# VTU Question paper solutions

## UNIT 1

**1. Explain fundamental difference between i) N/w OS and distributed OS ii) web bas ed and embedded computing. Jun 15/Jun14**

Network OS is used to manage Networked computer systems and create, maintain and transfer files in that Network. Distributed OS is also similar to Networked OS but in addition to it the platform on which it is running should have high configuration such as more capacity RAM, High speed Processor. The main difference between the DOS and the NOS is the transparent issue: Transparency: - How aware are users of the fact that multiple computers are being used?

• NETW OR K OS: - Users are aware where resources are located

- Network OS is built on top of centralized OS. Handles interfacing and coordination between local OS.

• DISTRIB UTED OS: - Designed to control and optimize operations and resources in distributed system.

web based computing is nothing but a transaction done by through online, n the trade n non tradingconcern done their business on through online n maintaining book of accounts through internet.An embedded system is a computer system designed to perform one or a few dedicated functions oftenwith real-time computing constraints.

**2. What do you mean by cooperating process? Describe its four advantages.**
**Jun 14/Jun 15**

**Cooperating Processes**

- **Independent** process cannot affect or be affected by the execution of another process
- **Cooperating** process can affect or be affected by the execution of another process Advantages of process cooperation
- Information sharing
- Computation speed-up
- Modularity

**3.What are different categories of system programs? Explain. Jun 14/Jan 15**

System programs provide a convenient environment for program development and execution. The canbe divided into:

a. File manipulation
b. Status information
c. File modification
d. Programming language support
e. Program loading and execution
f. Communications
g. Application programs
h. Provide a convenient environment for program development and execution
i. Some of them are simply user interfaces to system calls; others are considerably more complex

- File management - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- Status information
- Some ask the system for info - date, time, amount of available memory, disk space, number of users
- Others provide detailed performance, logging, and debugging information
- Typically, these programs format and print the output to the terminal or other output devices
- Some systems implement a registry - used to store and retrieve configuration information
- File modification
- Text editors to create and modify files
- Special commands to search contents of files or perform transformations of the text
- Programming- language support - Compilers, assemblers, debuggers and interpreters sometimes provided
- Program loading and execution- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language

- Communications - Provide the mechanism for creating virtual connections among processes, users, and computer systems
- ALLOW USERS TO SEND MESSAGES TO ONE ANOTHER'S SCREENS, BROWSE WEB PAGES, SEND electronicmail messages, log in remotely, transfer files from one machine to another.

**4. Define OS. Discuss its role from different perspectives. Jan16/Jun 14**

- An OS is an intermediary between the user of the computer & the computer hardware.
- It provides a basis for application program & acts as an intermediary between user of computer & computer hardware.
- The purpose of an OS is to provide a environment in which the user can execute the program in a convenient & efficient manner.
- OS is an important part of almost every computer systems.

  **User Views**:- The user view of the computer depends on the interface used.
- SOME USERS MAY USE PC'S. In this the system is designed so that only one user canutilize the resources and mostly for ease of use where the attention is mailnly onperformances and not on the resource utilization.
- Some users may use a terminal connected to a mainframe or minicomputers.
- Other users may access the same computer through other terminals. These users may share resources and exchange information. In this case the OS is designed to maximize resource utilization- so that all available CPU time, memory & I/O are used efficiently.
- Other users may sit at workstations, connected to the networks of other workstation and servers. In this case OS is designed to compromise between individual visibility &resource utilization.

  **System Views**:-We can view system as resource allocator i.e. a computer system has many resourcesthat may be used to solve a problem. The OS acts as a manager of these resources. TheOS must decide how to allocate these resources to programs and the users so that it canoperate the computer system efficiently and fairly.
- A different view of an OS is that it need to control various I/O devices & user programsi.e. an OS is a control program used to manage the execution of user program to prevent errors and improper use of the computer.

- Resources can be either CPU Time, memory space, file storage space, I/O devices and so on.

**5. List different services of OS. Explain. Jun15/jun16**

An OS provides services for the execution of the programs and the users of such programs.

The services provided by one OS may be different from other OS. OS makes the programming task easier.

The common services provided by the OS are

1. Program Execution:- The OS must able to load the program into memory & run that program. The program must end its execution either normally or abnormally.

2. I/O Operation:- A program running may require any I/O. This I/O may be a file or a specific device users cant control the I/O device directly so the OS must provide a means for controlling I/O devices.

3. File System Interface:- Program need to read or write a file. The OS should provide permission for the creation or deletion of files by names.

4. Communication:- In certain situation one process may need to exchange information with another process. This communication May takes place in two ways.

a. Between the processes executing on the same computer.

b. Between the processes executing on different computer that are connected by anetwork.

This communication can be implemented via shared memory or by OS.

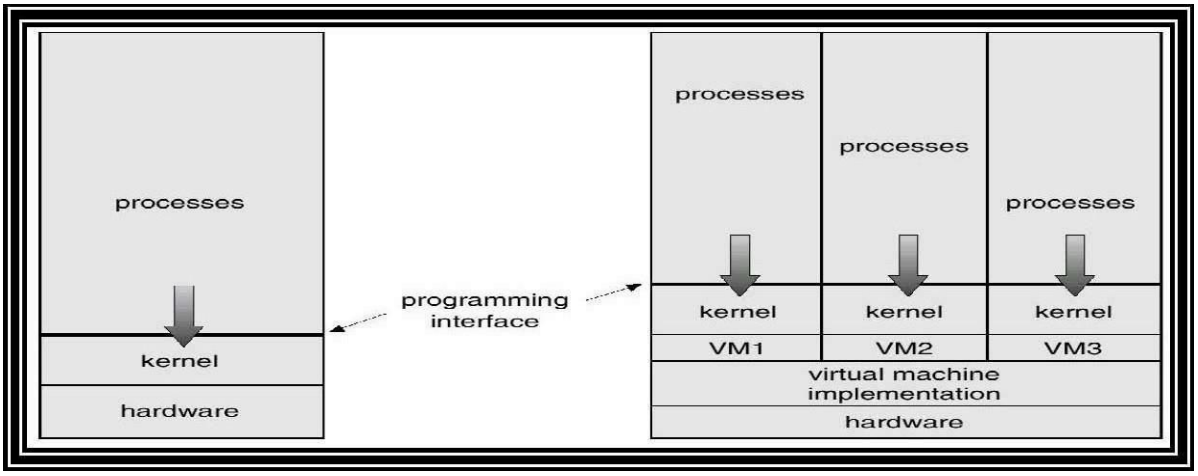
5. Error Detection:- Errors may occur in CPU, I/O devices or in M/y H/w. The OS constantly needs to be aware of possible errors. For each type of errors the OS should take appropriate actions to ensure correct & consistent computing.

**6. Explain the concept of virtual machines. Bring out its advantages. Jul 15/Jun 13/Jun16**

**Virtual Machines**

- A virtual machine takes the layered approach to its logical conclusion. It treats hardware andthe operating system kernel as though they were all hardware
- A virtual machine provides an interface *identical* to the underlying bare hardware
- The operating system host creates the illusion that a process has its own processor .

- Each guest provided with a (virtual) copy of underlying computerOPERATOR'S CONsole.




**Advantages and Disadvantages of VirtualMachines**

•The virtual- machine concept provides complete protection of system resources since each virtualmachine isisolated from all other virtual machines.This isolation, however, permits no direct sharing of resources.

•A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtualmachine, insteadof on a physical machineand so does not disrupt normal system operation.

•The virtual machineconcept is difficult toimplement due to the effort required to provide an*exact* duplicateto the underlying machine.

**7. Distinguish among following terminologies : Multiprogramming systems, multitasking systems,multiprocessor systems. Jun 14/Jan 15**

**Multi programmed System:-**

- If there are two or more programs in the memory at the same time sharing the processor, this is referred as multi programmed OS.
- It increases the CPU utilization by organizing the jobs so that the CPU will always have one job to execute.
- Jobs entering the systems are kept in memory.
- OS picks the job from memory & executes it.
-

**II. Multi Processor Systems:-**

- Multi processor systems include more than one processor in close communication.
- They share computer bus, the clock, m/y & peripheral devices.
- Two processes can run in parallel.

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

  **Response time** should be < 1 second
- Each user has at least one program executing in memory [**process**
- If several jobs ready to run at the same time [ **CPU scheduling**
- IF PROCESSES DON'T FIT IN MEMORY, **swapping** moves them in and out to run

**8. What is distributed operating system? What are the advantages of distributed operating system? Jun15/Jan 16**

A **distributed operating system** is a software over a collection of independent, networked, communicating, and physically separate computational nodes. Individual nodes each hold a specific software subset of the global aggregate operating system.

Advantages of distributed operating systems are:

1.Major breakthrough in microprocessor technology.

2. Incremental growth.

3. Reliability.

**9. What are system calls? With examples explain different categories of system call? Jan 15/Jun 15**

System provides interface between the process & the OS.

The calls are generally available as assembly language instruction & certain system allow system calls to be made directly from a high level language program.

Several languages have been defined to replace assembly language program.

System calls occuring different ways depending on the computer. Sometime more information is

needed to identify the desired system call. The exact type & amount of information needed may vary according to the Particular OS & call.

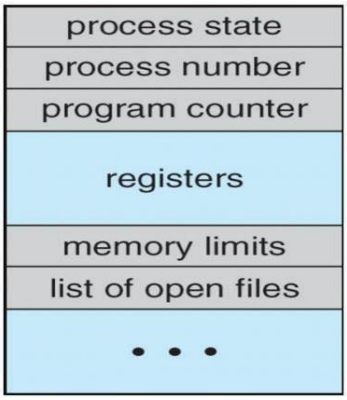
System calls may be grouped roughly into 5 categories

- .Process control.
- .File management.
- .Device management.
- .Informationmaintenance.
- .Communication.

## UNIT 2

**1. What do you mean by PCB? Where is it used? What are its contents? Explain.**

**Process Control Block (PCB) Jun 16/Jun 14**




Information associated with each process

- Process state
- Program counter
- CPU registers
- CPU scheduling information
- Memory-management information
- Accounting information
- I/O status information

A process in an operating system is represented by a data structure known as a process control block ( PCB) or process descriptor. The PCB contains important information about the specific process including

- The current state of the process i.e., whether it is ready, running, waiting, or whatever.
- Unique identification of the process in order to track "which is which" information.
- A pointer to parent process.
- Similarly, a pointer to child process (if it exists).

- The priority of process (a part of CPU scheduling information).
- Pointers to locate memory of processes.
- A register save area.

The PCB is a certain store that allows the operating systems to locate key information about a process.Thus, the PCB is the data structure that defines a process to the operating systems.

**2. Explain direct and indirect communications of message passing systems.**

**Direct Communication Jun 14/ Jan 15**

- Processes must name each other explicitly:
- **send** (*P*, *message*) – send a message to process P
- **receive**(*Q*, *message*) – receive a message from process Q
- Properties of communication link
- Links are established automatically
- A link is associated with exactly one pair of communicating processes
- Between each pair there exists exactly one link
- The link may be unidirectional, but is usually bi-directional

**Indirect Communication**

- Messages are directed and received from mailboxes (also referred to as ports)
- Each mailbox has a unique id
- Processes can communicate only if they share a mailbox
- Properties of communication link
- Link established only if processes share a common mailbox
- A link may be associated with many processes
- Each pair of processes may share several communication links
- Link may be unidirectional or bi-directional
- Operations
- create a new mailbox
- send and receive messages through mailbox
- destroy a mailbox

- Primitives are defined as:
- **send**(*A, message*) – send a message to mailbox A
- **receive**(*A, message*) – receive a message from mailbox A
- Mailbox sharing
- *P1, P2,* and *P3* share mailbox A

**3.Explain the diffrence between long term and short term and medium term schedulers Jan16/jun16**

1. **Long-term scheduler (or job scheduler) –** selects which processes should be brought intothe ready queue.

2. **Short-term scheduler (or CPU scheduler) –** selects which process should be executed nextand allocates CPU.

**3. Medium-term schedulers**

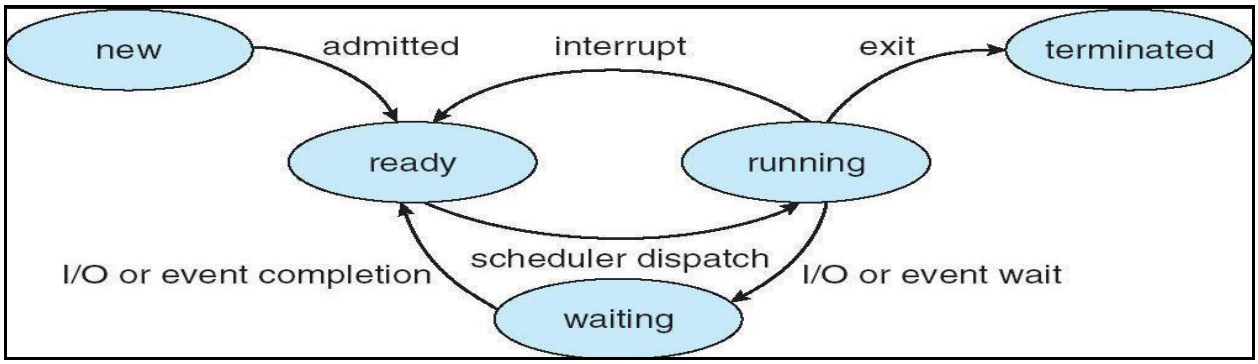
- Short-term scheduler is invoked very frequently (milliseconds) (must be fast)
- Long-term scheduler is invoked very infrequently (seconds, minutes) (may be slow)
- The long-term scheduler controls the *degree of multiprogramming*

Processes can be described as either:

- I/O-bound process – spends more time doing I/O than computations, many short CPU bursts
- CPU-bound process – spends more time doing computations; few very long CPU bursts

**4. What is process? Draw and explain process state diagram. Jan 15/Jun15**

Process is a dynamic entity. A process is a sequence of instruction execution process exists in alimited span of time. Two or more process may execute the same program by using its owndata & resources.




