UNIT-I (CO-I)

1.a) Outline The concept of Computer System Architecture?

A. A computer system can be organized in a number of different ways, which we can categorize roughly according to the number of general-purpose processors used.

1.3 Computer-System Architecture 13

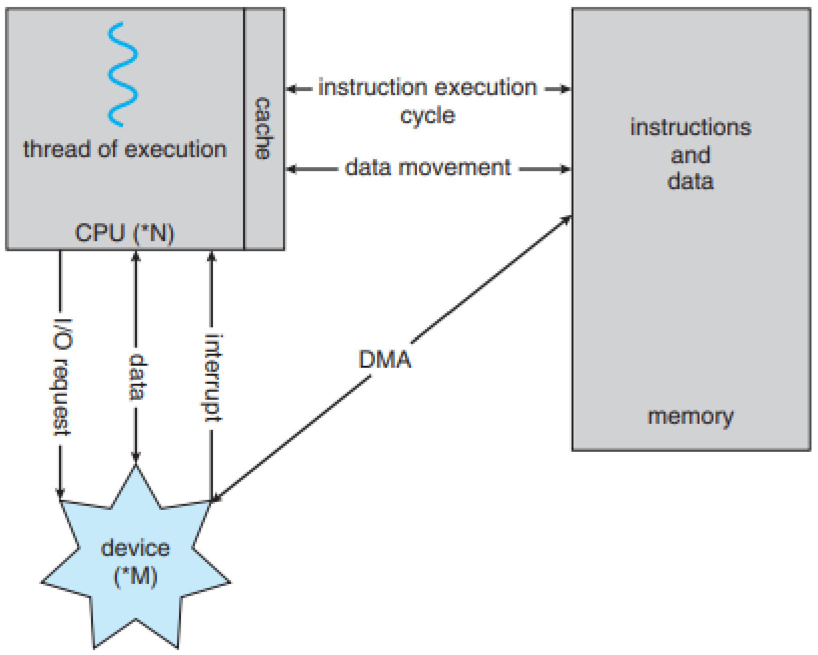


Figure 1.5 How a modern computer system works.

Single-Processor Systems

most computer systems used a single processor. On a singleprocessor system, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes.

Multiprocessor Systems

s. Multiprocessor systems first appeared prominently appeared in servers and have since migrated to desktop and laptop systems. Recently, multiple processors have appeared on mobile devices such as smartphones and tablet computers. Multiprocessor systems have three main advantages

. Increased throughput

. Economy of scale.

. Increased reliability

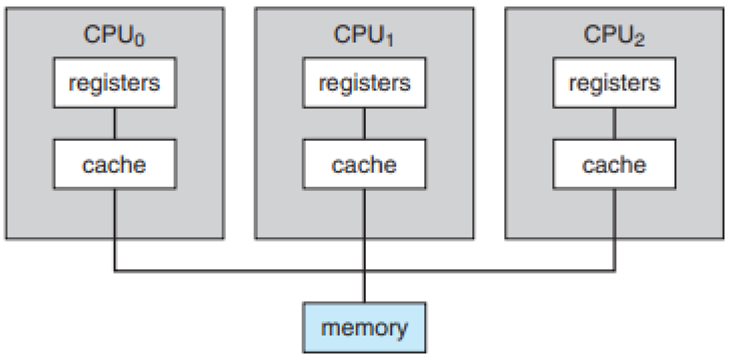


Figure 1.6 Symmetric multiprocessing architecture.

b) Explain in detail the Services of an Operating System?

A. An operating system provides services to both the users and to the programs.

It provides programs an environment to execute.’

It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system-

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

2. a) Write short notes on Distributed and multiprocessor Systems?

A. **Distributed System:**

A distributed system is a collection of physically separate, possibly heterogeneous, computer systems that are networked to provide the users with access to the various resources that the system maintains. Access to a shared resource increases computation speed, functionality, data availability, and reliability.

A in the simplest terms, is a communication path between two or more systems. Distributed systems depend on networking for their functionality.

A Local-area network (LAN) connects computers within a room, a floor, or a building. A N) usually links buildings, cities, or countries.

Multiprocessor Systems:

Within the past several years, multiprocessor systems (also known as parallel systems or multicore systems) have begun to dominate the landscape of computing. Such systems have two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices. Multiprocessor systems first appeared prominently appeared in servers and have since migrated to desktop and laptop systems. Recently, multiple processors have appeared on mobile devices such as smartphones and tablet computers.

Multiprocessor systems have three main advantages

- Increased throughput
- Economy of scale
- Increased reliability

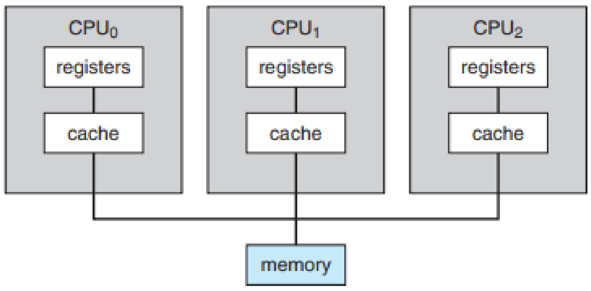


Figure 1.6 Symmetric multiprocessing architecture.

b) Present the types of System Calls in detail?

