

Lectures in

Operating Systems Concepts

Prepared by Assist . Prof .

Imad Matti

*AL-mamoon University College / Computer Science
Department*

2018

Operating Systems Concepts

Chapter 1

Introduction to O . S

Prepared by Assist . Prof.

Imad Matti

*AL-mamoon University College / Computer Science
Department*

2018

CHAPTER 1

1. Introduction to O/S.

1.1 O/S Definitions:

- 1- An operating system is a software that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.
- 2- O/S is resources manager, where the main resource it manage is computer H/W (processor, storage, I/O devices, communication devices, etc), and data.

1.2 Computer System Components:

- 1- The Hardware (CPU, Memory, I/O devices).
- 2- Operating System (O/S).
- 3- Application Programs (Assembler, Data base, Compilers, Text Editor).
- 4- The user (People, Machines, Other Computers).

1.3 O/S Goals:

- 1- Make computer system convenient to use.
- 2- Use the computer H/W in an efficient manner.

1.4 The O/S Functions:

- 1- Implementing the user interface.
- 2- Scheduling resources among users.
- 3- INPUT/OUTPUT Management.
- 4- Memory Management.
- 5- Process Management.
- 6- Processor Management.
- 7- Recovering from errors.
- 8- Accounting.

1.5 The O/S development history:

1.5.1 The 1940's and 1950's

- The programs were entered on punched cards.
- The first O/S implemented in 1950's, this system ran one job at a time.
- this type of O/S called Batch processing system.

1.5.2 O/S 1960'S

- Running several jobs at once.
- The O/S designers developed the concept of multiprogramming and software engineering.

1.5.3 O/S 1970'S

- Time-sharing system.

- Real-time application.

1.5.4 O/S 1980's

- Decade of PCs and workstation.
- Application software are available such as:
Word processing, database packages, and graphics
- E-mail.
- Client/ server model.

1.5.5 O/S 1990'S and beyond

- Distributed computing.
- Networks.

1.5.6 O/S Categories:

1- Batch System

In this type of O/S, users submit jobs on regular scheduling (daily, weekly, monthly) to a central place where the user of such system did not interact directly with C/S. The programs were entered on punched cards, and run one job at a time.

Advantages : Very simple.

Disadvantages :

- a. There is a delay between the job submission and the job completion (called **turnaround time**).
- b. The CPU is often idle, because the speeds of the mechanical I/O devices are slower than those of electronic devices.

Example: slower CPU executes thousands of instruction per second, while fast card reader read 1200 cards per minute(20 cards per second).

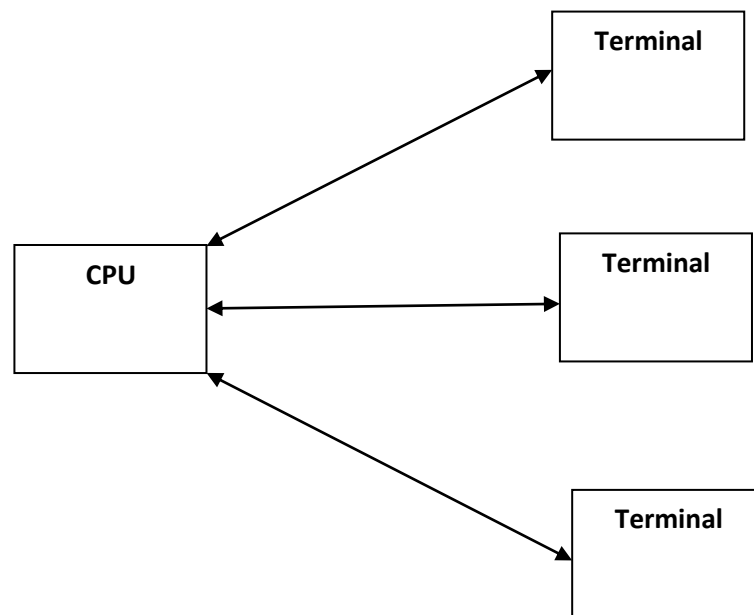
2- Time sharing system (Multi tasking system), (Interactive system).

It is an on-line communication between the user and the system, it allows many users simultaneously share the computer system where little CPU time is needed for each user.

Advantages:

- 1.Reduce the CPU idle time.
2. Minimize Response time.

Disadvantages: More complex, difficult and expensive to build.



Time-sharing system

3- Real-Time Systems:

Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.

Advantages: Critical tasks complete on time.

Example: Airline Reservation system.

4- Parallel systems:

It is multi processor system, where such systems have more than one processor in close communication sharing the computer Bus, the Clock, Memory, and peripheral devices.

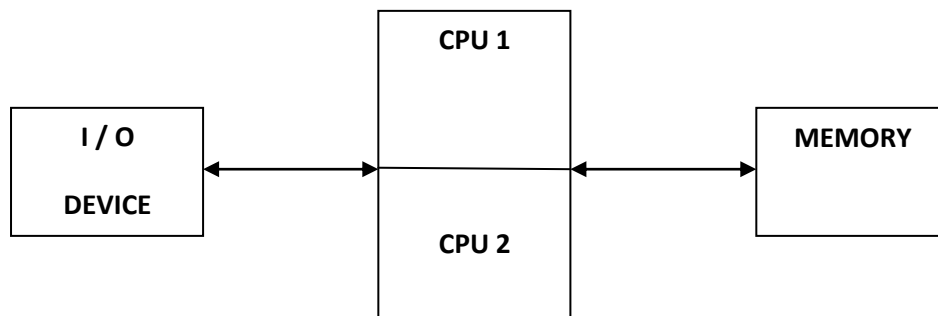
Advantages:

a. Increase Throughput.

(**Throughput:** is the number of jobs completed in unit of time.)

b. Increase reliability.

c. save money.



Parallel system layout

5- Distributed Systems:

It is multi processor system, the processors do not share memory and clock, each processor has its own local memory, the processors communicate with one another through communication lines, such as high speed buses, LAN, WAN.

Advantages:

a. Resource Sharing.

b. Computation speed up.

c. Communications.

d. Reliability.

6- Desktop systems:

Computer system dedicated to a single user.

7- Handheld systems:

- a. Personal Digital Assistants.
- b. Cellular telephones.

Issues:

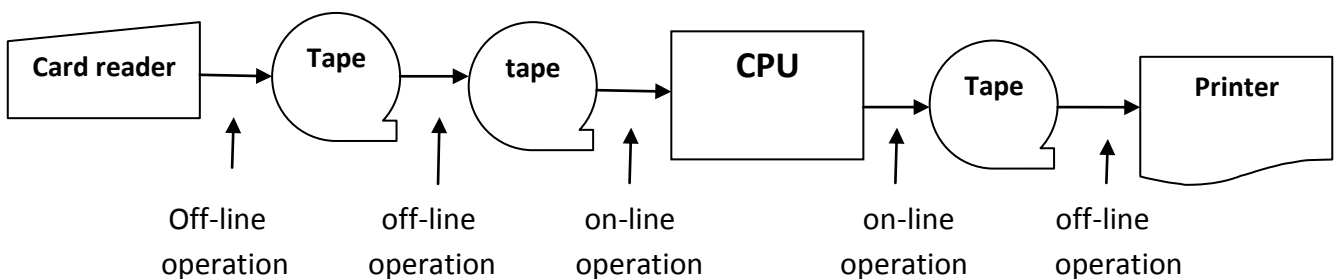
- 1. Limited memory.
- 2. Slow processors.
- 3. Small display screens.

1.7 Performance Development

1.7.1 On-Line and Off-Line operations

1. On-Line Operation: in which they are connected to the processor.

2. Off-Line Operation: in which they are not connected to the central C/S.



1.7.2 Buffering

A buffer : is an area of primary storage for holding data during I/O transfers.

There are two types of buffering:

1. The single buffer. (the CPU would be idle)
2. The double buffer. (the CPU will not be idle)

Advantages: In buffering the CPU and I/O are both busy.

1.7.3 Spooling (Simultaneous Peripheral Operation On- Line)

1. The Spooling Operation uses a disk as a very large buffer for reading and for storing output files.
2. Rather than the cards being read from the card reader directly into memory, the cards are read directly from the card reader onto the disk, the operating system satisfies its requests by reading from the disk, similarly when the job requests the printer to output a line.

Advantages:

Spooling can keep both CPU and the I/O devices working at much higher rates.

1.7.4 Multiprogramming

In multiprogramming system, when a job may have to wait for any reason such as an i/o, the o/s simply switches to and executes another job.

In a non-multiprogramming system (uni-programming), the CPU would sit idle. (**inefficient**)

Advantages:

1. Maximize CPU utilization. (The CPU will never be idle)
2. High and efficient CPU utilization.

CHAPTER 1 QUESTIONS

1. What is an Operating System ?
2. What are the **purpose** (**goals**) of an operating system?
3. In what system the CPU is often **idle** ? explain the reason.
4. In what kind of processing we can keep both the CPU and the I/O devices working at higher rates ? explain.
5. In what kind of system the **CPU will never be idle**? Explain how this system work, with drawing.
6. In what kind of system the **CPU** and **I/O** are both busy.
(e.g. what is the system that allows overlap operation with processing.)
7. Give three another names for Time-Sharing system.

8. What are the differences between **Batch-System** and **Time-Sharing System**?
9. What are the differences between **Parallel Systems** and **Distributed System** ?
10. What is **Real- Time System**? What are its applications?
11. What is the differences between **On-Line** and **Off-Line** operation? Give examples.
12. What is **Throughput**?
13. What is the difference between each two of the following:(وڌاري ۲۰۱۳)
1. Parallel system and Distributed system.
 2. Time-sharing system(Multi-tasking) and Multi-programming.
14. What is the reason for building parallel system?

15. Circle the correct answer to the following questions

1. distributed system is a collection of processers that:

- | | |
|---------------------------|--------------------------------|
| A. Share memory & clock | B. Do not share memory & clock |
| C. Share memory nor clock | D. Share clock nor memory |

2. Programs that do not require interaction or programs with long execution time may served well by:

- | | |
|------------------------|---------------------|
| A. Batch System | B. Real time System |
| C. Time-Sharing System | D. Parallel System |

Operating Systems Concepts

Chapter 2

Computer system operations

Prepared by Assist . Prof .

Imad Matti

*AL-mamoon University College / Computer Science
Department*

2018

CHAPTER 2

2. Computer System Operation

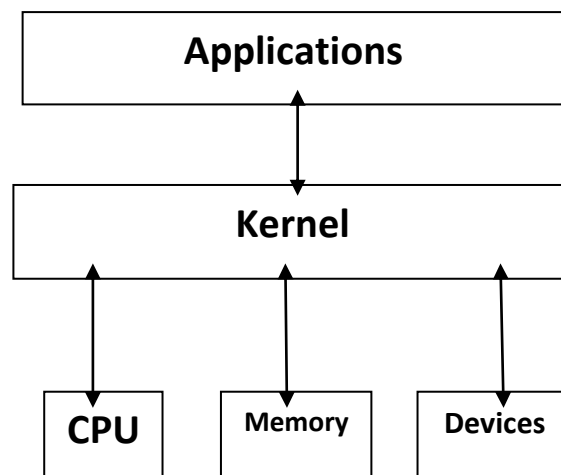
2.1 Bootstrap program: It is an initial program to run, its loaded at power-up or reboot. (stored in **ROM** or **EPROM** known as **firmware**)

2.2 Bootstrap program functions:

- 1- Initialize all aspects of the system from CPU registers to device controllers to memory contents.
- 2- Loads operating system **kernel** and starts execution.

- **Kernel:** Is the part of operating system that mediates access to the computer's resources including:

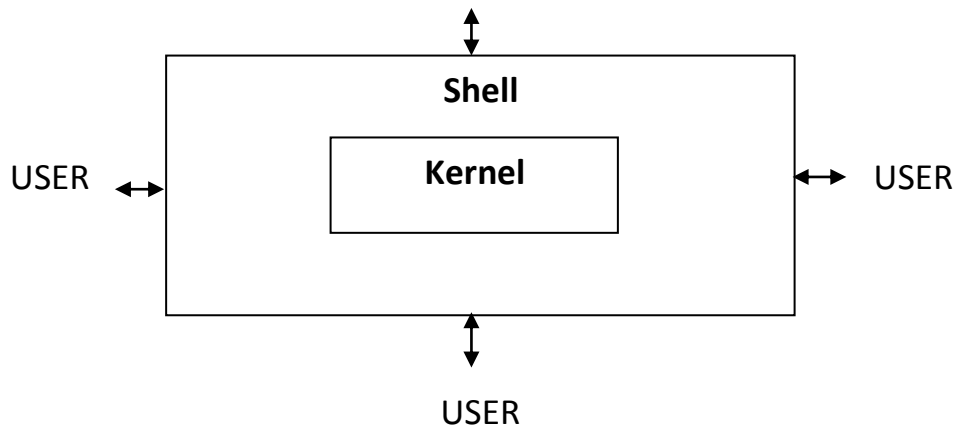
1. The central processing unit (CPU).
2. Random access memory.
3. Input/output (I/O) devices.



A kernel connects the application software to the hardware of a computer

Shell: Is a software that provides an access to operating system services.

USER



2.3 I/O Structure

A **device driver** (or **controller**): is the link between the operating system and the peripheral device.

→ Each I/O device is connected to the C/S through its **device driver**.

→ A device controller maintains local buffer storage and a set of special- purpose registers.

device driver function:

The operating system calls the driver, and the driver drives the device.

2.4 DMA Structure

- 1- Used for high-speed I/O devices.
- 2- Transfers **blocks** of data from buffer storage directly to main memory without CPU intervention.
- 3- Only one interrupt is generated per **block**, rather than one interrupt per **byte**.

