COS 318: Operating Systems

Overview

Kai Li and Andy Bavier
Computer Science Department
Princeton University

http://www.cs.princeton.edu/courses/archive/fall13/cos318/
Logistics

- **Precepts:**
  - Tue: 7:30pm-8:30pm, 105 CS building

- **Design review:**
  - Mon 9/23: 11am-7:40pm, 010 Friends center

- **Project 1 due:**
  - Sun 9/29 at 11:55pm

- **Reminder:**
  - Subscribe to the cos318 mailing list!

- **To do:**
  - Lab partner? Enrollment?
Who am I?

- A builder: practical, hands-on
- A philosopher?
- Search for beauty 😊
- Operating systems spans the spectrum

Abstract
Simple, powerful ideas

Concrete
Making things work
Today

- Overview of OS structure
- Overview of OS components
Hardware of A Typical Computer

- CPU
- Chipset
- Memory
- I/O bus
- ROM
- Network
Computing machinery

Analytical Engine (~1850) Charles Babbage

ENIAC (~1946) Eckert & Mauchly, UPenn

Johnniac (~1953) von Neumann, IAS
A Typical Computer System

- CPU
- Memory
  - Application
  - Operating System
  - ROM
  - BIOS
- Network
- OS
- Apps
- Data
Hardware Interrupts

- Raised by external events
- Interrupt handler is in the kernel
  - Switch to another process
  - Overlap I/O with CPU
  - …
- Eventually resume the interrupted process
Typical Unix OS Structure

Application

Libraries

Portable OS Layer

Machine-dependent layer

User level

Kernel level
Typical Unix OS Structure

Application

Libraries

Portable OS Layer

Machine-dependent layer

User function calls written by programmers and compiled by programmers.
Typical Unix OS Structure

- Application
- Libraries

- Portable OS Layer
- Machine-dependent layer

- Written by elves
- Objects pre-compiled
- Defined in headers
- Input to linker
- Invoked like functions
- May be “resolved” when program is loaded
Elves

GAAA!!! IT HURTS SO BAD!

THAT ARTIFICIAL DISPLAY OF PAIN WAS A REMINDER THAT SOFTWARE IS NOT CREATED BY MAGIC.

THE ELVES ARE GETTING UPPITY.
Pipeline of Creating An Executable File

- gcc can compile, assemble, and link together
- Compiler (part of gcc) compiles a program into assembly
- Assembler compiles assembly code into relocatable object file
- Linker links object files into an executable
- For more information:
  - Read man page of elf, ld, and nm
  - Read the document of ELF
On Unix, “loader” does the job
- Read an executable file
- Layout the code, data, heap and stack
- Dynamically link to shared libraries
- Prepare for the OS kernel to run the application
- E.g., on Linux, “man ld-linux”
What’s An Application?

- Four segments
  - Code/Text – instructions
  - Data – initialized global variables
  - Stack
  - Heap

- Why?
  - Separate code and data
  - Stack and heap go towards each other
Responsibilities

- **Stack**
  - Layout by compiler
  - Allocate/deallocate by process creation (fork) and termination
  - Names are relative to stack pointer and entirely local

- **Heap**
  - Linker and loader say the starting address
  - Allocate/deallocate by library calls such as malloc() and free()
  - Application program use the library calls to manage

- **Global data/code**
  - Compiler allocate statically
  - Compiler emit names and symbolic references
  - Linker translate references and relocate addresses
  - Loader finally lay them out in memory
Typical Unix OS Structure

- Application
- Libraries
- Portable OS Layer
- Machine-dependent layer

"Guts" of system calls
OS Service Examples

- Examples that are not provided at user level
  - System calls: file open, close, read and write
  - Control the CPU so that users won’t get stuck by running
    - while ( 1 );
  - Protection:
    - Keep user programs from crashing OS
    - Keep user programs from crashing each other

- System calls are typically traps or exceptions
  - System calls are implemented in the kernel
  - When finishing the service, a system returns to the user code
Typical Unix OS Structure

- Application
- Libraries

Portable OS Layer

Machine-dependent layer

- Bootstrap
- System initialization
- Interrupt and exception
- I/O device driver
- Memory management
- Mode switching
- Processor management
Software “Onion” Layers

- Applications
- Libraries
- OS Services
- Device
- Kernel
- Driver

User and Kernel boundary
Today

- Overview of OS structure
- Overview of OS components
Processor Management

- **Goals**
  - Overlap between I/O and computation
  - Time sharing
  - Multiple CPU allocations

- **Issues**
  - Do not waste CPU resources
  - Synchronization and mutual exclusion
  - Fairness and deadlock free
Memory Management

**Goals**
- Support programs to run
- Allocation and management
- Transfers from and to secondary storage

**Issues**
- Efficiency & convenience
- Fairness
- Protection

- Register: 1x
- L1 cache: 2-4x
- L2 cache: ~10x
- L3 cache: ~50x
- DRAM: ~200-500x
- Disks: ~30M x
- Archive storage: >1000M x
I/O Device Management

- **Goals**
  - Interactions between devices and applications
  - Ability to plug in new devices

- **Issues**
  - Efficiency
  - Fairness
  - Protection and sharing

![Diagram of I/O device management with User 1, User n, Library support, Driver, I/O device, and Driver, I/O device.]
File System

◆ Goals:
  ● Manage disk blocks
  ● Map between files and disk blocks

◆ A typical file system
  ● Open a file with authentication
  ● Read/write data in files
  ● Close a file

◆ Issues
  ● Reliability
  ● Safety
  ● Efficiency
  ● Manageability
Window Systems

- Goals
  - Interacting with a user
  - Interfaces to examine and manage apps and the system

- Issues
  - Direct inputs from keyboard and mouse
  - Display output from applications and systems
  - Division of labor
    - All in the kernel (Windows)
    - All at user level
    - Split between user and kernel (Unix)
Bootstrap

- Power up a computer
- Processor reset
  - Set to known state
  - Jump to ROM code (BIOS is in ROM)
- Load in the boot loader from stable storage
- Jump to the boot loader
- Load the rest of the operating system
- Initialize and run
- Question: Can BIOS be on disk?
Ways to Develop An Operating System

- A hardware simulator
- A virtual machine
- A good kernel debugger
  - When OS crashes, always goes to the debugger
  - Debugging over the network
- Smart people

1972

1998
Summary

- Interrupts
- User level vs. kernel level
- OS services
  - Processor
  - Memory
  - I/O devices
  - File system
  - Window system
- Booting the OS