Operating Systems Design

1. Introduction

Paul Krzyzanowski
pxk@cs.rutgers.edu
In the beginning…
“Preparing ENIAC for a series of runs was an incredibly involved process. First, detailed instructions had to be written defining the problem and a procedure for solving it. These instructions were programmed by adjusting switches manually and inserting thousands of cables into as many as forty large plug boards. A team of five operators might work several days on the external wiring and many more days searching for errors and correcting them.”

— Breakthrough to the Computer Age, Harry Wulforst, Charles Scribner’s & Sons Pub., 1982
Late 1940s – 1950s

- Stored program concept: reload a program
- Reusable code ("subroutines")
- IBM SHARE (Society to Help Alleviate Redundant Effort)
- The OS emerges
  - Batch systems
    - Branch to a location in the OS that would cause the next program to get loaded and run
  - Common I/O routines for device access
    - Precursor to device drivers
  - Programmatic transition to reduce overhead of starting new jobs
  - Job control languages to define resource needs
1960s

- **Goal:** improve throughput
  - Use every possible second of CPU time

- **Multiprogramming**
  - Keep several programs in memory at once; switch between them
  - Works because of the speed mismatch between I/O and CPU

- Conversational interaction (human I/O)

- Direct storage access (file systems)

- Transaction processing systems (SABRE)

- The **System Call** (Atlas Computer, Manchester)
1960s

• 1961: DEC PDP-1 – first minicomputer ($125,000+)

• 1964: IBM System/360
  – PCP/360: sequential jobs (batch)
  – MFT: Multiple job system, fixed number of tasks
  – MVT: Multiple jobs, variable number of tasks (direct memory)

• IBM 360 introduced:
  – Direct Address Translation
    (precursor of virtual memory & the Memory Management Unit)
  – Channels: specialized processors for transferring data between main memory and an I/O device
    (precursor of DMA)

• Time sharing: preemption
Late 1960s – 1970s

• 1968-1969:
  – User-friendly interfaces: mouse, windowing
  – Data networking

• 1970s: UNIX
  – Portable operating system
  – Written in a high level language

• 1972: Virtual Machines (VM/370)

• Microprocessors emerge
  – CP/M: dominant OS for 8080 family of machines
    • CCP: command interpreter
    • BDOS: file operations, printing, and console I/O
    • BIOS: character I/O, disk sector read/write
  – 1977: Apple II
1980s-1990s

• 1981: IBM PC
  – Open architecture; Microsoft OS
  – Only proprietary component was the BIOS

• 1982: BIOS was reverse engineered
  – PC clones (Compaq, Columbia, Dell, HP, …)

• Client-server networking
  – Network file systems

• Open Source Operating Systems
  – Linux, FreeBSD, NetBSD, OpenBSD

• Network PC, Thin clients

© 2010 Paul Krzyzanowski
2000s

- PC-based machine virtualization
  - Virtualization support added by Intel & AMD (2006)
  - Virtual machine migration
- Cloud computing, on-demand data centers
- Focus on mobility
  - iOS, Android, BlackBerry OS, Windows Mobile
- Security
  - Hardware authentication, Storage encryption, digital rights management
    - Trusted Platform Module
  - Personal firewalls
  - Address space layout randomization
The Operating System
What is an operating system?

• The first program
• A program that lets you run other programs
• A program that provides controlled access to resources:
  – CPU
  – Memory
  – Display, keyboard, mouse
  – Persistent storage
  – Network
The Operating System

- App
- App
- App
- App
- App

OS

- Ethernet
- Wi-Fi
- Disk
- USB
- etc.
Big Ideas In Operating Systems
The Big Ideas

- Interrupts
- Reusable code (→ I/O libraries → drivers)
- Subroutines
- Indirection
- Supervisor mode execution
- I/O processors (IBM’s channels, DMA, SCSI)
- I/O redirection (abstracting I/O)
- File systems
- Cache
- Virtual memory & virtual addresses
- Time sharing & preemption
- OS portability
- Multithreading
- Intelligent and discoverable I/O (e.g., PCI bus)
- Virtual machines (hypervisors)
Mechanisms & Policies
OS Mechanisms & Policies

• Mechanisms:
  – Presentation of a software abstraction:
    • Memory, data blocks, network access, processes

• Policies:
  – Procedures that define the behavior of the mechanism
    • Allocation of memory regions, replacement policy of data blocks
  – Permissions

• Keep mechanisms, policies, and permissions separate
Processes

• Mechanism:
  – Create, terminate, suspend, switch, communicate

• Policy
  – Who is allowed to create and destroy processes?
  – What is the limit?
  – What processes can communicate?
  – Who gets priority?
Threads

• Mechanism:
  – Create, terminate, suspend, switch, synchronize

• Policy
  – Who is allowed to create and destroy threads?
  – What is the limit?
  – How do you assign threads to processors?
  – How do you schedule the CPU among threads of the same process?
Virtual Memory

• Mechanism:
  – Logical to physical address mapping

• Policy
  – How do you allocate physical memory among processes and among users?
  – How do you share physical memory among processes?
  – Whose memory do you purge when you’re running low?
File Systems

• Mechanism:
  – Create, delete, read, write, share files
  – Manage a cache; memory map files

• Policy
  – What protection mechanisms do you enforce?
  – What disk blocks do you allocate?
  – How do you manage cached blocks of data (Per file? Per user? Per process?)
Messages

• Mechanism:
  – Send, receive, retransmit, buffer bytes

• Policy
  – Congestion control, dropping packets, routing, prioritization, multiplexing
Character Devices

• Mechanism:
  – Read, write, change device options

• Policy
  – Who is allowed to access the device?
  – Is sharing permitted?
  – How do you schedule device access?
The End