2.5 Storage structure

- 1- **Main memory**: large storage media that the CPU can access directly.
- 2- **Secondary Storage**: Extension of main memory that provides:
 - 1- large, and 2- **nonvolatile** storage capacity.
- 3- **Caching**: faster storage system.

2.5.1 Storage systems organized in hierarchy

- 1- Speed
- 2- cost
- 3- capacity

2.5.2 Storage-Device Hierarchy

2.6 Hardware Protection

To **improve** system utilization, the O/S share system resources among several programs simultaneously (Multi programming put several programs in memory at the same time).

This sharing create both **improve utilization** and **increase problems**.

2.6.1 Dual- Mode Operation

To ensure proper operation we must protect the O/S and all programs and their data from malfunctioning program.

Therefore we need two separate modes of operation:

1- User Mode.

2- Monitor Mode (system mode, supervisor mode, kernel mode).

A **bit** called mode bit is added to the H/W to indicate the current mode:

User mode → bit = **1**

Monitor mode → bit = **0**

The mode bit provides ability to distinguish when system is running user code or monitor mode.

2.6.2 I/O Protection

To prevent a user from performing illegal I/O, I/O instructions designed as **privilege** instructions.

- privilege instructions are executed only in monitor mode.

Thus users cannot issue I/O instructions directly, they must do it through the O/S.

2.6.3 Memory Protection

We can provide this protection by using two registers:

1- **Base register** : holds the smallest physical memory address.

2- **Limit register** : contains the size of the range.

2.6.4 CPU Protection

To provide this protection, we must prevent a user program from an infinite loop.

2.7 Operating System Services

- a. program execution.
- b. I/O operations.
- d. Error detection.
- e. Resource allocation.
- f. Accounting.
- g. Protection.

2.8 The User View

There are two methods of providing services:

- a- System calls.
- b- System programs.

2.8.1 System calls

Provides the interface between a running program and O/S.

- a- File manipulation.
- b- Device manipulation.
- c- Communication.

2.8.2 System programs

Solve common problems.

- a- File manipulation.
- b- Programming languages support.
- c- programming loading and execution.
- d- Communication.

2.9 The O/S View

1. O/S is event driven program:

If there are no jobs to execute, no I/O devices to service, and no user to respond, the O/S will sit quiet waiting for something to happen.

2. O/S is interrupt driven:

When an interrupt (or trap) occurs the H/W transfer control to O/S.

CHAPTER 2 QUESTIONS

1. What is Bootstrap program ? What are its functions ? (وزارت ٢٠١٦)
2. Draw the Storage-device Hierarchy? What are the factors that affect the organization in a hierarchy ? (وزارت ٢٠١٦).
3. Explain how the Dual-Mode operation protect the Hardware? (وزارت ٢٠١٦)
4. Explain Memory protection with drawing. (وزارت ٢٠١٦)
5. What is DMA ? What is its function?
6. What is Kernel (nucleus)? Why its ordinary maintained in primary storage?
(وزارت ٢٠٠٢)
7. What is Shell?

Operating Systems Concepts

Chapter 3

Operating system components

Prepared by Assist . Prof .

Imad Matti

*AL-mamoon University College / Computer Science
Department*

2018

CHAPTER 3

3. O/S System Components

3.1 Process Management

- **Process:** is a program in execution.
- **Process:** is a unit of work within the system.
- **Process:** is an active entity.
- **Program:** is a passive entity.
- **System processes:**
 - a- O/S processes.
 - b- User processes.

The O/S is responsible for the following activities in connection with process management:

- 1- Creation and deletion of both user and system processes.
- 2- Suspension and resumption processes.
- 3- Providing mechanisms for process synchronization.
- 4- Providing mechanisms for process communication.
- 5- Providing mechanisms for process deadlocks handling.

3.2 Memory Management

The O/S is responsible for the following activities in connection with memory management:

- 1- Keep track of which parts of memory are currently being used and by whom.

- 2- Decide which processes are to be loaded into memory when space becomes available.
- 3- Allocate and de-allocate memory space as needed.

3.3 File Management

- **File**: is a collection of related information defined by its creator.
- **Files** represent programs and data.

The O/S is responsible for the following activities in connection with file management:

- 1- Creation and deletion files.
- 2- Creation and deletion directories.
- 3- Mapping files onto secondary storage.
- 4- backup files on stable (nonvolatile) storage media.

3.4 I/O System Management

I/O Subsystem consists of:

- a- Memory management include buffering, catching and spooling.
- b- General device driver interface.
- c- Driver for hardware devices.

3.5 Secondary Storage Management.

The O/S is responsible for the following activities in connection with secondary storage management:

- a- Free Space management.
- b- Storage Allocation.
- c- Disk scheduling.

3.6 Networking

Collection of processes, each process has its local memory and clock, the processors communicates with one another through communication lines, such as high speed buses or telephone lines.

3.7 Protection

Controlling the access of program, processes.

3.8 Command Interpreter System.

Interface between the user and the O/S.

System Structure

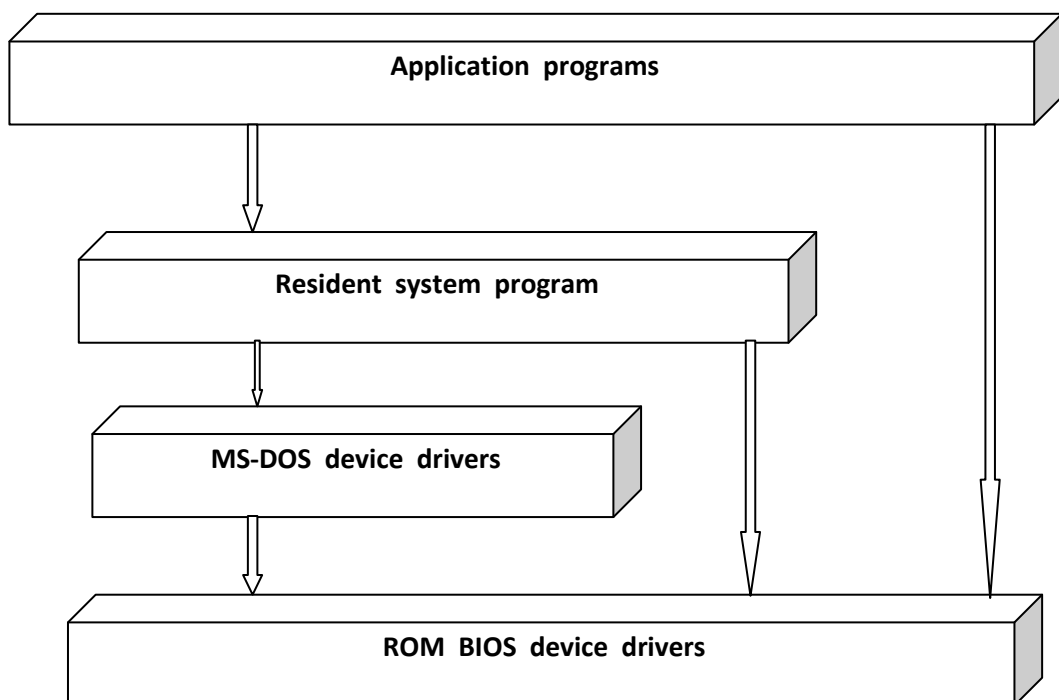
There are two approaches for the O/S structure:

2.1 Simple Structure

Small, simple, and limited systems.

The interfaces and levels of functionality are not well separated.

Example: MS-DOS.



2.2 Layered Approach

Consists of breaking the O/S into number of Layers (levels), each built on top of lower layers. The bottom layer (layer 0) is the H/W, the highest (layer N) is the user interface.

Layer 5:	user program
Layer 4:	buffering for input and output devices
Layer 3:	operator-console device driver
Layer 2:	memory management
Layer 1:	CPU scheduling
Layer 0:	Hardware

The layer structure

Advantages:

- a- modularity: The layers are selected such that each uses functions (operations) and services of only lower-level layers.
- b- simplifies debugging and system verification: The first layer can be debugged without any concern for the rest of the system.

CHAPTER 3 QUESTIONS

1. State five activities of File- management. (وزارې ٢٠١٦)
- 2.

Operating Systems Concepts

Chapter 4

Process management

Prepared by Assist . Prof.

Imad Matti

*AL-mamoon University College / Computer Science
Department*

2018

CHAPTER 4

4. Process Management

A computer system consists of a collection of processes:

- 1- O/S processes: execute system code.
- 2- User processes: execute user code.

4.1 Process Concept

A **Process**: is a program in execution.

Or **process**: is a unit of work.

Or **process**: is an active entity.

Program	Process
It is a passive entity	It is an active entity.
Stored in disk (i.e. file).	Stored in memory.
It is a sequence of instructions (static).	Sequence of actions (dynamic).

4.2 Process State

- 1- **New**: The process is being created.
- 2- **Ready**: The process is waiting to be assigned to a processor.
- 3- **Running**: instructions are being executed.
- 4- **Waiting**: The process is waiting for some event to occur.
- 5- **Terminated**: The process has finished execution.

4.3 PROCESS CONTROL BLOCK (PCB)

PCB: is a data structure containing all the necessary information for representing a process in the system.

It contains many pieces of information such as:

- 1- **Process Identifier:** ID number that identifies the process.
- 2- **Process state:** new, ready, running, waiting, or terminated.
- 3- **Program counter:** contains the address of the next instruction to be executed.
- 4- **CPU registers:** index registers, stack pointers, and general purpose registers.
- 5- **CPU scheduling:** process priority, and any other scheduling parameters.
- 6- **Memory management information:** value of base and limit registers.
- 7- **Accounting information:** amount of CPU and real time.
- 8- **I/O status information:** list of I/O devices, list of open files.

4.4 Process Scheduling

4.4.1 Scheduling: is a task by which the operating system decides to introduce new processes into the system.

4.4.2 Scheduling aims:

Maximize → CPU utilization, throughput.

Minimize → Response time, waiting time, and turnaround time.

4.4.3 Scheduling Criteria:

- a. CPU utilization: the percentage of the time CPU doing useful work to the total elapsed time.
- b. Throughput: is the total number of processes that complete their execution per unit of work.
- c. Turnaround time: is the total time between submission of a process and its completion.
- d. Waiting time: is the time the process remains in the ready queue.
- e. Response time: is the time from the submission of a request until the first response is produced.
- f. Balance: keep all parts of the system busy.

Processes can be described as :

1- **I/O-bound process**: spends more time doing I/O than computations.

2- **CPU-bound process**: spends more time doing computations.

4.5 Scheduling Levels

There are three levels (terms) of scheduling:(there are three types of schedulers)

4.5.1 Long -Term Scheduler: (or **job scheduler**) selects which processes should be brought from secondary storage devices (e.g. disk) into memory for execution.

L.T.S control the degree of multi programming.

L.T.S select a good process mix of I/O-bound process and CPU-bound process.

4.5.2 Short-Term Scheduler: (or CPU scheduler) selects which ready processes should be executed next and allocates CPU to it.

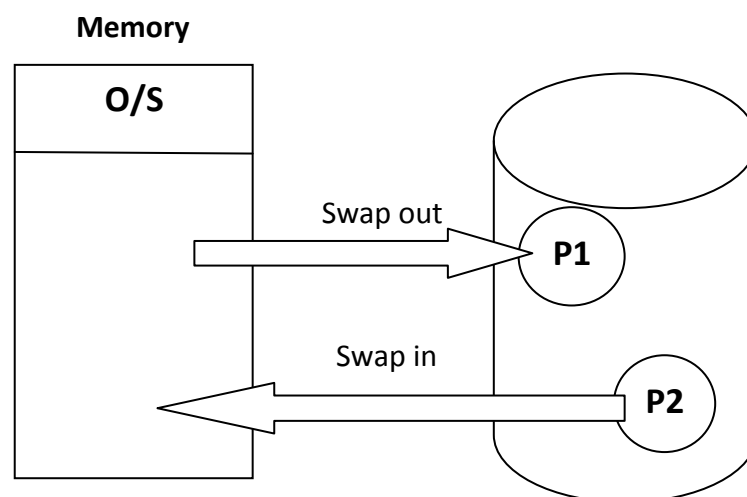
S.T.S select a good process mix of I/O-bound process and CPU-bound process.

4.5.3 Medium Term Scheduler: it removes processes from the memory, it reduces the degree of multiprogramming.

4.6 Context Switch: Switching the CPU to another process by saving the state of the old process and loading the saved state for the new process.

Disadvantages: **Context-switch time is Pure overhead** , because the system does no useful work while switching.

4.6.1 Swapping: removing a process from memory for some reason and later it can be reloaded into memory.



A process can be swapped out of memory to a backing store and then brought back into memory for continued execution.

Chapter 4 Questions

Q1: Fill in the blanks the following statements with **MAXIMIZE** or **MINIMIZE**:

- 1-The objective of the O/S is to ----- response time.
- 2- The objective of the O/S is to ----- waiting time.
- 3 The objective of the O/S is to ----- throughput.
- 4- the objective of the O/S is to ----- turnaround time.
- 5- the objective of multi programming is to ----- CPU utilization.

Operating Systems Concepts

Chapter 5

Interrupt processing

Prepared by Assist . Prof.

Imad Matti

*AL-mamoon University College / Computer Science
Department*

2018

Chapter 5

5. Interrupt Processing

5.1 Interrupt: is a signal to the processor indicating that an event was occurred and needs immediate attention.

→ The interrupt is generated by the H/W of O/S.

→ Each **interrupt** has its own **interrupt service routine**.

(interrupt handler routine)

→ An operating system is interrupt driven.

