

الجامعة التقنية الشمالية
الكلية التقنية / كركوك
قسم هندسة تقنيات الحاسوب
المادة: انظمة التشغيل
مدرسة المادة: د. امل سعيد طعمة



- Textbook:

Operating System Concepts -- Ninth Edition

A. Silberschatz, P.B. Galvin, and G. Gagne.

Operating systems are essential part of any computer system. Therefore, a course in operating systems is an essential part of any computer science education. The fundamental concepts of operating systems will be presented in this course.

The **syllabus** of the operating system course is as follows:

- Operating system overview
- Main frame systems
- Desktop systems
- Multiprocessor systems
- Distributed systems
- Clustered systems
- Real time systems
- Handheld systems
- Computing environment
- Computer system structure
- Hardware protection
- operating system structure
- operating system components
- operating system services
- processes
- process concepts
- operation on processes

- cooperating process
- threads
- CPU scheduling
- Memory Management
- Storage management
- Protection and Security

Objective of the course:

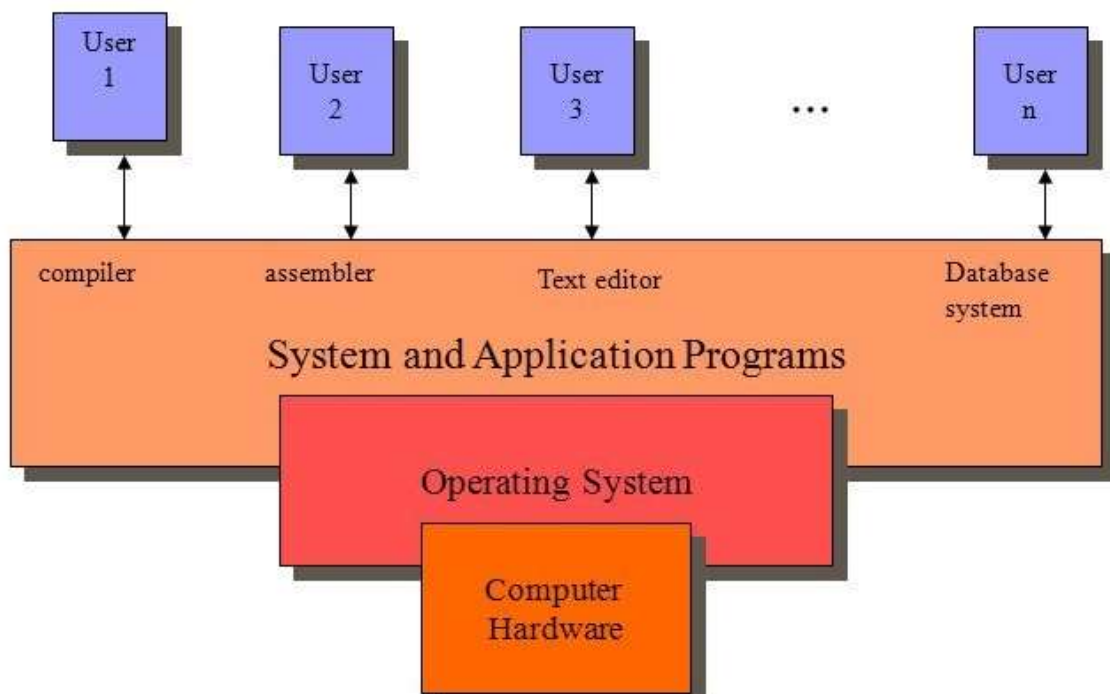
- To provide a general explanation of the component of operating systems.
- To provide the general organization of the computer systems and the relation between the computer structure and operating systems.

Chapter 1

Introduction

What is Operating System?

OS is the software (program) that manages the computer hardware. Therefore, it acts as an intermediary between a user of a computer and the computer hardware.



Computer systems can be divided into four components

- Hardware –provides basic computing resources CPU, memory, I/O devices
- Operating system-Controls and coordinates use of hardware among various applications and users

□ Application programs –Define the ways in which the system resources are used to solve the computing problems of the users

Word processors, compilers, web browsers, database systems, video games

□ Users

People, machines, other computers

Kernel: is the one program running at all times on the computer—usually called the kernel.

User View

The user view of computer varies by the interface being used. The operating systems are designed mostly for ease of use. Others are designed to maximize resource utilization. Other operating systems are designed to compromise between individual usability and resource utilization.

System view

From the computer's point of view, the OS is a

□ **resource allocator**

Manages all resources and decides between conflicting requests for efficient and fair resource use

□ **control program**

Controls execution of programs to prevent errors and improper use of the computer

Why should we study Operating Systems?

An operating system is needed for the following reasons:

- Need to understand interaction between the hardware and software
- Need to understand basic principles in the design of computer systems
- Efficient resource management, security, flexibility

Operating system roles

Referee

- Resource allocation among users, applications
- Isolation of different users, applications from each other
- Communication between users, applications

Illusionist

- Each application appears to have the entire machine to itself
- Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport

Glue

- Libraries, user interface widgets, ...
- Reduces cost of developing software

OS challenges

Reliability

- Does the system do what it was designed to do?

Availability

- What portion of the time is the system working?
- Mean Time To Failure (MTTF), Mean Time to Repair

Security

- Can the system be compromised by an attacker?