System calls can be grouped roughly into six major categories:

- process control
- file manipulation
- device manipulation
- information maintenance
- communication
- protection

UNIT-II (CO-2)

3. a) Write Short Notes on process Control block?

Process state
Process number
Program counter
Registers
Memory limits
List of open Files
……

Process Control Block

Process State

This specifies the process state i.e. new, ready, running, waiting or terminated.

Process Number

This shows the number of the particular process.

Program Counter

This contains the address of the next instruction that needs to be executed in the process.

Registers

This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

List of Open Files

These are the different files that are associated with the process

CPU Scheduling Information

The process priority, pointers to scheduling queues etc. is the CPU scheduling information that is contained in the PCB. This may also include any other scheduling parameters.

Memory Management Information

The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc.

I/O Status Information

This information includes the list of I/O devices used by the process, the list of files etc.

Accounting information

The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.

b) Explain FCFS, SJF and Priority Process scheduling algorithms in detail.

A**. First-Come, First-Served Scheduling:**

the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue. The code for FCFS scheduling is simple to write and understand.

Process	Burst Time -----
P1	24
p2	3
P3	3

If the processes ani ve in the order P1, P2, P3, and are served in FCFS order, we get the result shown in the following Gantt chart, which is a bar chart that illustrates a particular schedule, including the start and finish times of each of the participating processes: 0 24 27 30 The waiting time is 0 milliseconds for process P1, 24 milliseconds for process P2 , and 27 milliseconds for process P3 . Thus, the average waiting time is (0 + 24 + 27)/3 = 17 ncilliseconds. If the processes arrive in the order P2, P3 , P1, however, the results will be as shown in the following Gantt chart: 0 3 6 30 The average waiting time is now (6 + 0 + 3)/3 = 3 milliseconds. This reduction is substantial. Thus, the average waiting time under an FCFS policy is generally not minimal and d may vary substantially if the processes CPU burst times vary greatly.

Shortest-Job-First Scheduling

. This algorithm associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the process 190 Chapter 5 that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie. Note that a more appropriate term for this scheduling method would be the shortest-next-CPU-burst algorithm, because scheduling depends on the length of the next CPU burst of a process, rather than its total length. We use the term SJF because m.ost people and textbooks use this term to refer to this type of scheduling. As an example of SJF scheduling, consider the following set of processes, with the length of the CPU burst given in milliseconds:

The waiting time is 3 milliseconds for process P1, 16 milliseconds for process P2, 9 milliseconds for process P3, and 0 milliseconds for process P4 . Thus, the average waiting time is (3 + 16 + 9 + 0) I 4 = 7 milliseconds. By comparison, if we were using the FCFS scheduling scheme, the average waiting time would be 10.25 milliseconds.

Priority Scheduling

The SJF algorithm is a special case of the general priority scheduling algorithm. A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal-priority processes are scheduled in FCFS order. An SJF algorithm is simply a priority algorithm where the priority (p) is the inverse of the (predicted) next CPU burst. The larger the CPU burst, the lower the priority, and vice versa.

As an example, consider the following set of processes, assumed to have arrived at time 0 in the order P1, P2, · · ·, Ps, with the length of the CPU burst given in milliseconds: 5.3 193

Process	Burst Time	priority
pl	10	3
p2	1	1
p3	2	4
p4	1	5
P5	5	2

The average waiting time is 8.2 milliseconds. Priorities can be defined either internally or externally. Internally defined priorities use some nceasurable quantity or quantities to compute the priority of a process. For example, time limits, memory requirements, the number of open files, and the ratio of average I/0 burst to average CPU burst have been used in computing priorities.

4. a) Review in detail the operations on processes.

A. There are many operations that can be performed on processes. Some of these are process creation, process preemption, process blocking, and process termination. These are given in detail as follows –

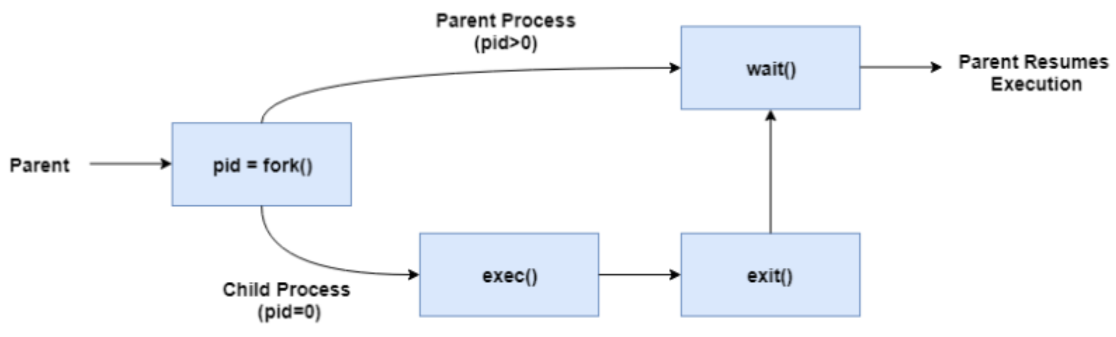
Process Creation

Processes need to be created in the system for different operations. This can be done by the following events –

- User request for process creation
- System initialization
- Execution of a process creation system call by a running process
- Batch job initialization

A process may be created by another process using fork(). The creating process is called the parent process and the created process is the child process. A child process can have only one parent but a parent process may have many children. Both the parent and child processes have the same memory image, open files, and environment strings. However, they have distinct address spaces.