5.2 Interrupt vector: contains the addresses of all service routines.

5.3 There are two types of interrupts:

5.3.1 Hardware generated interrupt.

Examples: 1. Timer.

2. Pressing a key on the keyboard.

3. Moving the mouse.

5.3.2 Software generated interrupt (trap or exception).

Examples: 1. Division by zero.

2. System calls.

3. Access to a bad memory address.

5.4 Interrupt Classes (types)

There are six interrupt classes:

5.4.1 SVC (Supervisor call) interrupts

- I/O request.
- obtaining more storage.

5.4.2 I/O interrupts

- I/O operation completes.
- I/O error.

5.4.3 External interrupts

5.4.4 Restart interrupts

- pressing the console restart bottom.

5.4.5 Program Check Interrupts

- Divide by zero.
- Arithmetic overflow.

5.4.6 Machine check interrupts

5.5 Preemptive and non Preemptive Scheduling

- **Preemptive scheduling:** the CPU can be taken away from the process.
- **Non preemptive scheduling:** the CPU cannot be taken away from the process.

5.6 Scheduling Algorithms

5.6.1 First- Come, First- Served (FCFS)

The process that requests the CPU first is allocated the CPU first.

Advantages: Simple and easy to understand.

Disadvantages: 1- The average waiting time is quite long.

2- Not useful for time-sharing system.

- FCFS is not optimal because the average waiting time is quite long.

- FCFS is not useful for time-sharing system because once the CPU is allocated to a process, that process keeps the CPU until it releases the CPU.

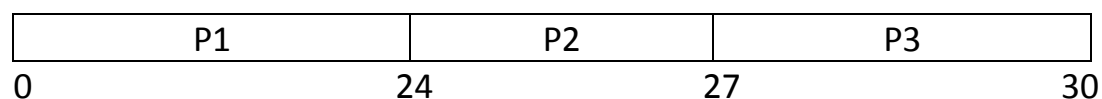
NOTE: FCFS is Non-Preemptive scheduling Algorithm.

Example 1: consider the following set of processes that arrive at time 0, with the length of the CPU burst time given in milliseconds:

Process	Burst time
P1	24
P2	3
P3	3

1. Draw Gantt Chart.
2. Find average waiting time.
3. Find average turnaround time.

The Gantt Chart is:



Waiting time = Start of execution time – arrival time.

Waiting time for P1 = (0 – 0) = 0 ms

Waiting time for P2 = (24 – 0) = 24 ms

Waiting time for P3 = (27 – 0) = 27 ms

Average waiting time: (0 + 24 + 27) / 3 = 17 ms

Turnaround time = waiting time + burst time.

Turnaround time for P1 = $0 + 24 = 24$ ms

Turnaround time for P1 = $24 + 3 = 27$ ms

Turnaround time for P1 = $27 + 3 = 30$ ms

Average turnaround time = $(24 + 27 + 30)/3 = 27$ ms

Example 2: Consider the following snapshot (table):

Process	Arrival time	Execution time
P0	0	5
P1	1	3
P2	2	8
P3	3	6

The Gantt Chart:

P0					P1					P2					P3									
0					5					8					16					22				

Waiting time for P0 = $(0 - 0) = 0$ ms

Waiting time for P1 = $(5 - 1) = 4$ ms

Waiting time for P2 = $(8 - 2) = 6$ ms

Waiting time for P3 = $(16 - 3) = 13$ ms

Average waiting time = $(0 + 4 + 6 + 13)/4 = 5.55$ ms

Example 3: Suppose 4 processes P1, P2, P3, P4 arrive to C/S at time (1, 2, 3, 4) ms respectively, their execution time (burst time) are (5, 3, 8, 6) ms respectively. Use FCFS scheduling, draw Gantt Chart and find average waiting time.

Gantt Chart:

idle	P1	P2	P3	P4	
0	1	6	9	17	23

Waiting time for P1 = $(1 - 1) = 0$ ms

Waiting time for P2 = $(6 - 2) = 4$ ms

Waiting time for P3 = $(9 - 3) = 6$ ms

Waiting time for P4 = (17 – 4) = 13 ms

Average waiting time = (0 + 4 + 6 + 13)/4 = 5.55 ms

Example 4: Suppose 4 processes P1, P2, P3, P4 arrive to C/S at time (6, 8, 5, 2) ms respectively, their execution time (burst time) are (7, 5, 6, 2) ms respectively. Use FCFS scheduling, draw Gantt Chart and find average waiting time.

Gantt Chart:

Idle	idle	P4	idle	P3	P1	P2	
0	1	2	4	5	11	18	23

Waiting time for P4 = (2 – 2) = 0 ms

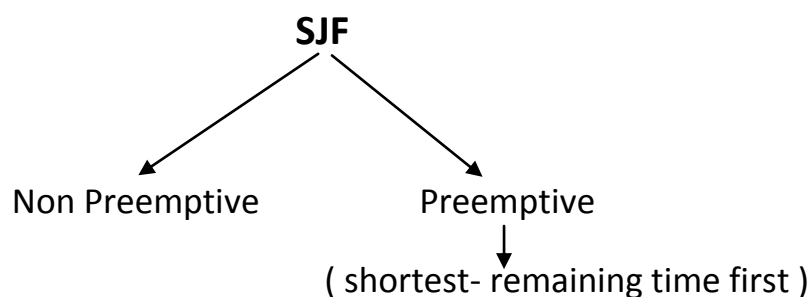
Waiting time for P3 = (5 – 5) = 0 ms

Waiting time for P1 = (11 – 6) = 5 ms

Waiting time for P2 = (18 – 8) = 10 ms

Average waiting time = (0 + 0 + 5 + 10)/4 = 3.75 ms

5.6.2 Shortest-Job-First Scheduling (SJF)



Advantages: SJF is optimal, because it gives the minimum A.W.T
(e.g. minimize waiting time)

Disadvantages:

- 1- The need to Know the length of the next CPU burst.
- 2- Not suitable for **interactive system** and **short-term scheduling**,
Because there is no way to know the length of the next CPU burst.
- 3- Impossible to implement.

NOTE: SJF is used frequently in **long-term scheduling**.

Example 5: consider the following set of processes that arrive at time 0, with the length of the CPU burst time given in milliseconds:

Process	Burst time
P1	6
P2	8
P3	7
P4	3

Use SJF scheduling (non-preemptive):

1. Draw Gantt Chart.
2. Find average waiting time.

Gantt Chart:

P4	P1	P3	P2	
0	3	9	16	24

Waiting time for P4 = $(0 - 0) = 0$ ms

Waiting time for P1 = $(3 - 0) = 3$ ms

Waiting time for P3 = $(9 - 0) = 9$ ms

Waiting time for P2 = $(16 - 0) = 16$ ms

Average waiting time = $(0 + 3 + 9 + 16) / 4 = 7$ ms

Example 6: Consider the following snapshot (table):

Process	Arrival time	Burst time
P0	0	5
P1	1	3
P2	2	8
P3	3	6

Use non-preemptive SJF scheduling:

1. Draw Gantt Chart.
2. Find average waiting time.

The Gantt Chart:

P0	P1	P3	P2
0	5	8	14
			22

Waiting time for P0 = $(0 - 0) = 0$ ms

Waiting time for P1 = $(5 - 1) = 4$ ms

Waiting time for P3 = $(8 - 3) = 5$ ms

Waiting time for P2 = $(14 - 2) = 12$ ms

Average waiting time = $(0 + 4 + 5 + 12)/4 = 5.25$ ms

Example 7: Consider the following snapshot (table):

Process	Arrival time	Burst time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Use **preemptive** SJF scheduling:

1. Draw Gantt Chart.
2. Find average waiting time.

The Gantt Chart:

P1	P2	P4	P1	P3
0	1	5	10	17
				26

Waiting time for P1 = $(0 - 0) + (10 - 1) = 9$ ms

Waiting time for P2 = $(1 - 1) = 0$ ms

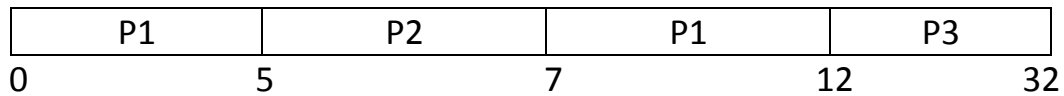
Waiting time for P4 = $(5 - 3) = 2$ ms

Waiting time for P3 = $(17 - 2) = 15$ ms

Average waiting time = $(9 + 0 + 2 + 15)/4 = 26/4 = 6.5$ ms

Example 8: Suppose 3 processes P1, P2, P3 arrive to C/S at time (0, 5, 2) ms respectively, their execution time (burst time) are (10, 2, 20) ms respectively. Use preemptive SJF scheduling, draw Gantt Chart and find average waiting time.

The Gantt Chart:



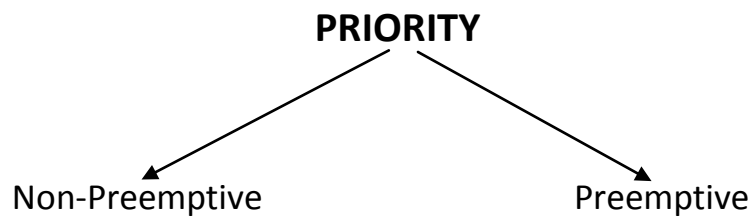
Waiting time for P0 = (0 – 0) + (7- 5) = 2 ms

Waiting time for P2 = (5 – 5) = 0 ms

Waiting time for P3 = (12 – 2) = 10 ms

Average waiting time = (2 + 0 + 10)/3 = 12/3 = 4 ms

5.6.3 Priority Scheduling



- A priority is an integer number associated with each process.
- SJF is a special case of the priority scheduling alg.
- Equal priority processes are scheduled in FCFS.
- Low numbers represent low priority, others use low numbers for high priority.
- We assume low numbers represent high priority.
- The CPU is allocated to the process with the **highest priority**.

Disadvantages: Starvation or indefinite blocking (major problem).

Starvation: a process that is ready to run but lacking CPU.

The solution to this problem is **aging**.

Aging: is a technique of gradually increase the priority of processes that wait in the system for a long time.

Example 9: consider the following set of processes that arrive at time 0, with the length of the CPU burst time given in milliseconds, Consider also low number is high priority.

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Use **non preemptive priority** scheduling:

1. Draw Gantt Chart.

2. Find average waiting time.

The Gantt Chart:

P2	P5	P1			P3	P4
0	1	6	16	18	19	

Waiting time for P2 = $(0 - 0) = 0$ ms

Waiting time for P5 = $(1 - 0) = 1$ ms

Waiting time for P1 = $(6 - 0) = 6$ ms

Waiting time for P3 = $(16 - 0) = 16$ ms

Waiting time for P4 = $(18 - 0) = 18$ ms

Average waiting time = $(0 + 1 + 6 + 16 + 18) / 5 = 8.2$ ms

Example 10: Consider the following snapshot (table):

Process	Arrival time	Burst time	Priority
P1	1	4	4
P2	2	3	3
P3	3	3	1
P4	4	5	2

Consider also low number is high priority.

Use **non preemptive priority** scheduling:

1. Draw Gantt Chart.
2. Find average waiting time.

The Gantt Chart:

Idle	P1	P3	P4	P2	
0	1	5	8	13	16

Waiting time for P1 = $(1 - 1) = 0$ ms

Waiting time for P3 = $(5 - 3) = 2$ ms

Waiting time for P4 = $(8 - 4) = 4$ ms

Waiting time for P2 = $(13 - 2) = 11$ ms

Average waiting time = $(0 + 2 + 4 + 11) / 4 = 17 / 4 = 4.25$ ms

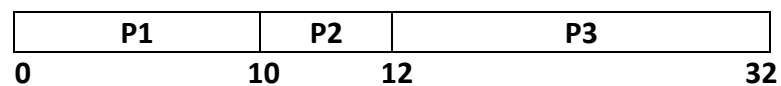
Example 11: consider the following snapshot:

Process	Arrival time	Burst time	Priority
P1	0	10	12
P2	5	2	0
P3	2	20	2

Use **non preemptive highest priority first** scheduling:

1. Draw Gantt Chart only.

The Gantt Chart is:



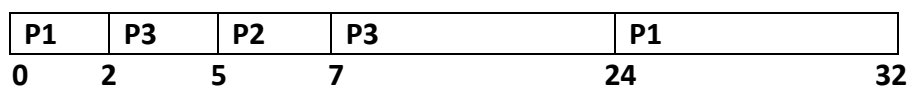
Example 12: consider the following snapshot:

Process	Arrival time	Burst time	Priority
P1	0	10	12
P2	5	2	0
P3	2	20	2

Use **preemptive highest priority first** scheduling:

1. Draw Gantt Chart.
2. Find average waiting time.

The Gantt Chart is:



Waiting time for P1 = $(0 - 0) + (24 - 2) = 22$ ms

Waiting time for P3 = $(2 - 2) + (7 - 5) = 2$ ms

Waiting time for P2 = $(5 - 5) = 0$ ms

Average waiting time = $(22 + 2 + 0)/3 = 24/3 = 8$ ms

5.6.4 Round-Robin Scheduling (RR)

- RR is designed for time-sharing systems.
- RR is similar to FCFS but RR is preemptive.
- The ready queue is circular.
- Each processor gets a small unit of CPU time (time quantum).

Disadvantages:

- The average waiting time is quite long.
- If time quantum (T.Q) is large → FIFO queue.
- If T.Q is small → context switch overhead increase.