Privacy

- Data is accessible only to authorized users

Performance

- Latency/response time
- How long does an operation take to complete?
- Throughput
- How many operations can be done per unit of time?
- Overhead
- How much extra work is done by the OS?
- Fairness
- How equal is the performance received by different users?
- Predictability
- How consistent is the performance over time?

Portability

- For programs: Application programming interface (API)
- For the kernel: Hardware abstraction layer

Historical development of Operating System

Operating systems and computer architecture have influenced each other. To facilitate the use of the hardware, researchers developed operating systems. In the following historical review, we will notice the mutual effect between operating systems and computer hardware which led to developments in both sides.

1- Mainframe systems

Mainframe systems were the first computers to achieve commercial and scientific applications. M-F systems grow on three stages:

A- Batch systems

Early computers were physically large machines run from a **console**. The input devices were card readers and tape drives. The output devices were line printer, card punches. In this type of computer systems, the **operator** batch +together jobs with similar needs and ran through the computer as group.

The operating system was simple, always resident in memory, and its major task was to transfer control automatically from one job to the next.

There was a lack of interaction between user and his job (program). CPU is often **idle** because speed of mechanical I/O devices is slower than electronic CPU.

B-Multi-programmed systems

The introduction of disk technology helps to read from cards to disk, and OS reads/writes the job from/to disk.

The Multi-programmed operating system is the first one which makes a decision for the users to switch among jobs. Making this decision is called job scheduling or spooling in order to increase **CPU utilization**.

C-Time-shared systems

*The CPU executes multiple jobs by switching among them, but the switches occurred so frequently the users can interact with each program while it is running. A little CPU time is needed for each user (time slice).

*Time-shared Operating Systems allows many user programs (processes) to share the computer simultaneously. Users can interact with their programs. Keyboard is used for input, Monitor display is used for output.

*In time sharing OS, virtual memory is used to perform job swapping in memory management environment in order to use programs larger than physical memory.

2- Desktop systems

The operating systems of desktop systems were neither multi-user nor multitasking. Operating systems have changed with time; instead of maximizing CPU and peripheral utilization, the systems improved to maximize user convenience and responsiveness.

3-Multiprocessor Systems (Parallel systems or tightly coupled systems)

Such systems have more than one processor in close communication sharing the computer bus, the clock, and sometimes memory and peripheral devices.

Multiprocessor systems have three main advantages

- 1- Increase throughput.
- 2- Economy of scale by sharing peripherals, power supply.
- 3- Increased reliability by fault tolerant or graceful degradation.

This ability to continue providing service proportional to the level of surviving hardware is called “graceful degradation” is also called “fault tolerant”.

There are different architectures for multiprocessor systems. Types are dual core, quad core, and arranged in master-slave manner

4- Distributed Systems

A network is a communication path between two or more systems. Distributed systems depend on networking for their functionality. Using communicates, distributed systems are able to share computational tasks, and provide a rich set of set of feature to users.

- ☐ client-server systems
- ☐ peer-to-peer systems

Some operating system benefits from ideas of networking and distributed systems in build network operating system. Types are LAN, MAN, WAN networks.

5- Clustered Systems

Like parallel systems, clustered systems gather together multiple CPUs to accomplish computational work, they composed of two or more individual systems coupled together. The general accepted definition is that clustered computers share storage and is closely linked via LAN networking. Clustering is usually performed to provide high availability.

6- Real-Time Systems

Special purpose operating system, it is used when there are rigid time requirements on the operation of a processor or the flow of data, thus it is often used as a control device in dedicated application. Sensors collect data, computer analyzes and process data.

Such systems: medical imaging, industrial control.

Real time system need that the processing must be done within the defined time constraints or the system will fail.

7- Handheld Systems

Handheld systems include personal digital assistants (PDAs) such as cellular phones. Developers of handheld systems and applications face many challenges (due to the limited size of such devices) such as speed of processor, limited size of memory, and small display screen.

Chapter 2

Computer System Structures

Computer System Operation:

OS is a **resource allocator**: Manages all resources and Decides between conflicting requests for efficient and fair resource use. OS is a **control program**: Controls execution of programs to prevent errors and improper use of the computer. “The one program running at all times on the computer” is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.

Bootting:

It is the operation of bringing operating system kernel from the secondary storage and put it in main storage to execute it in CPU. There is a program bootstrap which is performing this operation when computer is powered up or rebooted.

Bootstrap software: it is an initial program and simple it is stored in read-only memory (ROM) such as firmware or EPROM within the computer hardware.

Jobs of Bootstrap program:

- 1- Initialize all the aspect of the system, from CPU registers to device controllers to memory contents.
- 2- Locate and load the operating system kernel into memory then the operating system starts executing the first process, such as “init” and waits for some event to occur.

