change file mode bits

SYNOPSIS …

- man -k chmod
- man -s 2 chmod
- Online
  - http://man.he.net/
  - http://www.tldp.org/guides.html
  - http://man7.org/linux/man-pages/index.html

Homework Assignment (Exercise)

- **Write a C/C++ program to read command line arguments and just print them.**
- **Read “man getopt” (all sections).**
  - What sections of the manual has a getopt section?
  - What types of strings does getopt usually process?

QA

The Dark Ages - JCL

```
//EXAMPLE    JOB     DONOVAN,    T168,1,100,0
//STEP1      EXEC    FORTRAN, NOPUNCH
           READ 9100,N
           DO 100 I = 1,N
           I2 = I*I
           I3 = I*I*I
      100  PRINT 9100, I, I2, I3
     9100  FORMAT (3I10)
           END
/*
//STEP2      EXEC LOAD
/*
//STEP3      EXEC OBJECT
          10
/*
```

- Darkages ~ 1972
- JCL (Job Control Language)
- Batch Processing
  - Job card?

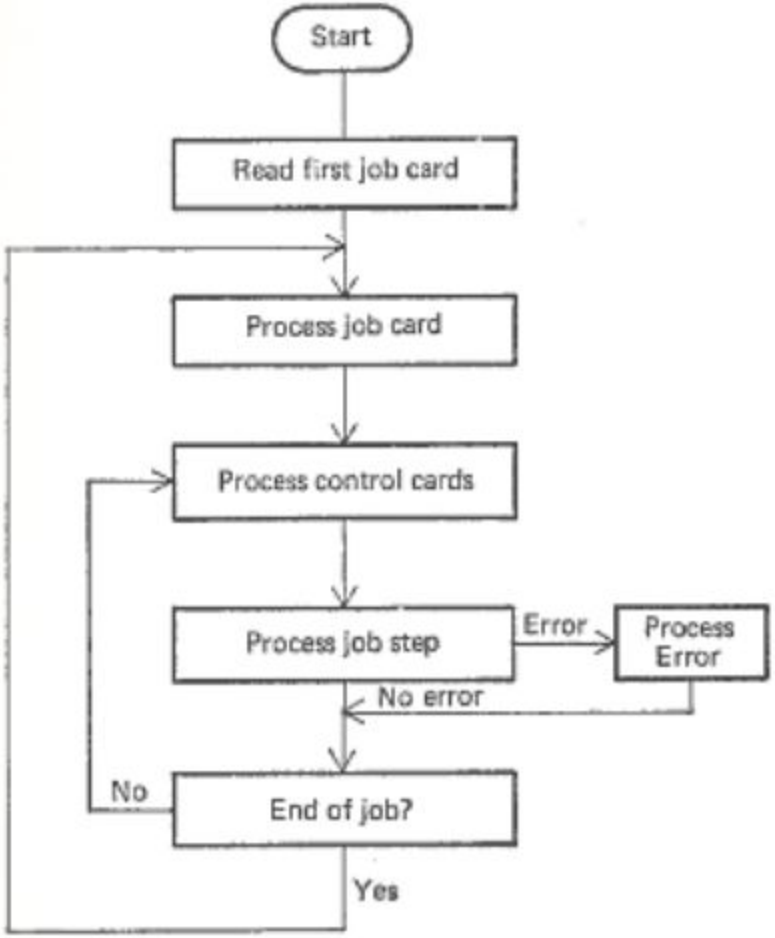


FIGURE 1.4 Main loop of a simple batch monitor system

References

- **Text: The Linux Programming Interface by M. Kerrisk**
  - http://man7.org/tlpi/
- **Other Books**
  - Adv Prog in Unix Environment (APUE) by Richard Stevens (Rago 2nd ed)
    - http://www.kohala.com/start/apue.html
    - http://www.apuebook.com/ 2nd edition site – code etc
  - C by Kernighan and Ritchie (K&R) 2nd edition
    - http://en.wikipedia.org/wiki/The_C_Programming_Language
    - http://clc-wiki.net/wiki/K%26R2_solutions
    - http://cm.bell-labs.com/who/dmr/