- **New State** The process being created.
- **Terminated State** The process has finished execution.
- **Blocked (waiting) State** When a process blocks, it does so because logically it cannotcontinue, typically because it is waiting for input that is not yet available. Formally, a processis said to be blocked if it is waiting for some event to happen (such as an I/O completion)before it can proceed. In this state a process is unable to run until some external event happens.
- **Running State** A process is said t be running if it currently has the CPU, that is, actuallyusing the CPU at that particular instant.
- **Ready State** A process is said to be ready if it use a CPU if one were available . It is runnablebut temporarily stopped to let another process run.

**5. Define IPC.whatare different methods used for logical implementations of message passing systems. Jun 14/ Jan 15**

There are two types of communication models

**Direct Communication**

1. Processes must name each other explicitly:

x send (*P*, *message*) – send a message to process P x receive(*Q, message*) – receive a message from process Q

2. Properties of communication link x Links are established automatically x A link is associated with exactly one pairof communicating processes x Between each pair there exists exactly one link x usually bi-directional

**Indirect Communication**

1. Messages are directed and received from mailboxes (also referred to as ports) x Each mailbox has a unique id

**6. Discuss common ways of establishing relationship between user and kernel thread**

**User-Level Threads**

1. Thread management done by user-level threads library
2. Three primary thread libraries:

-> POSIX Pthreads

-> Win32 threads -

> Java threads

**Kernel-Level Threads**

1. Supported by the Kernel
2. Examples

->Windows XP/2000 ->Solaris

->Linux

->Tru64 UNIX

->Mac OS X

In this method, the kernel knows about and manages the threads. No runtime system is needed in thiscase. Instead of thread table in each process, the kernel has a thread table that keeps track of all threadsin the system. In addition, the kernel also maintains the traditional process table to keep track ofprocesses. Operating Systems kernel provides system call to create and manage threads.

**7. Explain multithreading models. Jan 16.**

**TYPES OF MULTITHREADING MODELS**

•Many-to-One

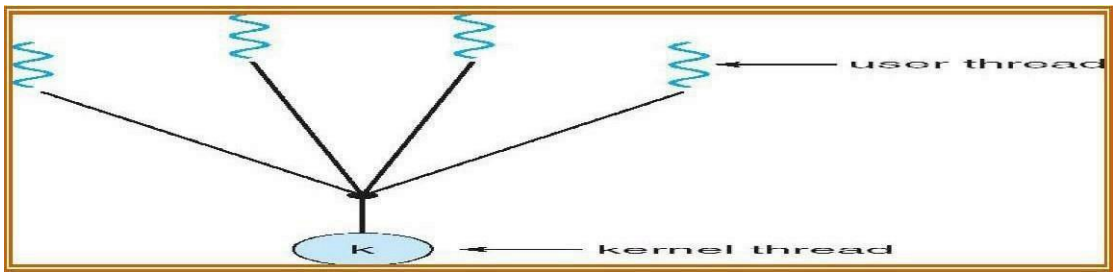
•One-to-One

•Many-to-Many

**Many-to-One**

Many user-level threads mapped to single kernel thread

->Examples: ->Solaris Green Threads, ->GNU Portable Threads



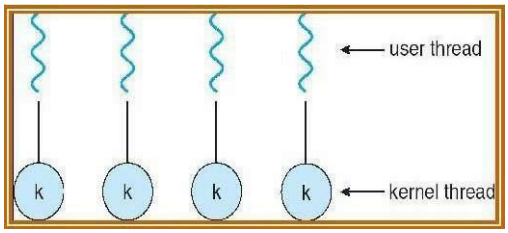

**One-to-One**

1. Each user-level thread maps to kernel thread

•Examples Windows NT/XP/2000

•Linux

•Solaris 9 and later

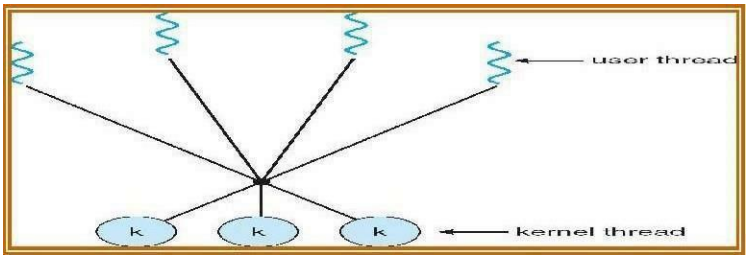



**Many-to-ManyModel**

1.Allows many user level threads to be mapped to many kernel threads.

2.Allows the operating system to create a sufficient number of kernel threads. 3.Solaris prior toversion9.

4.WindowsNT/2000 with the *ThreadFiber*package.



**UNIT 3:**

**1. What are semaphores? Explain two primitive semaphore operations. What are its advantages? Jun 15/jan16**

A semaphore is a protected variable whose value can be accessed and altered only by the operations Pand V and initialization operation called 'Semaphoiinitislize'.

Binary Semaphores can assume only the value 0 or the value 1 counting semaphores also calledgeneral semaphores can assume only nonnegative values.

The P (or wait or sleep or down) operation on semaphores S, written as P(S) or wait (S), operates asfollows:

P(S): IF S > 0

THEN S := S - 1

ELSE (wait on S)

The V (or signal or wakeup or up) operation on semaphore S, written as V(S) or signal (S), operatesas follows:

V(S): IF (one or more process are waiting on S)

THEN (let one of these processes proceed)

ELSE S := S +1

**2. Explain any one synchronization problem for testing newly proposed sync scheme Jun 14/jun16**

**Classical Problems of Synchronization**

1. Bounded-Buffer Problem
2. Readers and Writers Problem
3. Dining-Philosophers Problem

**Bounded-Buffer Problem**

1. *N* buffers, each can hold one item
2. Semaphore mutex initialized to the value 1
3. Semaphore full initialized to the value 0

4. Semaphore empty initialized to the value N.

5. The structure of the producer process

```
while (true) {
        // produce an
    item wait (empty);
    wait (mutex);
        // add the item to the
    buffer signal (mutex);
    signal (full);
}
```

6. The structure of the consumer process

```
    while (true) {
        wait (full);
        wait (mutex);
            // remove an item from
        buffer signal (mutex);
        signal (empty);
    // consume the removed item
}
```

**3. Explain three requirements that a solution to critical –section problem must satisfy. Jun 15/ Jun 16**

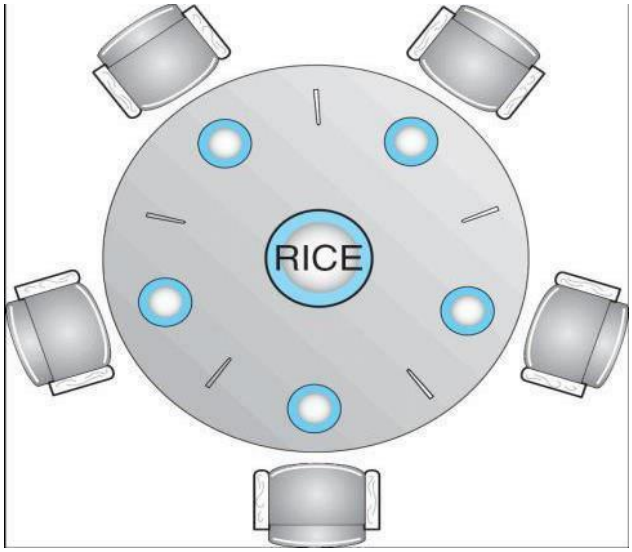
Solution to Critical-Section Problem

1. Mutual Exclusion - If process Pi is executing in its critical section, then no other processes can be executing in their critical sections

2. Progress - If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely

3. Bounded Waiting - A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted

**4. State dining PHILOSO PH ER'S problem and give a solution using semaphores. Write structure of philosopher. Jun 15**




**Dining-Philosophers Problem**

1. Shared data

oBowl of rice (data set)

oSemaphore chopstick [5] initialized to

1 2. The structure of Philosopher *i*:

```
While (true) {
        wait ( chopstick[i] );
        wait ( chopStick[ (i + 1) % 5]
                ); // eat
        signal ( chopstick[i] );
        signal (chopstick[ (i + 1) % 5] );
```

// think

}

**5. What do you mean by binary semaphore and counting semaphore? With C struct, explain implementation of wait() and signal. Jun 14/Jan 14. Semaphore as General Synchronization Tool. Jun 14/Jan 15**

- Counting semaphore – integer value can range over an unrestricted domain
- Binary semaphore – integer value can range only between 0
- and 1; can be simpler to implement
- Also known as mutexlocksnCan implement a counting semaphore S as a binary semaphore
- Provides mutual exclusionSemaphoremutex;

// initialized to do {

wait (mutex);

// Critical Section

signal (mutex);

// remainder section

} while (TRUE);

**6. Describe term monitor. Explain solution to dining philosophers. Jun 14.**

1. High-level abstraction that provides a convenient and effective mechanism for process synchronization

2. Only one process may be active within the monitor at a time

monitor monitor-name

{

// shared variable declarations

PROCED UR E P1 (…) { …. }

…

PROCED UR EPN (…) {…… }

```
INITIA L IZ A TIO N  CODE  ( ….) { … }
…
}
}
```

**Solution to Dining**

**Philosophers** monitor DP

```
{
enum { THINKING; HUNGRY, EATING) state [5] ;
condition self [5];
void pickup (int i) {
state[i]                =
HUNGRY; test(i);
if (state[i] != EATING) self [i].wait;
}
void putdown (int i) {
state[i] = THINKING;
// test left and right
neighbors test((i + 4) % 5);
test((i + 1) % 5);
}
void test (int i) {
if ( (state[(i + 4) % 5] != EATING)
&& (state[i] == HUNGRY) &&
(state[(i + 1) % 5] != EATING) )
{ state[i] = EATING ;
self[i].signal () ;
}
}
initialization_code() {
for (int i = 0; i < 5; i++)
```

state[i] = THINKING;

}

}

->Each philosopher *I* invokes the operations pickup()

and putdown() in the following sequence:

dp.pickup (i)

EAT dp.putdown (i)

**7. Explain synchronization? Jun 14/Jan 15**

**Synchronization Hardware**

- Many systems provide hardware support for critical section code
- Uniprocessors – could disable interrupts
- Currently running code would execute without preemption
- Generally too inefficient on multiprocessor systems
  - Operating systems using this not broadly scalable
- Modern machines provide special atomic hardware instructions
  - Atomic = non-interruptable
- Either test memory word and set value Or swap contents of two memory words

**8. What are semaphores? Explain solution to producer-cons umer problem using semaphores Jan 16**

**Producer-Consumer Problem Using Semaphores**

The Solution to producer-consumer problem uses three semaphores, namely, full, empty and mutex.

The semaphore 'full' is used for counting the number of slots in the buffer that are full. The 'empty' forcounting the number of slots that are empty and semaphore 'mutex' to make sure that the producer andconsumer do not access modifiable shared section of the buffer simultaneously.

**Initialization**

- Set full buffer slots to 0.
  i.e., semaphore Full = 0.
- Set empty buffer slots to N.
  i.e., semaphore empty = N.
- For control access to critical section set mutex to 1. i.e., semaphore mutex = 1.

```
Producer ( )
WHILE (true)
produce-Item (
); P (empty);
P (mutex);
enter-Item ( )
V (mutex)
V (full);
Consumer ( )
WHILE
(true) P (full)
P (mutex);
remove-Item (
); V (mutex);
V (empty);
consume-Item (Item)
```

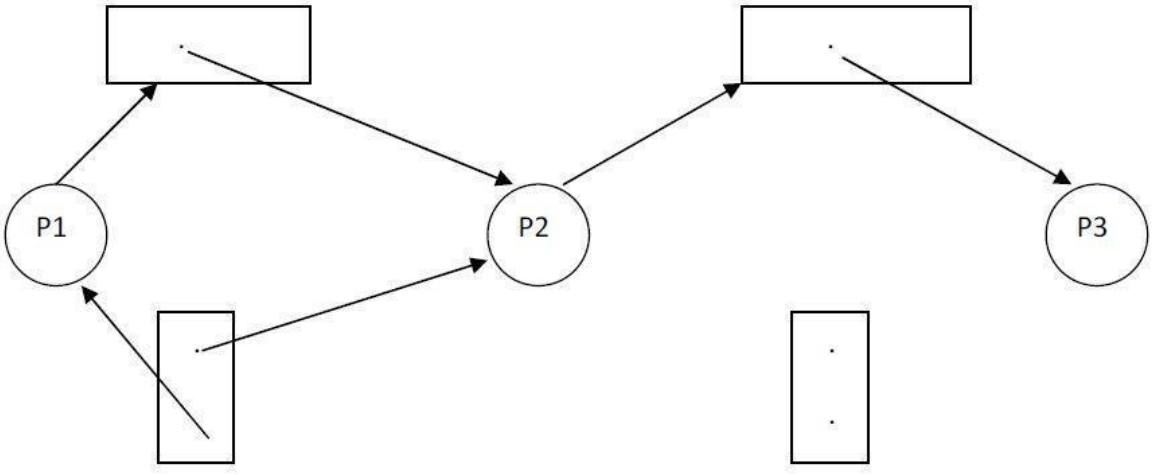
**UNIT 4**

**1. Why is deadlock state more critical than starvation? Describe resource allocation graph with a deadlock, with a cycle but no deadlock. Jun 16/Jun 14**

When processes request a resource and if the resources are not available at that time the process enters into waiting state. Waiting process may not change its state because there sources they are requested are held by other process. This situation is called deadlock.