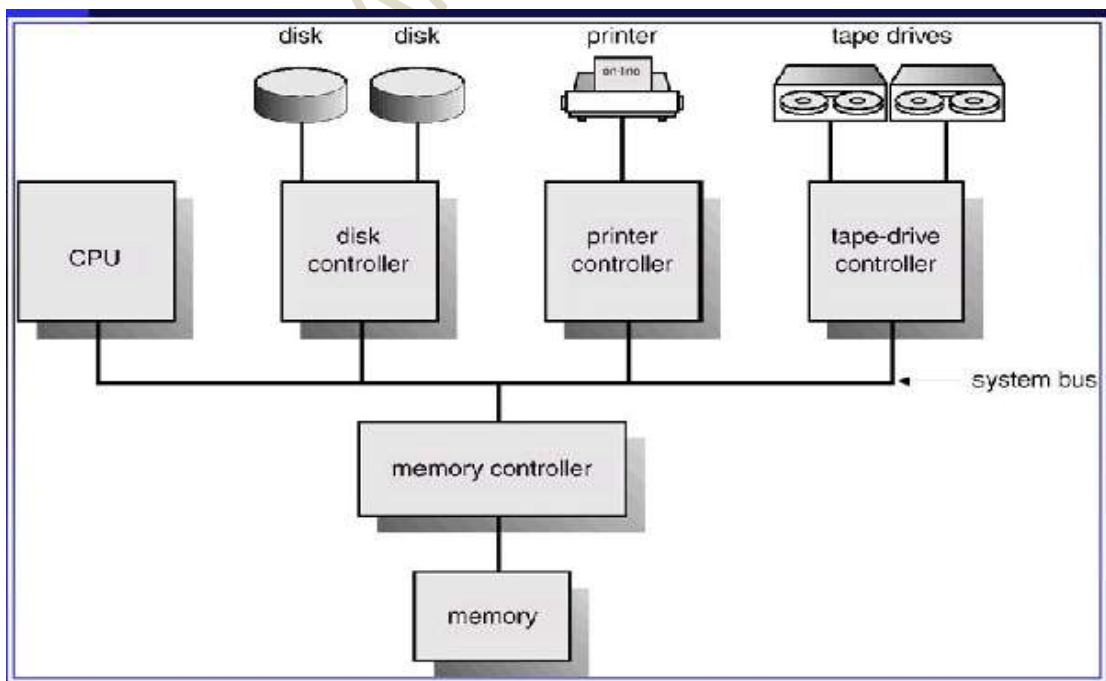
Types of events are either software events (**system call**) or hardware events (signals from the hardware devices to the CPU through the system bus and known as an **interrupt**).

Note: all modern operating system are “interrupt driven”.

I/O structure

A modern, general-purpose computer system consists of CPU and a number of device controllers that connected through a common bus that provides access to shared memory system, CPU other devices can execute concurrently competing for memory cycles.

- 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



Trap (exception): it is a software-generated interrupt caused either by an error (ex: division by zero or invalid memory access) or by a specific request from a user program that an operating system service be performed.

Interrupt vector (IV): it is a fixed locations (an array) in the low memory area (first 100 locations of RAM) of operating system when the interrupt occur the CPU stops what it's doing and transfer execution to a fixed location (IV) contain starting address of the interrupt service routine(ISR), on completion the CPU resumes the interrupted computation.

Interrupt Service Routine: is it a routine provided to be responsible for dealing with the interrupt.

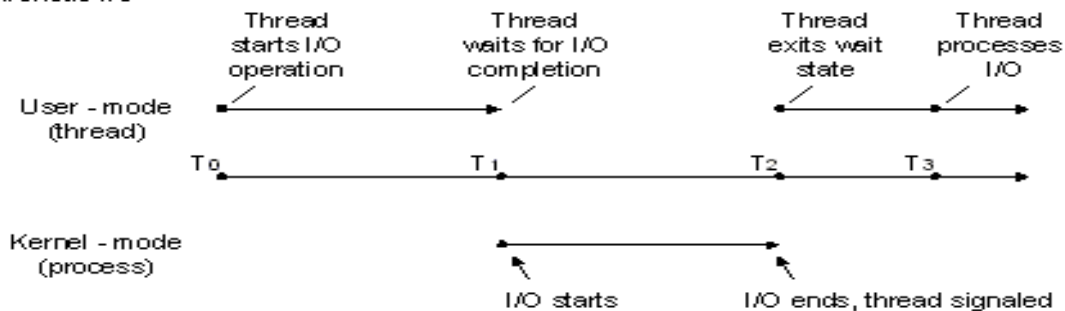
I/O interrupt

To start an I/O operation, it accomplishes this by interrupt. Two courses of action are possible. Synchronous I/O and asynchronous I/O. Asynchronous I/O is also referred to as overlapped I/O.

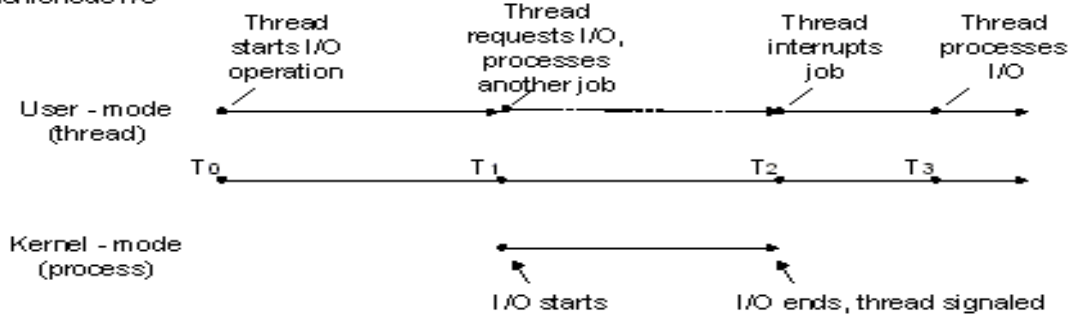
In *synchronous file I/O*, a thread starts an I/O operation and immediately enters a wait state until the I/O request has completed. A thread performing *asynchronous file I/O* sends an I/O request to the kernel by calling an appropriate function. If the request is accepted by the kernel, the calling thread continues processing another job until the kernel signals to the thread that the I/O operation is complete. It then interrupts its current job and processes the data from the I/O operation as necessary.

