MANCHESTER 1824
The University of Manchester

COMP25111: Operating Systems

Lecture 19: Linux Case Study

Oscar Palomar

School of Computer Science, University of Manchester

Autumn 2016

Overview & Learning Outcomes

1. Background
2. Shell
3. Components: layers & managers
   - Scheduling
   - Memory
   - Files
   - Input/Output

UNIX History & Motivation

(weak pun on “MULTICS”)
Initially (1969) single-user, soon (1973) multi-user timesharing system
Written in C (developed at Bell Labs to support Unix)
Fundamental Architectural Design finished in 1978
POSIX standards 1988, 1992,3,5, 2001,4,8

by programmers, for programmers
small, modular, clean design
UI consistency, brevity
source distribution

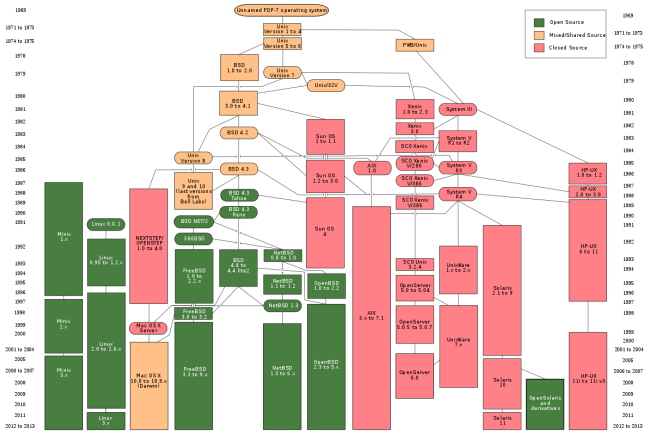
Unifying Themes

Everything is a file

Composition & re-usability:
– shell, I/O redirection, pipes, filters

Simplicity & minimality:
– 1 reusable best tool for each purpose

Development (wikimedia: Unix_history-simple.png)



Shell

user-level process, executes programs
(command interpreter, CLI)

– reads next user command
– searches path for program
– forks child process & execs program
– waits for termination of child

fork & exec (MOS figs 10-4, 10-7)

```
pid = fork();
if (pid < 0)       { /* error */ }
else if (pid > 0) { /* parent*/ }
else /* pid==0 */ { /* child */ }

while (1) {
  type_prompt();
  read_command(&command, &params);
  pid = fork();
  if (pid < 0)       { /* error */ }
  else if (pid > 0) waitpid(-1, &status, 0);
  else /* pid==0 */ execve(command, params, 0);
}
```


I/O redirection

processes start with 3 open files: standard input, output, error

Can redirect to/from files:
e.g. `compile <input >output 2>errors`
or `compile <input &>combined-output`

or another process in a pipeline:
e.g. `ls -la | grep Nov | grep 23`
`pipe()` system call

Interfaces (MOS fig. 10-1)

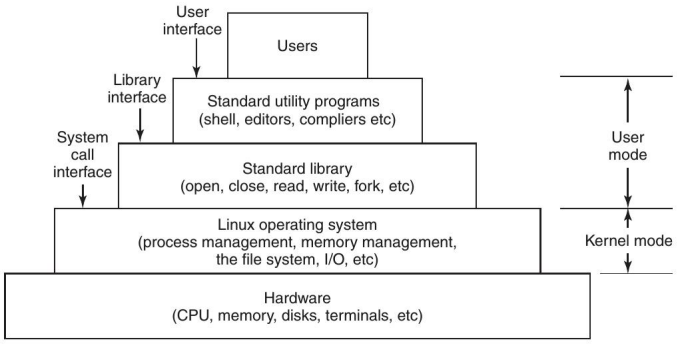


Figure 10-1. The layers in a Linux system.

Architecture – Overview

User level (non-privileged):
user processes = application, library etc.

Programmer Interface:
– System programs (e.g., mkdir, rm, cp, ...)
– System calls (file manipulation; process control; information)

Kernel level (privileged):
managers (file, process, memory, ...) & device drivers

Kernel layers (MOS fig. 10-3)

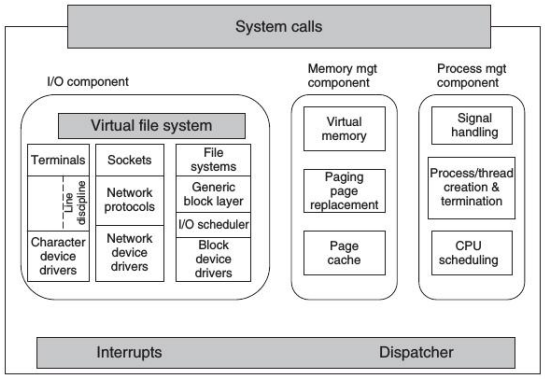


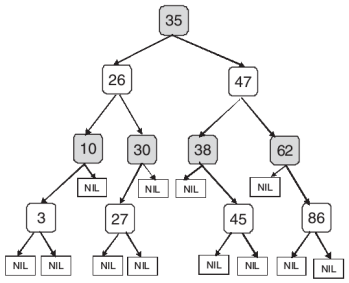
Figure 10-3. Structure of the Linux kernel

Process

"process descriptor" (see handout 5 – PCB)
– task_struct
– unique PID
– address space
– UFIDs (handout 16)
– signal handling vector
– UID & GIDs (User/Group ID)
– scheduling priorities
– thread(s)

Properties inherited from parent
– fork, execve, clone...