**Resource Allocation Graph**:-

- Deadlocks are described by using a directed graph called system resource allocation graph. The graph consists of set of vertices (v) and set of edges (e).
- The set of vertices (v) can be described into two different types of nodes P={P1,P2……..PN}I .E ., SET Consisting of all active processes and R={R 1,R2 … … … .R N }I. E ., SET CONSISTING OF all resource types in the system.
- A directed edge from process Pi to resource type Rj denoted by Pi Ri indicates that Pirequested an instance of resource Rj and is waiting. This edge is called Request edge.
- A directed edge Ri Pj signifies that resource Rj is held by process Pi. This is calledAssignment edge.




- If the graph contain no cycle, then no process in the system is deadlock. If the graph contains acycle then a deadlock may exist.

If each resource type has exactly one instance than a cycle implies that a deadlock hasoccurred. If each resource has several instances then a cycle do not necessarily implies that adeadlock has occurred.

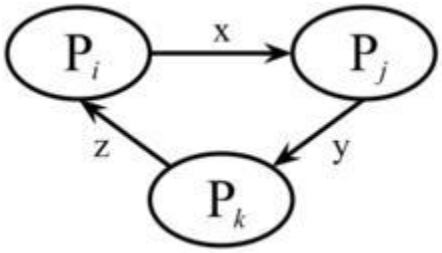
**2. What are two options for breaking deadlock? Jan 16/Jun 14**

Two methods

1)Deadlock avoidance

2)Deadlock prevention

A **wait-for graph** in computer science is a directed graph used for deadlock detection in operating Systems and relational database systems.




In computer science, a system that allows concurrent operation of multiple processes and locking of resources and which does not provide mechanisms to avoid or prevent deadlock must support a mechanism to detect deadlocks and an algorithm for recovering from them.

One such deadlock detection algorithm makes use of a wait-for graph to track which other processes a process is currently blocking on. In a wait-for graph, processes are represented as nodes, and an edge from process $P_i$ to $P_j$ implies $P_j$ is holding a resource that $P_i$ needs and thus $P_i$ is waiting for $P_j$ to release its lock on that resource. If the process is waiting for more than a single resource to become available (the trivial case), multiple edges may represent a conjunctive (and) or disjunctive (or) set of different resources or a certain number of equivalent resources from a collection. In the conjunctive case, graph cycles imply the possibility of a deadlock, whereas in the disjunctive case knots are indicative of deadlock possibility. There is no simple algorithm for detecting the possibility of deadlock in the final case.

**3. Solve the deadlock to find safe or unsafe state Jun 15 /Jan 15**

Total resources in system:A B C D:6 5 7 6

Available system resources are:A B C D:3 1 1 2

| Allocated | Maximum | Need |
|---|---|---|
| A B C D | A B C D | A B C D |
| P1 1 2 2 1 | P1 3 3 2 2 | P1 2 1 0 1 |
| P2 1 0 3 3 | P2 1 2 3 4 | P2 0 2 0 1 |
| P3 1 2 1 0 | P3 1 3 5 0 | P3 0 1 4 0 |

We can show that the state given in the previous example is a safe state by showing that it is possible for each process to acquire its maximum resources and then terminate.

1. P1 acquires 2 A, 1 B and 1 D more resources, achieving its maximum
   - [available resource: <3 1 1 2> - <2 1 0 1> = <1 0 1 1>]
   - The system now still has 1 A, no B, 1 C and 1 D resource available
2. P1 terminates, returning 3 A, 3 B, 2 C and 2 D resources to the system
   - [available resource: <1 0 1 1> + <3 3 2 2> = <4 3 3 3>]
   - The system now has 4 A, 3 B, 3 C and 3 D resources available
3. P2 acquires 2 B and 1 D extra resources, then terminates, returning all its resources
   - [available resource: <4 3 3 3> - <0 2 0 1>+<1 2 3 4> = <5 3 6 6>]
   - The system now has 5 A, 3 B, 6 C and 6 D resources
4. P3 acquires 1 B and 4 C resources and terminates
   - [available resource: <5 3 6 6> - <0 1 4 0> + <1 3 5 0> = <6 5 7 6>
   - The system now has all resources: 6 A, 5 B, 7 C and 6 D
5. Because all processes were able to terminate, this state is safe

**4. Describe necessary conditions for a deadlock situation to arise. Jun 16/Jan 15**

Necessary Conditions:- deadlock situation can occur if the following 4 conditions occur simultaneously in a system:-

1. Mutual Exclusion:- Only one process must hold the resource at a time. If any other process requestsfor the resource, the requesting process must be delayed until the resource has been released.

2. Hold and Wait:- A process must be holding at least one resource and waiting to acquire additionalresources that are currently being held by the other process.

3. No Preemption:- RESOURCES CAN'T BE PREEMPTED I.E., ONLY THE PROCESS HOLDING THE RESOURCES mustrelease it after the process has completed its task.

4. Circular Wait:- A SET {P0,P1… … ..PN} OF WAITING PROCESS MUST EXIST SUCH THAT P0 IS WAITING for aresource i.e., held by P1, P1 is waiting for a resource i.e., held by P2. Pn-1 is waiting for resourceheld by process Pn and Pn is waiting for the resource i.e., held by P1.

**5. Explain different methods to handle deadlocks. Jun 15/Jan16**

Ignoring deadlock

In this approach, it is assumed that a deadlock will never occur. This is used when the time intervals between occurrences of deadlocks are large and the data loss incurred each time is tolerable.

Detection

Under deadlock detection, deadlocks are allowed to occur. Then the state of the system is examined to detect that a deadlock has occurred and subsequently it is corrected. An algorithm is employed that tracks resource allocation and process states, it rolls back and restarts one or more of the processes in order to remove the detected deadlock. Detecting a deadlock that has already occurred is easily possible since the resources that each process has locked and/or currently requested are known to the resource scheduler of the operating system.

After a deadlock is detected, it can be corrected by using one of the following methods:

1. Process Termination: One or more process involved in the deadlock may be aborted. We can choose to abort all processes involved in the deadlock. This ensures that deadlock is resolved with certainty and speed. But the expense is high as partial computations will be

lost. Or, we can choose to abort one process at a time until the deadlock is resolved. This approach has high overheads because after each abort an algorithm must determine whether the system is still in deadlock. Several factors must be considered while choosing a candidate for termination, such as priority and age of the process.

2. Resource Preemption: Resources allocated to various processes may be successively preempted and allocated to other processes until the deadlock is broken.

Prevention

Main article: Deadlock prevention algorithms

Deadlock prevention works by preventing one of the four Coffman conditions from occurring.

- Removing the mutual exclusion condition means that no process will have exclusive access to a resource. This proves impossible for resources that cannot be spooled. But even with spooled resources, deadlock could still occur. Algorithms that avoid mutual exclusion are called non-blocking synchronization algorithms.
- The hold and wait or resource holding conditions may be removed by requiring processes to request all the resources they will need before starting up. This advance knowledge is frequently difficult to satisfy and, in any case, is an inefficient use of resources. Another way is to require processes to request resources only when it has none. Thus, first they must release all their currently held resources before requesting all the resources they will need from scratch. This too is often impractical. It is so because resources may be allocated and remain unused for long periods. Also, a process requiring a popular resource may have to wait indefinitely, as such a resource may always be allocated to some process, resulting in resource starvation.[
- The no preemption condition may also be difficult or impossible to avoid as a process has to be able to have a resource for a certain amount of time, or the processing outcome may be inconsistent or thrashing may occur. However, inability to enforce preemption may interfere with a priority algorithm. Preemption of a "locked out" resource generally implies a rollback, and is to be avoided, since it is very costly in overhead. Algorithms that allow preemption include lock-free and wait-free algorithms and optimistic concurrency control.
- The final condition is the circular wait condition. Approaches that avoid circular waits include disabling interrupts during critical sections and using a hierarchy to determine a partial ordering of resources. If no obvious hierarchy exists, even the memory address of resources has been used to determine ordering and resources are requested in the increasing order of the enumeration. Dijkstra's solution can also be used.

Deadlock can be avoided if certain information about processes are available to the operating system before allocation of resources, such as which resources a process will consume in its lifetime. For every resource request, the system sees whether granting the request will mean that the system will enter an unsafe state, meaning a state that could result in deadlock. The system then only grants requests that will lead to safe states.[1] In order for the system to be able to determine whether the next state will be safe or unsafe, it must know in advance at any time:

- resources currently available
- resources currently allocated to each process
- resources that will be required and released by these processes in the future

**6. Explain different methods to recover deadlocks.**
**Jan16/Jan 15**

**1.** Abort all deadlocked processes – this is the method adopted in most general purpose systems.

**2.** Re-start all deadlocked processes, however, this method may lead straight back to the original Deadlock.

**3.** Successfully (one at a time) abort deadlocked processes until Deadlock no longer exist. The order in which this is done should reduce resources already used.

**4.** Successfully (one at a time) pre-empty resources from deadlocked processes until Deadlock no longer exist. The order of pre-empting should be such that to minimize the cost.

## UNIT 5

**1) What is paging and swapping? Jun 15/Jan 16**

Swapping is a technique of temporarily removing inactive programs from the memory of the system.

A process can be swapped temporarily out of the memory to a backing store and then brought backin to the memory for continuing the execution. This process is called swapping.

*Eg*:- In a multi-programming environment with a round robin CPU scheduling whenever the time quantum expires then the process that has just finished is swapped out and a new process swapsin to the memory for execution.

A variation of swap is priority based scheduling. When a low priority is executing and if a high priority process arrives then a low priority will be swapped out and high priority is allowed for execution. This process is also called as Roll out and Roll in.

Normally the process which is swapped out will be swapped back to the same memory space that is occupied previously. This depends upon address binding.

If the binding is done at load time, then the process is moved to same memory location.

If the binding is done at run time, then the process is moved to different memory location. This is because the physical address is computed during run time.

Swapping requires backing store and it should be large enough to accommodate the copies of all memory images.

The system maintains a ready queue consisting of all the processes whose memory images are on the backing store or in memory that are ready to

run. Swapping is constant by other factors:-

- To swap a process, it should be completely idle.
- A process may be waiting for an i/o operation. If the i/o is asynchronously accessing the user memory for i/o buffers, then the process cannot be swapped.

**Paging**:-

- Paging is a memory management scheme that permits the physical address space of a process to be non-contiguous. Support for paging is handled by hardware.
- It is used to avoid external fragmentation.
- Paging avoids the considerable problem of fitting the varying sized memory chunks on to the backing store.
- When some code or date residing in main memory need to be swapped out, space must be found on backing store.

**2) With a diagram discuss the steps involved in handling a page fault? Jun 14**

The step for handling page fault is straight forward and is given below:-

1.We check the internal table of the process to determine whether the reference made is valid or invalid .

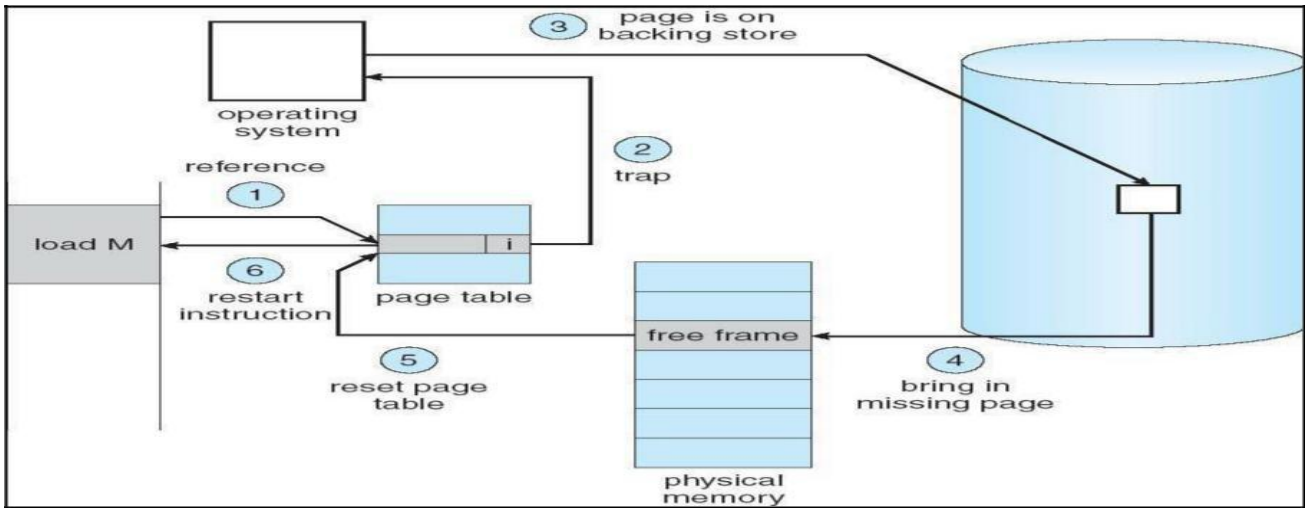
2.If invalid terminate the process,. If valid, then the page is not yet loaded and we now page it in. 3.We find a free frame.

4. We schedule disk operation to read the desired page in to newly allocated frame.

5.When disk reed is complete, we modify the internal table kept with the process to indicate that the page is now in memory.

6.We restart the instruction which was interrupted by illegal address trap. The process can now access the page.

In extreme cases, we start the process without pages in memory. When the OS points to the instruction of process it generates a page fault. After this page is brought in to memory the process continues to execute, faulting as necessary until every demand paging i.e., it never brings the page in to memory until it is required.




1.Save the user registers and process state.

2.Determine that the interrupt was a page fault.

3.Checks the page references were legal and determine the location of page on disk. 4.Issue a read from disk to a free frame.

5.If waiting, allocate the CPU to some other user. 6.Interrupt from the disk.

