CSCE-352 Chapter 1
Pt. 1

- What OS Do?
- Some Computer-System Organization
- OS Structure
- OS Operations

✎ General Info

- Some computers **have no UIs**
- Users want the system to be **convenience**, **easy to use** and **good performance**
- I/O devices and CPU can **execute concurrently**
- Each **device controller** has a local **buffer**
- CPU moves data from/to main memory to/from local buffers (buffer is a small memory)
- **Device controller** informs CPU that it has finished its operation by causing an **interrupt**

◍ Definition

A **program** that acts as an **intermediate** between **user** and **hardware**.

◍ Goals

- Execute programs
- Make computer convenient
- Efficient use of hardware

◍ Further Info

- **OS is a resource allocator**: **Manages all resources** and decides between conflict requests **for efficient resource use**
- **OS is a control program**: **Controls execution** of programs to prevent errors and improve the use of the computer
- OS has no universally accepted definition
- OS depends on the point of view

Operating System

◍ OS Structure

Multiprogramming (Batch system): needed for efficiency.

- A single user cannot keep CPU and IO devices busy all times
- Multiprogramming organize jobs so CPU execute one at a time
- Subset of total jobs in system is kept in memory
- One job is selected and run via **job scheduling**; when it has to wait, the OS switches to another job

Timesharing (multitasking): is logical extension in which CPU **switches jobs**, so user can interact with each job while it is running, creating **interactive** computing.

- less than 1 second → the **Response Time**
- User has at least one program executing in memory → called **Process**
- If several jobs ready to run at the same time → use **CPU Scheduling**
- If a process doesn't fit in memory → use **Swapping**
- Allows process to execute not completely in memory → called **Virtual Memory**

◍ OS Operations

Interrupt driven

- Hardware interrupted by one of the devices
- Software interrupted by exceptions or traps

Dual-mode operation: allows OS to protect itself and other system components.

- Like user mode and kernel mode
- Mode bit (provided by hardware)
- Some instructions designated as privileged, only executable in kernel mode
- CPUs increasingly support multi-mode operations

Multiprocessors systems

Also known as parallel systems, tightly-coupled systems

Are growing in use and importance

types of Multiprocessing:

1. **Asymmetric**: each processor is assigned to a specie task
2. **Symmetric**: each processor performs all tasks

Advantages:

1. Increased throughput
2. Economy of scale
3. Increased reliability

Computer system can be divided into 4 components:

Hardware
Basic computing resources: CPU, memory, I/O devices

OS
Controls and coordinates use of hardware among various applications and users

Application Programs
Used to solve the computing problems of the users: Word, games, and web browsers

User

Every Program is either:

- System Program
- Application Program

Except the Kernel.

→ **Kernel is the only program running all time**

Interrupt transfers control, through the **interrupt vector**, the **interrupt vector** contains the addresses of the service

- Why?
  To dedicate the address of the previous operation
- A **trap** or **exception** is a generated interrupt, caused by error or user
- OS is interrupt driven
- Types or methods of interrupt:
  Polling
  Vectored interrupt system

YouTube channel link: Soft Conclusion - YouTube

CSCE-352 Chapter 1
Pt. 2

- Some Computer-System Organization
- Computer-System Architecture
- Process Management
- Memory Management
- Protection and Security
- Open-Source OS

Memory Management

- Memory management determines what's in memory and when
  - thus, optimize CPU utilization and computer response
- To execute a program, the instructions must be in memory
- Data (or part) needed by the program, must be in memory
- Memory management activities:
  - Keeping track of which parts of memory are being used and by who
  - Deciding which processes and data to move into and out of memory
  - Allocating and deallocating memory space as needed

OS activities in the context of **memory management** include:

- Creating and deleting files and directories
- Mapping files onto secondary storage
- Backup files onto stable (non-volatile) storage media

✎ Information about storage and memory

Storage systems organized in hierarchy:
- Cost
- Speed
- Volatility

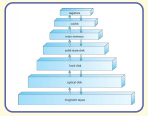
Direct Memory Access (**DMA**): Used for high-speed I/O devices, to transmit information close to memory speeds

Device Driver: Provides uniform interface between controller and kernel. In each device controller there is a device driver

Open-source OS is made available in source-code format

Counter to copy protection and Digital Rights Management (DRM)

Examples include Linux and UNIX



Storage structure (memory types):

- **Main memory**
  Large storage, that the CPU can access directly, random access and volatile
- **Secondary storage**
  Extension of main memory, that provides large nonvolatile storage
- **Hard disks**
  Rigid metal or glass platters covered with magnetic recording material
- **Solid-state disks**
  Faster than hard disks, nonvolatile

Caching is copying information **to faster storage**; **main memory** can be viewed as a cache for **secondary storage**.

```
if (info in cache) {
use info directly from cache
}
else if (info not in cache ) {
copy info into cache
}
```

caching is an important principle, performed at many levels in a computer (in hardware, operating system, or software)

Process needs **resources** to accomplish its task, like CPU or Initialized data

A **process** is a program in execution, and it is a unit of work within the system

Program is a **passive entity**, **Process** is an **active entity**

Two types of process:

- Single-threaded process: has one **program counter** specifying **location** of **next instruction** to execute
- Multi-threaded process: has one **program counter** per thread

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting user or system processes
- Suspend and resume processes
- Provide mechanisms for process synchronization, communication, and for deadlock handling

Protection is any mechanism for **controlling access** of a process or user to resources, defined by the OS

Security is the **defense** of the system, against internal and external attacks

Computing Environments

1. Distributed computing collection of separate systems, networked (linked) together.

Network: a communications path, like:
- Local Area Network (LAN)
- Wide Area Network (WAN)
- Metropolitan Area Network (MAN)
- Personal Area Network (PAN)

Network Operating System: provides features between systems across network.
- Allows systems to exchange messages
- Illusion of a single system (to user)

2. Client-Server Computing a servers that respond to a request generated by client.

- **Compute-server system** provides an interface to client to request a service.
- **File-server system** provides an interface for clients to store and retrieve files.