Scheduling



(b) Per CPU red-black tree in the CFS scheduler

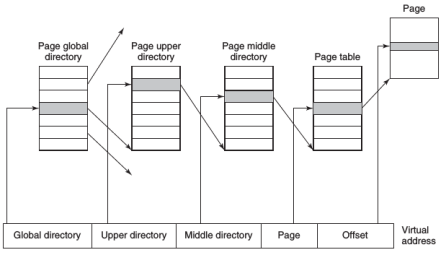
MOS4 10-10

Priorities
– "Real-time" high priority (0-99), FIFO or Round-Robin
– Timesharing: low priority (100-139)
– ±20 (nice)
Completely Fair Scheduler – translates priorities into "speed"
– O(1) in older versions

Delayed jobs:
`at 0630 myjob` for one-off deferral
`cron` for regular jobs (hourly, daily, weekly, monthly, etc)

Virtual Memory

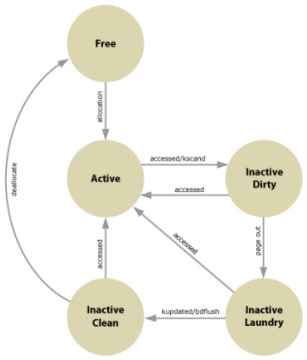
Paged (e.g. 4kB – now also huge pages)
"segments": Text (code), Data (& heap), Stack
x86: 1GB Kernel, 3GB user; x86-64: 128TB each
can be shared; copy-on-write
memory-mapped files
Page Tables (MOS fig 10-17)
– 2-level: Intel x86
– 3-level: DEC Alpha (43-bit virtual addresses); x86-PAE
– 4-level: Intel x86-64 (48-bit virtual addresses)



Paging

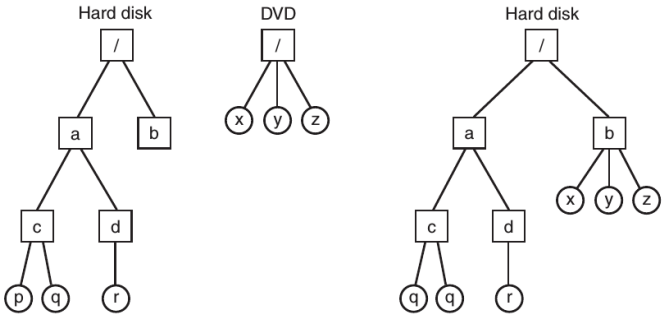
Demand-driven
Pages pre-cleaned

LRU approximation with second-chance
Reference bit (set when page accessed)
Round-robin check pages (& clear ref)



(www.redhat.com/magazine/001nov04/features/vm)

`mount` (MOS4 fig 10-25)



inode

(see handout 16)

Owner, last time info, permissions, size, links to the file

15 content pointers to disk blocks:
– 12 pointers to direct blocks
– 1 single-indirect; 1 double-indirect; 1 triple-indirect

ext2: groups inodes, which point to nearby blocks

ext3: + journaling
ext4: journaling, +extent

Protection Model

Process Concepts: UID GID

File Attributes: RWXRWXRWX
Owner, Group, Other – Read, Write, Execute

Primarily files, directories

Same mechanisms used by devices, network connections, ...

Security:
user particulars in /etc/passwd
Holes: read about the Morris Internet Worm (1988)

I/O and Device Drivers

Drivers are privileged code (not user-supplied)

user-access to devices via special files:
e.g. `/dev/fd` `/dev/tty`

Can be character or block devices

Summary of key points

1. Background
2. Shell
3. Components: layers & managers
   - Scheduling
   - Memory
   - Files
   - Input/Output

Your Questions

Exam Questions

In Unix, what is the use of the shell variable PATH? (2 marks)

Briefly explain how a shell implements a pipe between two commands (2 marks)

Briefly explain how Unix implements input redirection in a shell command. (2 marks)

Briefly explain that a Unix shell does to execute the following command /bin/who $>$ myfile (2 marks)

Glossary

I/O redirection
shell
pipe
filter
fork
exec (execve)
waitpid
2-level (high & low) scheduling
character & block devices

Reading

newer OSC/J: Ch. 21

older OSC/J: Ch. 20

MOS: Ch. 10

https://www.kernel.org/doc/

David A Rusling, The Linux Kernel
http://tldp.org/LDP/tlk/tlk-toc.html
(Linux 2.0.33, 1999)