Example 13: consider the following set of processes that arrive at time 0, with the length of the CPU burst time given in milliseconds:

Process	Burst time
P1	53
P2	17
P3	68
P4	24

Use RR scheduling with time quantum (T.Q) = 20 :

1. Draw Gantt Chart.
2. Find average waiting time.

The Gantt Chart is:

P1	P2	P3	P4	P1	P3	P4	P1	P3	P3	
0	20	37	57	77	97	117	121	134	154	162

Waiting time for P1 = (0 - 0) + (77 - 20) + (121 - 97) = 81 ms

Waiting time for P2 = (20 - 0) = 20 ms

Waiting time for P3 = (37 - 0) + (97 - 57) + (134 - 117) = 94 ms

Waiting time for P4 = $(57 - 0) + (117 - 77) = 97$ ms

Average waiting time = $(81 + 20 + 94 + 97)/4 = 292/4 = 73$ ms

Example 14: Consider the following snapshot:

Process	Arrival time	Burst time
P0	0	5
P1	1	3
P2	2	8
P3	3	6

Use **RR** Scheduling with T.Q = 3 ms

1. Draw Gantt Chart.
2. Find average waiting time.

Gantt Chart:

P0	P1	P2	P3	P0	P2	P3	P2	
0	3	6	9	12	15	18	21	24

Waiting time for P0 = $(0 - 0) + (12 - 3) = 9$ ms

Waiting time for P1 = $(3 - 1) = 2$ ms

Waiting time for P2 = $(6 - 2) + (15 - 9) = 10$ ms

Waiting time for P3 = $(9 - 3) + (18 - 12) = 12$ ms

Average waiting time = $(9 + 2 + 10 + 12)/4 = 8.25$ ms

5.6.5 Multilevel Feedback Queue Scheduling

-- Three queues:

Q0 – RR with time quantum 8 milliseconds

Q1 – RR with time quantum 16 milliseconds

Q2 – FCFS

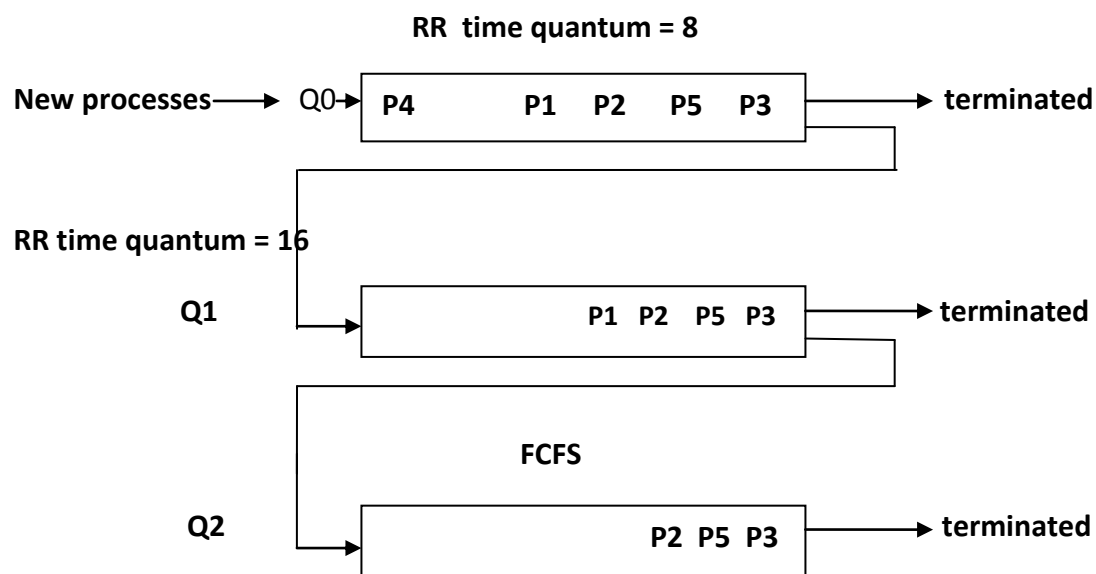
- A new job enters queue Q0 receives 8 ms, if it does not finish in 8 ms, job is moved to Q1.
- At Q1 job receives 16 additional ms, if it still does not complete, it is moved to Q2.

Example 15:

Consider a MLFBQ with three queues numbered from 0 – 2, where queue 0 is RR with T.Q = 8 ms, queue 1 is RR with T.Q = 16 ms, and FCFS for queue 2. Assume we have the following five processes their arrival time and burst time as indicated in the table below. Draw the Gantt Chart to show the execution of each process and calculate the A.W.T and A.T.A.T.

Process	Arrival time	Burst time
P1	10	20
P2	5	30
P3	0	15
P4	35	7
P5	3	28

The solution:



The Gantt Chart:

P3	P5	P2	P1	P3	P4	P5	P2	P1	P3	P5	P2
0	8	16	24	32	35	42	58	74	86	90	100

$$W.T(P1) = (24 - 10) + (74 - 32) = 14 + 42 = 56 \text{ ms}$$

$$W.T(P2) = (16 - 5) + (58 - 42) + (94 - 74) = 11 + 16 + 20 = 47 \text{ ms}$$

$$W.T(P3) = (32 - 8) + (86 - 35) = 24 + 51 = 75 \text{ ms}$$

$$W.T(P4) = (35 - 35) = 0 \text{ ms}$$

$$W.T(P5) = (8 - 3) + (42 - 16) + (90 - 58) = 5 + 26 + 32 = 63 \text{ ms}$$

$$A.W.T = (56 + 47 + 75 + 0 + 63) / 5 = 48.2 \text{ ms}$$

Chapter 5 Questions

Q1: Consider the following set of processes:

Process	Arrival time	CPU burst time	Priority
P1	2	27	3
P2	10	22	4
P3	15	5	2
P4	20	14	2
P5	30	12	1

A. Draw Gantt Chart for all processes using FCFS, SJF, SRJF, RR (time quantum = 10)

Note: small priority no. = high priority.

B. Compute average waiting time and turn-around time for each algorithm in (A).

Q2: Consider the following table:

Process	A . T	B . T
P1	2	53
P2	4	17
P3	1	68
P4	3	24

1-Use R.R with T.Q=20 , Draw Gantt Chart only.

2-When the round robin (RR) becomes FCFS? Give example.

Q3: Consider the following table:

Process	Arrival time	Burst time
P1	10	20
P2	5	30
P3	0	15
P4	38	7
P5	3	28

Use round robin (RR) scheduling with T.Q= 12 Draw Gantt Chart only.

Q4: Consider the following snapshot(table):-

Process	Arrival time	Burst time	Priority
P1	0	4	4
P2	1	3	3
P3	2	3	1
P4	3	5	2

1- Use non-preemptive HPF scheduling , draw only Gantt Chart.

2- Use preemptive HPF scheduling , draw only Gantt Chart.

Q5: Consider the following snapshot (table):-

Process	Arrival Time	Burst Time
P1	0	3
P2	1	6
P3	4	4
P4	6	2

1-Use non-preemptive SJF scheduling , Draw only Gantt Chart.

2-Use preemptive SJF scheduling , Draw only Gantt Chart.

Operating Systems Concepts

Chapter 6

deadlocks

Prepared by Assist . Prof.

Imad Matti

AL-mamoon University College / Computer Science

Department

2018

Chapter 6

6. Deadlocks

Deadlock: Two or more processes are unable to proceed because each process is waiting indefinitely for one of the others to do something.

6.1 Resources states:

- 1. Busy:** being used by another process.
- 2. Free:** Ready to be used by any process.
- 3. Broken:** Cannot be used by any process.

6.2 Resources types:

CPU, memory, I/O devices, Files.

6.3 Resources instances:

If the system has two CPU, then the resource type CPU has two instances.

If the system has five printers then the resource type printer has five instances.

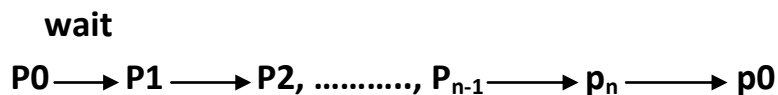
6.4 A process may utilize a resource in only the following sequence:

- 1. Request:** A process must request a resource before using it.
- 2. Use:** The resource is allocated to the process.
- 3. Release:** The process releases the resource.

6.5 Deadlock Necessary Conditions

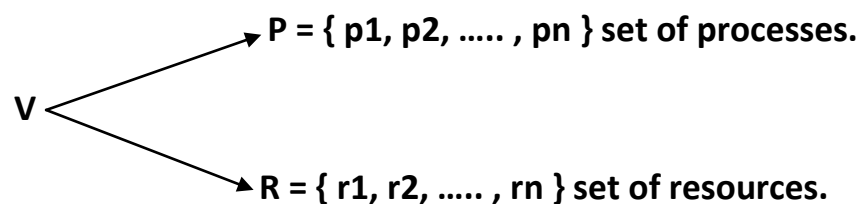
- 1. Mutual exclusion:** There exist at least one non-sharable resource that is held by a process.

2. Hold and wait: a process holding at least one resource and waiting to additional resources.
3. No preemption: Resources cannot be preempted.
4. Circular wait: There must exist a set $\{p_0, p_1, \dots, p_n\}$ of waiting processes such that:



6.6 Resource allocation graph (RAG)

$$RAG = (V, E)$$



E partitioned into two types:

1. $P_i \rightarrow R_j$ Process P_i request (wait)an instance of type R_j .
2. $R_j \rightarrow P_i$ an instance of resource type R_j has been allocated to process P_i .

Pictorially, we represent:

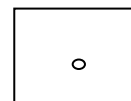
- Each **process** P_i as a **circle**.



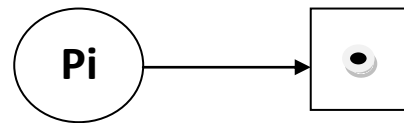
- Each **resource** type R_j as a **square**.



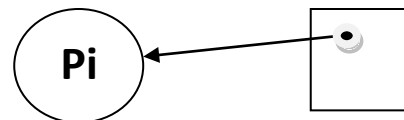
- Each instance of a resource is represented as a dot within the square.



- **Request edge:** a process request a resource. (**wait** state)



- **Assignment edge:** a process **hold** one instance of a resource.



Example 1: Consider the following system:

$P = \{P_1, P_2, P_3\}$ set of processes

$R = \{R_1, R_2, R_3, R_4\}$ set of resources

$E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$

R_1 one instance.

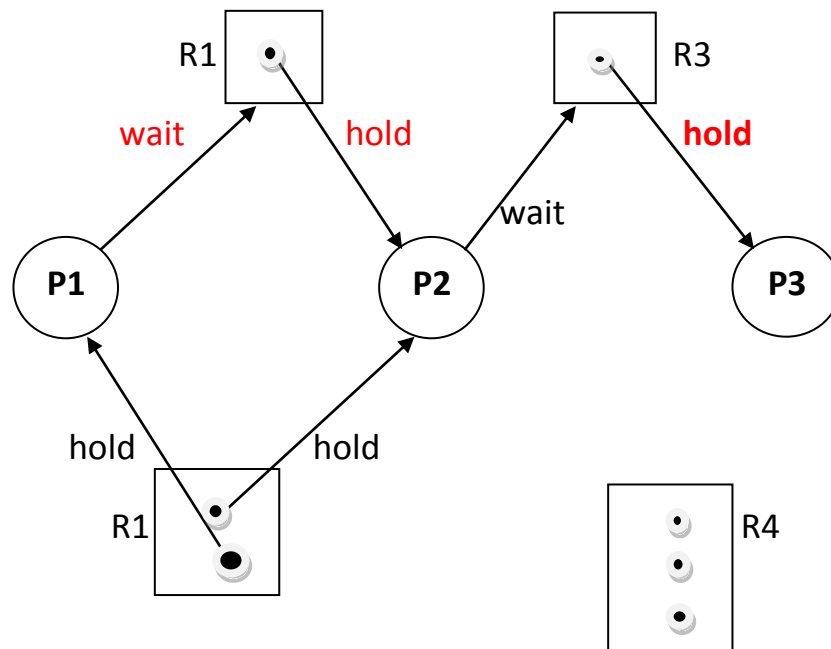
R_2 two instances.

R_3 one instance.

R_4 three instances.

1. Draw Resource Allocation Graph (RAG)

2. Is the system Deadlock or not Deadlock? Explain, why?



1. Resource Allocation Graph

2. The system is not in a deadlock, because the process P3 is holding instance of R3 but it is not waiting for any resource.

6.7 Basic Facts:

1. If the graph contains no cycle \longrightarrow No deadlock.
2. If the graph contains a cycle:
 - a. If only one instance per resource type \longrightarrow deadlock occurred.
 - b. If several instances per resource type \longrightarrow deadlock may exist.

