COP 4600 (Sec 1): Operating Systems

Lecture 2

Instructor: Joohan Lee
jlee@cs.ucf.edu

School of Computer Science
University of Central Florida

Course Overview

• Introduction
  – What is an operating system and its structure
• Hardware/Software Concept
  – Hardware components
  – Software overview
• Process and Threads
  – Process/Thread concepts and management
• Asynchronous Concurrent Execution
  – Mutual exclusion and semaphores
• Concurrent Programming
  – Condition variables and monitors
• Deadlock
  – Deadlock problems and solutions

Chapter 1 – Introduction to Operating Systems

Outline
1.1 Introduction
1.2 What is an Operating System?
1.3 Early History: The 1940s and 1950s
1.4 The 1960s
1.5 The 1970s
1.6 The 1980s
1.7 History of the Internet and World Wide Web
1.8 The 1990s
1.9 2000 and Beyond
1.10 Application Bases
1.11 Operating System Environments
1.12 Operating System Components and Goals
  1.12.1 Core Operating System Components
  1.12.2 Operating System Goals

Outline (continued)
1.13 Operating System Architectures
  1.13.1 Monolithic Architecture
  1.13.2 Layered Architecture
  1.13.3 Microkernel Architecture
  1.13.4 Networked and Distributed Operating Systems

Current Status

• Unprecedented growth of computing during the past several decades.
  – Consider the Moore’s Rule
• Faster computers
  – Desktop workstations execute billions of instructions per second (BIPS)
  – Supercomputers can execute over a trillion instructions per second
• Computers are now employed in almost every aspect of life.
  – Computers, PDA, MP3 player, cell phone, automobiles, air plane, refrigerator, telephone switches, etc
Objectives

- After reading this chapter, you should understand:
  - what an operating system is.
  - a brief history of operating systems.
  - a brief history of the Internet and the World Wide Web.
  - core operating system components.
  - goals of operating systems.
  - operating system architectures.

What is an Operating System?

- Some years ago
  - an operating system was defined as the software that controls the hardware
  - (ex) RC4000 real-time control system
- Landscape of computer systems has evolved significantly, requiring a more complicated definition.
- Applications are now designed to execute concurrently.

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Let’s look at the requirements and expectations on computer systems by the user community and the low-level capabilities of existing hardware
- Bridge the “Semantic Gap” between hardware and application
- Software that enables applications to interact with a computer’s hardware
- A layer of software that separates applications from the hardware they access and provides services that allow each application to execute safely and effectively

- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
- Use the computer hardware in an efficient manner.

Computer System Components

1. Hardware
   provides basic computing resources (CPU, memory, I/O devices)
2. Operating system
   controls and coordinates the use of the hardware among the various application programs for the various users
3. Applications programs
   define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs)
4. Users
   people, machines, other computers

Bridging the Semantic Gap

- Hardware capabilities are very low level
  - Arithmetic and logical operators
  - Comparison of two bit-strings
  - Branching, reading, and writing bytes
- User needs to think in terms of problem to be solved
  - High-level data structures and corresponding operations
  - Simple, uniform interfaces to subsystems,
  - Treat programs and data files as single entities
- Use software to bridge this gap
  - Language processors (e.g., assemblers, compilers, interpreters).
  - Editors and text processors, linkers and loaders.
  - Application programs, utility and service programs.
  - Operating Systems

The role of OSs

- Bridge Hardware/Application Gap
  - Machine instruction vs high level operation
  - compiler bridges gap
  - Linear memory vs data structures
  - compiler bridges gap
  - Limited CPU & memory vs more needed
  - OS bridges gap
  - Secondary memory devices vs files
  - OS bridges gap
  - I/O devices vs high level I/O commands
  - OS bridges gap
Multiprocessor Systems

- **Shared memory machine**
  - Data can be shared by shared memory and shared variables
  - Unrealistic for a large number of processors

- **Distributed memory multiprocessor**
  - Each processor has its own private memory and uses message passing for data communication via interconnection network.
Three views of OSs

- **OS is an extended machine**
  - Principle of abstraction hides complexity
  - OS provides high level operations using lower level operations
- **OS is a virtual machine**
  - Principle of virtualization supports sharing
  - OS provides virtual CPU, memory, devices
- **OS is a resource manager**
  - Balance overall performance with individual needs (response time, deadlines)
  - Efficient use of shared resources

OS as Extended Machines

- Layers of software to hide complexity of the lower level operations
- Floppy disk drives, Hard disk drives, CR-ROMs, network devices, parallel/serial interface, USB interface, firewire interface, ...
  - What if each vendor makes their own interface operations for those different types of devices?