A diagram that demonstrates process creation using fork() is as follows –

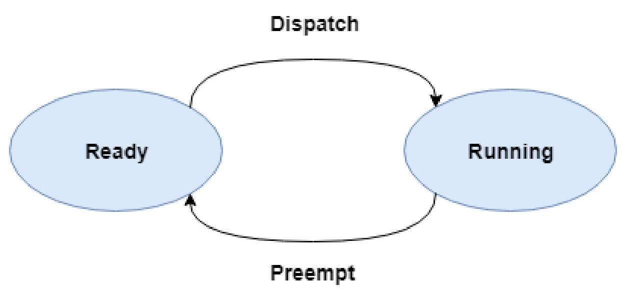


Process Creation Using fork()

Process Preemption

An interrupt mechanism is used in preemption that suspends the process executing currently and the next process to execute is determined by the short-term scheduler. Preemption makes sure that all processes get some CPU time for execution.

A diagram that demonstrates process preemption is as follows –

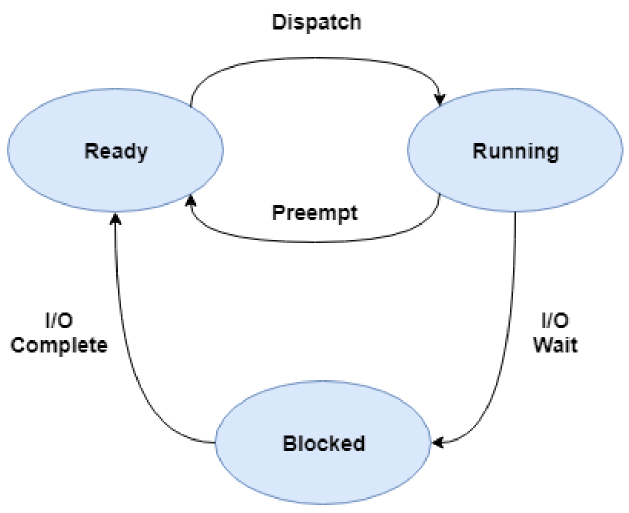


Process Preemption

Process Blocking

The process is blocked if it is waiting for some event to occur. This event may be I/O as the I/O events are executed in the main memory and don't require the processor. After the event is complete, the process again goes to the ready state.

A diagram that demonstrates process blocking is as follows –



Process Blocking

Process Termination

After the process has completed the execution of its last instruction, it is terminated. The resources held by a process are released after it is terminated.

A child process can be terminated by its parent process if its task is no longer relevant. The child process sends its status information to the parent process before it terminates. Also, when a parent process is terminated, its child processes are terminated as well as the child processes cannot run if the parent processes are terminated.

b) Briefly write about Multithreading models.

A. s. User threads are supported above the kernel and are managed without kernel support, whereas kernel threads are supported and managed directly by the operating system. Virtually all contemporary operating systems-including Wiridows XP, Linux, Mac OS X, Solaris, and Tru64 UNIX (formerly Digital UNIX)-support kernel threads. Ultimately, a relationship must exist between user threads and kernel threads. In this section, we look at three common ways of establishing such a relationship.

1 Many-to-One Model

The many-to-one model (Figure 4.5) maps many user-level threads to one kernel thread.

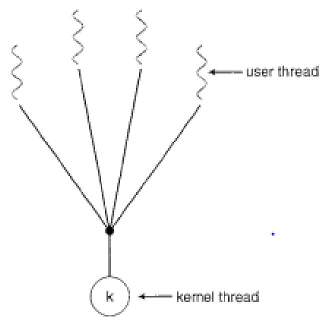


Figure 4.5 Many-to-one model.

2 One-to-One Model

The one-to-one model maps each user thread to a kernel thread.

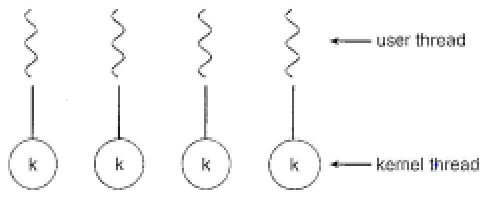


Figure 4.6 One-to-one model.

3 Many-to-Many Model

The many-to-many model multiplexes many user-level threads to a smaller or equal number of kernel threads.