Example 2: Consider the following system:

$P = \{P1, P2, P3\}$ set of processes

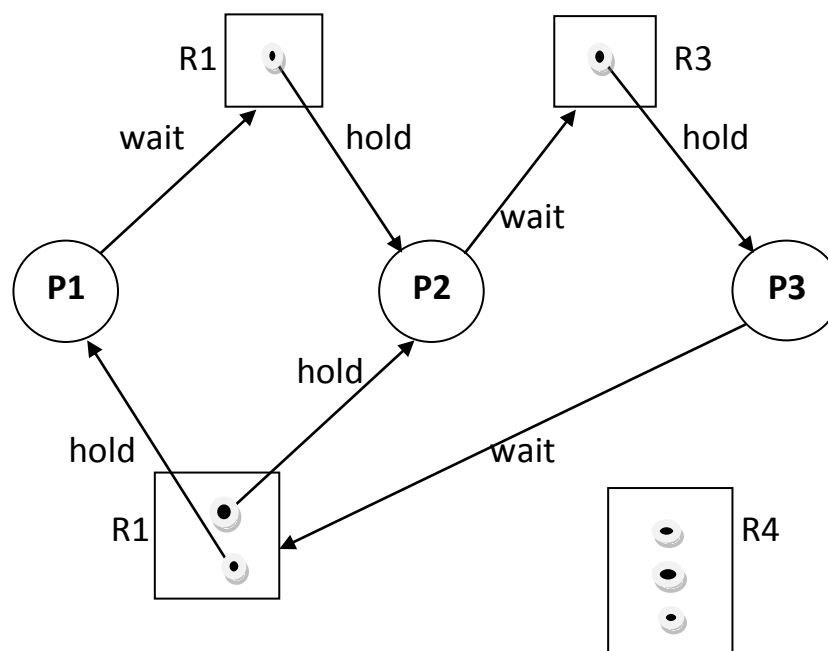
$R = \{R1, R2, R3, R4\}$ set of resources

$E = \{P1 \rightarrow R1, P2 \rightarrow R3, R1 \rightarrow P2, R2 \rightarrow P2, R2 \rightarrow P1, R3 \rightarrow P3, P3 \rightarrow R2\}$

R1 one instance.
 R2 two instances.
 R3 one instance.
 R4 three instances.

1. Draw Resource Allocation Graph (RAG)

2. Is the system Deadlock or not Deadlock? Explain, why?



1. Resource Allocation Graph.

2. The system is in a deadlock, because:

- a. Mutual exclusion.
- b. All the processes, P1, P2, P3 are hold and wait.
- c. No preemption.
- d. Two cycles exist:
 1. P1 --> R1 --> P2 --> R3 --> P3 --> R2 --> P1
 2. P2 --> R3 --> P3 --> R2 --> P2

NOTE: Example 2 is RAG with a cycle and with a deadlock.

Example 3: Consider the following system:

$P = \{P1, P2, P3, P4\}$ set of processes

$R = \{R1, R2\}$ set of resources

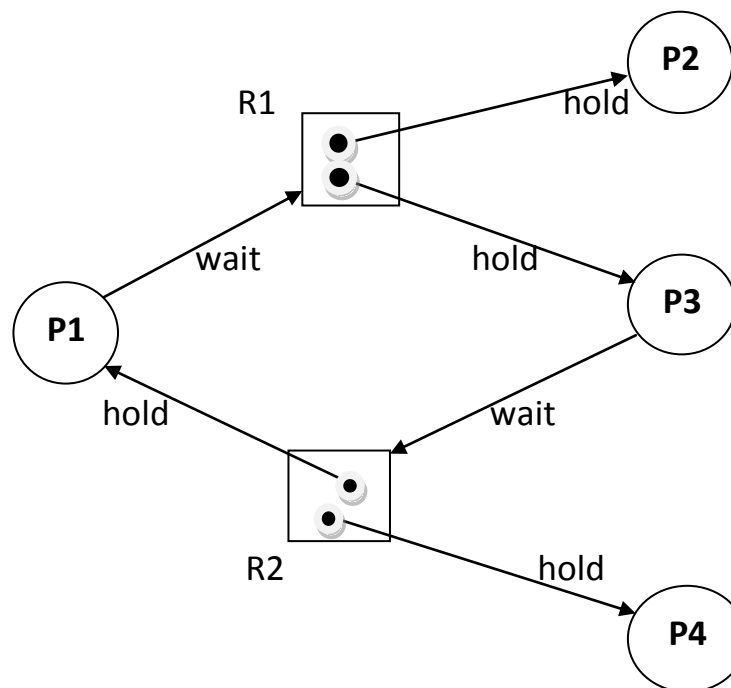
$E = \{P1 \rightarrow R1, R1 \rightarrow P2, R1 \rightarrow P3, P3 \rightarrow R2, R2 \rightarrow P4, R2 \rightarrow P1\}$

R1 two instances.

R2 two instances.

1. Draw Resource Allocation Graph (RAG)

2. Is the system Deadlock or not Deadlock? Explain, why?



1. The Resource Allocation Graph (RAG)

2. The system is not in a deadlock, because the processes P2 and P4 are both holding instance but not waiting for any resource.

NOTE: Example 3 is RAG with a cycle but no deadlock.

Example 4: Consider the following snapshot (table), assume 4 processes (P1, P2, P3, P4 , 4 resources types (R1, R2, R3, R4). Each resource type has only one instance.

Process	Resource Allocation	Resource Request
---------	---------------------	------------------

P1	R1	R4
P2	R3	R2
P3	R4	R3
P4	R2

1. Draw the resource allocation graph (RAG).
2. Is the system in a deadlock or not? What is the sequence of process execution?

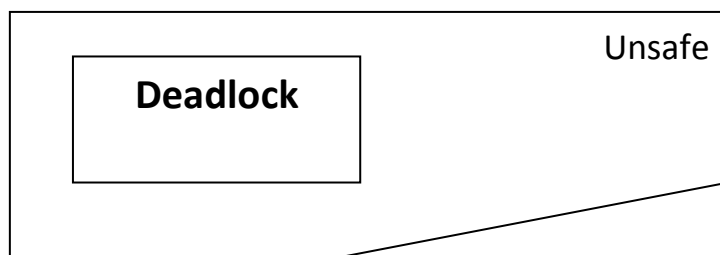
6.8 Methods of Handling Deadlocks:

1. **Deadlock Prevention:** By ensuring that at least one of the deadlock conditions does not hold at all time.
2. **Deadlock Avoidance:** Ensure that a system will never enter an unsafe state.
3. **Deadlock Detection:** Detect deadlock when it occurs and try to recover.
4. **Recovery from Deadlock:** There are two options for breaking a deadlock.
 - a. **Process termination by killing a process:**
 - Kill all deadlock processes.
 - Kill one process at a time until the deadlock is eliminated.
 - b. **Resource preemption to eliminate deadlock.**

Safe state: The system can allocate resources to each process and still avoid a deadlock.

Safe state —————> no deadlock

Unsafe state —————> possible deadlock



6.9 Safety Algorithm

1. Let Work and Finish be vectors of length m and n, respectively.
Initialize:
 Work = Available;
 Finish [i] = false for i = 0, 1, ... n-1
2. Find an i such that both:
 - a. Finish [i] = false
 - b. Need [i] ≤ WorkIf no such i exists, go to step 4
3. Work = Work + Allocation[i]
 Finish[i] = true
 Go to step 2
4. If Finish[i] = true for all i, then the system is in a safe state

Example 4: Consider a system with 5 processes P0 through P4 and 3 resources types A, B, and C such that:
A has 10 instances, B has 5 instances, C has 7 instances
Snapshot at time T1:

	Max	Allocation
--	-----	------------

Process	A	B	C	A	B	C
P0	7	5	3	0	1	0
P1	3	2	2	2	0	0
P2	9	0	2	3	0	2
P3	2	2	2	2	1	1
P4	4	3	3	0	0	2

1. use Banker algorithm, is the system in a safe state or not? Give the safety process sequence.
2. Suppose that process P1 request for (1, 0, 2), can this request be granted immediately ?
3. Suppose that process P4 request for (3, 3, 0), can this request be granted immediately ?
4. Suppose that process P0 request for (0, 2, 0), can this request be granted immediately ?

Example 5: Suppose a system with 12 magnetic tape drives and three processes at time T1:

Process	Maximum Need	Current Need
P0	10	5
P1	4	2
P2	9	2

- a. Is the system safe or unsafe? Give the safety sequence.
- b. Suppose that process P2 requests and is allocated 1 more tape drive, is the system still safe or no longer in a safe state? Why?

Example 6: Consider a system with 5 processes P0 through P4 and 3 resources types A, B, and C such that:

A has 7 instances, B has 2 instances, C has 6 instances.

Suppose that at time T_0 , we have the following Snapshot:

2. Suppose that process P2 request for (0, 0, 1), can this request be granted immediately ?

Example 7: Consider the following system snapshot:

	Allocation		Request		Total Resources	
Process	A	B	A	B	A	B
P1	1	0	0	1	10	3
P2	5	1	3	2		
P3	2	1	1	1		
P4	0	1	2	0		

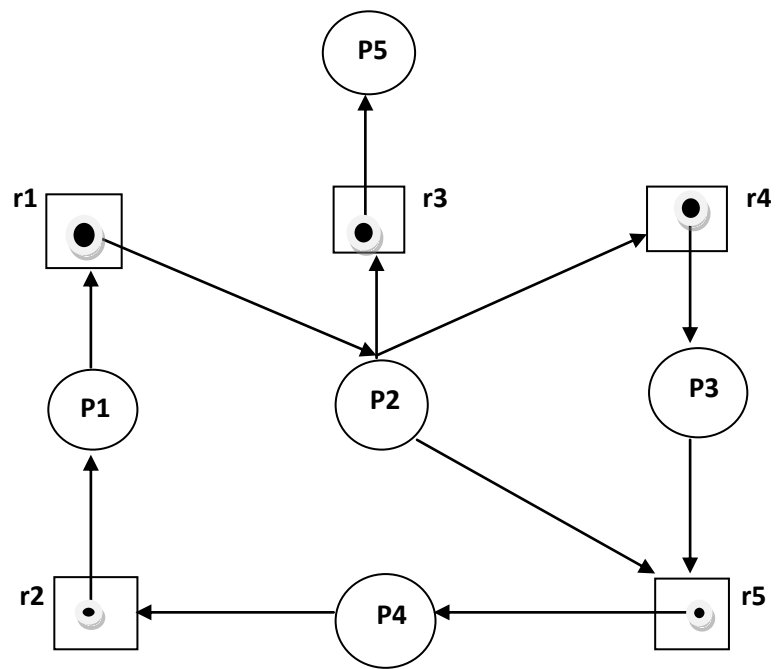
Is the system in a deadlock state? Show the processes sequence.

6.10 Wait-for graph

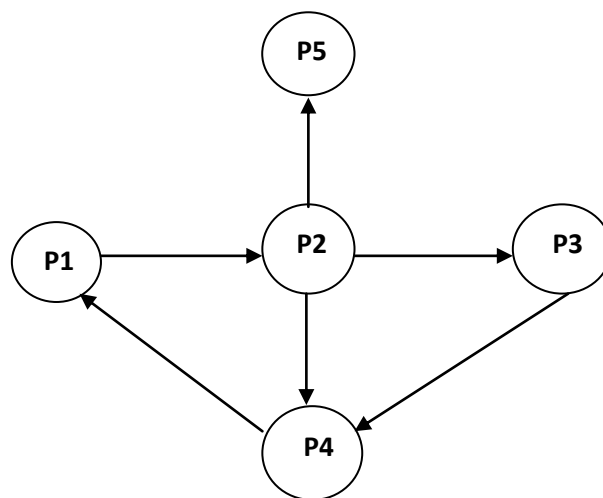
If all resources have only a single instance, we can define a graph called a Wait-for graph. This graph is obtained from the resources allocation graph (RAG), by removing the nodes of type resource and collapsing the edges.

A **deadlock exists** in the system if and only if the **wait-for graph** contains a **cycle**.

Example 7: Consider the following resource allocation graph (RAG):



Resource Allocation Graph (RAG)



(Wait-for graph)

Q1: The operating system contain 3 resource types, their instances are 7, 7, 10 respectively. The current resource allocation state is shown below:

Process	Current Allocation			Maximum Need		
	R1	R2	R3	R1	R2	R3
P1	2	2	3	3	6	8
P2	2	0	3	4	3	3
P3	1	2	4	3	4	4

- Find the available resources.
- What is the content of the matrix need?
- Is the current allocation in a safe state? Give the safety sequence.

Q2: Referring to example 7, which process can be killed to eliminate from deadlock?

Q3: For a system with the following processes and resources information:

Resources Name	Printer	Plotter	Disk	Flash
No. of instances	7	5	6	5

Process	Allocation				Max need			
	Printer	Plotter	Disk	Flash	Printer	Plotter	Disk	Flash
P1	2	1	0	1	4	2	5	1
P2	1	0	1	0	2	3	2	2
P3	2	1	2	0	2	4	3	1

State whether the system safe or unsafe. Give the safety sequence of processes if it is safe.

Q4: For a system with the following set of processes and resources information: (2016)

$P = \{P1, P2, P3, P4\}$

$R = \{R1, R2, R3\}$

$E = \{P1 \rightarrow R1, P2 \rightarrow R2, P3 \rightarrow R3, R1 \rightarrow P4, R1 \rightarrow P2, R2 \rightarrow P3, R3 \rightarrow P1\}$

Resources instances:

2 instances of resource type R1.

1 instance of resource type R2.

1 instance of resource type R3.

1. Draw RAG.

2. Is system safe or unsafe? (Draw all possible cycles).

Q5: Consider a system with processes P0 through p5 and three resource types A, B, C. Resource type A has 6 instances, Resource type B has 4 instances, Resource type C has 8 instances. Suppose that, at time T0, the following snapshot of the system has been taken: (2016)

	Allocation			Max		
	<u>A</u>	<u>B</u>	<u>C</u>	<u>A</u>	<u>B</u>	<u>C</u>
P0	2	1	0	3	4	2
P1	0	0	2	0	1	4
P2	1	0	1	3	2	2
P3	0	1	1	0	1	2
P4	2	0	1	3	1	1
P5	0	0	1	2	0	3

a. By using banker's algorithm, show that if the above system in safe or unsafe state.

b. Suppose now that the process P1 makes the request (0, 1, 1), can this request be granted immediately? Explain your answer.