In situations where an I/O request is expected to take a large amount of time, such as a refresh or backup of a large database or a slow communications link, asynchronous I/O is generally a good way to optimize processing efficiency.

Synchronous I/O



Asynchronous I/O



A **thread** is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

Direct Memory Access 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.

Storage Structure

Main memory – only large storage media that the CPU can access directly Random access

Secondary storage – extension of main memory that provides large nonvolatile storage capacity

Caching

Faster storage (cache) checked first to determine if information is there If it is, information used directly from the cache (fast) If not, data copied to cache and used there .

Chapter 3

Protection

Hardware protection:

Many programming errors are detected by the hardware. These errors are normally handled by the operating system. If a user program fails in executing an illegal instruction or access memory that is not in user address space, then the hardware will trap to the OS. The trap transfers control through interrupt vector to the OS.

A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other program to execute incorrectly.

Dual-Mode Operation:

To ensure proper operation, we must protect the operating system and all other programs and their data from any malfunctioning program.

The approach taken by many operating systems provides hardware support that allows us to differentiate among various modes of execution. We need two modes of operation: user mode and monitor mode.

A **bit**, called the **mode bit** is added to the hardware of the computer to indicates the current mode: monitor (0) or user (1) with mode bit we could distinguish between a task that is executed on behalf of the operating system , and one that is executed on behalf of the user. As system boot time, the hardware starts in monitor mode; the os is then loaded and starts user processes in user mode. Whenever an interrupt

occurs, the hardware switches from user to monitor mode by change the bit mode state to 0.

The dual mode provides protection from **errant users** to access **privileged instructions**.

I/O Operation Protection:

A user program may disrupt the normal operation of the system by issuing illegal I/O instruction. We can use various mechanisms to ensure that such disruption cannot take place in the system.

One of them is by defining all I/O instructions to be privileged instructions. Thus users cannot issue I/O instructions directly. They must do it through the operating system, by executing a system call to request that the operating system performing I/O in its behalf. The operating system, executing in monitor mode, check that the request is valid, and (if the request is valid) does the I/O requested. The operating system then returns to the user.

Memory Protection:

To insure correct operation, we must protect the interrupt vector and interrupt service routine from modification by a user program. This protection must be provided by the hardware, we need the ability to determine the range of legal addresses that the program may access, and to protect the memory outside that space. We could provide the protection by using two registers:

Base register holds the smallest legal physical memory address.

Limit register: contains the size of the range.

This protection is accomplished by the CPU hardware comparing every address generated in user mode with the registers. Any attempt by a program executing in user mode to access monitor memory or other users' memory results in a trap to the monitor, which treats the attempts as a fatal error.

CPU Protection:

In addition to protecting I/O and memory we must insure that the operating system maintains control. We must prevent the user from getting stuck in an infinite loop or not calling system services, and never returning control to the operating system. To accomplish this goal, we can use a **timer**.

Timer can be set to interrupt the computer after a specified period. The period may be **fixed** (for example, 1/60 second) or **variable** (for example, from 1 millisecond to 1 second) A variable timer is generally implemented by a fixed rate clock and a counter.

We can use the timer to prevent a user program from running too long. Simple technique is to initialize a counter with the amount of time that a program is allowed to run.

Amore common use of timer is to implement **time sharing**. In the most case, the timer could be set to interrupt every N millisecond, where N is the **time slice** that each user is allowed to execute before the next user get control of the CPU. The operating system is invoked to perform housekeeping tasks.

This procedure is known as a **context switching**, following a context switch, the next program continues with its execution from the point at which it left off.

Chapter 4

Operating System Structure

An OS may be viewed from several points. One is by examining the services it provides. Second, by looking at the interface that it makes available to users and programmers. Third, by disassembling the system into components and their interconnections.

1. System Components

Many modern computer systems share the goal of supporting the following components:

- **Process management**

A process can be thought of a program in execution. A process needs certain resources to accomplish its task. Also the process has various initialization values.

A process is the unit of work in a system. Such a system consists of a collection of processes, some of which are system processes others are user processes. All processes execute concurrently by multiplexing the CPU among them.

The OS responsible for the following activities in connection with process management:

- Creation and deletion both user and system processes.
- Suspending and resuming processes.
- Providing mechanisms for process synchronization.
- Providing mechanisms for process communication.
- Providing mechanisms for deadlock handling.

▪ **Main Memory Management**

The main memory is the central to the operation of a modern computer system. For a program to be executed it must mapped to absolute addresses and loaded to the MM.