7.Save the registers and process states of other users.

8.Determine that the interrupt was from the disk.

9.Correct the page table and other table to show that the desired page is now in memory. 10.Wait for the CPU to be allocated to this process again.

11.Restore the user register process state and new page table then resume the interrupted instruction.

**3.What is address binding? Explain the concept of dynamic relocation of addresses? Jun15**

**Address Binding**:-

- Programs are stored on the secondary storage disks as binary executable files.
- When the programs are to be executed they are brought in to the main memory and placed within aprocess.
- The collection of processes on the disk waiting to enter the main memory forms the input queue.
- One of the processes which are to be executed is fetched from the queue and placed in the mainmemory.
- During the execution it fetches instruction and data from main memory. After the process terminatesit returns back the memory space.
- During execution the process will go through different steps and in each step the address isrepresented in different ways.
- In source program the address is symbolic.
- The compiler converts the symbolic address to re-locatable address.
- The loader will convert this re-locatable address to absolute address.

Binding of instructions and data can be done at any step along the way:-

1.Compile time:- If we know whether the process resides in memory then absolute codecan be generated. If the static address changes then it is necessary to re-compile thecode from the beginning.

2.Load time:- IF THE COMPILER DOESN'T KNOW whether the process resides in memory thenit generates the re-locatable code. In this the binding is delayed until the load time. 3.Execution time:- If the process is moved during its execution from one memory segment to another then the binding is delayed until run time.

**Dynamic re-location using a re-location registers**

- The above figure shows that dynamic re-location which implies mapping from virtualaddresses space to physical address space and is performed by the hardware at run time.
- Re-location is performed by the hardware and is invisible to the user dynamic relocationmakes it possible to move a partially executed process from one area of memory to anotherwithout affecting.

**Dynamic Loading**:-

- For a process to be executed it should be loaded in to the physical memory. The size of the process islimited to the size of the physical memory.
- Dynamic loading is used to obtain better memory utilization.
- In dynamic loading the routine or procedure will not be loaded until it is called.
- Whenever a routine is called, the calling routine first checks whether the called routine is already loaded or not. If it is not loaded it cause the loader to load the desired program in to the memory and updates the programs address table to indicate the change and control is passed to newly called routine.
- Advantage:-
- Gives better memory utilization.
- Unused routine is never loaded.
- Do not need special operating system support.
- This method is useful when large amount of codes are needed to handle in frequently occurring cases.

**4. Define external fragmentation? what are the causes? Jun 14/ Jan 15**

- External Fragmentation exists when there is enough memory space exists to satisfy the request, but it not contiguous i.e., storage is fragmented in to large number of small holes.
- External Fragmentation may be either minor or a major problem.
- One solution for over-coming external fragmentation is compaction. The goal is to move all the free memory together to form a large block. Compaction is not possible always. If

the re-location is static and is done at load time then compaction is not possible. Compaction is possible if the re-location is dynamic and done at execution time.

- Another possible solution to the external fragmentation problem is to permit the logical address space of a process to be non-contiguous, thus allowing the process to be allocated physical memory whenever the latter is available.

**5.what is paging ?explain the paging hardware? Jun 15/Jan 16**

**Paging**:-

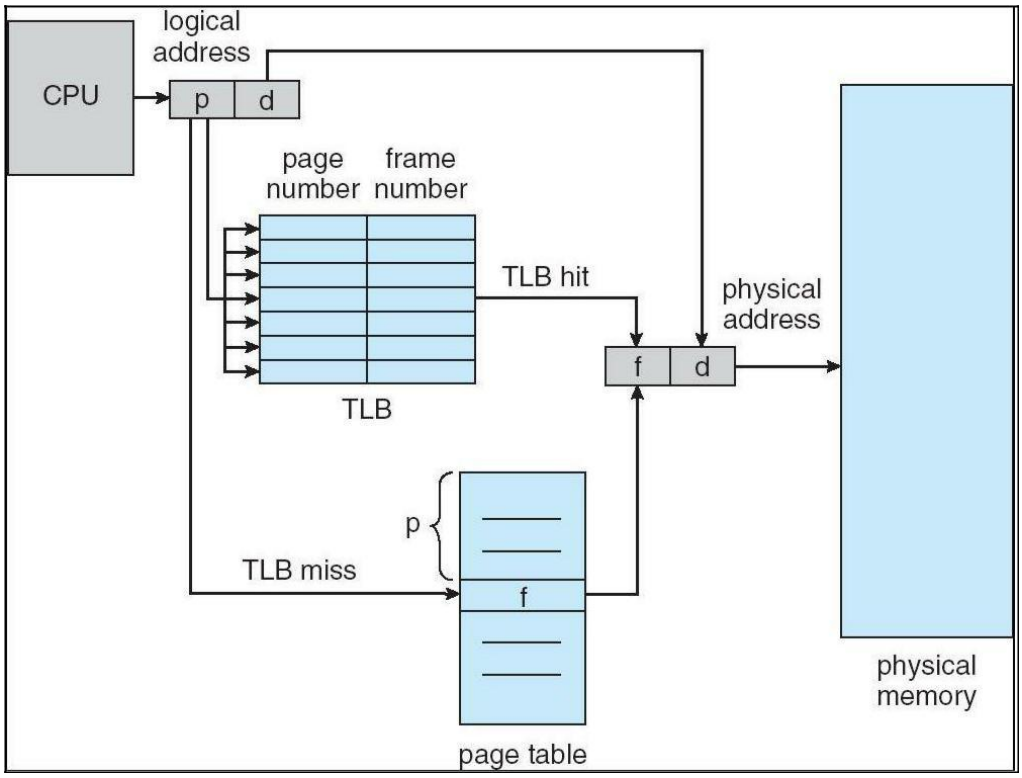
- Paging is a memory management scheme that permits the physical address space of a process to benon-contiguous. Support for paging is handled by hardware.
- It is used to avoid external fragmentation.
- Paging avoids the considerable problem of fitting the varying sized memory chunks on to the backingstore.
- When some code or date residing in main memory need to be swapped out, space must be found onbacking store.

**Hardware Support for Paging**:-

The hardware implementation of the page table can be done in several ways:-

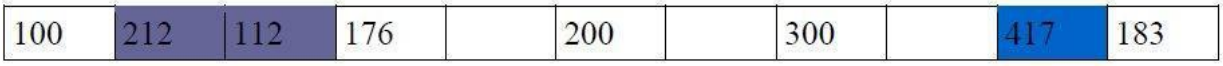
1. The simplest method is that the page table is implemented as a set of dedicated registers. Theseregisters must be built with very high speed logic for making paging address translation . Everyaccessed memory must go through paging map. The use of registers for page table is satisfactory if thepage table is small.
2. If the page table is large then the use of registers is not visible. So the page table is kept in the mainmemory and a page table base register [PTBR] points to the page table. Changing the page tablerequires only one register which reduces the context switching type. The problem with this approach isthe time required to access memory location. To access a location [i] first we have to index the pagetable using PTBR offset. It gives the frame number which is combined with the page offset to produce the actual address. Thus we need two memory accesses for a byte.
3. The only solution is to use special, fast, lookup hardware cache called translation look aside buffer[TLB] or associative register.

4. TLB is built with associative register with high speed memory. Each register contains two paths akey and a value.



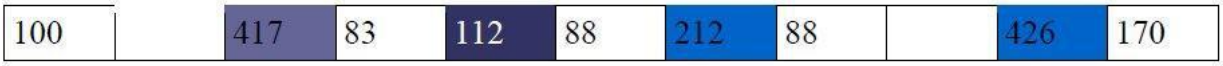

**6.Memory partitions of 100kb,500 kb,200 kb,300kb,600 kb are available how would best worst, first fit algorithm to place processes 212,417,112,426 in order. Which is the best algorithm? Jun16/Jun 14**

First fit

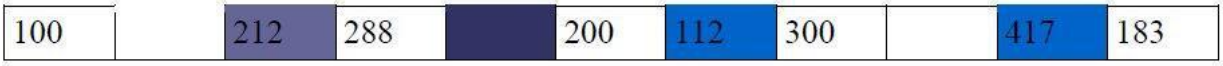



There is no free space to insert 426.

Best fit




Worst fit




There is no free space to insert 426.

**7)Differentiate between internal and external fragmentation? Jun15/Jan 15**

**Fragmentation**:-

- Memory fragmentation can be of two types:-
  - Internal Fragmentation
  - External Fragmentation
- In Internal Fragmentation there is wasted space internal to a portion due to the fact that block ofdata loaded is smaller than the partition.
- *Eg*:- If there is a block of 50kb and if the process requests 40kb and if the block is allocated to theprocess then there will be 10kb of memory left.
- External Fragmentation exists when there is enough memory space exists to satisfy the request, butit not contiguous i.e., storage is fragmented in to large number of small holes.
- External Fragmentation may be either minor or a major problem.
- One solution for over-coming external fragmentation is compaction. The goal is to move all the freememory together to form a large block. Compaction is not possible always. If the re-location is staticand is done at load time then compaction is not possible. Compaction is possible if the re-location isdynamic and done at execution time.

- Another possible solution to the external fragmentation problem is to permit the logical addressspace of a process to be non-contiguous, thus allowing the process to be allocated physical memorywhenever the latter is available.

**8)consider the reference stream 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6.how many page faults while using fcfs and lru? Jun 14** **Jun 14**

FCFS

| frames | 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 2 | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | | | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 4 | | | | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 5 | | | | | | | 5 | 5 | 5 | 5 | 5 | 5 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| pf | * | * | * | * | | | * | * | | * | * | * | * | | | | | | | |

PAGE FAULT=10

LRU

| frames | 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | | | 3 | 3 | 3 | 3 | 3 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 4 | | | | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 5 | | | | | | | 5 | 5 | 5 | 5 | 5 | 5 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| pf | * | * | * | * | | | * | * | | | | * | * | | | | | | | |

PAGE FAULT=8

**9. What are the methods of handling the page faults? Jun 16**

The steps for handling page fault is straight forward and is given below:-
We check the internal table of the process to determine whether the reference made is valid or invalid.If invalid terminate the process,. If valid, then the page is not yet loaded and we now page it in. We find a free frame. We schedule disk operation to read the desired page in to newly allocated frame.

**10.Consider the following reference string?1,2,3,4,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6**

**How many page faults using 2 frames? Jun 16/ Jan 15**

| frames | 1 | 2 | 3 | 4 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 3 | 3 | 1 | 1 | 6 | 6 | 1 | 1 | 1 | 7 | 7 | 3 | 3 | 1 | 1 | 1 | 6 |
| 2 | | 2 | 2 | 4 | 4 | 5 | 5 | 2 | 2 | 2 | 3 | 3 | 6 | 6 | 2 | 2 | 2 | 3 | 3 |
| pf | * | * | * | * | * | * | * | * | * | | * | * | * | * | * | * | | * | * |

Page fault =17

**11.What is thrashing?what are causes? Jun 14**

**Thrashing**

- If the number of frames allocated to a low-priority process falls below the minimum numberrequired by the computer architecture then we suspend the process execution.
- A process is thrashing if it is spending more time in paging than executing.
- If the processes do not have enough number of frames, it will quickly page fault. During this it mustreplace some page that is not currently in use. Consequently it quickly faults again and again. Theprocess continues to fault, replacing pages for which it then faults and brings back. This high pagingactivity is called thrashing. The phenomenon of excessively moving pages back and forth b/w memory and secondary has been called thrashing.

**Cause of Thrashing**

- Thrashing results in severe performance problem.
- The operating system monitors the cpu utilization is low. We increase the degree of multiprogramming by introducing new process to the system.
- A global page replacement algorithm replaces pages with no regards to the process to which theybelong.
- The figure shows the thrashing
- As the degree of multi programming increases, more slowly until a maximum is reached. If thedegree of multi programming is increased further thrashing sets in and the cpu utilization dropssharply.
- At this point, to increases CPU utilization and stop thrashing, we must increase degree of multiprogramming. We can limit the effect of thrashing by using a local replacement algorithm. To prevent thrashing, we must provide a process as many frames as it needs.

## UNIT 6

**1. Explain the following i)file types ii)file operation iii)file attributes? Jun 16/Jan 16/ Jan 15**

A file has a certain defined structures according to its type:-

1. Text file:- Text file is a sequence of characters organized in to lines.

2. Object file:- Object file is a sequence of bytes organized in to blocksunderstandable by the systems linker.

3. Executable file:- Executable file is a series of code section that the loader can bring in to memory and execute.

4. Source File:- Source file is a sequence of subroutine and function, each of which are further organized as declaration followed by executable statements.

**File Attributes**:- File attributes varies from one OS to other. The common file attributes are:

1. Name:- The symbolic file name is the only information kept in human readable form.

2. Identifier:- The unique tag, usually a number, identifies the file within the file system. It is the non-readable name for a file.

3. Type:- This information is needed for those systems that supports different types.

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

5. Size:- The current size of the file and possibly the maximum allowed size are included in this attribute.

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

7. Time, data and User Identification:- This information must be kept for creation, last modification and last use. These data are useful for protection, security and usage monitoring.

**File Operation**:-

File is an abstract data type. To define a file we need to consider the operation that can be performed on the file.

Basic operations of files are:-

1. Creating a file:- Two steps are necessary to create a file. First space in the file system for file is found . Second an entry for the new file must be made in the directory. The directory entry records the name of the file and the location in the file system.

2. Writing a file:- System call is mainly used for writing in to the file. System call specify the name of the file and the information i.e., to be written on to the file. Given the name the system search the entire directory for the file. The system must keep a write pointer to the location in the file where the next write to be taken place.

3. Reading a file:- To read a file system call is used. It requires the name of the file and the memory address. Again the directory is searched for the associated directory and system must maintain a read pointer to the location in the file where next read is to take place.