Q6: Consider the following system snapshot:

Process	Max Need				Allocation				Avaliable			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	5	2	0
P1	1	7	5	0	1	0	0	0				
P2	2	3	5	6	1	3	5	4				
P3	0	6	5	2	0	6	3	2				
P4	0	6	5	6	0	0	1	4				

a. What is the total resource types of A, B, C, and D?

b. Is the system in a safe state? In what way?

Q7: Draw 2 R.A.G both have a cycle in it, but one of them is deadlock, while the other one is not deadlock.

Q8: Consider a system with the following sets:

$P=\{p1,p2,p3,p4\}$ set of processes

$r=\{r1,r2,r3,r4\}$ set of resources

$r1=r3=1$ instance, $r2=2$ instance, $r4=3$ instance.

$E=\{p1 \rightarrow r1, p2 \rightarrow r3, r1 \rightarrow p2, r2 \rightarrow p2, r2 \rightarrow p1, r3 \rightarrow p3\}$.

1. Draw resource allocation graph (RAG) for the system.
2. What is the state of the system? Explain your answer.

Q9: Consider the following system:

$P = \{ p1 , p2 , p3 , p4 \}$ set of processes.

$R = \{ r1 , r2 , r3 \}$ set of resources.

Where $r1 = r2 = r3 = 1$ instances. and

$E = \{ r1 \rightarrow p1 , p3 \rightarrow r1 , p1 \rightarrow r2 , r2 \rightarrow p2 , p2 \rightarrow r3 , r3 \rightarrow p3 , p4 \rightarrow r2 , p4 \rightarrow r3 , p2 \rightarrow r1 \}$. set of edges.

- (a): Draw Resource Allocation Graph (RAG).
- (B): Can we convert this graph to Wait-for Graph?. If yes convert it, if no give a reason.
- (c): Is the system in a deadlock state ?.

Operating Systems Concepts

Chapter 7

Memory management

Prepared by Assist . Prof.

Imad Matti

*AL-mamoon University College / Computer Science
Department*

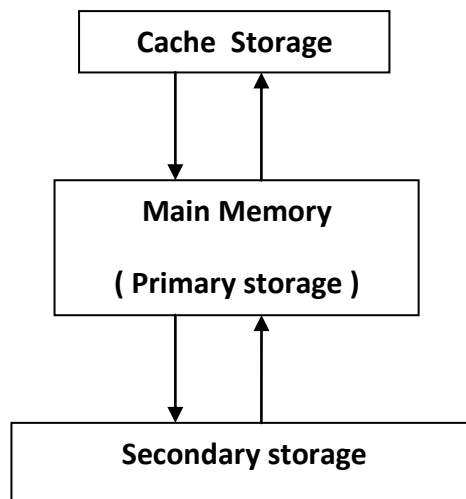
2018

Chapter 7

Memory Management

7. Introduction

Programs and data need to be in main storage in order to be executed, And if they do not needed immediately, they may be kept on secondary storage media such as tapes or disk until needed and then brought into the main storage for execution.



(Storage Organization)

Logical (virtual) address: address generated By CPU.

Physical address: address seen by the memory unit.

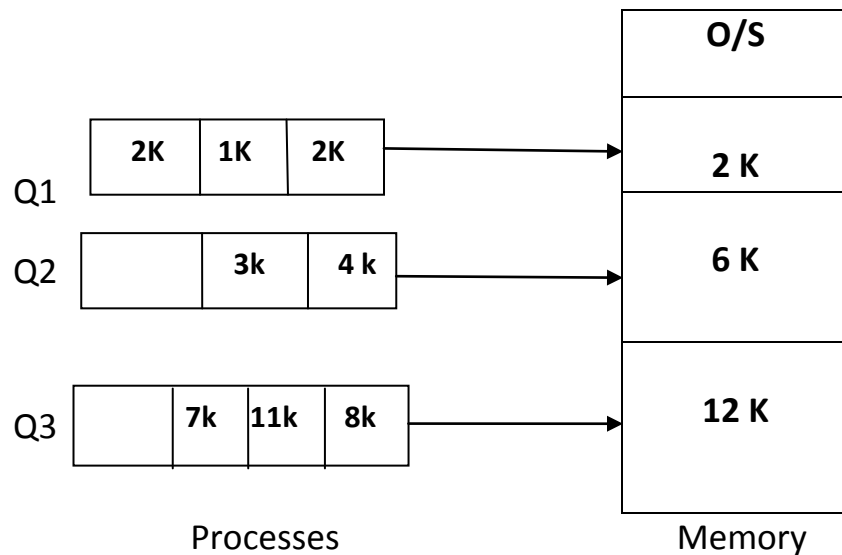
7.1 Multiple Partition Allocation

There are two memory management schemes:

7.1.1 Multiple contiguous fixed partition (MFT)

The memory is divided into a number of fixed sized partitions. Each one contain exactly one process.

Example 1: Consider the following:



Disadvantages:

The problem with MFT is **internal** and **external fragmentation**.

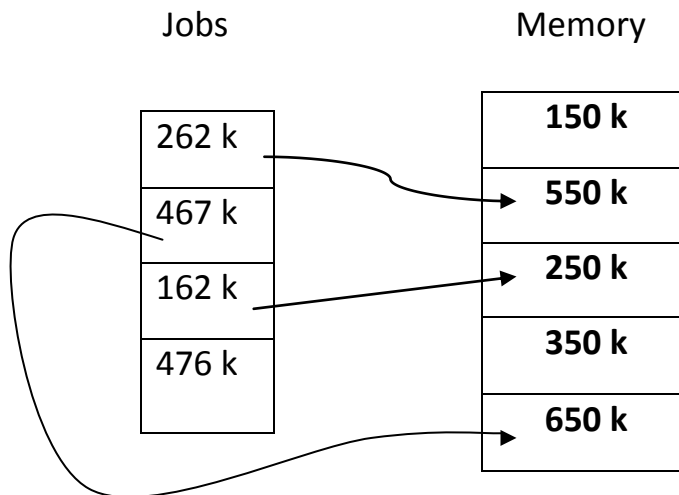
The solution to this problem is **MVT**.

7.1.2 Storage placement strategies

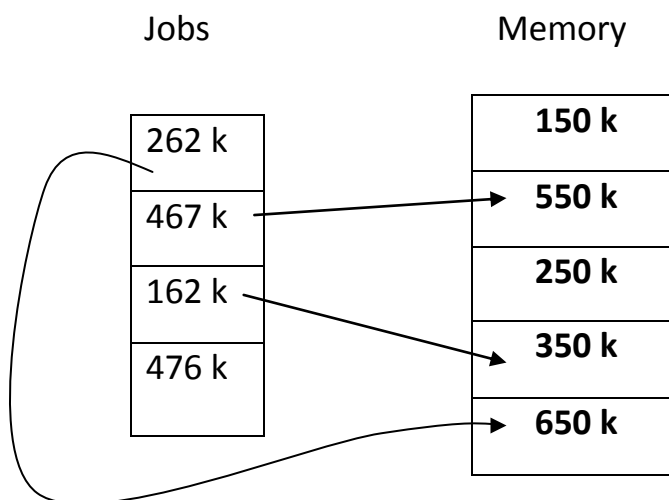
- First-Fit** strategy: Allocate job in the **first** hole in memory that is big enough.
- Best-Fit** strategy: Allocate job in the **smallest** hole in memory big enough.
- Worst-Fit strategy**: Allocate job in the largest hole in the memory.

Example 2: Given memory partitions of 150 k, 550 k, 250 k, and 650 k (in order). How would each of the First fit, Worst fit, and Best fit algorithms place processes (jobs) of 262 k, 467 k, 162 k, and 476 k (in order)?

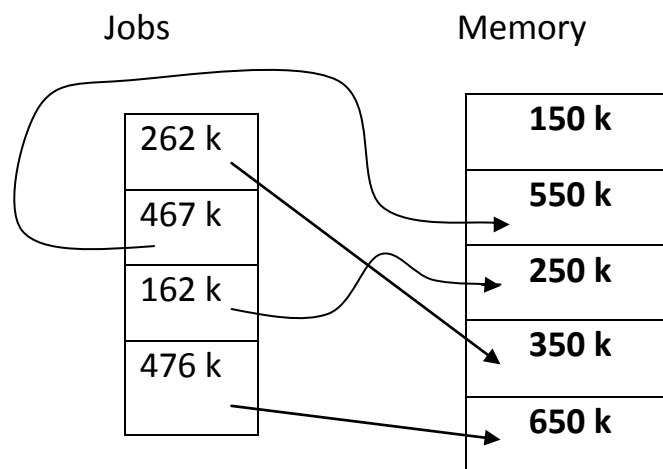
1. First fit:



2. Worst fit:



3. Best fit:



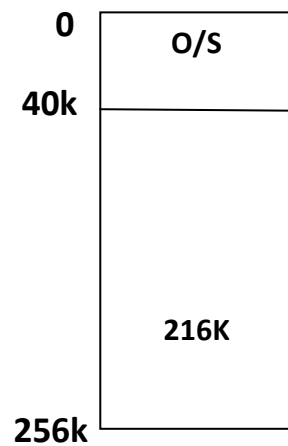
7.1.3 Multiple Variable Partition (MVT)

In MVT, initially all memory is available for user programs and is considered as one large block. When the job arrives and needs memory we search for a hole in memory large enough for this job.

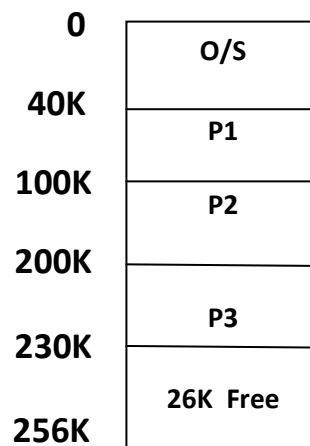
Advantages: minimize internal and external fragmentation.

Example 3: assume we have 256k memory available, and O/S require 40k, with job queue FCFS scheduling, and the following system snapshot, allocate memory to jobs 1, 2, 3, 4, and 5.

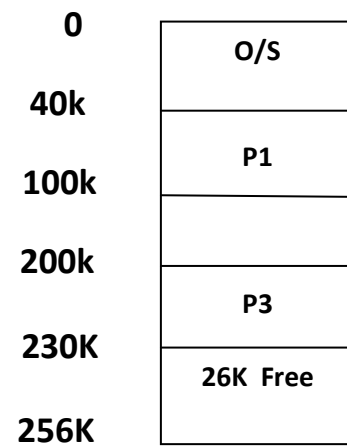
Process	Size	Time
P1	60k	10
P2	100k	5
P3	30k	20
P4	70k	8
P5	50k	15



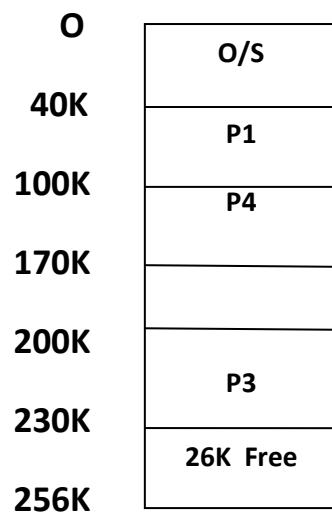
(a)



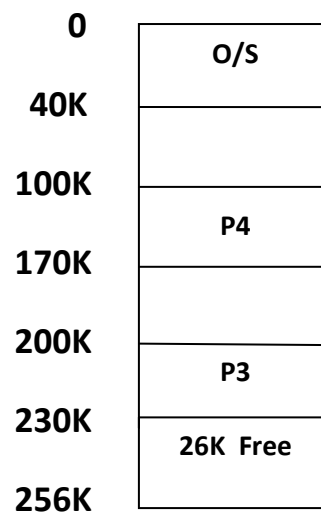
(b)



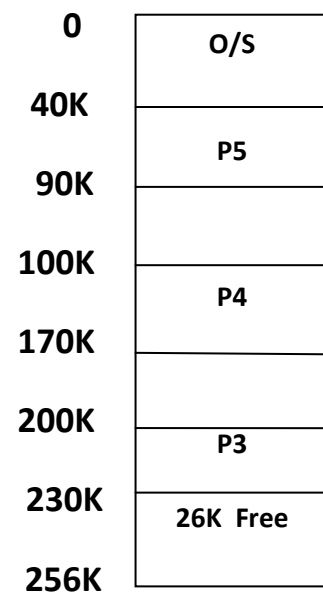
(c)



(d)



(e)



(f)

7.1.4 Fragmentation

There are two types of fragmentation:

1. Internal fragmentation(IF):

some portions of memory are left unused, and it cannot be used by any process.

2. External fragmentation(EF):

Occurs when a region is unused and available but too small for any waiting job.

$$F = IF + EF$$

Disadvantage of fragmentation: Memory waste.