The OS responsible for the following activities in connection with MM management:

- Keeping track of which parts of memory are currently being used and by whom.
- Deciding which processes are to be loaded into memory when memory space become available.
- Allocating and deallocating memory space as needed.

▪ **File Management**

For convenient use of the computer, the OS provides a uniform logical view of information storage. The OS abstracts from the physical properties of its storage device to define the logical storage unit, the file.

A file is a collection of related information defined by its creator. These files are organized in directories to ease their use.

The OS responsible for the following activities in connection with file management:

- Creating and deleting **files**.
- Creating and deleting **directories**.
- Supporting primitives for manipulating files and directories.
- Mapping files onto secondary storage.
- Backing up files on stable storage media.

▪ **I/O System Management**

One of the purposes of OS is to hide the peculiarities of specific hardware devices. The OS responsible for the following activities in connection with I/O system management:

- A memory management component that includes buffering, caching and spooling.
- A general device driver interface.
- Drivers for specific hardware devices.

▪ **Secondary Storage Management**

The computer system must provide secondary storage to back up main memory because that are hold by MM are lost when power is switched of f and the MM is too small to accommodate all data programs. The OS responsible for the following activities in connection with disk management:

- Free space management
- Storage allocation
- Disk scheduling

▪ **Networking**

A distributed system collects physically separate heterogeneous system into a single coherent system, providing the user with the access to various resources that the system maintain. Access to a shared resource allows computation speed up, increase functionality, increase data ability, and enhances reliability.

▪ **Protection System**

Protection is any mechanism for controlling the access programs, processes, or users to the resources defined by the computer system. This mechanism must provide means for specification of the controls to be imposed and means for enforcement. Protection can improve reliability by detecting latent errors at the interfaces between component subsystems.

▪ **Command Interpreter System**

Command Interpreter System is the interface between the user and the OS. Some of these Command Interpreter System are user friendly such as mouse based window and menus. In other shells commands are typed on a keyboard.

2. Operating System Services

An operating system provides an environment for the execution of programs. It provides certain services to programs and to the users of these programs. The specific services provided differ from one operating system to another but we can identify common classes. These services are provided **for the convenience of the programmer**, to make the programming task easier.

1. Program execution
2. I/O operation
3. File system manipulation
4. Communications
5. Error detection
6. Resource allocation

- 7. Accounting.
- 8. Protection

3. System Calls

System calls provide the interface between a process and the operating system. These calls are generally available as assembly language instructions and they are usually listed in the various manuals used by assembly language.

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

4. System Programs

System programs provide a convenient environment for program development and execution. Some of them are simply user interfaces to system calls others are considerably more complex. They can be divided into these categories:

- File management
- Status information
- File modification
- Programming language support
- Program loading and execution
- Communications

System Structure

A system as large and complex as a modern operating system must be engineered carefully if it is to function properly and to be modified easily. There are three different system structures:

- Simple structure
- Layered Approach
- Microkernel

System Design and Implementation

The problems and steps of system design and implementation are as follows:

- Design Goals
- Mechanisms and Policies
- Implementation

CHAPTER 5

Processes

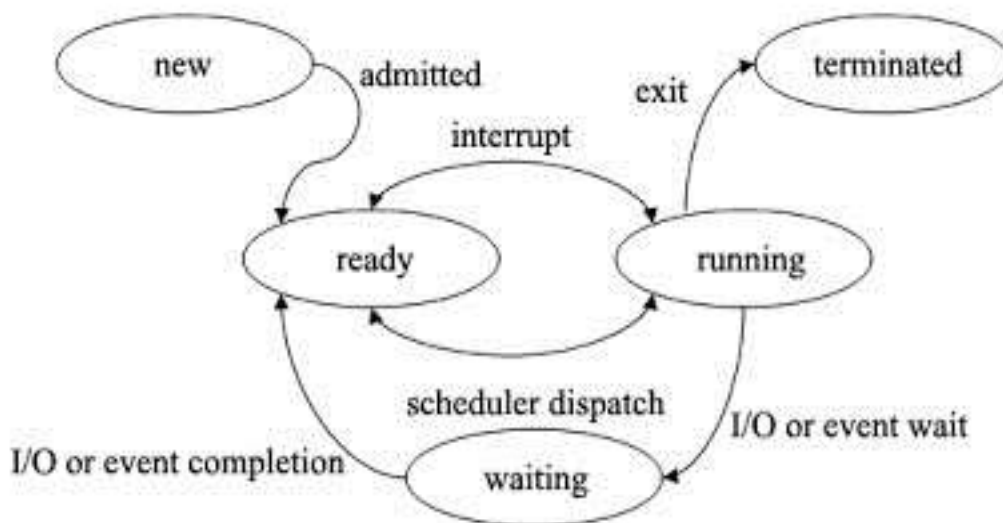
Process Concept

A process is a program in execution. A process is more than the program code, which is sometimes known as the text section. It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers.

Process state

The state of a process is defined in part by the current activity of the process. Each process may be in one of the following states:

- New: the process is being created.
- Running: instructions are being executed.
- Waiting: the process is waiting for some event to occur (I/O or reception signal).
- Ready: the process is waiting to be assigned to a processor.
- Terminated: the process has finished execution.