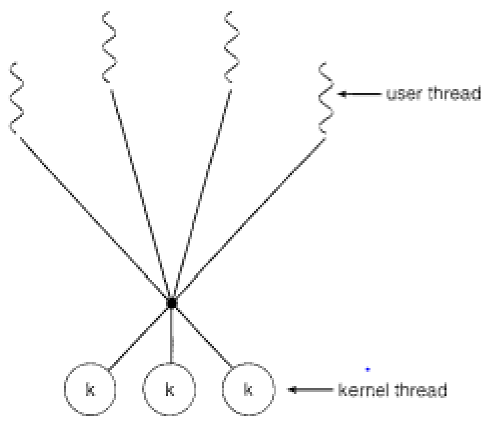


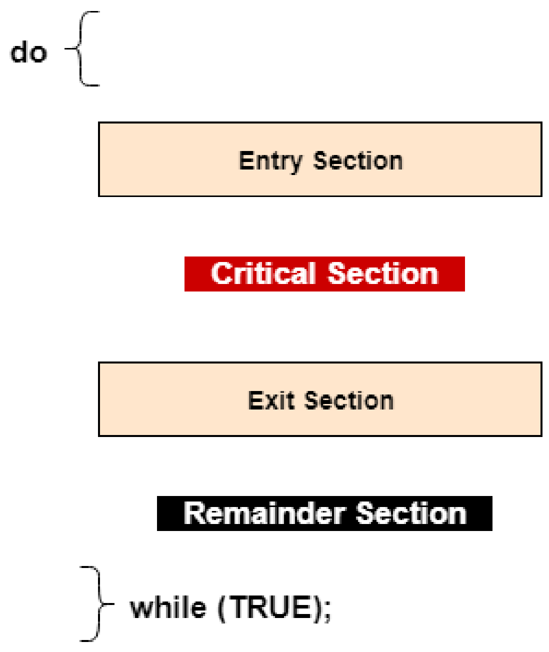
Figure 4.7 Many-to-many model.

UNIT-III (CO-3)

5. a) What is Critical Section Problem, what are the requirements for solving it and what are the approaches to handle it?

A. The critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical sections.

A diagram that demonstrates the critical section is as follows −



In the above diagram, the entry section handles the entry into the critical section. It acquires the resources needed for execution by the process. The exit section handles the exit from the critical section. It releases the resources and also informs the other processes that the critical section is free.

Solution to the Critical Section Problem

The critical section problem needs a solution to synchronize the different processes. The solution to the critical section problem must satisfy the following conditions –

- **Mutual Exclusion**

  Mutual exclusion implies that only one process can be inside the critical section at any time. If any other processes require the critical section, they must wait until it is free.

- **Progress**

  Progress means that if a process is not using the critical section, then it should not stop any other process from accessing it. In other words, any process can enter a critical section if it is free.

- **Bounded Waiting**

  Bounded waiting means that each process must have a limited waiting time. Itt should not wait endlessly to access the critical section.

b) How to prevent a deadlock? Explain.

A. r a deadlock to occur, each of the four necessary conditions must hold. By ensuring that at least one of these conditions cannot hold, we can prevent the occurrence of a deadlock. We elaborate on this approach by examining each of the four necessary conditions separately.

- **Mutual Exclusion:**The mutual-exclusion condition must hold for nonsharable resources.
- **Hold and Wait:**To ensure that the hold-and-wait condition never occurs in the system, we must guarantee that, whenever a process requests a resource, it does not hold any other resources
- **No Preemption**: The third necessary condition for deadlocks is that there be no preemption of resources that have already been allocated. To ensure that this condition does not hold, we can use the following protocol.
- **Circular Wait:**The fourth and final condition for deadlocks is the circular-wait condition. One way to ensure that this condition never holds is to impose a total ordering of all resource types and to require that each process requests resources in an increasing order of enumeration.

6 a) Semaphores in brief

Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization.

The definitions of wait and signal are as follows –

- **Wait**

  The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

```
wait(S)
```

```
{
  while(S<=0);


  S--;
}
```

- **Signal**

  The signal operation increments the value of its argument S.

```
signal(S)
{
  S++;
}
```

Types of Semaphores

There are two main types of semaphores i.e. counting semaphores and binary semaphores. Details about these are given as follows −

- **Counting Semaphores**

  These are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources. If the resources are added, semaphore count automatically incremented and if the resources are removed, the count is decremented.

- **Binary Semaphores**

  The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0. It is sometimes easier to implement binary semaphores than counting semaphores.

Advantages of Semaphores

Some of the advantages of semaphores are as follows −

- Semaphores allow only one process into the critical section. They follow the mutual exclusion principle strictly and are much more efficient than some other methods of synchronization.
- There is no resource wastage because of busy waiting in semaphores as processor time is not wasted unnecessarily to check if a condition is fulfilled to allow a process to access the critical section.
- Semaphores are implemented in the machine independent code of the microkernel. So they are machine independent.

b) Discuss Bankers Algorithm With example

A. The resource-allocation-graph algorithm is not applicable to a resourceallocation system with multiple instances of each resource type. The deadlockavoidance algorithm that we describe next is applicable to such a system but is less efficient than the resource-allocation graph scheme. This algorithm is commonly known as the banker's algorithm. The name was chosen because the algorithm. could be used in a banking system to ensure that the bank never allocated its available cash in such a way that it could no longer satisfy the needs of all its customers.Several data structures must be maintained to implement the banker's algorithm. These data structures encode the state of the resource-allocation system. We need the following data structures, where n is the number of processes in the system and m is the number of resource types:

- Available.
- Max.
- Allocation.
- Need.

Example: Consider a system that contains five processes P1, P2, P3, P4, P5 and the three resource types A, B and C. Following are the resources types: A has 10, B has 5 and the resource type C has 7 instances.