4. Delete a file:- System will search for the directory for which file to be deleted. If entry is found it releases all free space. That free space can be reused by another file.

5. Truncating the file:- User may want to erase the contents of the file but keep its attributes. Rather than forcing the user to delete a file and then recreate it, truncation allows all attributes to remain unchanged except for file length.

6. Repositioning within a file:- The directory is searched for appropriate entry and the current file position is set to a given value. Repositioning within a file does not need to involve actual i/o. The file operation is also known as file seeks.

In addition to this basis 6 operations the other two operations include appending new information to the end of the file and renaming the existing file. These primitives can be combined to perform other two operations.

**2.Explain the method used for implementing directories? Jun 16/Jan 16**

A file has a certain defined structures according to its type:-

- Text file:- Text file is a sequence of characters organized in to lines.
- Object file:- Object file is a sequence of bytes organized in to blocks understandable by the systems linker.
- Executable file:- Executable file is a series of code section that the loader can bring in to memory andexecute.
- Source File:- Source file is a sequence of subroutine and function, each of which are further organizedas declaration followed by executable statements.
- Size:- The current size of the file and possibly the maximum allowed size are included in this attribute.
- Protection:-Access control information determines who can do reading, writing, execute and so on.
- Time, data and User Identification:- This information must be kept for creation, last modification and last use. These data are useful for protection, security and usage monitoring.

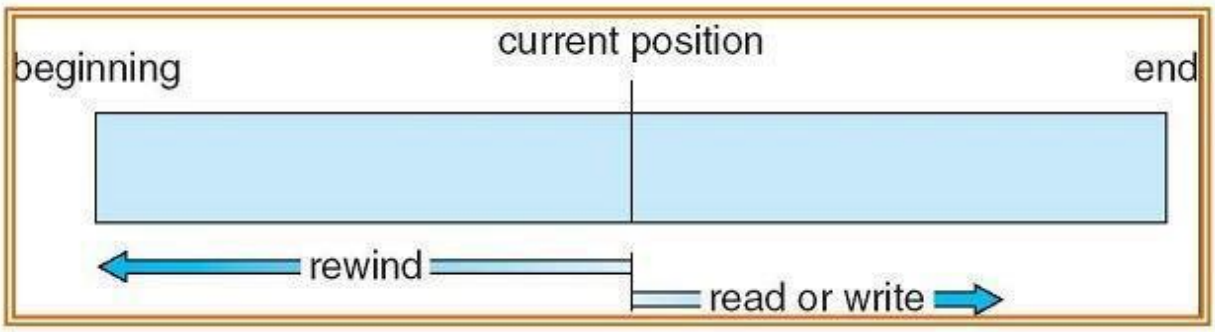
**3 Describe various file access methods? Jun 15/ Jan 15**

**Access Methods**:

1. **Sequential Access**:-

Sequential access is the simplest access method. Information in the file is processed in order, onerecord after another. Editors and compilers access the files in this fashion.

Normally read and write operations are done on the files. A read operation reads the next portion ofthe file and automatically advances a file pointer, which track next track.Write operation appends to the end of the file and such a file can be next to the beginning. Sequential access depends on a tape model of a file.




2. **Direct Access**:-

Direct access allows random access to any file block. This method is based on disk model of a file.Afile is made up of fixed length logical records. It allows the program to read and write records rapidlyin any order.A direct access file allows arbitrary blocks to be read or written.

*Eg*:-User may need block 13, then read block 99 then write block 12.

For searching the records in large amount of information with immediate result, the direct accessmethod is suitable. Not all OS support sequential and direct access. Few OS use sequential access andsome OS uses direct access. It is easy to simulate sequential access on a direct access but the reverse isextremely inefficient.

**Indexing Method**:-

- The index is like an index at the end of a book which contains pointers tovarious blocks.
- To find a record in a file, we search the index and then use the pointer to accessthe file directly and to find the desired record.
- With large files index file itself can be very large to be kept in memory. One solution to create anindex to the index files itself. The primary index file would contain pointer to secondary index fileswhich would point to the actual data items.

Two types of indexes can be used:-

a. Exhaustive index:- Contain one entry for each of record in the main file. An indexitself is organized as a sequential file.

b. Partial index:- Contains entries to records where the field of interest exists withrecords of variable length, soma record will not contain an fields. When a newrecord is added to the main file, all index files must be updated.

**4. Explain the file system mounting operation. Jan 15**

**File System Mounting**:-

The file system must be mounted before it can be available to processes on the system

The procedure for mounting the file is:

a. The OS is given the name of the device and the location within the file structure atwhich to attach the file system (mount point).A mount point will be an emptydirectory at which the mounted file system will be attached.

*Eg*:- On UNIX a file system containing users home directory might be mounted as /home then toaccess the directory structure within that file system. We must precede the directory names as/home/jane.

b. Then OS verifies that the device contains this valid file system. OS uses devicedrivers for this verification.

c. Finally the OS mounts the file system at the specified mount point.

**File System Structure**:-

Disks provide bulk of secondary storage on which the file system is maintained. Disks have twocharacteristics:-

a. They can be rewritten in place i.e., it is possible to read a block from the disk to modify the block and to write back in to same place.

b. They can access any given block of information on the disk. Thus it is simple to access any file either sequentially or randomly and switching from one file to another.

The lowest level is the i/o control consisting of device drivers and interrupt handless to transfer theinformation between memory and the disk system. The device driver is like a translator. Its input is ahigh level command and the o/p consists of low level hardware specific instructions, which are usedby the hardware controllers which interface I/O device to the rest of the system.

The basic file system needs only to issue generic commands to the appropriate device drivers to readand write physical blocks on the disk.

The file organization module knows about files and their logical blocks as well as physical blocks . Byknowing the type of file allocation used and the location of the file, the file organization module cantranslate logical block address to the physical block address. Each logical block is numbered 0 to N.

Since the physical blocks containing the data usually do not match the logical numbers, So atranslation is needed to locate each block. The file allocation modules also include free space managerwhich tracks the unallocated blocks and provides these blocks when requested.

**File System Implementation**:-

- File system is implemented on the disk and the memory.
- The implementation of the file system varies according to the OS and the file system, but thereare some general principles.

If the file system is implemented on the disk it contains the following information:-

1. **Boot Control Block**:- can contain information needed by the system to boot an OS from thatpartition. If the disk has no OS, this block is empty. It is the first block of the partition. InUFS boot block, In NTFS partition boot sector.
2. **Partition control Block**:- contains partition details such as the number of blocks in partition,size of the blocks, number of free blocks, free block pointer, free FCB count and FCBpointers. In NTFS master file tables, In UFS super block.
3. Directory structure is used to organize the files.
4. An FCB contains many of the files details, including file permissions, ownership, size,location of the data blocks. In UFS inode, In NTFS this information is actually stored withinmaster file table.

Structure of the file system management in memory is as follows:-

a. An in-memory partition table containing information about each mounted information.

b. An in-memory directory structure that holds the directory information of recently accessed directories

c. The system wide open file table contains a copy of the FCB of each open file as well asother information.

d. The per-process open file table contains a pointer to the appropriate entry in the system wideopen file table as well as other information.

**5. Mention different file attributes and file types? Jun 15**

A file has a certain defined structures according to its type:-

1. Text file:- Text file is a sequence of characters organized in to lines.
2. Object file:- Object file is a sequence of bytes organized in to blocksunderstandable by the systems linker.
3. Executable file:- Executable file is a series of code section that the loader canbring in to memory and execute.
4. Source File:- Source file is a sequence of subroutine and function, each ofwhich are further organized as declaration followed by executable statements.

**File Attributes**:-

File attributes varies from one OS to other. The common file attributes are:

1. Name:- The symbolic file name is the only information kept in human readableform.
2. Identifier:- The unique tag, usually a number, identifies the file within the filesystem. It is the non-readable name for a file.
3. Type:- This information is needed for those systems that supports differenttypes.
4. Location:- This information is a pointer to a device and to the location of the file on that device.
5. Size:- The current size of the file and possibly the maximum allowed size areincluded in this attribute.
6. Protection:-Access control information determines who can do reading, writing,execute and so on.
7. Time, data and User Identification:- This information must be kept for creation,last modification and last use. These data are useful for protection, security andusage monitoring.

**6. How free space is managed? Explain.**
**Jun 14/ Jun 15**

Another important aspect of disk management is keeping track of and allocating free space.

- Bit Vector

  One simple approach is to use a bit vector, in which each bit represents a disk block, set to 1 if free or 0 if allocated.Fast algorithms exist for quickly finding contiguous blocks of a given size

  The down side is that a 40GB disk requires over 5MB just to store the bitmap.

- Linked List

  A linked list can also be used to keep track of all free blocks.

  Traversing the list and/or finding a contiguous block of a given size are not easy, but fortunately are not frequently needed operations. Generally the system just adds and removes single blocks from the beginning of the list.The FAT table keeps track of the free list as just one more linked list on the table.

- Grouping

  A variation on linked list free lists is to use links of blocks of indices of free blocks. If a block holds up to N addresses, then the first block in the linked-list contains up to N-1 addresses of free blocks and a pointer to the next block of free addresses.

- Counting

  When there are multiple contiguous blocks of free space then the system can keep track of the starting address of the group and the number of contiguous free blocks. As long as the average length of a contiguous group of free blocks is greater than two this offers a savings in space needed for the free list.

- Space Maps

  Sun's ZFS file system was designed for huge numbers and sizes of files, directories, and even file systems.

  The resulting data structures could be very inefficient if not implemented carefully. For example, freeing up a 1 GB file on a 1 TB file system could involve updating thousands of blocks of free list bit maps if the file was spread across the disk.

  ZFS uses a combination of techniques, starting with dividing the disk up into of meta slabs of a manageable size, each having their own space map.

Free blocks are managed using the counting technique, but rather than write the information to a table, it is recorded in a log-structured transaction record. Adjacent free blocks are also coalesced into a larger single free block.

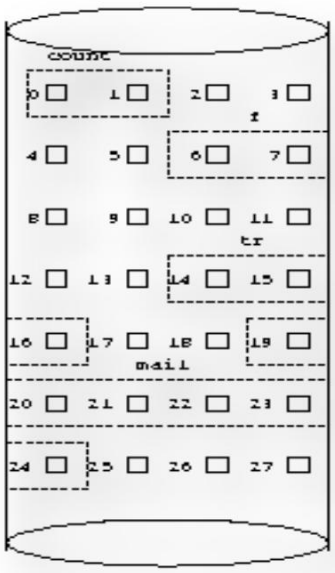
An in-memory space map is constructed using a balanced tree data structure, constructed from the log data.

**7. What are the three methods for allocating disk space? Explain. Jun 15**

**Disk Allocation Methods**

a)contiguous allocation

- each file occupies a set of consecutive addresses on disk
- each directory entry contains:
- file name
- starting address of the first block
- block address = sector id (e.g., block = 4K)
- length in blocks
- usual dynamic storage allocation problem
- use first fit, best fit, or worst fit algorithms to manage storage
- if the file can increase in size, either
- leave no extra space, and copy the file elsewhere if it expands
- leave extra space




DIRECTORY

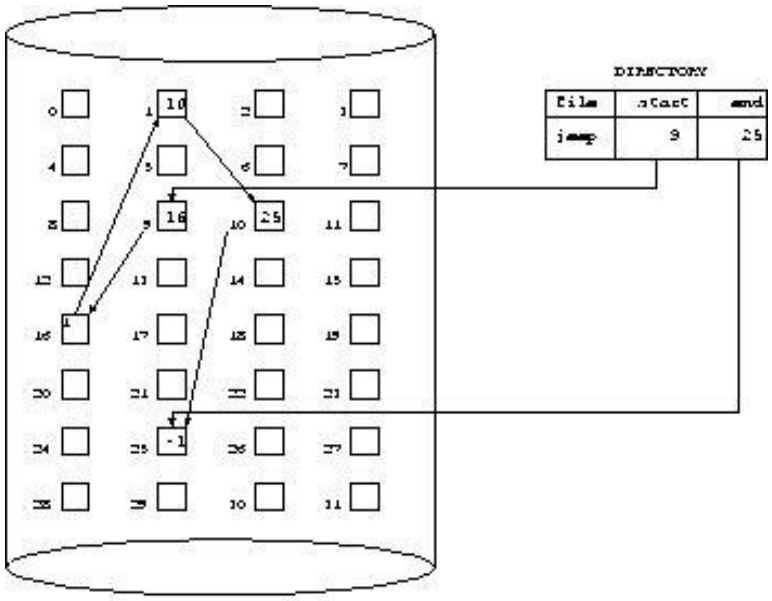
| file | start | length |
|---|---|---|
| count | 0 | 2 |
| tr | 14 | 3 |
| mail | 19 | 6 |
| f | 6 | 2 |

Contiguous allocation of disk space
Silberschatz, p. 388

b)linked allocation

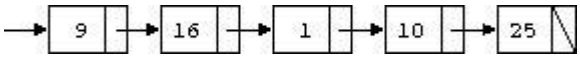
- each data block contains the block address of the next block in the file
- each directory entry contains:
- file name
- **block address**: pointer to the first block

- sometimes, also have a pointer to the last block



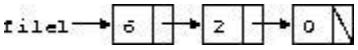

Linked allocation of disk space (Silberschatz, p. 390)

- a view of the linked list




c)indexed allocation

- store all pointers together in an index table
- the index table is stored in several **index blocks**
- assume index table has been loaded into main memory The index has one entry for each block on disk.