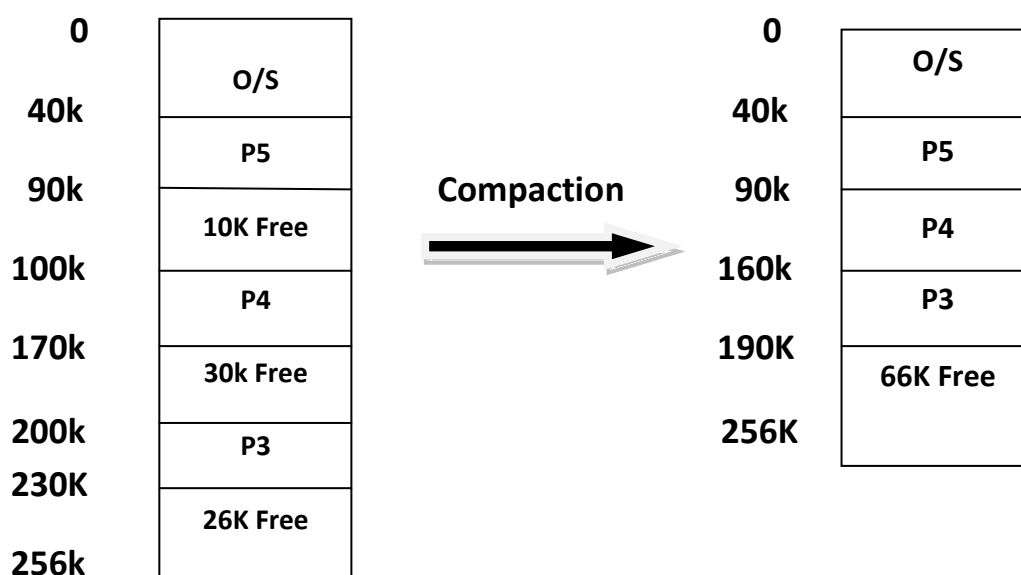
Solutions to the fragmentation problem:

1. Compaction.
2. Paging.

7.1.5 Compaction

Is to move all jobs towards one end of memory, all holes move in the other direction, producing one large hole of available memory.

Example 4:



7.1.6 Paging Method

In paging the physical memory is broken into blocks called frames. Logical memory is also broken into blocks called pages. When a process is to be executed its pages are loaded into any available memory frames.

Example 5: A user program consists of 16 instructions, each instruction takes one byte. Suppose page size = frame size = 4 bytes. The physical memory size is 512 bytes.

- a. Calculate the number of pages in the logical map.
- b. Calculate the number of frames in the physical map.
- c. Draw the logical map.
- d. Suppose page 1 is allocated in frame 20, and page 2 is allocated in frame 22, calculate the physical addresses of the logical addresses of page 1 and page 2.
- e. Draw the physical map.

SOLUTION:

a. Program size = number of program instructions × instruction size.

$$= 16 \times 1 \text{ byte} = 16 \text{ bytes.}$$

Number of pages = program size ÷ page size.

$$= 16 \div 4 = 4 \text{ pages in the logical map.}$$

b. Number of frames = physical memory size ÷ page size.

$$= 512 \div 4 = 128 \text{ frames in the physical map.}$$

Page number	Logical address	displacement	Instructions
Page 0	0	0	X1
	1	1	X2
	2	2	X3
	3	3	X4
Page 1	4	0	X5
	5	1	X6
	6	2	X7
	7	3	X8
Page 2	8	0	X9
	9	1	X10
	10	2	X11
	11	3	X12

Page 3	12	0	X13
	13	1	X14
	14	2	X15
	15	3	X16

(Logical map)

c.

Page number	Frame number
1	20
2	22

Page table

Physical address = (frame number × page size) + displacement.

Physical address(x5) = (20 × 4) = 80 + 0 = 80

Physical address(x6) = (20 × 4) = 80 + 1 = 81

Physical address(x7) = (20 × 4) = 80 + 2 = 82

Physical address(x8) = (20 × 4) = 80 + 3 = 83

Physical address(x9) = (22 × 4) = 88 + 0 = 88

Physical address(x5) = (22 × 4) = 88 + 1 = 89

Physical address(x5) = (22 × 4) = 88 + 2 = 90

Physical address(x5) = (22 × 4) = 88 + 3 = 91

Physical address	Displacement	instructions	Frame number
			0
		
80	0	X5	20
81	1	X6	
82	2	X7	
83	3	X8	
		
88	0	X9	22
89	1	X10	
90	2	X11	
91	3	X12	
		
			127

d. Physical map

7.1.7 Segmentation

Segmentation: is memory management scheme that supports user view of memory.

Example 6: Consider a program consists of five segments
S0 = 600kB, S1 = 14kB, S2 = 100kB, S3 = 580kB,
S4 = 96Kb.

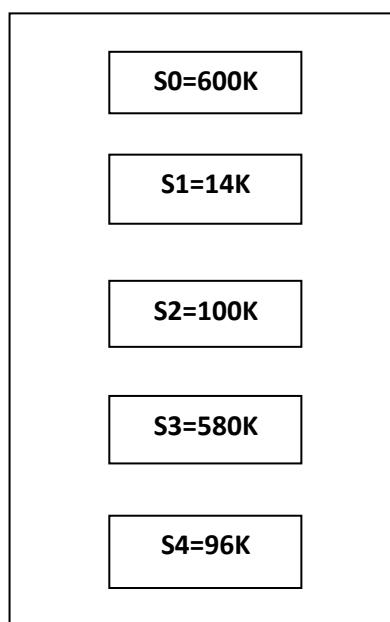
Assume at that time the available free space partitions of memory are:

1200-1850kb, 50-160kb, 220-250kb, 2500-3200kb

- a. Allocate space for each segment in the order given above.
- b. Draw the logical, physical maps and segment table.
- c. Calculate the external and internal fragmentations, where if the remainder of partition < 10 bytes leave it as internal fragmentation and use the BEST-FIT strategy to allocate space for each segment.
- d. What are the physical addresses in memory for the following logical addresses:
(1) 0.580 (2) 1.17 (c) 2.96 (d) 4.112 (e) 3.420

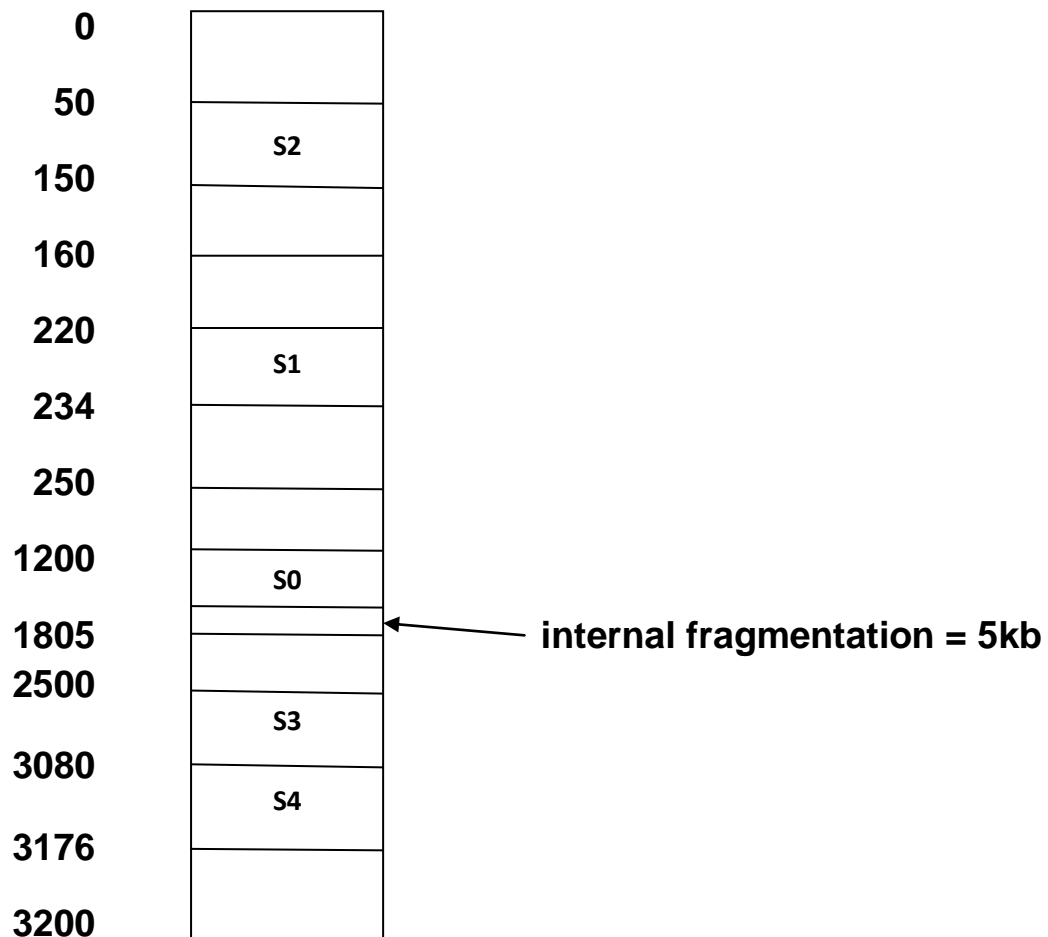
The solution:

The logical map and segment table before allocation are:



Segment number	Limit register	Base register
0	600	1200
1	14	220
2	100	20
3	580	2500
4	96	3080

The segment table



Physical memory

d.

segment number	offset
s	d

if $d < \text{limit}$

then:

physical address = base address + d

else error (no physical address)

(1) 0.580

S=0 d=580

d < limit

580 < 600 --> true

Physical address = 1200 + 580 = 1780

(2) 1.17

S=1 d=17

d < limit

17 < 14 --> false --> error (no physical address)

(3) 2.96

S = 2 d = 96

d < limit

96 < 100 --> true

Physical address = 50 + 96 = 146

(4) and (5) home work

Chapter 7 Questions

Q1: Given a snapshot of a memory management table and an input queue as shown in the following: (وزاري ٢٠١٦)

Memory Management Table	Process Name	Input Queue or (process requirements)
10 KB	P1	10 KB
30 KB	P2	20 KB
20 KB	P3	15 KB
40 KB	P4	30 KB
50 KB	P5	25 KB

Show how First Fit, Best Fit, and Worst Fit algorithms would allocate the processes in the input queue in the spaces shown in the memory management table.

Q2: A computer system with a memory consists of the following hole sizes in the following memory order: 10 KB , 4 KB , 20 KB , 18 KB , 7 KB , 9 KB , 12 KB, 15 KB.

Which hole is taken for successive block request of program size request of 14 KB , 9 KB , 11 KB for :

1. First-fit 2. Best-fit 3. Worst-fit. Draw the solution for each one.

Q3: A user program consists of 26 instructions, each instruction takes one byte. Suppose page size = frame size = 4 bytes. The physical memory size is 512 bytes.

- Calculate the number of pages in the logical map.**
- Calculate the number of frames in the physical map.**
- Draw the logical map.**
- Suppose page 0 is allocated in frame 10, and page 3 is allocated in frame 30, calculate the physical addresses of the logical addresses of page 0 and page 3.**
- Draw the physical map.**

Q4: A user program consists of 16 instructions, each instruction takes two bytes. Suppose page size = frame size = 8 bytes. The physical memory size is 512 bytes.

- a. Calculate the number of pages in the logical map.
- b. Calculate the number of frames in the physical map.
- c. Draw the logical map.
- d. Suppose page 0 is allocated in frame 10, and page 3 is allocated in frame 30, calculate the physical addresses of the logical addresses of page 0 and page 3.
- e. Draw the physical map.

Q5: A user program of size 35 bytes. Suppose page size = frame size = 8 bytes. The physical memory size is 8192 bytes.

- a. Calculate the number of pages in the logical map.
- b. Calculate the number of frames in the physical map.
- c. Draw the logical map.
- d. Suppose page 1 is allocated in frame 10, and page 3 is allocated in frame 30, calculate the physical addresses of the logical addresses of page 1 and page 3.
- e. Draw the physical map.

Q6: Given free memory partitions of

100k, 500k, 200k, 700k, and 600k (in order),

How would each of the First-fit, Best-fit, and Worst-fit algorithms place processes of 212k, 417k, 112k, and 426k (in order)?

Which algorithm makes the most efficient use of memory?

Q7: A process uses six pages numbered (0, 1, 2, 3, 4, 5) during its execution. The pages 0, 2, 4, and 5 are loaded in main memory at page frames 4, 2, 5, and 1 respectively. Assume the page size is 4 bytes and the main memory of size 32 bytes is used to map the pages.

- a. Construct the page map table.
- b. Find the physical addresses for the following logical addresses.
(0,2), (1,3), (2,0), (4,1), (3,3), and (5,2)

Q8: Consider the following part of process segment table:

Segment number	Limit	Base address
0	1100	1200
1	500	4400
2	700	3500
3	950	2200
4	1200	7500

What are the physical addresses for the following logical addresses?

a. (0.1050) b. (1.550) c. (2.770) d. (3.900) e. (4.1100)

Q9: Consider a system in which memory consists of the following hole sizes in the following order:

20kb, 4kb, 25kb, 9kb, 12kb, 7kb, 10kb, and 28kb.

Which hole is taken for successive block request of

22kb, 10kb, 8kb.

For First-fit, Best-fit, and Worst-fit?

Q10: A user program consists of 30 instructions, each instruction takes one byte. Suppose page size = frame size = 9 bytes. The physical memory size is 512 bytes. (2015)

- Calculate the number of pages in the logical map.**
- Calculate the number of frames in the physical map.**
- Draw the logical map.**
- Suppose page 0 is allocated in frame 10, and page 2 is allocated in frame 15, calculate the physical addresses of the logical addresses of page 0 and page 2.**
- Draw the physical map.**

Q11: Given memory partitions as shown below. How would each of the First fit, Worst fit, and Best fit algorithms place processes (jobs) of 5k, 10 k, and 20k(in order), in the holes (partitions) of the memory?

Used	Hole	Used	Hole	Used	Hole	Used	Hole	Used	Hole
10k	10k	20k	30k	10k	5k	60k	15k	20k	20k

Operating Systems Concepts

Chapter 8

Virtual memory

Prepared by Assist . Prof.

Imad Matti

*AL-mamoon University College / Computer Science
Department*

2018

CHAPTER 8

Virtual Memory

8.1 Virtual memory: is a technique that allows the execution of processes which are not completely available in memory.

Advantages:

1. programs can be larger than physical memory.
2. program is not required to be loaded fully in main memory.
3. The ability to execute a program that is only partially in memory.
4. Less number of I/O would be needed to load or swap each user program into memory.
5. Increase CPU utilization and throughput.

