Random Stuff

- No tutorials or comp3891/comp9283 lecture this week
- Release the warm-up exercise tomorrow
- New weeks tutorial questions – probably tomorrow also
Introduction to Operating Systems

Chapter 1 – 1.3
Chapter 1.5 – 1.9
Learning Outcomes

• High-level understand what is an operating system and the role it plays
• A high-level understanding of the structure of operating systems, applications, and the relationship between them.
• Some knowledge of the services provided by operating systems.
• Exposure to some details of major OS concepts.
What is an Operating System?
Viewing the Operating System as an Abstract Machine

- Extends the basic hardware with added functionality
- Provides high-level abstractions
  - More programmer friendly
  - Common core for all applications
- It hides the details of the hardware
  - Makes application code portable
Viewing the Operating System as a Resource Manager

- Responsible for allocating resources to users and processes
- Must ensure
  - No Starvation
  - Progress
  - Allocation is according to some desired policy
    - First-come, first-served; Fair share; Weighted fair share; limits (quotas), etc...
  - Overall, that the system is efficiently used
Traditional View: the Operating System as the Privileged Component

Privileged Mode

Operating System

Requests (System Calls)

User Mode

Applications

Applications

Applications
Kernel

- Portion of the operating system that is running in *privileged mode*
- Usually resident in main memory
- Contains fundamental functionality
  - Whatever is required to implement other services
  - Whatever is required to provide security
- Contains most-frequently used functions
- Also called the nucleus or supervisor
The Operating System is Privileged

- Applications should not be able to interfere or bypass the operating system
  - OS can enforce the “extended machine”
  - OS can enforce its resource allocation policies
  - Prevent applications from interfering with each other

- Note: Some Embedded OSs have no privileged component, e.g. PalmOS
  - Can implement OS functionality, but cannot enforce it.

- Note: Some operating systems implement significant OS functionality in user-mode, e.g. User-mode Linux
Structure of a Computer System

User Mode

- Application
- System Libraries

Kernel Mode

- Operating System
- Hardware
Structure of a Computer System

User Mode

- Application
- System Libraries

Kernel Mode

- Operating System

Hardware

Interacts via load and store instructions to CPU and device registers, and interrupts
Structure of a Computer System

User Mode
- Application
- System Libraries

Interaction via function calls to library procedures

Kernel Mode
- Operating System
- Hardware
Structure of a Computer System

User Mode

Application

System Libraries

Interaction via System Calls

Kernel Mode

Operating System

Hardware
A note on System Libraries

• System libraries are just that, libraries of support functions (procedures, subroutines)
  – Only a subset of library functions are actually systems calls
    • strcmp(), memcpy(), are pure library functions
    • open(), close(), read(), write() are system calls
  – System call functions are in the library for convenience
Operating System

Objectives

• Convenience
  – Make the computer more convenient to use

• Abstraction
  – Hardware-independent programming model

• Efficiency
  – Allows the computer system to be used in an efficient manner

• Ability to evolve
  – Permit effective development, testing, and introduction of new system functions without interfering with existing services

• Protection
Services Provided by the Operating System

• Program development
  – Editors, compilers, debuggers
    • Not so much these days
• Program execution
  – Load a program and its data
• Access to I/O devices
• Controlled access to files
  – Access protection
• System access
  – User authentication
Services Provided by the Operating System

• Error detection and response
  – internal and external hardware errors
    • memory error
    • device failure
  – software errors
    • arithmetic overflow
    • access forbidden memory locations
  – operating system cannot grant request of application
Services Provided by the Operating System

• Accounting
  – collect statistics
  – monitor performance
  – used to anticipate future enhancements
  – used for billing users
Operating System Software

- Fundamentally, OS functions the same way as ordinary computer software
  - It is a program that is executed (just like apps)
  - It has more privileges
- Operating system relinquishes control of the processor to execute other programs
  - Reestablishes control after
    - System calls
    - Interrupts (especially timer interrupts)
Cross Compiler

C5161 - gcc
Major OS Concepts

- Processes
- Concurrency and deadlocks
- Memory management
- Files
- Information Security and Protection
- Scheduling and resource management
Processes

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of resource ownership
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources
  - Nowadays the execution abstraction is separated out: *Thread*
  - Single process can contain many threads
Process

• Consist of three segments
  – Text
    • contains the code (instructions)
  – Data
    • Global variables
  – Stack
    • Activation records of procedure
    • Local variables

• Note:
  – data can dynamically grow up
  – The stack can dynamically grow down
Process

- Consists of three components
  - An executable program
    - text
  - Associated data needed by the program
    - Data and stack
  - Execution context of the program
    - All information the operating system needs to manage the process
      - Registers, program counter, stack pointer, etc…
    - A multithread program has a stack and execution context for each thread
Multiple processes creates concurrency issues

(a) A potential deadlock. (b) an actual deadlock.
Memory Management

- The view from thirty thousand feet
  - Process isolation
    - Prevent processes from accessing each other’s data
  - Automatic allocation and management
    - Don’t want users to deal with physical memory directly
  - Protection and access control
    - Still want controlled sharing
  - Long-term storage
  - OS services
    - Virtual memory
    - File system
Virtual Memory

- Allows programmers to address memory from a logical point of view
  - Gives apps the illusion of having RAM to themselves
  - Logical addresses are independent of other processes
  - Provides isolation of processes from each other
- Can overlap execution of one process while swapping in/out others.
Virtual Memory Addressing

Figure 2.10  Virtual Memory Addressing