Introduction

• Course Objectives
  – explore the why?’s and “how?’s of operating system design and implementation

• Key Areas
  – OS Structures
  – Process Management
  – Storage Management
  – I/O Systems
  – Protection and Security

• Slides based on
  – “Operating Systems Concepts” by Silberschatz et al
Introduction/Grading

- A midterm (90 min. short answer or multiple choice exam during class hours), and a final exam (two hours). Each exam is worth 20% of the final grade.
- Two homework assignments (each contributing 30% of the final grade). Homeworks will be due two weeks after they are assigned, at the beginning of class. Programming assignments must be handed in via email.
- Assignment prerequisites:
  - Good knowledge of C, Unix.

Computer Systems

- Web, Banking, Airlines
- Compilers, Editors, CLI’s
- Operating System
- Machine Language
- Microarchitecture
- Physical Devices

<table>
<thead>
<tr>
<th>Application Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Programs</td>
</tr>
<tr>
<td>Hardware</td>
</tr>
</tbody>
</table>

Computer system components
What is an Operating System

• Extending the machine
  – life is too short to deal directly with hardware
  – hardware changes programs (hopefully) remain

  → OS acts as an intermediary between the programmer and the hardware.

• Resource Manager
  – multi user - multi tasking
  – housekeeping
  – resource scheduling

  → By channeling all activities through the OS, we allow it to manage all pieces of a complex system

Key Requirements

• Closed Systems
  – reliability
  – availability
  – serviceability

• Open Systems
  – security
  – ease of use
  – flexibility
Historical Perspective

- the more it changes the more it stays the same

(French proverb)

Operating System Definitions

- **kernel** - the OS core, runs all the time
- **device drivers** - integral parts of the kernel, or loadable modules that control the hardware devices
- **applications** - programs that run outside the kernel space
- **multitasking** - ability to execute more than one program at a time.
- **multiuser** - ability to support more than one user at a time
Memory Layout of a Simple System

- what happens when the OS grows?

Multi-tasking

- Several tasks are kept in memory at the same time and the CPU is shared between them.
Multi-tasking Prerequisites

- I/O must be handled by the Operating System.
- Memory management - the OS must allocate memory to the various tasks and ensure that they do not interfere with one another.
- CPU scheduling - the system must chose among several tasks that are ready to run.
- Allocation of devices.
  - dead lock
  - spooling

Parallel Systems

- Parallelism is often inherent in problems
  - e.g. matrix operations
  - mandelbrot set
- Various forms of parallelism exists
  - array processors
  - tightly coupled multiprocessors
  - loosely coupled multiprocessors (clusters)
- Advantages
  - transcend limits of uniprocessor
  - increased reliability
Parallel Systems (Cont.)

- Symmetric Multiprocessors (SMP)
  - each processor runs the same copy of the OS
  - many processes can run at once
  - most modern OSs support SMP

- Asymmetric Multiprocessing
  - each processor is assigned a specific task; master processor allocates work to slaves
  - very old idea (channel controllers in System/360)

Clustering

- Clustering allows two or more systems to share resources
- Provides high reliability
- **Asymmetric Clustering**: one machine runs the application, while the others are standing by.
- **Symmetric Clustering**: all machines run the application
Real - Time Systems

- Used to control dedicated applications
  - automation
  - measurement
  - avionics
  - multimedia
- Well defined fixed-time constraints
- Real-time systems may be hard or soft real - time.
  - key criterion is maximum time that may elapse before an event is serviced.
  - impacts scheduling and interrupt processing

Embedded Systems

- special purpose systems
  - industrial process controllers
  - home bread cookers
  - cell phones
- special requirements
  - hostile environment
  - real-time constraints
  - reliability - availability
  - unattended operation
  - special peripherals
Questions

• what are the main purposes of an operating system?
• what is the main advantage of multi-tasking?
• what are the differences between symmetric and asymmetric multiprocessor systems?
• what are the tradeoffs inherent in handheld computers?

Computer System Structures

• Computer System Operation
• I/O Structure
• Storage Hierarchy
• Hardware Protection
• General System Architecture
Computer System Architecture

- CPU
- Disk controller
- Printer controller
- Tape controller
- Memory controller
- Memory
- System bus

Computer System Operation

- I/O devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers.
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*. 
Common Function of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
- Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.
- A trap is a software-generated interrupt caused either by an error or a user request.
- An operating system is interrupt driven.

