ECE595: Introduction to Operating Systems

Y. Charlie Hu

1/11/2017

Roadmap

- Course administration
- Operating System components

About this course…

Principles of OS design
- Some theory
- Some rational
- Lots of practice

Goals
- Understand OS design decisions
- Basis for future learning

To achieve the goals:
- Learn concepts in class
- Get hands “dirty” in labs

Expect (some) pain

Somewhat fast pace

Some heavy programming projects

Some difficult (abstract) concepts

Mechanics – Course Staff

Y. Charlie Hu, ychu@ecn, MSEE 232
Office hours: Wedn 1:30-3:30pm and by appt.

No TA

A Helper with programming projects

5

Mechanics – General Info

- Course home page: engineering.purdue.edu/~ece595

- Instructor/Staff email: ece595@ecn.purdue.edu

6

Mechanics – Q & A

- Questions of general interests → mailing list
- Other questions → ece595@ecn

- Announcements → mailing list

7

Mechanics – Textbook

Operating System Concepts
Silberschatz, Galvin, and Gagne, 8th (7th, 6th) edition

Explains concepts very well

Some papers – will be available from class web page

8

Mechanics – Lecture Notes

If available, will be provided on the web

Not necessary self-contained, complete, or coherent

Not a substitute for attending classes

9

Mechanics - Projects

- 4 Programming projects
  - **Use DLXOS, simulated on Linux machines**
  - Build key components of a mini-OS (DLXOS)
- 1st not graded (DLXOS tutorial)
- 3~4 weeks each (excl. spring break)
  - Explained in designated Friday lectures
  - due: *Sunday midnights*
  - no extensions
- Work in pairs (optional to work on your own)
  - Be decent to each other!

10

Mechanics - Exams

- Midterm: close-book,
  - Mar 9, Thursday 7-9pm (before Spring break)
- Final: close-book
- Short answers, some proof, some design (derivation), at most 1 programming problem

11

Mechanics – Grading

- Programming projects (50%: 15%-15%-20%)
- Midterm exam (25%)
- Final exam (25%)
- Bonus pop quizzes (up to 10%)

12

Academic Integrity

- Programming assignments
  - Ask instructor/helper for clarification
  - Each team must write their own solution
  - No discussion of or sharing of specific code or written answers is allowed
  - Any sources used outside of textbook/handouts/ lectures must be explicitly acknowledged

ABET outcomes

- ABET, incorporated as the Accreditation Board for Engineering and Technology, is a non-governmental organization that accredits post-secondary education programs in "applied science, computing, engineering, and engineering technology".
- By March 9: document on mapping of questions in labs/exams to ABET outcomes
- By March 9: Remediation homework

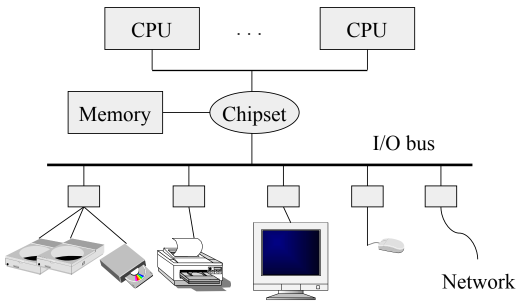
Homework

- Find a project partner and email 595@ecn by Jan 22

Roadmap

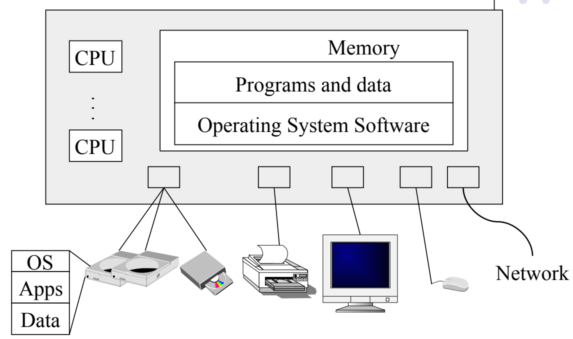
- Course administration
- Operating System components