OS as Virtual Machines

- Creates the illusion of having multiple copies of the same physical resource
  - Think about the fact that many users can use the CPU, memory, and network device as if there is only one user without considering other people
  - (eg) however, you may voluntarily yield your resource to other people if you are generous
  - UNIX: nice, bg, sleep, thread_yield....
- Virtual memory, spooling

OS as Resource Managers

- Applications have different requirements
  - Interactive: shell, interactive games, Windows GUI
  - CPU intensive: scientific computation
  - I/O intensive: database, FTP, E-mail, editors
- Maintain higher resource utilization and reduce the response time

PC Hardware Organization
1.3 Early History: The 1940s and 1950s
- Operating systems evolved through several phases
  - 1940s
    - Early computers did not include operating systems
  - 1950s
    - Executed one job at a time
    - Included technologies to smooth job-to-job transitions
    - Single-stream batch-processing systems
    - Programs and data submitted consecutively on tape

---

Batch Operating Systems
- Batch OSs (Non-interactive computing)
  - Used batches of punched cards for each program
  - No interaction with a user

---

1.4 The 1960s
- Timesharing systems
  - Developed to support many simultaneous interactive users
  - Turnaround time was reduced to minutes or seconds
    - Time between submission of job and the return of its results
  - Real-time systems
    - Supply response within certain bounded time period
    - Improved development time and methods
    - MIT used CTSS (Compatible Time Sharing System) system to develop its own successor, Multics
    - TSS (Time Sharing System), Multics and CP/CMS (Control Program/Conversational Monitor System) all incorporated virtual memory
    - Address more memory locations than actually exist

---

Multiprogramming
- Basic problem:
  - Some programs are compute-bound, some I/O-bound
  - Even “balanced” programs are balanced only over time
  - No one program can make full use of the system

---
Computer is Lazy!!

• Human World
  – Rocket: 1000 Mph (500 times faster)
  – Boeing 747: 600 Mph (300 times faster)
  – Car: 70 Mph (35 times faster)
  – Human: 2 Mph

• Computer World
  – CPU speed: 1 GHz → 10^9 clock ticks/sec
    → 1 ns / clock tick
    → 10 ns / instruction cycle
  – Memory speed: effective memory access time 100 ns
    → 10 times slower
  – Hard disk speed: 1 MB/sec (10^6 bytes/sec)
    → writing 1KB takes 1,000,000ns → 100,000 times slower
    → Work for an hour and sleep for 100,000 hours (11.4 years)

Multiprogramming

• Solution: Multiprogramming
  – Have more than one active (running) program in memory at any one time

Multiprogramming requires
  – Bridging the semantic gap
  – Sharing resources among different programs
  – Hiding from each program the fact of this sharing

Multiprogramming

Sequential Execution: Job A is done first, then Job B starts running

Multiprogramming: Job A and B are interleaved sharing the resources

Higher throughput

OS Evolution and Concepts

• Multiprogramming Systems
  – Overlap CPU and I/O
  – Protection
    • Privacy and integrity of use process
    • Synchronization and Communication
      • Different processes compete for a common resource that need to be read or modified concurrently
    • Dynamic Memory Management
    • Swapping and paging
  – Interactive OSs
    – Throughput: #job finished / time
    – Response time: Time-sharing (quantum) → reduce the response time

1.5 The 1970s

• Primarily multimode timesharing systems
  – Supported batch processing, timesharing and real-time applications
  – Personal computing only in incipient stages
    • Fostered by early developments in microprocessor technology

• Department of Defense develops TCP/IP
  – Standard communications protocol
  – Widely used in military and university settings
  – Security problems
    • Growing volumes of information passed over vulnerable communications lines.