- better than linked allocation if we want to seek a particular offset of a file because many links are stored together instead of each one in a separate block
- SGG call this organization a ``linked'' scheme, but I call it an ``indexed'' scheme because an index is kept in main memory.
- problem: index is too large to fit in main memory for large disks
- FAT may get really large and we may need to store FAT on disk, which will increase access time
- e.g., 500 Mb disk with 1 Kb blocks = 4 bytes * 500 K = 2Mb entries

**UNIT 7**

**1. Describe the access matrix model used for protection purpose? Jan 16/Jun 14**

Access Matrix

_ View protection as a matrix (access matrix)

_ Rows represent domains

_ Columns represent objects

_ Access (i, j) is the set of operations that a processexecuting in Domain i can invokeon Object j

Access Matrix

| domain \ object | $F_1$ | $F_2$ | $F_3$ | printer |
|---|---|---|---|---|
| $D_1$ | read | | read | |
| $D_2$ | | | | print |
| $D_3$ | | read | execute | |
| $D_4$ | read write | | read write | |

Use of Access Matrix

_ IF A PROCESS IN DOMAIN DI TRIES TO DO "OP" ON OBJECT OJ, THEN "OP" MUST BE IN THE ACCESS matrix.Canbe expanded to dynamic protection.

Operations to add, delete access rights. Special access rights:

owner of Oi

copyo p from Oi to Oj

control – Di can modify Dj access rights

transfer – switch from domain Di to Dj

Access matrix design separates mechanism from policy.

Mechanism

Operating system provides access-matrix + rules.

If ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced.

Policy

User dictates policy.

Who can access what object and in what mode.

**2. Explain various disk scheduling algorithms? Jun 15/Jun 16**

Disk Scheduling

The amount of head movement needed to satisfy a series of i/o request can affectthe performance. If the desired drive and the controller are available the request can be servicedimmediately. If the device or controller is busy any new requests for service will be placed on thequeue of pending requests for that drive when one request is complete the OS chooses which pendingrequest to service next.

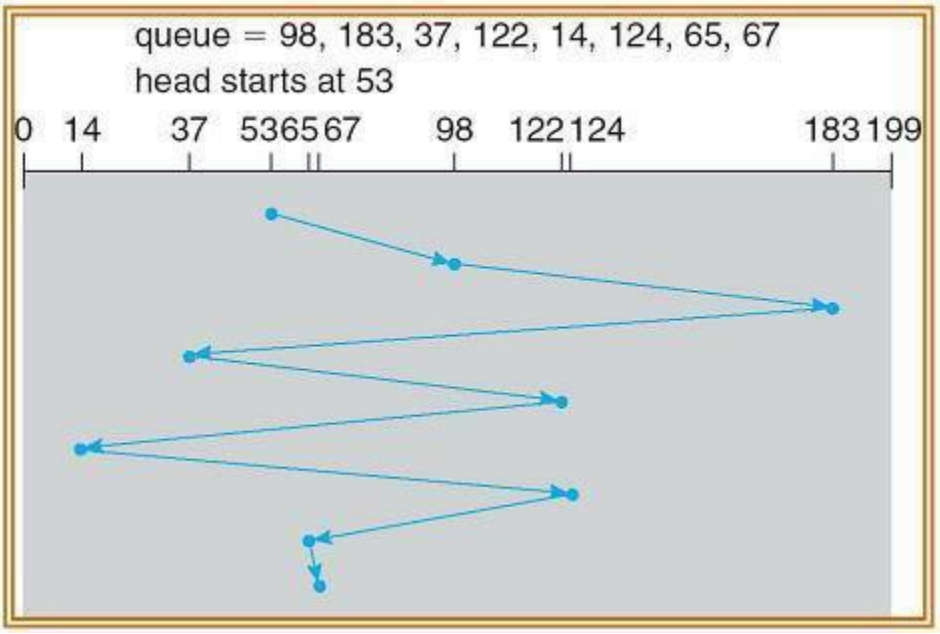
Different types of scheduling algorithms are as follows:

FCFS scheduling algorithm:

This is the simplest form of disk scheduling algorithm. This services the request in the order they arereceived. This algorithm is fair but do not provide fastest service. It takes no special time to minimizethe overall seek time.

Eg:- consider a disk queue with request for i/o to blocks on cylinders.

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




If the disk head is initially at 53, it will first move from 53 to 98 then to 183 and then to 37, 122, 14,124, 65, 67 for a total head movement of 640 cylinders.

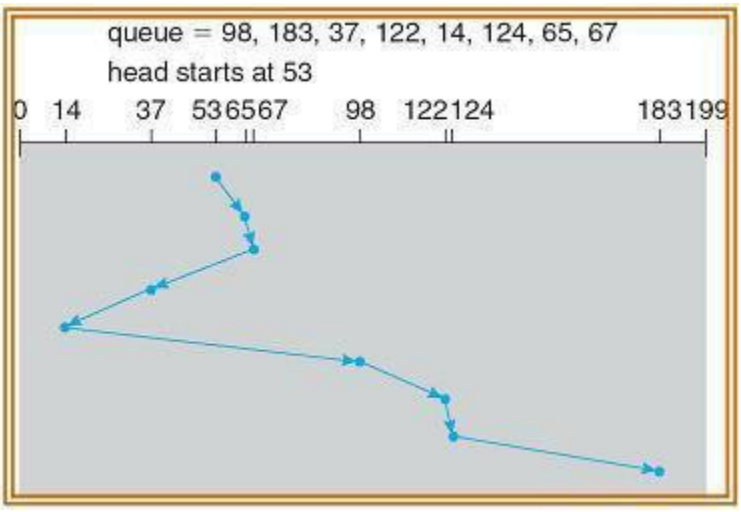
The wild swing from 122 to 14 and then back to 124 illustrates the problem with this schedule. If the requests for cylinders 37 and 14 could be serviced together before or after 122 and 124 the total head movement could be decreased substantially and performance could be improved.

SSTF (Shortest seek time first) algorithm:

This selects the request with minimum seek time from the current head position.

Since seek time increases with the number of cylinders traversed by head, SSTF chooses the pendingrequest closest to the current head position.

Eg:- :- consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14, 124, 65,67



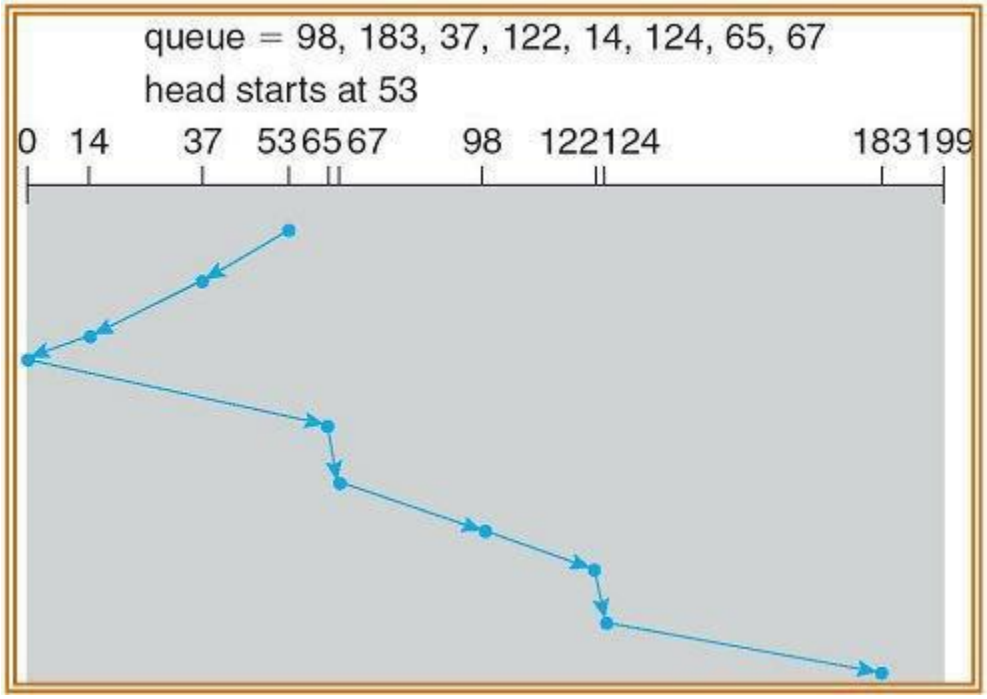

If the disk head is initially at 53, the closest is at cylinder 65, then 67, then 37 is closer then 98 to 67.So it services 37, continuing we service 14, 98, 122, 124 and finally 183.

The total head movement is only 236 cylinders. SSTF is essentially a form of SJF and it may causestarvation of some requests. SSTF is a substantial improvement over FCFS, it is not optim al .

SCAN algorithm:

In this the disk arm starts at one end of the disk and moves towards the other end,servicing the request as it reaches each cylinder until it gets to the other end of the disk. At the otherend, the direction of the head movement is reversed and servicing continues.

Eg:- :- consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14, 124, 65,67

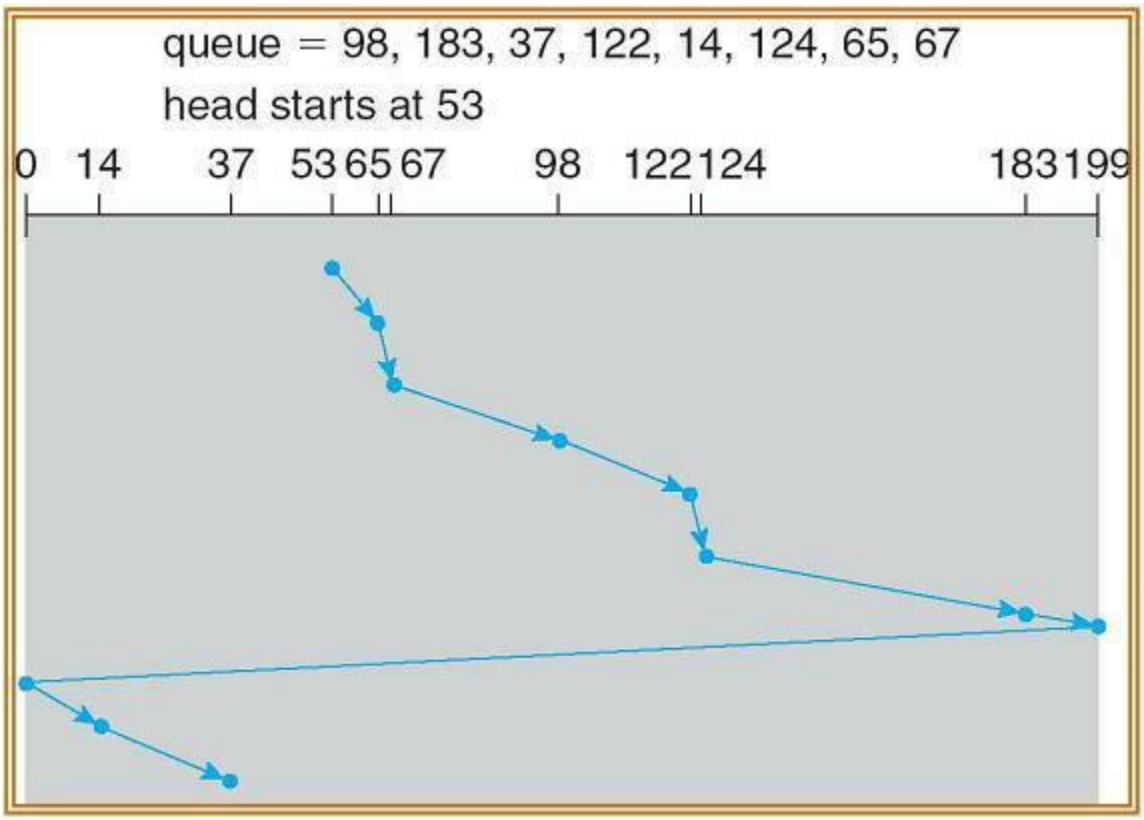



If the disk head is initially at 53 and if the head is moving towards 0, it services 37 and then 14. Atcylinder 0 the arm will reverse and will move towards the other end of the disk servicing 65, 67, 98,122, 124 and 183. If a request arrives just in from of head, it will be serviced immediately and therequest just behind the head will have to wait until the arms reach other end and reverses direction.The SCAN is also called as elevator algorithm.

C-SCAN (Circular scan) algorithm:

C-SCAN is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from end of the disk to the other servicing the request along the way. When the headreaches the other end, it immediately returns to the beginning of the disk, without servicing anyrequest on the return. The C-SCAN treats the cylinders as circular list that wraps around from the finalcylinder to the first one.