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - polling
  - vectored interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt.
I/O Structure

- **Synchronous**: After I/O starts, control returns to user program only upon I/O completion.
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access).
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing.

- **Asynchronous**: After I/O starts, control returns to user program without waiting for I/O completion.
  - System call – request to the operating system to allow user to wait for I/O completion.
  - Device-status table contains entry for each I/O device indicating its type, address, and state.
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

Device Status Table

<table>
<thead>
<tr>
<th>Device</th>
<th>Status</th>
<th>Request Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>Idle</td>
<td></td>
</tr>
<tr>
<td>Laser Printer</td>
<td>Busy</td>
<td>Request for laser printer address: xyz, length: 1374</td>
</tr>
<tr>
<td>Mouse</td>
<td>Idle</td>
<td></td>
</tr>
<tr>
<td>Disk Unit 1</td>
<td>Idle</td>
<td></td>
</tr>
<tr>
<td>Disk Unit 2</td>
<td>Busy</td>
<td>Request for du2 block: abc operation: read address: yyy length: 5560</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Request for du2 block: ghi operation: write address: xxx length: 1024</td>
</tr>
</tbody>
</table>
DMA Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds.
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Only on interrupt is generated per block, rather than the one interrupt per byte.

Storage Structure

- **Main memory** – only large storage media that the CPU can access directly.
- **Secondary storage** – extension of main memory that provides large nonvolatile storage capacity.
- **Magnetic disks** – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into tracks, which are subdivided into sectors.
  - The disk controller determines the logical interaction between the device and the computer.
Moving Head Disk Assembly

- CHS addressing (cylinder, head, sector)

Storage Hierarchy

- Storage systems organized in hierarchy:
  - Speed
  - Cost
  - Volatility
- Caching – copying information into faster storage system; main memory can be viewed as a last cache for secondary storage.
Caching

- Use of high-speed memory to hold recently-accessed data.
  - Requires a *cache management* policy.
- Caching introduces another level in storage hierarchy.
  - This requires data that is simultaneously stored in more than one level to be *consistent*. 
Backup

• Manual transfer of information from on-line to off-line storage
• Applications
  – disaster recovery
  – data protection
  – archiving
• Why backup when you have RAID?
• Backup techniques
  – live data
  – quiescent data

Backup (cont)

• Types of backup
  – full backup
  – incremental
• Backup considerations
  – how long do we want to keep the data
  – what do we do with the backups (storage - disposal)
  – is keeping just the data sufficient?
Hardware Protection

- Dual Mode Operation
- I/O Protection
- Memory Protection
- CPU Protection

Dual Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- Provide hardware support to differentiate between at least two modes of operations.
  1. User mode – execution done on behalf of a user.
  2. Monitor mode (also kernel mode or system mode) – execution done on behalf of operating system.
Dual Mode Operation (cont)

- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1).
- When an interrupt or fault occurs hardware switches to monitor mode.

```
    monitor  set user mode  user
  interrupt/fault
```

*Privileged instructions* can be issued only in monitor mode.

I/O Protection

- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode (i.e., a user program that, as part of its execution, stores a new address in the interrupt vector).
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - **Base register** – holds the smallest legal physical memory address.
  - **Limit register** – contains the size of the range
- Memory outside the defined range is protected.

Use of A Base and Limit Register
Hardware Protection

- When executing in monitor mode, the operating system has unrestricted access to both monitor and user’s memory.
- The load instructions for the base and limit registers are privileged instructions.

CPU Protection

- **Timer** – interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs
- Timer commonly used to implement time sharing.
- Time also used to compute the current time.
- Load-timer is a privileged instruction.
- Timer may also be used to implement **watchdog reset**.
Questions

- Compare prefetching with spooling
- How does the distinction between monitor and user modes affect system security?
- Can reentrant code have static structures?
- Why are caches useful? What problems do they solve?
- Some CPUs provide more than two modes of operation, what are the possible uses of these multiple modes?