3. Peer-to-Peer is another model of distributed system.

- **P2P** does not distinguish clients and servers.
- All nodes are considered peers.
- Each node acts as client, server or both.

4. Virtualization allows operating systems to run other OSes.

How? OS (natively compiled) running guest OS (also natively compiled).

Emulation used when CPU type is different from target type (slowest method).

VMM (virtual machine Manager) provides virtualization services, also VMM can run natively

In **Virtualization** there is **host** OS and **guest** OS

Some common use cases:
- Developing apps for multiple OSes
- Testing applications

YouTube channel link: Soft Conclusion - YouTube

CSCE-352 Chapter 2
Pt. 1

OS Services, Design, Implementation, System Calls, and System Programs

Types of **System Programs**:

- File Management
- File Modification
- Status Information
- Programming-Language Support
- Program Loading and Execution
- Communications
- Background Services

System Programs

Provide a convenient environment for program **development** and **execution**

Application programs

Run by users, not considered part of OS

Launched by command line, mouse click, finger poke

OS Design and Implementation

- D&I of OS is **not solvable**
- Internal structure of OSes can vary
- Start the design by defining: **goals** and **specifications**
- D&I is affected by choice of hardware
- Early OSes in assembly language, then system programming languages (like Algol, PL/1), now mix of languages (lowest levels in assembly, main body in C, systems programs in C, C++)
- High-level language easier to port to other hardware (but slower)
- **Emulation** can allow an OS to run on non-native hardware

CLI or **command interpreter** → allows direct command entry.

- Implemented in kernel, or by system program
- Sometimes multiple flavors implemented → shells
- Fetch a command from user and executes it
- Built-in commands, or names of programs

GUI → User-friendly desktop metaphor interface.

- Usually mouse, keyboard, and monitor
- Icons represent files, programs, or actions
- Invented at Xerox PARC

Touchscreen Interfaces

- Virtual keyboard for text entry
- Include voice commands

User Interface

Almost all operating systems have a user interface (UI), such as:

Command-Line (CLI)
Graphics User Interface (GUI)
Batch

Protection and Security

The owners of information stored in a multiuser or networked computer system may want to control that information

File-system Manipulation

Some programs need to read, write, create, delete, or search files and directories, as well as managing permissions

Program Execution

System must be able to load a program into memory and run it

OS provide an execution environment and **services** to programs and users.

OS services provides helpful functions to user

9 examples of OS services:

I/O Operations

Accounting

There are 2 types of goals in D&I of OS:

- **User goals** – operating system should be convenient to use, easy to learn, reliable, safe, and fast
- **System goals** – OS should be easy to design, implement, maintain, flexible, reliable, error-free, and efficient

- **Policy**: What will be done?
- **Mechanism**: How to do it?

Policies decide what will be done
Mechanisms determine how to do something

System Calls

Programming interface to the services, provided by the OS

Typically written in a high-level language (C or C++)

Most programs use a high-level **Application Programming Interface** (**API**) rather than direct **System Call**

Types of **System Calls**:

- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection

Communications

Processes may exchange information, on the same computer or between computers over a network

Error Detection

OS needs to be constantly aware of possible errors

Resource Allocation

When multiple users or multiple jobs running concurrently, resources must be allocated to each of them

Resource = Hardware or Software

YouTube channel link: Soft Conclusion - YouTube

CSCE-352 Chapter 2
Pt. 2

OS Structures, Debugging, Generation, and System Boot

Operating System Structures

OS is a very large program

Various ways to structures:

♜ 1. Simple structure

Example → MS-DOS

Written to provide the **most functionality** in the **least space**

Not divided into modules

♜ 2. Non Simple Structure

Example → UNIX

Limited by hardware functionality, and has two parts: **System programs** and **Kernel**

♜ 3. Layered

The operating system is **divided into a number of layers**

The bottom layer is the hardware; the highest is the UI

Each layer uses function and services from the lower layer

With modularity

♜ 4. Microkernel

Example → Mach

Moves as much from the kernel into user space

Benefits:
Easier to extend, and to port to a new architecture, also more reliable and secure

♜ 5. Modules

Example → Linux, Solaris

Many modern OSes implement **loadable kernel modules**

- Use object-oriented approach
- Each component is separate
- Talks to each over a known interfaces
- Each is loadable as needed in kernel

Similar to layers but with more flexible

♜ 6. Hybrid Systems

Example → Apple Mac OS X

Most modern OSes are not one pure model, they are Hybrid Systems

Combining multiple approaches to address performance, security, usability needs

♜ 7. Monolithic

Example → Windows

Operating System Generation

Operating systems designed to run on any class of machines

SYSGEN is a program obtains information concerning the **specific configuration** of the **hardware** system

Operating-System Debugging

- Debugging is finding and fixing errors, or bugs
- OS generate log files, contains error information

- Failure of an **application** → generate **core dump**
- Failure of **OS** → generate **crash dump**

System Boot

When power initialized on system, execution starts at a **fixed memory location**

OS must be **available to hardware** so hardware can start it

Firmware **ROM** is used to hold initial boot code

Small piece of code called **bootstrap loader**, stored in **ROM** or **EEPROM**, locates the kernel, loads kernel to memory, and start the system

When kernel loads, system runs