Eg:- :- consider a disk queue with request for i/o to blocks on cylinders. 98, 183, 37, 122, 14, 124, 65,67



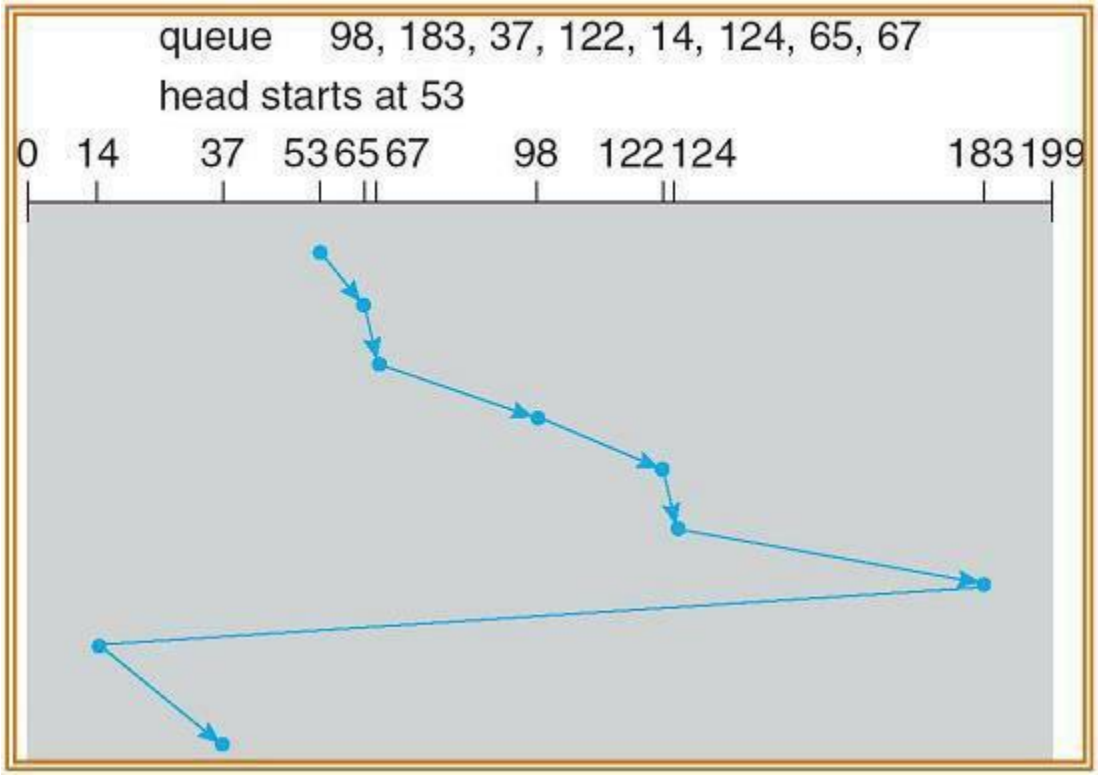

Look Scheduling algorithm:-

Both SCAN and C-SCAN move the disk arm across the full width of the disk. In practice neither ofthe algorithms is implemented in this way.

The arm goes only as far as the final request in each direction. Then it reverses, without going all theway to the end of the disk. These versions of SCAN and C-SCAN are called Look and C-Lookscheduling because they look for a request before continuing to move in a given direction.

Eg:- :- consider a disk queue with request for i/o to blocks on cylinders.

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




**3. Explain the access matrix structure employed in protection domain? Jun 16/ Jan 15**

Access Matrix

_ View protection as a matrix (access matrix)

_ Rows represent domains

_ Columns represent objects

_ Access (i, j) is the set of operations that a processexecuting in Domain i can invoke on Object j

Use of Access Matrix

_ If a process in Domain Di tries to DO "OP" ON OBJECT OJ, THEN "OP" MUST BE IN THE ACCESS matrix.Canbe expanded to dynamic protection.

Operations to add, delete access rights. Special access rights:

owner of Oi

copyo p from Oi to Oj

control – Di can modify Dj access rights

transfer – switch from domain Di to Dj

Access matrix design separates mechanism from policy.

Mechanism

Operating system provides access-matrix + rules.

If ensures that the matrix is only manipulated byauthorized agents and that rules are strictly enforced.

Policy

User dictates policy.

Who can access what object and in what mode

**4.What is protection goals and principles? Jun 15/ Jan 15**

**Goals of Protection**

- Obviously to prevent malicious misuse of the system by users or programs. See chapter 15 for a more thorough coverage of this goal.
- To ensure that each shared resource is used only in accordance with system *policies*, which may be set either by system designers or by system administrators.
- To ensure that errant programs cause the minimal amount of damage possible.
- Note that protection systems only provide the *mechanisms* for enforcing policies and ensuring reliable systems. It is up to administrators and users to implement those mechanisms effectively.

**Principles of Protection**

- The ***principle of least privilege*** dictates that programs, users, and systems be given just enough privileges to perform their tasks.
- This ensures that failures do the least amount of harm and allow the least of harm to be done.
- For example, if a program needs special privileges to perform a task, it is better to make it a SGID program with group ownership of "network" or "backup" or some other pseudo group, rather than SUID with root ownership. This limits the amount of damage that can occur if something goes wrong.

- Typically each user is given their own account, and has only enough privilege to modify their own files.
- The root account should not be used for normal day to day activities - The System Administrator should also have an ordinary account, and reserve use of the root account for only those tasks which need the root privileges.

**5. Differentiate between mechanism and policy. Jun 15**

Policy: a set of ideas or a plan of what to do.

Mechanism: a process, technique, or system for achieving a result.

A mechanism is more about 'how' a particular task

where as a policy is more about 'what' needs to be done.

For example, consider an office with several employees. The office may have a policy like "All the employees need to be authenticated before they enter the office. "

As you can see, the policy just describes what needs to be done without delving much on how it can be achieved. This policy can be enforced by one or more of the below given mechanisms:

Each employee needs to swipe his/her identity card on an RFID reader. The door will be unlocked only when a valid identity card is swiped.

Using a retina/fingerprint scanner, a biosensor will authenticate every employee entering the office.

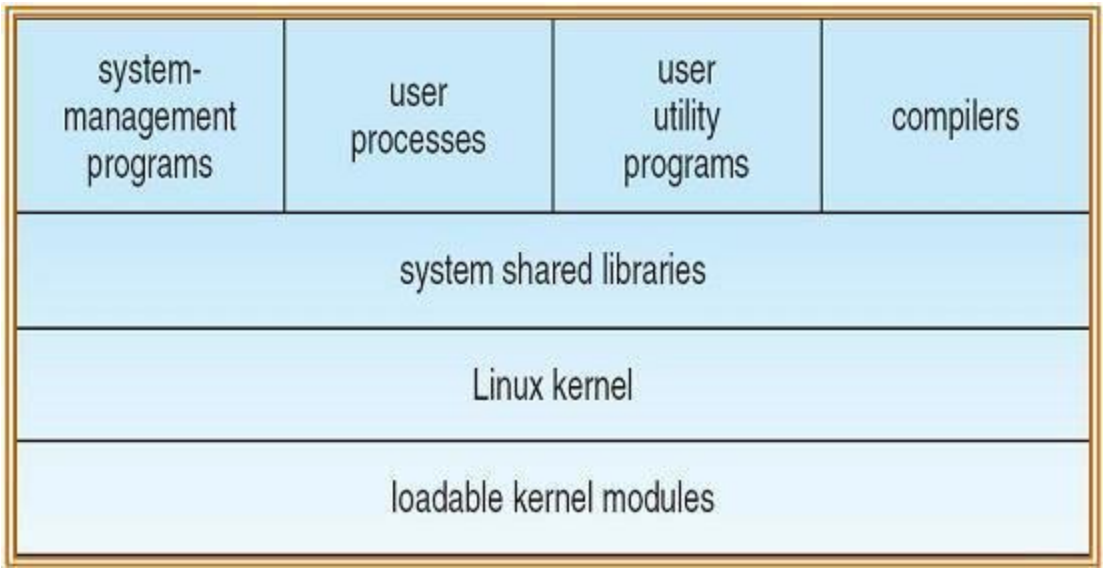
**6. Write short notes on Revocation of access rights. Jan 16/ Jan 15**

In a dynamic protection system, we may sometimes need to revoke access rights to objects shared by different users. Various questions about revocation may arise:

- **Immediate versus delayed**. Does revocation occur immediately, or is it delayed? If revocation is delayed, can we find out when it will take place?
- **Selective versus general**. When an access right to an object is revoked, does it affect *all* the users who have an access right to that object, or can we specify a select group of users whose access rights should be revoked?
- **Partial versus total**. Can a subset of the rights associated with an object be revoked, or must we revoke all access rights for this object?

**UNIT 8**

**1. Write short note on components of Linux system? Jun 15/Jun 14**




Like most UNIX implementations, Linux is composed of three main bodies of code;
the mostimportant distinction between the kernel and all other components
The kernel is responsible for maintaining the important abstractions of the operating system

1. Kernel code executes in *kernel mode* with full access to all the physical resources of the computer
2. All kernel code and data structures are kept in the same single address space

The system libraries define a standard set of functions through which applications interact with thekernel, and which implement much of the operating-system functionality that does not need the full privileges of kernel code.

**2. Explain the process management model of linux operating system? Jun 15/Jan 16/ Jan 15**

UNIX process management separates the creation of processes and the running of a new program into two distinct operations.

i. The fork system call creates a new process
ii. A new program is run after a call to execve

Under UNIX, a process encompasses all the information that the operating system must maintain track the context of a single execution of a single program

UNDER LINUX, PROCESS PROPERTIES FALL INTO THREE GROUPS: THE PROCESS'S IDENTITY, ENVIRONMENT, AND context.

Process ID (PID). The unique identifier for the process; used to specify processes to the operating system when an application makes a system call to signal, modify, or wait for another process Credentials. Each process must have an associated user ID and one or more group IDs that determine THE PROCESS'S RIGHTS TO ACCESS SYSTEM RESOURCES AND FILES Personality. Not traditionally found on UNIX systems, but under Linux each process has anassociated personality identifier that can slightly modify the semantics of certain system calls

1. Used primarily by emulation libraries to request that system calls be compatible with certain specific flavors of UNIX
2. THE PROCESS'S ENVIRONMENT IS INHERITED FROM ITS PARENT, AND IS COMPOSED OF TWO NULL-terminated vectors
3. The argument vector lists the command-line arguments used to invoke the running program conventionally starts with the name of the program itself
4. THE ENVIRONMENT VECTOR IS A LIST OF "NAME=VALUE" PAIRS THAT ASSOCIATES NAMED environment variables with arbitrary textual values

- PASSING ENVIRONMENT VARIABLES AMONG PROCESSES AND INHERITING VARIABLES BY A PROCESS'S children areflexible means of passing information to components of the user-mode system software
- The environment- variable mechanism provides a customization of the operating system that can be seton a per-process basis, rather than being configured for the system as a whole
- The (constantly changing) state of a running program at any point in time
- The scheduling context is the most important part of the process context; it is the information that thescheduler needs to suspend and restart the process

The kernel maintains accounting information about the resources currently being consumed by eachprocess, and the total resources consumed by the process in its lifetime .

The file table is an array of pointers to kernel file structures

- When making file I/O system calls, processes refer to files by their index into this table
- Whereas the file table lists the existing open files, thefile-system context applies to requests to open new files
- The current root and default directories to be used for new file searches are stored here
- The signal-HANDLER TABLE DEFINES THE ROUTINE IN THE PROCESS'S ADDRESS SPACE TO BE CALLED when specific signals arrive.The virtual- memory context of a process describes the full contents of the its private address space

**3. What are the two file system models adopted in linux operating system? Jun 16**

To the user, Linux 'S FILE SYSTEM APPEARS AS A HIERARCHICAL DIRECTORY TREE OBEYING UNIX semantics Internally, the kernel hides implementation details and manages the multiple different file systems via an abstraction layer, that is, the virtual file system.

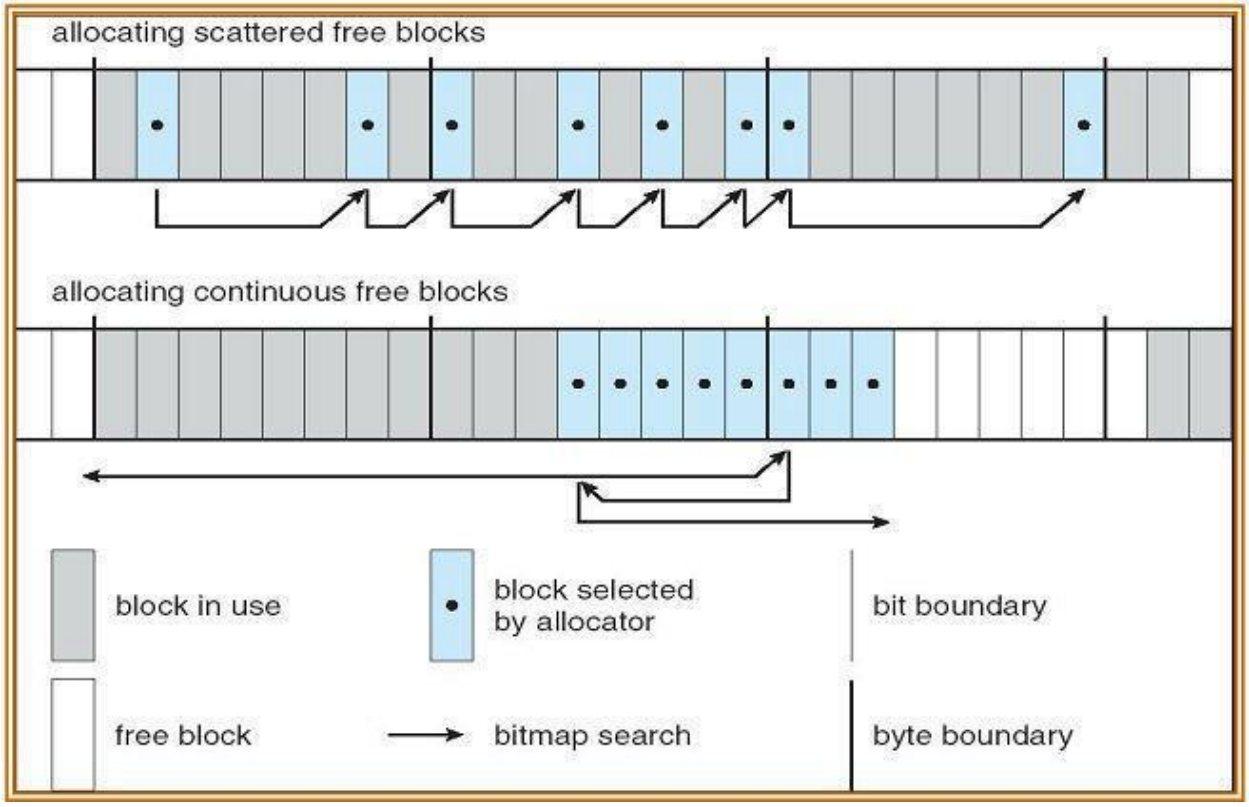
- The Linux VFS is designed around object-oriented principles and composed of two components:

i. A set of definitions that define what a file object is allowed to look like the inode-object and the file-object structures represent individual files □the *file system object* represents an entire file system.

ii. A layer of software to manipulate those objects uses a mechanism similar to that of BSD Fast File System (ffs) for locating data blocks belonging to a specific file the main differences between ext2fs and ffs concern their disk allocation policies

In the disk is allocated to files in blocks of 8Kb, with blocks being subdivided into fragments of 1Kb to store small files or partially filled blocks at the end of a file does not use fragments; it performs its allocations in smaller units default block size on ext2fs is 1Kb, although 2Kb and 4Kb blocks are also supported.