8.2 Page-Replacement Algorithms

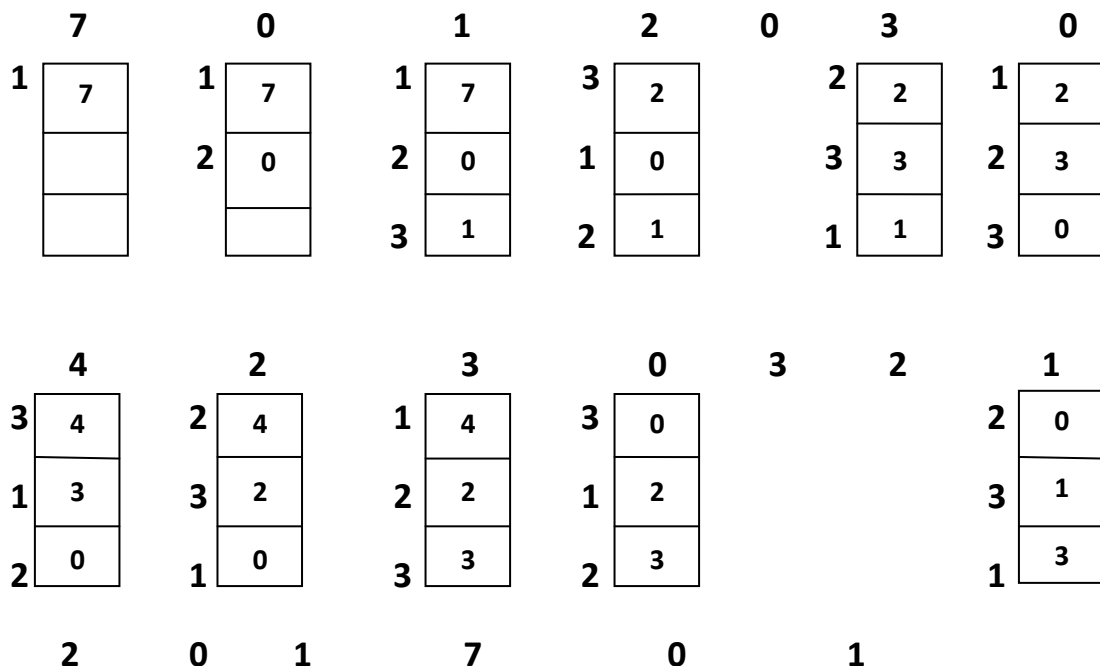
There are many different page-replacement algorithms:

8.2.1 FIFO Algorithm

When the page must be replaced, the oldest page is chosen.

Example 1: Consider the following page reference string, use three frames are initially empty, show how the page faults are brought into these frames, using FIFO algorithm.

Page reference string: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



1	0	3	7	2	7	1	7
2	1	1	1	3	0	2	0
3	2	2	2	1	2	3	1

Page Faults = 15

Advantages:

1. simple.
2. easy to understand and program.

Disadvantages:

1. performance is not always good.
2. it suffer from Belady's anomaly.

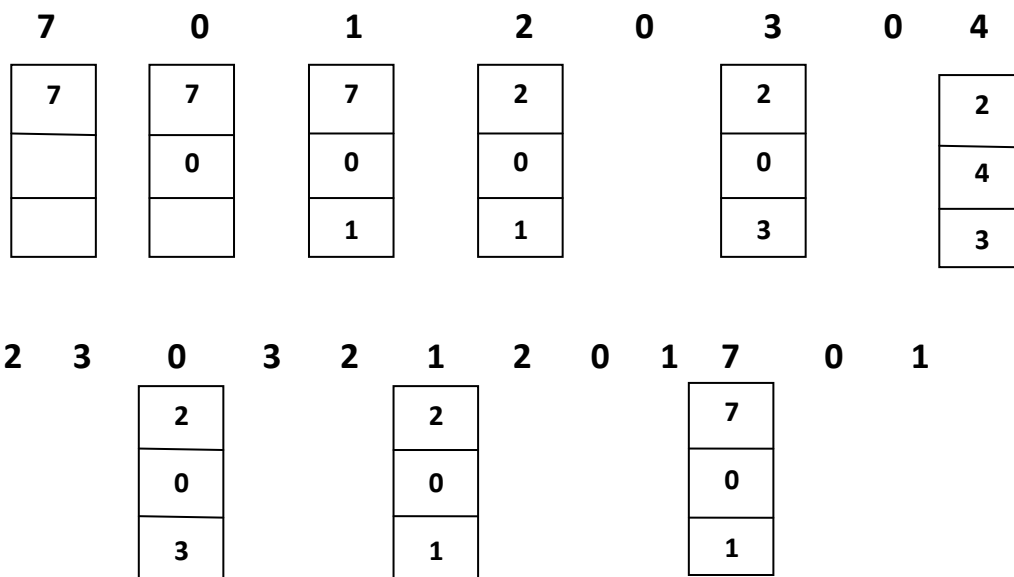
Belady's anomaly: it means the page-faults rate may increase as the number of free frames increase.

8.2.2 Optimal Algorithm

Replacing the page that will not be used for the longest period of time.

Example 2: Consider the following page reference string, use three frames are initially empty, show how the page faults are brought into these frames, using optimal algorithm.

Page reference string: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1



Page faults = 9

Advantages:

1. not suffer from Belady's anomaly.
2. Lowest page fault rate.

Disadvantages: difficult to implement because it requires future knowledge of the reference string.

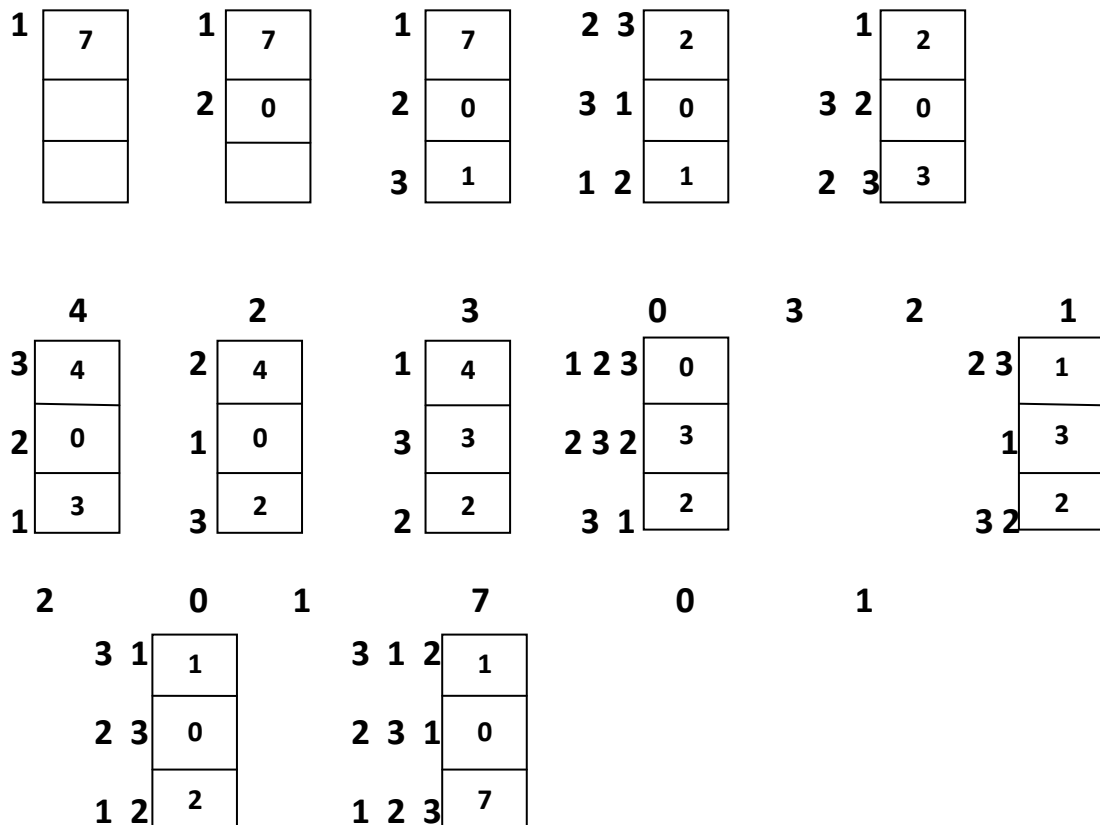
8.2.3 LRU Algorithm (least recently used)

When a page must be replaced LRU chooses that page that has not been used for the longest period of time.

Example 3: Consider the following page reference string, use three frames are initially empty, show how the page faults are brought into these frames, using LRU algorithm.

Page reference string: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

7 0 1 2 0 3 0



Number of page faults = 12

Advantages: quite good.

Disadvantages: how to implement LRU.

8.3 Allocation of frames

Example 4: Assume a memory of 62 frames and two processes

P1 = 10K, and P2 = 127K. Find the number of frames allocated for each process.

Suppose s_i is virtual memory for process p_i

Suppose a_i is number of frames allocated for process p_i

Suppose m is the memory size

Solution:

$$m = 62$$

$$s_1 = p_1 = 10k$$

$$s_2 = p_2 = 127k$$

$$S = \sum S_i = s_1 + s_2 = 10 + 127 = 137$$

$$a_i = (s_i / s) \times m$$

$$a_1 = (10 / 137) \times 62 = 4 \text{ frames}$$

$$a_2 = (127 / 137) \times 62 = 57 \text{ frames.}$$

8.4 Thrashing: A process is thrashing if it is spending more time paging than executing.

8.5 Disk Scheduling

Several algorithms exist to schedule the servicing of disk I/O request:

8.5.1 FCFS

Example 5: Suppose that a disk drive has 200 cylinders numbered from 0 to 199, and the drive is currently serving a request at cylinder (head start) 53. The queue of pending request in order is 98, 183, 37, 122, 14, 124, 65, 67.

1. Draw the direction of the head moving by using FCFS algorithm.
2. Calculate the total head movements.

Solution:

Queue = 98, 183, 37, 122, 14, 124, 65, 67

Head = 53

$$98 - 53 = 45$$

$$183 - 98 = 85$$

$$183 - 37 = 146$$

$$122 - 37 = 85$$

$$122 - 14 = 108$$

$$124 - 14 = 110$$

$$124 - 65 = 59$$

$$67 - 65 = 2$$

$$\text{Total head movements} = 45+85+146+85+108+110+59+2=640 \text{ cylinders}$$

8.5.2 SSTF

Example 6: Suppose that a disk drive has 200 cylinders numbered from 0 to 199, and the drive is currently serving a request at cylinder (head start) 53. The queue of pending request in order is 98, 183, 37, 122, 14, 124, 65, 67.

1. Draw the direction of the head moving by using SSTF algorithm.
2. Calculate the total head movements.

Solution:

Queue = 98, 183, 37, 122, 14, 124, 65, 67

Head = 53

$$65 - 53 = 12$$

$$67 - 65 = 2$$

$$67 - 37 = 30$$

$$37 - 14 = 23$$

$$98 - 14 = 84$$

$$122 - 98 = 24$$

$$124 - 122 = 2$$

$$183 - 124 = 59$$

Total head movements=12+2+30+23+84+24+2+59=236 cylinders

8.5.3 SCAN

Example 7: Suppose that a disk drive has 200 cylinders numbered from 0 to 199, and the drive is currently serving a request at cylinder (head start) 53. The queue of pending request in order is 98, 183, 37, 122, 14, 124, 65, 67.

1. Draw the direction of the head moving by using SCAN algorithm.
2. Calculate the total head movements.

Solution:

Queue = 98, 183, 37, 122, 14, 124, 65, 67

Head = 53

8.5.4 C-SCAN

Example 8: Suppose that a disk drive has 200 cylinders numbered from 0 to 199, and the drive is currently serving a request at

cylinder (head start) 53. The queue of pending request in order is 98, 183, 37, 122, 14, 124, 65, 67.

1. Draw the direction of the head moving by using C-SCAN algorithm.
2. Calculate the total head movements.

Solution:

Queue = 98, 183, 37, 122, 14, 124, 65, 67

Head = 53

8.5.5 C-LOOK

Example 9: Suppose that a disk drive has 200 cylinders numbered from 0 to 199, and the drive is currently serving a request at cylinder (head start) 53. The queue of pending request in order is 98, 183, 37, 122, 14, 124, 65, 67.

1. Draw the direction of the head moving by using C-LOOK algorithm.
2. Calculate the total head movements.

Solution:

Queue = 98, 183, 37, 122, 14, 124, 65, 67

Head = 53

Q1: Suppose that a disk drive has 200 cylinders numbered from 0 to 199, and the drive is currently serving a request at cylinder (head start) 60. The queue of pending request in order is 101, 186, 40, 125, 17, 127, 68, 70.

- 1. Draw the direction of the head moving by using FCFS and SSTF algorithm.**
- 2. Calculate the total head movements for each algorithm.**

Q2: Consider the following page reference string:

21, 14, 15, 16, 14, 17, 14, 18, 16, 17, 14, 17

With a memory that can hold only 4 pages. How many page faults and page replacements occur when using FIFO and LRU algorithms.

Q3: Consider the following page reference string:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

With a memory that can hold 3 and 4 pages. How many page faults and page replacements occur when using FIFO, OPTIMAL, and LRU algorithms.

Q4: Consider the following page reference string: (2016)

120, 113, 114, 115, 113, 116, 113, 117, 115, 116, 113, 116

With a memory that can hold only 3 pages. How many page faults and page replacements occur when using FIFO and OPTIMAL algorithms.