Process	**Allocation**			**Max**		
	A	**B**	**C**	**A**	**B**	**C**
P1	0	1	0	7	5	3
P2	2	0	0	3	2	2
P3	3	0	2	9	0	2
P4	2	1	1	2	2	2
P5	0	0	2	4	3	3

Answer the following questions using the banker's algorithm:

1. What is the reference of the need matrix?
2. Determine if the system is safe or not.
3. What will happen if the resource request (1, 0, 0) for process P1 can the system accept this request immediately?

Ans. 2: Context of the need matrix is as follows:

Need [i] = Max [i] - Allocation [i]
Need for P1: (7, 5, 3) - (0, 1, 0) = 7, 4, 3
Need for P2: (3, 2, 2) - (2, 0, 0) = 1, 2, 2
Need for P3: (9, 0, 2) - (3, 0, 2) = 6, 0, 0

Need for P4: (2, 2, 2) - (2, 1, 1) = 0, 1, 1
Need for P5: (4, 3, 3) - (0, 0, 2) = 4, 3, 1

Process	**Need**		
	A	**B**	**C**
P1	7	4	3
P2	1	2	2
P3	6	0	0
P4	0	1	1
P5	4	3	1

Hence, we created the context of need matrix.

Ans. 2: Apply the Banker's Algorithm:

Available Resources of A, B and C are 3, 3, and 2.

Now we check if each type of resource request is available for each process.

Step 1: For Process P1:

Need <= Available

7, 4, 3 <= 3, 3, 2 condition is **false**.

So, we examine another process, P2.

Step 2: For Process P2:

Need <= Available

1, 2, 2 <= 3, 3, 2 condition **true**

New available = available + Allocation

(3, 3, 2) + (2, 0, 0) => 5, 3, 2

Similarly, we examine another process P3.

Step 3: For Process P3:

P3 Need <= Available

6, 0, 0 < = 5, 3, 2 condition is **false**.

Similarly, we examine another process, P4.

Step 4: For Process P4:

P4 Need <= Available

0, 1, 1 <= 5, 3, 2 condition is **true**

New Available resource = Available + Allocation

5, 3, 2 + 2, 1, 1 => 7, 4, 3

Similarly, we examine another process P5.

Step 5: For Process P5:

P5 Need <= Available

4, 3, 1 <= 7, 4, 3 condition is **true**

New available resource = Available + Allocation

7, 4, 3 + 0, 0, 2 => 7, 4, 5

Now, we again examine each type of resource request for processes P1 and P3.

Step 6: For Process P1:

P1 Need <= Available

7, 4, 3 <= 7, 4, 5 condition is **true**

New Available Resource = Available + Allocation

7, 4, 5 + 0, 1, 0 => 7, 5, 5

So, we examine another process P2.

Step 7: For Process P3:

P3 Need <= Available

6, 0, 0 <= 7, 5, 5 condition is true

New Available Resource = Available + Allocation

7, 5, 5 + 3, 0, 2 => 10, 5, 7

UNIT-IV (CO-4)

7. a) Explain the concept of paging in detail.

A. Paging is a memory-management scheme that permits the physical address space a process to be non-contiguous. Paging avoids external fragmentation and the need for compaction. It also solves the considerable problem of fitting memory chunks of varying sizes onto the backing store; most memorymanagement schemes used before the introduction of paging suffered from this problem. The problem arises because, when some code fragments or data residing in main memory need to be swapped out, space must be found on the backing store. The backing store has the same fragmentation problems discussed in connection with main memory, but access is much slower, so compaction is impossible. Because of its advantages over earlier methods, paging in its various forms is used in most operating systems.

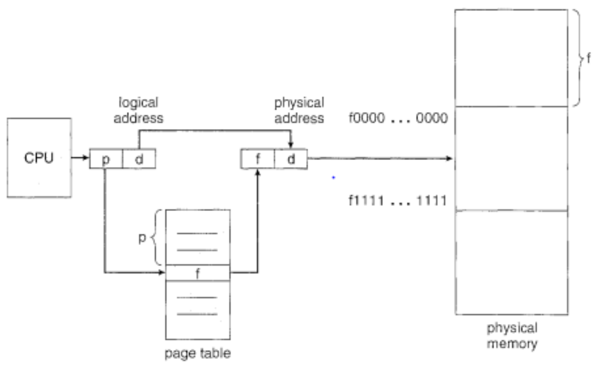


Figure 8.7 Paging hardware.

b) What are the common attributes and operations of a File?

A. A file is a named collection of related information that is recorded on secondary storage.

File Attributes

Name. The symbolic file name is the only information kept in humanreadable form.

Identifier. This unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file.

Type. This information is needed for systems that support different types of files.

Location. This information is a pointer to a device and to the location of the file on that device.

Size. The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.

Protection. Access-control information determines who can do reading, writing, executing, and so on.