Process Control Block

Each process is represented in the OS by a process control block (PCB). A PCB contains many pieces of information associated with a specific process, such as:

- Process states: may be new, ready, running, waiting, terminated.
- Program counter: indicates the address of next instruction to execute.
- CPU registers: registers vary in number and type due to computer architecture. They include accumulator, index register, stack pointer, general purpose reg.
- CPU scheduling information: includes process priority, pointers to scheduling queues, and parameters.
- Memory management information: it may be the value of base and limit reg, page or segment tables.
- Accounting information: includes the amount of CPU and real time used, time limits, account numbers.
- I/O status information: list of I/O devices, list of open files.

Pointer	Process state
Process number	
Program counter	
Registers	
Memory limits	
List of open files	
...	

Process Scheduling

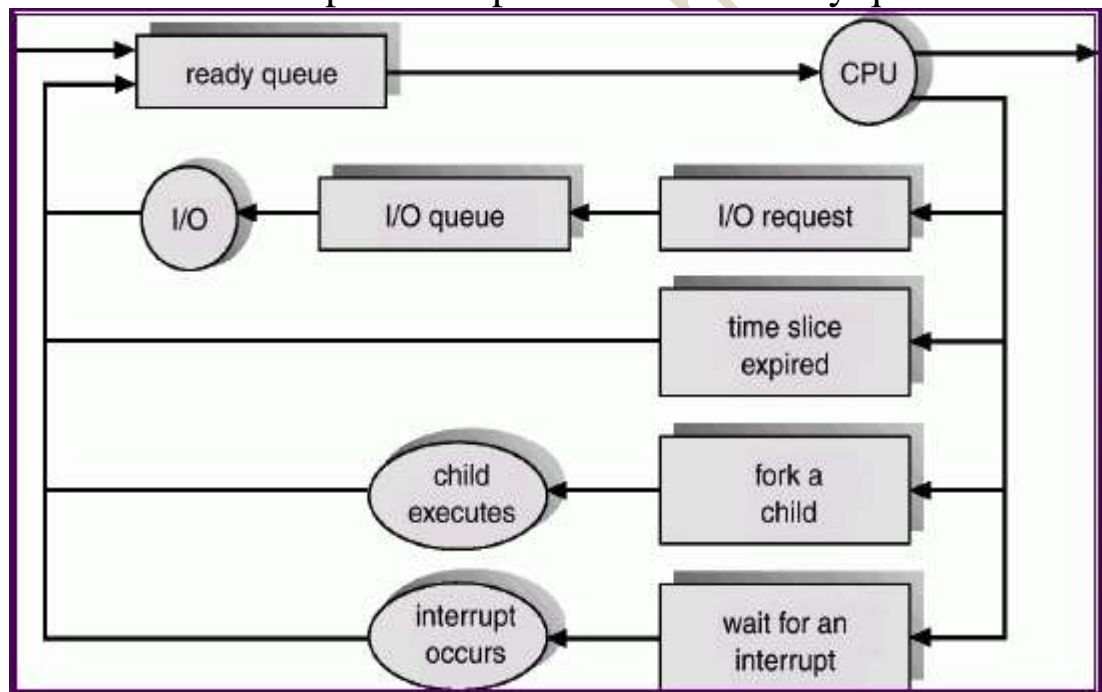
A uniprocessor system can have only one running process. If more processes exist, as in multiprogramming system, there will be only

one process running and the rest must wait until the CPU is free and can be rescheduled.

▪ Scheduling Queues

A new process as enter the system is put in a queue called ready queue. It waits in the ready queue until it is selected for execution. Once the process is assigned to the CPU and it is executing, one of the several event could occur:

- The process could issue an I/O request, and then be placed in an I/O queue.
- The process could create a new sub-process and wait for the termination.
- The process could be removed forcibly from the CPU, as a result of an interrupt and be put back in the ready queue.



Scheduler

A process migrates between the various scheduling queues throughout its lifetime. The operating system must select processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler. There are two types of scheduling algorithms categorized according to the frequency of their execution.

- Long term scheduler (job scheduler) which selects a process from the job pool and load them into the MM.
- Short term scheduler (CPU scheduler) which select a process from the ready queue and allocate it to the CPU.

Context Switch

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the process. This task is known as a context switch.

Operation on Processes

The process in the system can execute concurrently, and they must be created and deleted dynamically.

▪ Process Creation

A process may create several new processes during the course of execution. The creating process is called a parent process, whereas the new processes are called the children.

When a process is created it obtains various resources and initialization values that may be passed along from the parent process to the child process.

▪ Process Termination

A process terminates when it finishes executing its final statement and asks the operating system to delete it. At that point the process may return data to its parent process and the OS deallocate all the physical and logical resources that are previously allocated to that process.

Chapter 8

Memory management

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. A program resides on a disk as a binary executable file. To be executed, the program must be brought into memory and placed within a process. The processes on the disk that are waiting to be brought into memory for execution form the input queue.

We usually want several user processes to reside in memory at the same time. We therefore need to consider how to allocate available memory to the processes that are in the input queue waiting to be brought into memory. In **contiguous** memory allocation, each process is contained in a single section of memory that is contiguous to the section containing the next process.