A Typical Computer from a Hardware Point of View



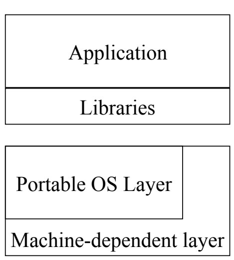
17

A Typical Computer System: adding software



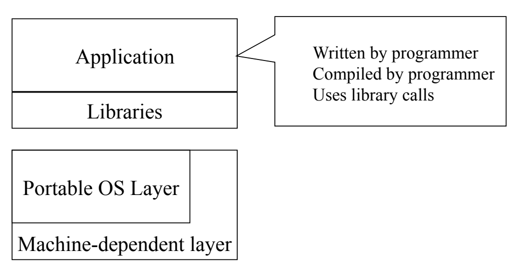
18

Typical OS Structure



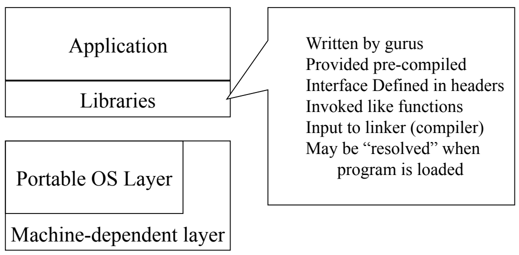
19

Typical Unix OS Structure



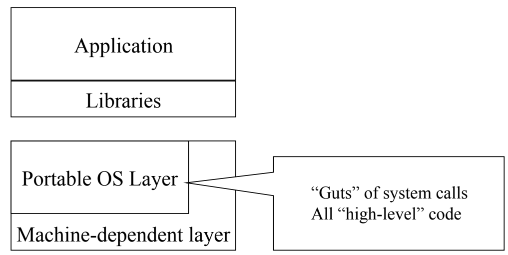
20

Typical Unix OS Structure



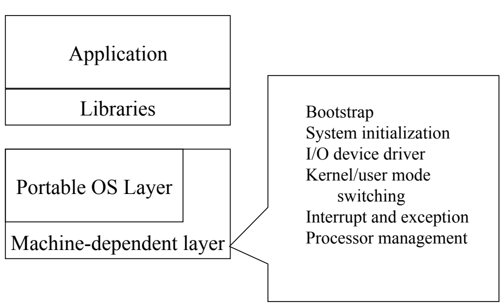
21

Typical Unix OS Structure



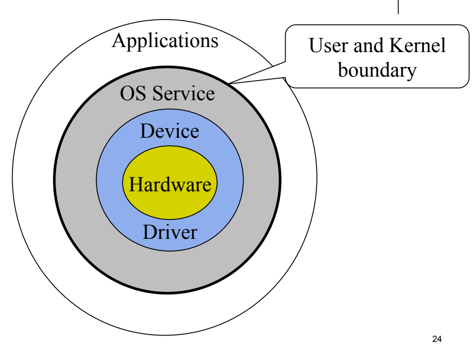
22

Typical Unix OS Structure



23

Another Look: UNIX “Onion”



24

Operating system components

- Process management
- Main-memory management
- Secondary-storage management
- File management

- I/O system management
- Networking
- Command-interpreter system

Process management

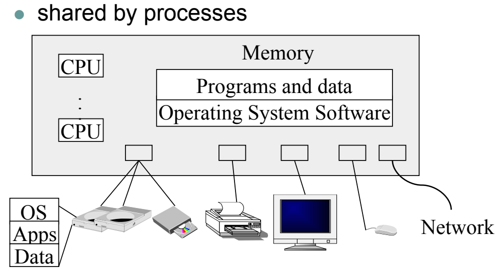
- A *process* is a program in execution
  - Process is *active*, program is *passive*
  - A process needs resources to accomplish its task:
    - CPU time
    - Memory
    - Files
    - I/O devices

Process management

- The OS is responsible for:
  - Process creation and deletion
  - Process suspension and resumption
  - Provision of mechanisms for:
    - Process communication
    - Process synchronization (e.g. parallel computing)
      - E.g.10 processes together adding up 10 billion numbers
      - Lab2
    - Deadlock detection and handling

Main-memory management

- Memory: a *volatile* storage device
  - shared by the CPU and I/O devices
  - shared by processes



Main-memory management

- The OS is responsible for:
  - Allocate and free memory space as needed
    - Keep track of which parts of memory are currently being used and by whom
  - Protection
    - How to do it fast?
  - Support larger address space than physical memory
    - malloc(2G) on a 1G machine?

Secondary-storage management

- Main memory (*primary storage*) is volatile and small → OS needs a *secondary storage* to back up main memory
- Most modern systems use disks as the principle on-line storage medium
- The OS is responsible for:
  - Allocating/freeing disk storage
  - Disk scheduling
    - FIFO can be horrible
    - How does the elevator move?

File management

- *A file* is a logical entity
  - It is a collection of related information defined by its creator.
- The OS is responsible for:
  - File creation and deletion
  - Directory creation and deletion
  - Mapping files onto secondary storage
  - Support of primitives for manipulating files and directories (e.g. ls, ls -l)
  - Sharing and protection

I/O subsystem management

- I/O devices of many flavors
  - Storage devices: drives, tapes
  - Transmission devices: NIC, modems
  - Human-interface devices: screen, keyboard, mouse
- I/O subsystem hides peculiarities of I/O devices
- I/O subsystem consists of:
  - A uniform device-driver interface (wrapper)
  - Drivers for specific hardware devices
  - Buffering / spooling / caching (access locality)

Networking (distributed systems)

- A *distributed system*:
  - A collection of machines that do not share memory or a clock
  - Connected through a network to provide user access to various system resources
  - Examples: Internet, WWW, banking, peer-to-peer, cloud, …
  - Allows computation speed-up, increased data availability, enhanced reliability, low latency, …
- The OS is responsible for
  - Communication between processes
  - Reliability – reliable protocols on unreliable Internet

Command-interpreter system

- Interface between user and OS
- Many commands are given to the OS by control statements which deal with:
  - process creation and management (a.out, kill –9)
  - secondary-storage management (partition disk)
  - file system
    - access (ls, cat)
    - protection (chmod 700 *, umask 077)
  - networking (telnet, ssh)
  - monioring (ps, top, vmstat, netstat)

Command-interpreter system (cont.)

- The program that reads and interprets control statements is called the *command-line interpreter* (or the *shell* in UNIX)
- What is the shell program's structure like?
- Two prevalent approaches to the interface
  - moused-based windows-and-menus (Mac, Windows)
  - command line (UNIX)

Roadmap for ECE595 lectures (labs)

Introduction to OS components
Individual components
- Process management
- Memory management
- File management
- Secondary-storage management

- (Device management)
- (shell)
- (Networking)

Hardware support for OS interspersed (before relevant topics)

Common Themes in our journey

- Functionality, performance, tradeoff if applicable
- Software vs. hardware vs. hybrid solutions

Questions?

- Reading assignment:
  - OSC, Chapters 1-2, by Friday
- Find a lab partner and email ece595@ecn
  - By Jan 22