Ext2fs uses allocation policies designed to place logically adjacent blocks of a file into physically adjacent blocks on disk, so that it can submit an I/O request for several disk blocks as a single peration




The proc file system

The proc file system does not store data, rather, its contents are computed on demand according to user file I/O requests proc must implement a directory structure, and the file contents within; it must then define a unique and persistent inode number for each directory and files it contains It uses this inode number to identify just what operation is required when a user tries to read from a particular file inode or perform a lookup in a particular directory inode .When data is read from one of these files, proc collects the appropriate information, formats it into text form and places it into the requesting PROCESS'S READ BUFFER.

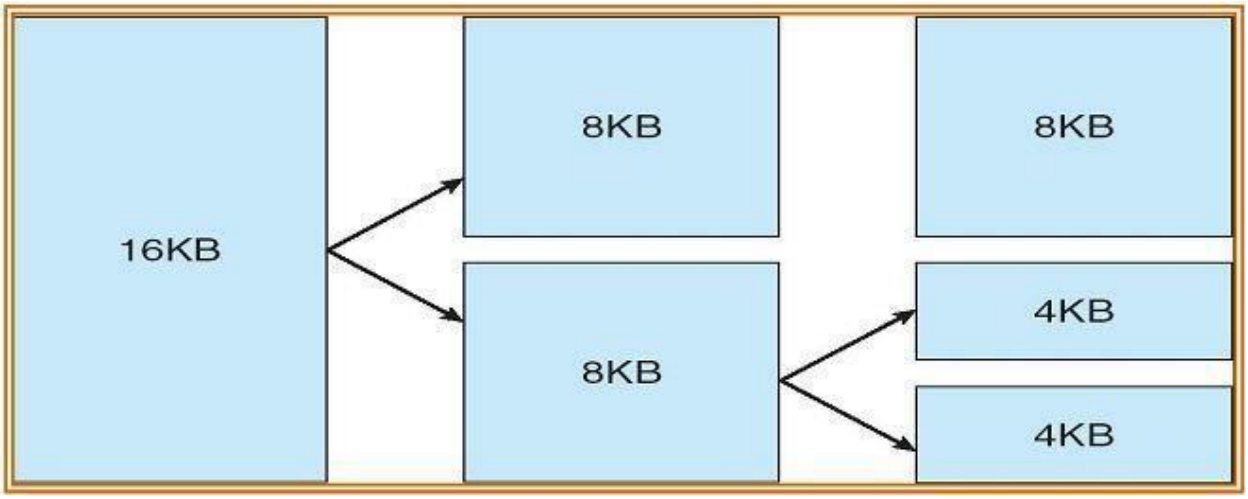
**4.Write notes on buddy system of memory management in unix? Jun 15**

LINUX'S PHYSICAL MEMORY- management system deals with allocating and freeing pages, groups ofpages, and small blocks of memory It has additional mechanisms for handling virtual memory,memory mapped into the address space of running processes.

Splits memory into 3 different zones due to hardware characteristics

| zone | physical memory |
|---|---|
| ZONE_DMA | < 16 MB |
| ZONE_NORMAL | 16 .. 896 MB |
| ZONE_HIGHMEM | > 896 MB |

**Splitting of Memory in a Buddy Heap**

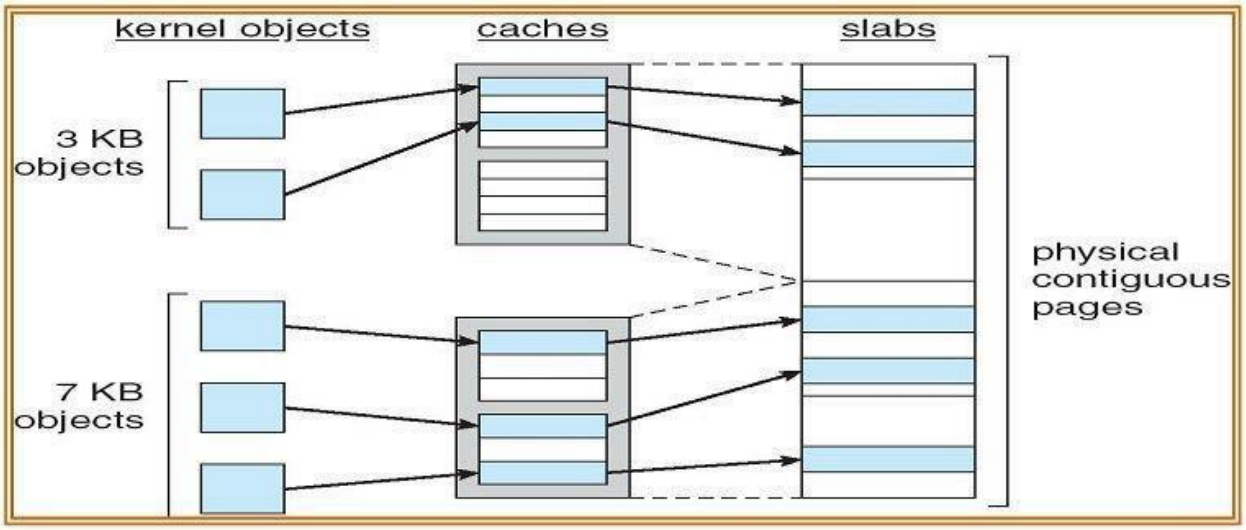



The page allocator allocates and frees all physical pages; it can allocate ranges of physically contiguous pages on request.

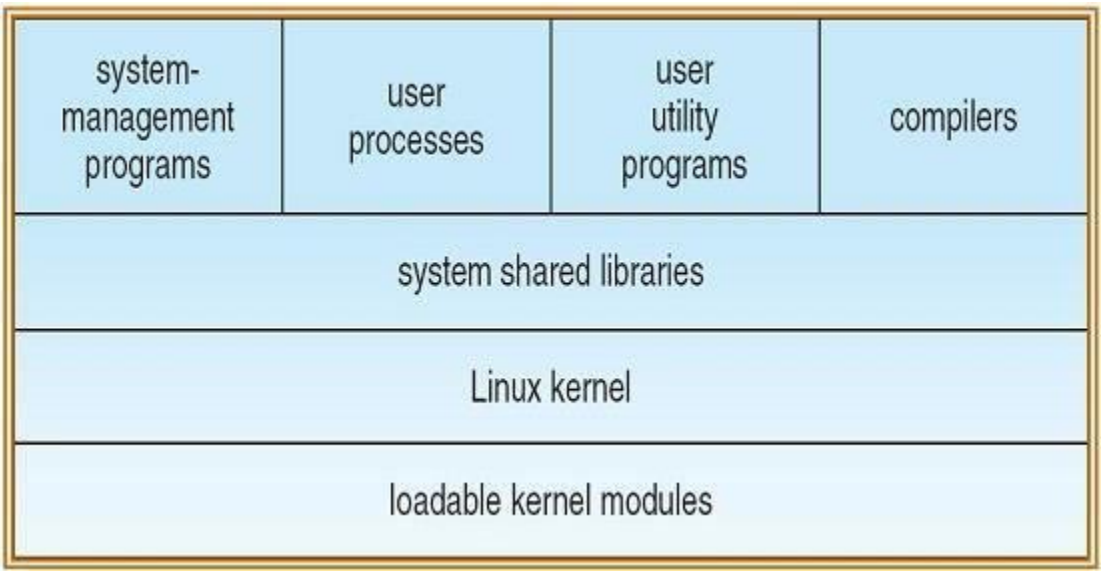
The allocator uses a buddy-heap algorithm to keep track of available physical pages

Each allocatable memory region is paired with an adjacent partner

Whenever two allocated partner regions are both freed up they are combined to form a larger region If a small memory request cannot be satisfied by allocating an existing small free region, then a larger free region will be subdivided into two partners to satisfy the request Memory allocations in the Linux kernel occur either statically (drivers reserve a contiguous area of memory during system boot time) or dynamically (via the page allocator)




**5.Discuss the various components of linux system? Jun 15/Jun 14 Jul 15/Jun 14**




Like most UNIX implementations, Linux is composed of three main bodies of code; the most important distinction between the kernel and all other components

The kernel is responsible for maintaining the important abstractions of the operating system Kernel code executes in *kernel mode* with full access to all the physical resources of the computer All kernel code and data structures are kept in the same single address space The

system libraries define a standard set of functions through which applications interact with the kernel, and which implement much of the operating-system functionality that does not need the full privileges of kernel code The system utilities perform individual specialized management tasks

**6. Interprocess communication in linux system? Jan 16**

Like UNIX, Linux informs processes that an event has occurred via signals There is a limited number of signals, and they cannot carry information: Only the fact that a signal occurred is available to a process The Linux kernel does not use signals to communicate with processes with are running in kernel mode, rather, communication within the kernel is accomplished via scheduling states and wait, queue structures.

Signals

Signals are a way of sending simple messages to processes. Most of these messages are already defined and can be found in <linux/signal.h>. However, signals can only be processed when the process is in user mode. If a signal has been sent to a process that is in kernel mode, it is dealt with immediately on returning to user mode.

Pipes

The common Linux shells all allow redirection. For example

$ ls | pr | lpr

pipes the output from the ls command listing the directory's files into the standard input of the prcommand which paginates them. Finally the standard output from the pr command is piped into the standard input of the lpr command which prints the results on the default printer. Pipes then are unidirectional byte streams which connect the standard output from one process into the standard input of another process. Neither process is aware of this redirection and behaves just as it would normally. It is the shell that sets up these temporary pipes between the processes.

Semaphores

In its simplest form a semaphore is a location in memory whose value can be tested and set by more than one process. The test and set operation is, so far as each process is concerned, uninterruptible or atomic; once started nothing can stop it. The result of the test and set operation is the addition of the current value of the semaphore and the set value, which can be positive or

negative. Depending on the result of the test and set operation one process may have to sleep until the semphore's value is changed by another process. Semaphores can be used to implement *critical regions*, areas of critical code that only one process at a time should be executing.

Message Queues

Message queues allow one or more processes to write messages that will be read by one or more reading processes. Linux maintains a list of message queues, the msgque vector: each element of which points to a msqid_ds data structure that fully describes the message queue. When message queues are created, a new msqid_ds data structure is allocated from system memory and inserted into the vector.

**7. What do you mean by cloning? How is it achieved in Linux systems? Jan 16/ Jan 15**

The main use of **clone**() is to implement threads: multiple threads of control in a program that run concurrently in a shared memory space. When the child process is created with **clone**(), it executes the function fn(arg). (This differs from fork (2), where execution continues in the child from the point of the fork(2) call.) The fn argument is a pointer to a function that is called by the child process at the beginning of its execution. The *arg* argument is passed to the *fn* function.

When the fn (arg) function application returns, the child process terminates. The integer returned by fn is the exit code for the child process. The child process may also terminate explicitly by calling exit (2) or after receiving a fatal signal .

**8. What are the design principles of Linux OS? Jun15/Jun16/ Jan 15**

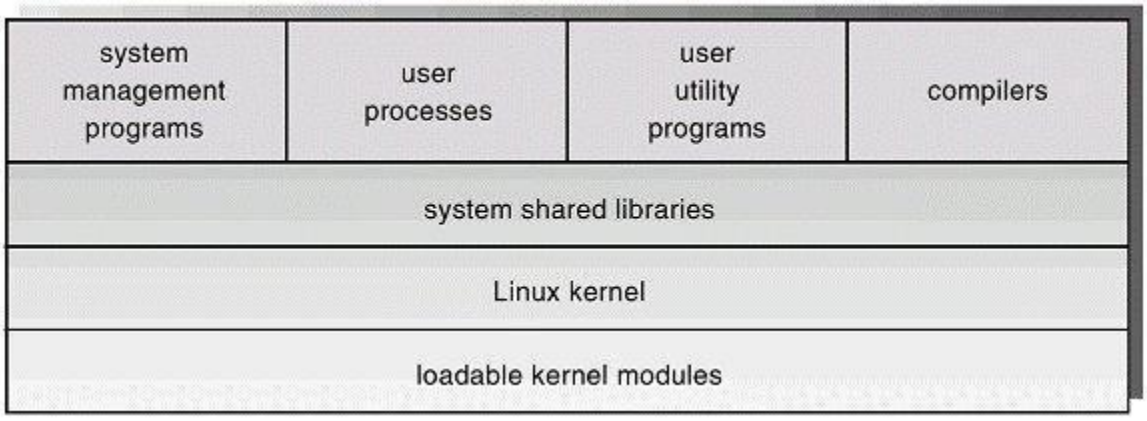
Design Principles

Linux is a multi-user, multitasking system with a full set of UNIX-compatible tools..

Its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model.

The Linux programming interface adheres to the SVR4 UNIX semantics, rather than to BSD behavior.

Components of a Linux System




Like most UNIX implementations, Linux is composed of three main bodies of code; the most important distinction between the kernel and all other components.

The kernel is responsible for maintaining the important abstractions of the operating system.

1. Kernel code executes in kernel mode with full access to all the physical resources of the computer.
2. All kernel code and data structures are kept in the same single address space.

The system libraries define a standard set of functions through which applications interact with the kernel, and which implement much of the operating-system functionality that does not need the full privileges of kernel code.