Logical vs. Physical Address Space

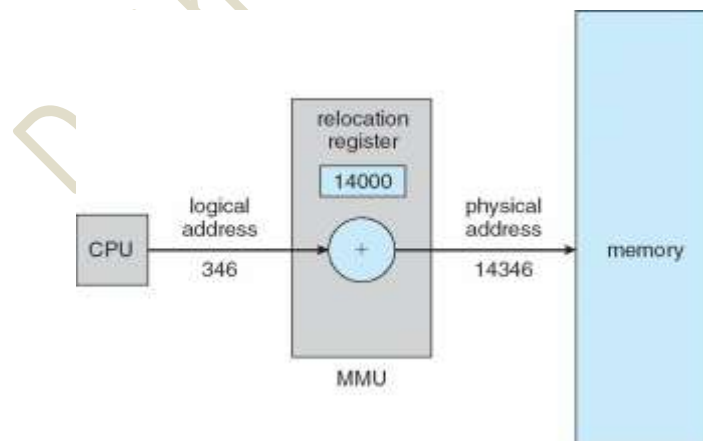
The set of all logical addresses generated by a program is referred to as a **logical address space**. The set of all physical addresses corresponding to these logical addresses is referred to as a **physical address space**.

- Logical address generated by the CPU; also referred to as virtual address
- Physical address: address seen by the memory unit

Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme.

The runtime mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device. MMU uses following mechanism to convert virtual address to physical address:

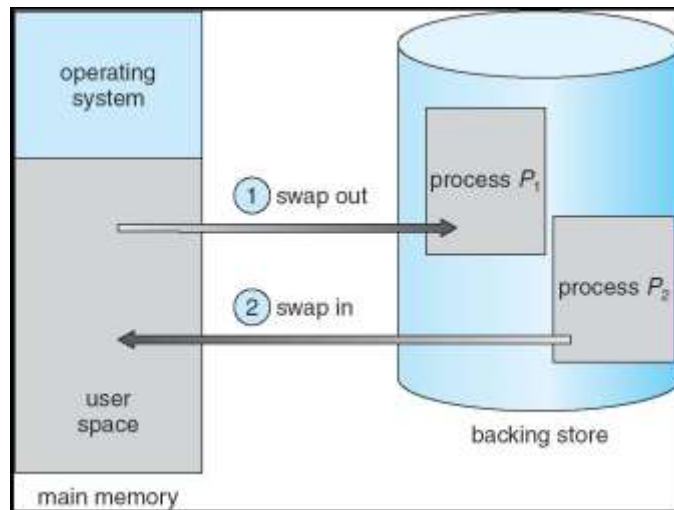
- The value in the base register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
- The user program deals with virtual addresses; it never sees the real physical addresses.



Swapping

Swapping is mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other

processes. At some later time, the system swaps back the process from the secondary storage to main memory. Swapping is also known as a technique for memory compaction.



The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.

Memory Allocation

Main memory usually has two partitions –

- **Low Memory** – Operating system resides in this memory.
- **High Memory** – User processes are held in high memory.

Single partition allocation: One of the simplest methods for allocating memory is to divide memory into several fixed-sized partitions. Each partition may contain exactly one process. Thus, the degree of multiprogramming is bound by the number of partitions. In this multiple partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.

Multiple partition allocation: In the variable-partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.

Initially, all memory is available for user processes and is considered one large block of available memory, a hole. Memory contains a set of holes of various sizes.

This procedure concerns how to satisfy a request of size n from a list of free holes. There are many strategies used to select a free hole from the set of available holes.

- **First fit.** Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or at the location where the previous first-fit search ended. We can stop searching as soon as we find a free hole that is large enough.
- **Best fit.** Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.

- **Worst fit.** Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.

Fragmentation

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation. Fragmentation is of two types: –

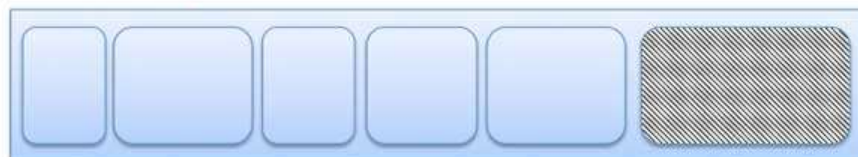
External fragmentation: Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used

Internal fragmentation: Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process