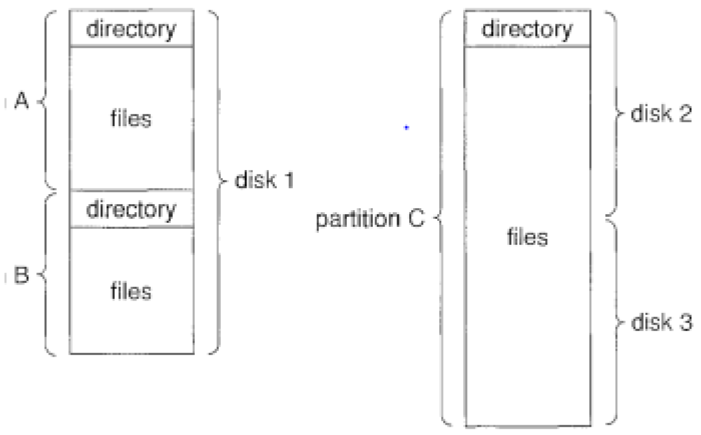
Time, date, and user identification. This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

File Operations

- Creating a file.
- Writing a file.
- Reading a file
- Repositioning within a file.
- Deleting a file.
- Truncating a file.

8 a) Briefly explain the Directory Structure in File System.

A. There are typically thousand, millions, and even billions of files within a computer. Files are stored on random-access storage devices, including hard disks, optical disks, and solid state (memory-based) disks. A storage device can be used in its entirety for a file system. It can also be subdivided for finer-grained control.Partitioning is useful for limiting the sizes of individual file systems, putting multiple file-system types on the same device, or leaving part of the device available for other uses, such as swap space or unformatted disk



Directory Overview

- Search for a file
- Create a file.
- Delete a file.
- List a directory.
- Rename a file
- Traverse the file system

3 Single-level Directory

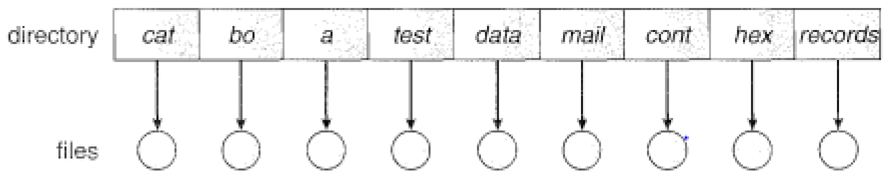


Figure 10.8 Single-level directory.

Two-Level Directory

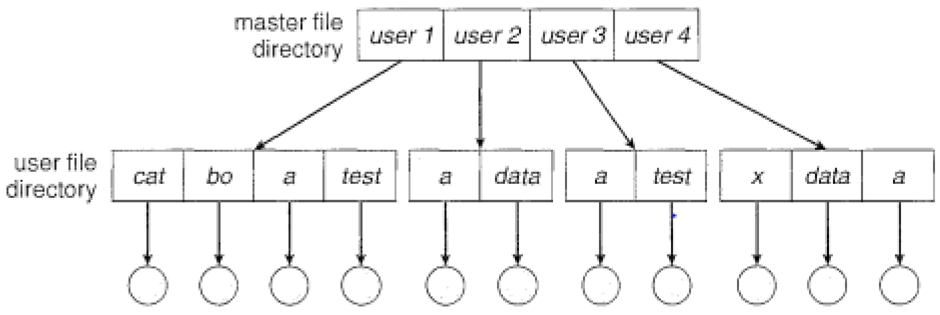


Figure 10.9 Two-level directory structure.

b) Analyze FIFO and LRU page replacement algorithm in detail.

A. **FIFO Page Replacement**

The simplest page-replacement algorithm is a first-in, first-out (FIFO) algorithm. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. Notice that it is not strictly necessary to record the time when a page is brought in. We can create a FIFO queue to hold all pages in memory. We replace the page at the head of the queue. When a page is brought into memory, we insert it at the tail of the queue.

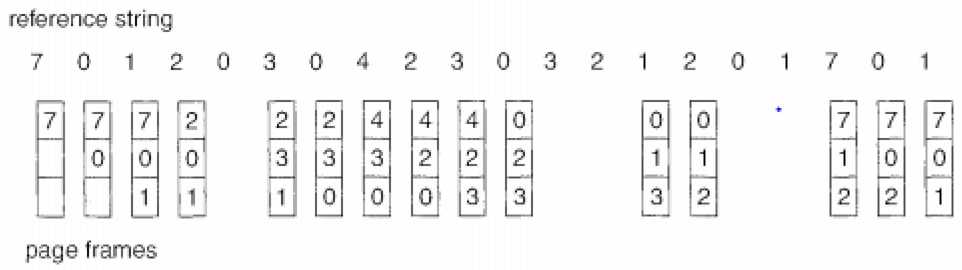


Figure 9.12 FIFO page-replacement algorithm.

The FIFO page-replacement algorithm is easy to Lmderstand and program. However, its performance is not always good. On the one hand, the page replaced may be an initialization module that was used a long time ago and is no longer needed. On the other hand, it could contain a heavily used variable that was initialized early and is in constant use.

LRU Page Replacement

If the optimal algorithm is not feasible, perhaps an approximation of the optimal algorithm is possible. The key distinction between the FIFO and OPT algorithms

LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses the page that has not been used for the longest period of time. We can think of this strategy as the optimal page-replacement algorithm looking backward in time, rather than forward.