General System Architecture

- System Components
- Operating System Services
- System Calls
- System Programs
- System Structure
- Virtual Machines
- System Design and Implementation
- System Generation
Common System Components

- Process Management
- Main Memory Management
- File Management
- I/O System Management
- Secondary Management
- Networking
- Protection System
- Command-Interpreter System

Process Management

- A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion.
  - process suspension and resumption.
  - Provision of mechanisms for:
    - process synchronization
    - process communication
Main-Memory Management

- Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents in the case of system failure.
- The operating system is responsible for the following activities in connection with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and deallocate memory space as needed.

File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- The operating system is responsible for the following activities in connection with file management:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - File backup on stable (nonvolatile) storage media.
I/O System Management

- The I/O system consists of:
  - A buffer-caching system
  - A general device-driver interface
  - Drivers for specific hardware devices

Secondary-Storage Management

- Since main memory (primary storage) is volatile and too small to accommodate all data and programs permanently, the computer system must provide secondary storage to back up main memory.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling
Networking (Distributed Systems)

- A distributed system is a collection processors that do not share memory or a clock. Each processor has its own local memory.
- The processors in the system are connected through a communication network.
- Communication takes place using a protocol.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability

Protection System

- Protection refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
- The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.
  - provide a means of enforcement.
Command-Interpreter System

• Many commands are given to the operating system by control statements which deal with:
  – process creation and management
  – I/O handling
  – secondary-storage management
  – main-memory management
  – file-system access
  – protection
  – networking

Command-Interpreter System
(Cont.)

• The program that reads and interprets control statements is called variously:
  – command-line interpreter
  – shell (in UNIX)
• Its function is to get and execute the next command statement.
Operating System Services

- Program execution - system capability to load a program into memory and to run it.
- I/O operations - since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.
- File-system manipulation - program capability to read, write, create, and delete files.
- Communications - exchange of information between processes executing either on the same computer or on different systems tied together by a network. Implemented via *shared memory* or *message passing*.
- Error detection - ensure correct computing by detecting errors in the CPU and memory hardware, in I/O devices, or in user programs.

Additional Operating System Functions

- Additional functions exist not for helping the user, but rather for ensuring efficient system operations.
  - Resource allocation - allocating resources to multiple users or multiple jobs running at the same time.
  - Accounting - keep track of and record which users use how much and what kinds of computer resources for account billing or for accumulating usage statistics.
  - Protection - ensuring that all access to system resources is controlled.
System Calls

- System calls provide the interface between a running program and the operating system.
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C, C++)
- Three general methods are used to pass parameters between a running program and the operating system.
  - Pass parameters in *registers*.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
  - *Push* (store) the parameters onto the stack by the program, and pop off the *stack* by the operating system.

Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications
System Programs

• System programs provide a convenient environment for program development and execution. The can be divided into:
  – File manipulation
  – Status information
  – File modification
  – Programming language support
  – Program loading and execution
  – Communications
  – Application programs
• Most users’ view of the operation system is defined by system programs, not the actual system calls.

MS-DOS System Structure

• MS-DOS – written to provide the most functionality in the least space
  – Not divided into modules
  – Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated
UNIX System Structure

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts.
  - Systems programs
  - The kernel
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.
Microkernel System Structure

- Moves as much from the kernel into “user” space.
- Communication takes place between user modules using message passing.
- Benefits:
  - easier to extend a microkernel
  - easier to port the operating system to new architectures
  - more reliable (less code is running in kernel mode)
  - more secure

System Design Goals

- User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast.
- System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient.
Mechanisms and Policies

• Mechanisms determine how to do something, policies decide what will be done.
• The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later.

System Implementation

• Traditionally written in assembly language, operating systems can now be written in higher-level languages.
• Code written in a high-level language:
  – can be written faster.
  – is more compact.
  – is easier to understand and debug.
• An operating system is far easier to port (move to some other hardware) if it is written in a high-level language.
System Generation (SYSGEN)

• Operating systems are designed to run on any of a class of machines; the system must be configured for each specific computer site.
• SYSGEN program obtains information concerning the specific configuration of the hardware system.
• Booting – starting a computer by loading the kernel.
• Bootstrap program – code stored in ROM that is able to locate the kernel, load it into memory, and start its execution.