Fragmented memory before compaction



Memory after compaction



External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

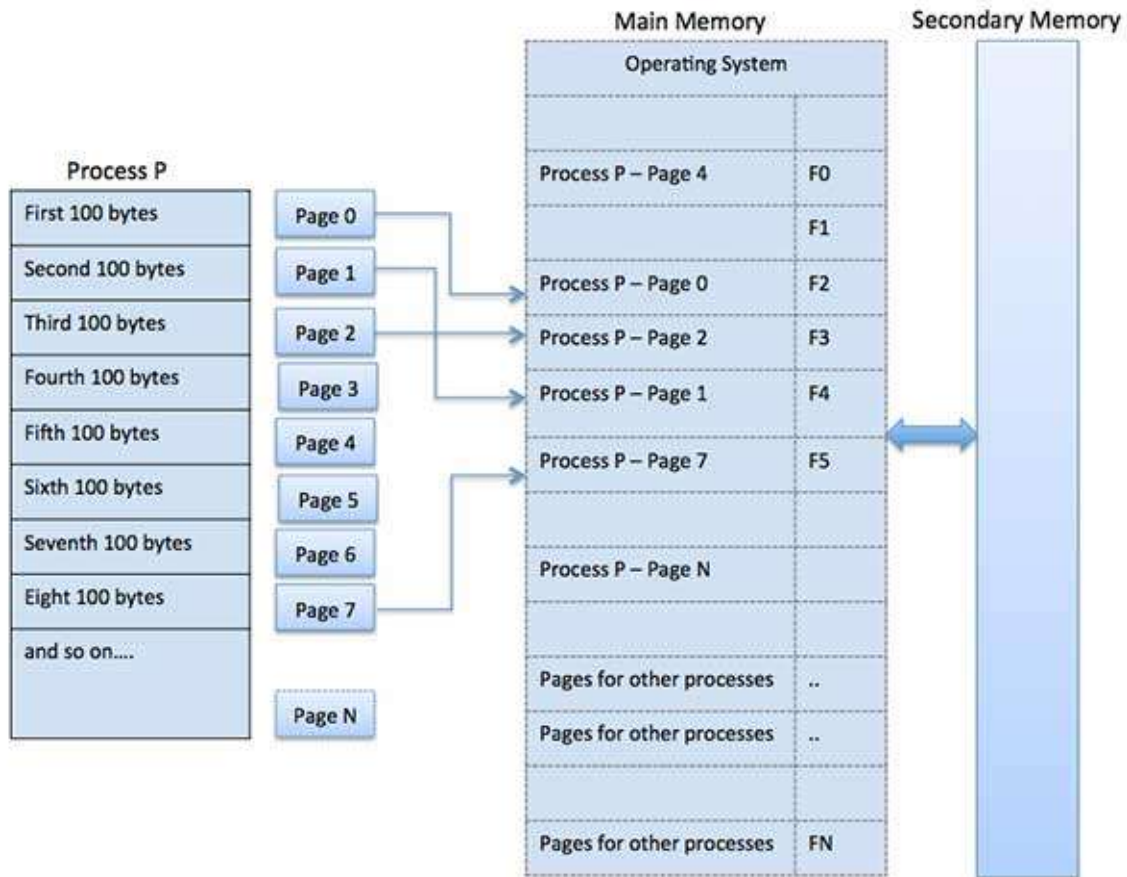
The internal fragmentation can be reduced by effectively assigning the smallest partition but large enough for the process.

Paging

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



Address Translation

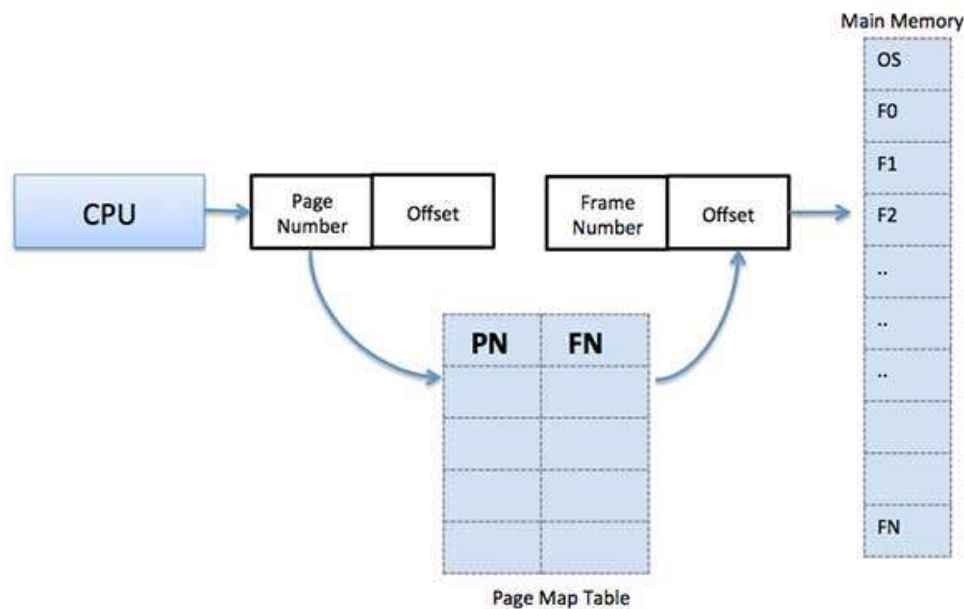
Page address is called logical address and represented by page number and the offset.

$$\text{Logical Address} = \text{Page number} + \text{page offset}$$

Frame address is called physical address and represented by a frame number and the offset.

$$\text{Physical Address} = \text{Frame number} + \text{page offset}$$

A data structure called **page map table** is used to keep track of the relation between a page of a process to a frame in physical memory.



When the system allocates a frame to any page, it translates this logical address into a physical address and creates entry into the page table to be used throughout execution of the program.

When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture. When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program. Here is a list of advantages and disadvantages of paging –

- Paging reduces external fragmentation, but still suffers from internal fragmentation.
- Paging is simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.
- Page table requires extra memory space, so may not be good for a system having small RAM.

Segmentation

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

When a process is to be executed, its corresponding segmentation is loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory. Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the

segment. A reference to a memory location includes a value that identifies a segment and an offset.