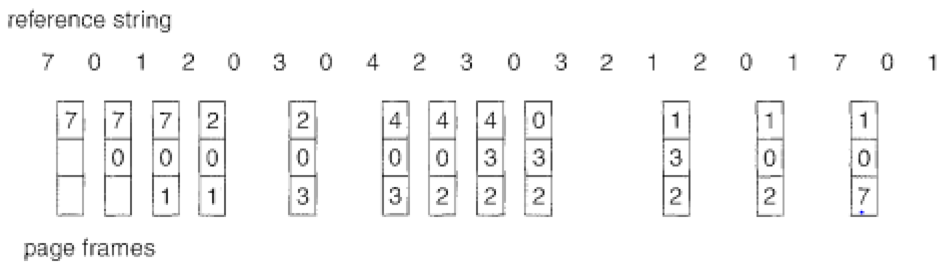


Figure 9.15 LRU page-replacement algorithm.

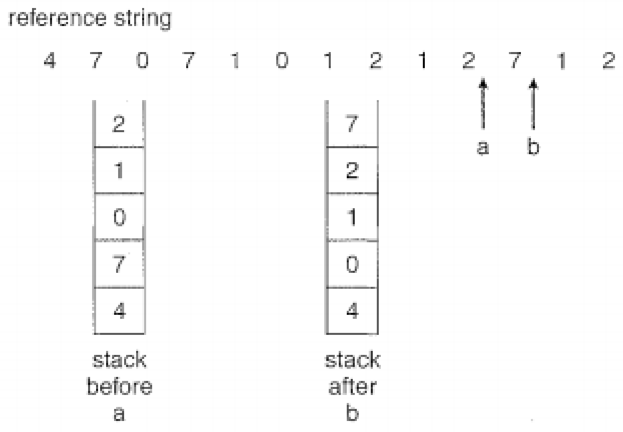


Figure 9.16 Use of a stack to record the most recent page references.

UNIT-V (CO-5)

9. a) Explain Magnetic Disk structure in Mass Storage System

A. Magnetic Disks

Magnetic disk provides the bulk of secondary storage for modern computer systems. Conceptually, disks are relatively simple (Figure 12.1). Each disk platter has a flat circular shape, like a CD.

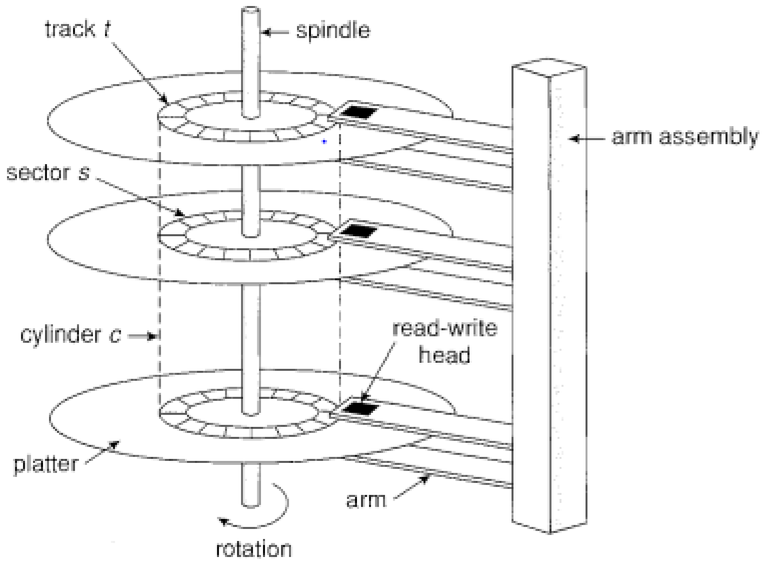


Figure 12.1 Moving-head disk mechanism.

[illegible]

The set of tracks that are at one arm position makes up a There may be thousands of concentric cylinders in a disk drive, and each track may contain hundreds of sectors. The storage capacity of common disk drives is measured gigabytes.

When the disk is in use, a drive motor spins it at high speed. Most drives rotate 60 to 200 times per second. Disk speed has two parts. The is the rate at which data flow between the drive and the computer. The sometimes called the consists of the time necessary to move the disk arm to the desired cylinder, called the and the time necessary for the desired sector to rotate to the disk head, called the Typical disks can transfer several megabytes of data per second, and they seek times and rotational latencies of several milliseconds.

b) Summarized difference program threats.

Processes, along with the kernel, are the only means of accomplishing work on a computer. Therefore, writing a program that creates a breach of security, or causing a normal process to change its behavior and create a breach, is a common goal of crackers. In fact even most nonprogram security events have as their goal causing a program threat.

Trojan Horse

Trap Door

Logic Bomb

Stack and Buffer Overflow

Viruses

10. a) Evaluate any two Disk Scheduling algorithm with example

Disk Scheuling:

One of the responsibilities of the operating system is to use the hardware efficiently. For the disk drives, meeting this responsibility entails having

FCFS Scheduling

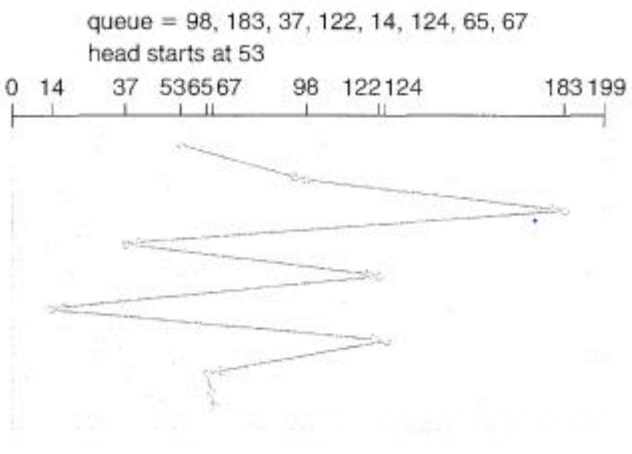


Figure 12.4 FCFS disk scheduling.