1.6 The 1980s

• 1980s
  – Decade of personal computers and workstations
  – Early PC OSs
    • Single user single program system → later extended to support multiprograms
    • TSR (Terminate and Stay Resident) programs
    • No major innovations except for GUI
  – Computing distributed to sites at which it was needed
    • Client/Server model
  – Personal computers proved relatively easy to learn and use
    • Graphical user interfaces (GUI)
  – Transferring information between computers via networks became more economical and practical
1.6 The 1980s

- Client/server computing model became widespread
  - Clients request various services
  - Servers perform requested services
- Software engineering field continued to evolve
  - Major thrust by the United States government aimed at tighter control of Department of Defense software projects
  - Realizing code reusability
  - Greater degree of abstraction in programming languages
  - Multiple threads of instructions that could execute independently

1.7 History of the Internet and World Wide Web

- Advanced Research Projects Agency (ARPA)
  - Department of Defense
  - In late 1960s, created and implemented ARPAnet
- Grandparent of today's Internet
- Networked main computer systems of ARPA-funded institutions
- Capable of near-instant communication via e-mail
- Designed to operate without centralized control

1.7 History of the Internet and World Wide Web

- Transmission Control Protocol/Internet Protocol
  - Set of rules for communicating over ARPANet
  - TCP/IP manages communication between applications
  - Ensure that messages routed properly from sender to receiver
  - Error-correction
  - Later opened to general commercial use

1.8 The 1990s

- Hardware performance improved exponentially
  - Inexpensive processing power and storage
  - Execute large, complex programs on personal computers.
  - Economical machines for extensive database and processing jobs
  - Mainframes rarely necessary
  - Shift toward distributed computing rapidly accelerated
  - Multiple independent computers performing common task
- Operating system support for networking tasks became standard
  - Increased productivity and communication
  - Security threats
- Microsoft Corporation became dominant
  - Windows operating systems
    - Employed many concepts used in early Macintosh operating systems
    - Enabled users to navigate multiple concurrent applications with ease.
    - 1981 : MS-DOS
    - 1990 : Windows 3.0
    - 1993 : Windows 3.1, Windows NT
    - 1995 : Windows 95
1.8 The 1990s

- Object technology became popular in many areas of computing
  - Many applications written in object-oriented programming languages
    - For example, C++ or Java
    - Improve reusability
  - Object-oriented operating systems (OOOS)
    - Objects represent components of the operating system
    - Concepts such as inheritance and interfaces
      - Exploited to create modular operating systems
      - Easier to maintain and extend than systems built with previous techniques

- Most commercial software sold as object code (binaries)
  - The source code not included
  - Enables vendors to hide proprietary information and programming techniques

- Free and open-source software became increasingly common in the 1990s
  - Open-source software distributed with the source code
  - Allows individuals to examine and modify software
    - Linux operating system and Apache Web server both open-source
  - Richard Stallman launched the GNU project
    - Recreate and extend tools for AT&T’s UNIX operating system
    - GNU is Not UNIX
    - He disagreed with concept of paying for permission to use software

1.8 The 1990s

- Open Source Initiative (OSI)
  - www.opensource.org
  - Founded to further benefits of open-source programming
  - Facilitates enhancements to software products
  - Permissions anyone to test, debug and enhance applications
  - Increases chance that subtle bugs will be caught and fixed
    - Crucial for security errors which need to be fixed quickly
  - Individuals and corporations can modify the source
    - Create custom software to meet needs of certain environment
  - Open source operating systems
    - LINUX, FreeBSD, OpenBSD

1.9 2000 and Beyond

- Middleware
  - Links two separate applications
    - Often over a network and between incompatible machines
    - Particularly important for Web services
      - Simplifies communication across multiple architectures

- Web services
  - Encompass set of related standards
  - Ready-to-use pieces of software on the Internet
  - Enable any two applications to communicate and exchange data

- Multiprocessor and network architecture
  - New hardware and software design techniques
  - Massive parallelism
    - More CPUs

- Portability among different OSs
  - Portable Operating System Interface (POSIX)

- Computing on mobile devices