SSTF Scheduling

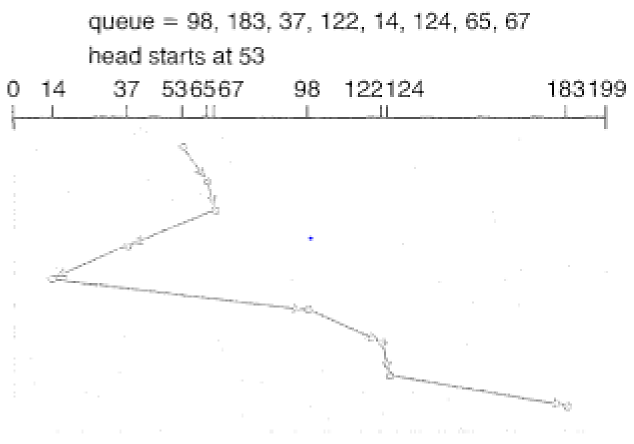


Figure 12.5 SSTF disk scheduling.

SCAN Scheduling

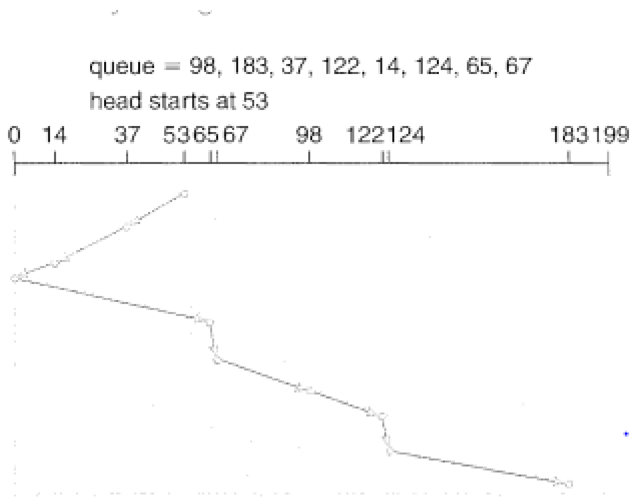


Figure 12.6 SCAN disk scheduling.

C-SCAN Scheduling

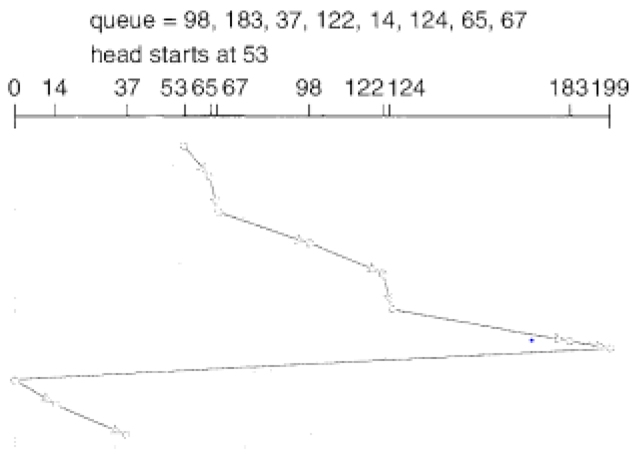
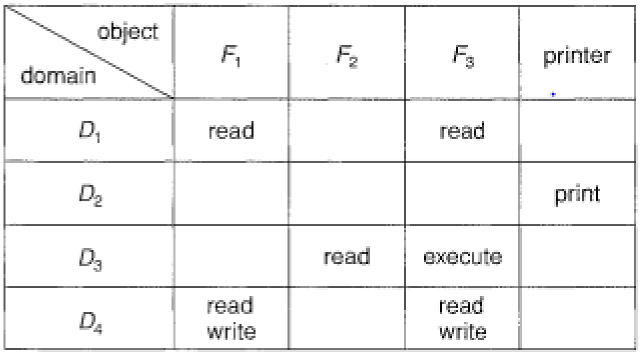


Figure 12.7 C-SCAN disk scheduling.

b) Review several methods of implementing Access Matrix.

A. The rows of the access matrix represent domains, and the columns represent objects. Each entry in the matrix consists of a set of access rights. Because the column defines objects explicitly, we can omit the object name from the access right



object domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Figure 14.3 Access matrix.

The access matrix can implement policy decisions concerning protection. The policy decisions involve which rights should be included in the (i,j)th entry. We must also decide the domain in which each process executes. This last policy is usually decided by the operating system. The users normally decide the contents of the access-matrix entries. When a user creates a new object Oi, the column Oi is added to the access matrix with the appropriate initialization entries, as dictated by the creator. The user may decide to enter some rights in some entries in cohum1 j and other rights in other entries, as needed.

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Figure 14.4 Access matrix of Figure 14.3 with domains as objects.

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

Figure 14.5 Access matrix with *copy* rights.