

1. What is an operating system?

- a) collection of programs that manages hardware resources
- b) system service provider to the application programs
- c) interface between the hardware and application programs
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: An Operating System acts as an intermediary between user/user applications/application programs and hardware. It is a program that manages hardware resources. It provides services to application programs.

2. To access the services of operating system, the interface is provided by the

-
- a) System calls
 - b) API
 - c) Library
 - d) Assembly instructions

[View Answer](#)

Answer: a

Explanation: To access services of the Operating System an interface is provided by the System Calls. Generally, these are functions written in C and C++. Open, Close, Read, Write are some of most prominently used system calls.

3. Which one of the following is not true?

- a) kernel is the program that constitutes the central core of the operating system
- b) kernel is the first part of operating system to load into memory during booting
- c) kernel is made of various modules which can not be loaded in running operating system
- d) kernel remains in the memory during the entire computer session

[View Answer](#)

Answer: c

Explanation: Kernel is the first program which is loaded in memory when OS is loading as well as it remains in memory till OS is running. Kernel is the core part of the OS which is responsible for managing resources, allowing multiple processes to use the resources and provide services to various processes. Kernel modules can be loaded and unloaded in run-time i.e. in running OS.

4. Which one of the following error will be handle by the operating system?

- a) power failure
- b) lack of paper in printer
- c) connection failure in the network
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: All the mentioned errors are handled by OS. The OS is continuously monitoring all of its resources. Also, the OS is constantly detecting and correcting errors.

5. What is the main function of the command interpreter?

- a) to get and execute the next user-specified command
- b) to provide the interface between the API and application program
- c) to handle the files in operating system
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: The main function of command interpreter is to get and execute the next user-specified command. Command Interpreter checks for valid command and then runs that command else it will throw an error.

6. In Operating Systems, which of the following is/are CPU scheduling algorithms?

- a) Round Robin
- b) Shortest Job First
- c) Priority
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: In Operating Systems, CPU scheduling algorithms are:

- i) First Come First Served scheduling
- ii) Shortest Job First scheduling
- iii) Priority scheduling
- iv) Round Robin scheduling
- v) Multilevel Queue scheduling
- vi) Multilevel Feedback Queue scheduling

All of these scheduling algorithms have their own advantages and disadvantages.

7. If a process fails, most operating system write the error information to a

- a) log file
- b) another running process
- c) new file
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: If a process fails, most operating systems write the error information to a log file. Log file is examined by the debugger, to find out what is the actual cause of that particular problem. Log file is useful for system programmers for correcting errors.

8. Which facility dynamically adds probes to a running system, both in user processes and in the kernel?

- a) DTrace
- b) DLocate
- c) DMap
- d) DAdd

[View Answer](#)

Answer: a

Explanation: A facility that dynamically adds probes to a running system, both in user process and in the kernel is called DTrace. This is very much useful in troubleshooting kernels in real-time.

9. Which one of the following is not a real time operating system?

- a) VxWorks
- b) QNX
- c) RTLinux
- d) Palm OS

[View Answer](#)

Answer: d

Explanation: VxWorks, QNX & RTLinux are real-time operating systems. Palm OS is a mobile operating system. Palm OS is developed for Personal Digital Assistants (PDAs).

10. The OS X has _____

- a) monolithic kernel
- b) hybrid kernel
- c) microkernel

d) monolithic kernel with modules

[View Answer](#)

Answer: b

Explanation: OS X has a hybrid kernel. Hybrid kernel is a combination of two different kernels. OS X is developed by Apple and originally it is known as Mac OS X.

1. The systems which allow only one process execution at a time, are called _____

- a) uniprogramming systems
- b) uniprocessing systems
- c) unitasking systems
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: Those systems which allows more than one process execution at a time, are called multiprogramming systems. Uniprocessing means only one processor.

2. In operating system, each process has its own _____

- a) address space and global variables
- b) open files
- c) pending alarms, signals and signal handlers
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: In Operating Systems, each process has its own address space which contains code, data, stack and heap segments or sections. Each process also has a list of files which is opened by the process as well as all pending alarms, signals and various signal handlers.

3. In Unix, Which system call creates the new process?

- a) fork
- b) create
- c) new
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: In UNIX, a new process is created by fork() system call. fork()

system call returns a process ID which is generally the process id of the child process created.

4. A process can be terminated due to _____

- a) normal exit
- b) fatal error
- c) killed by another process
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: A process can be terminated normally by completing its task or because of fatal error or killed by another process or forcefully killed by a user. When the process completes its task without any error then it exits normally. The process may exit abnormally because of the occurrence of fatal error while it is running. The process can be killed or terminated forcefully by another process.

5. What is the ready state of a process?

- a) when process is scheduled to run after some execution
- b) when process is unable to run until some task has been completed
- c) when process is using the CPU
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: Ready state of the process means process has all necessary resources which are required for execution of that process when CPU is allocated. Process is ready for execution but waiting for the CPU to be allocated.

6. What is interprocess communication?

- a) communication within the process
- b) communication between two process
- c) communication between two threads of same process
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: Interprocess Communication (IPC) is a communication mechanism that allows processes to communicate with each other and synchronise their actions without using the same address space. IPC can be achieved using shared memory and message passing.

7. A set of processes is deadlock if _____
- a) each process is blocked and will remain so forever
 - b) each process is terminated
 - c) all processes are trying to kill each other
 - d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: Deadlock is a situation which occurs because process A is waiting for one resource and holds another resource (blocking resource). At the same time another process B demands blocking a resource as it is already held by a process A, process B is waiting state unless and until process A releases occupied resource.

8. A process stack does not contain _____
- a) Function parameters
 - b) Local variables
 - c) Return addresses
 - d) PID of child process

[View Answer](#)

Answer: d

Explanation: Process stack contains Function parameters, Local variables and Return address. It does not contain the PID of child process.

9. Which system call can be used by a parent process to determine the termination of child process?
- a) wait
 - b) exit
 - c) fork
 - d) get

[View Answer](#)

Answer: a

Explanation: wait() system call is used by the parent process to determine termination of child process. The parent process uses wait() system call and gets the exit status of the child process as well as the pid of the child process which is terminated.

10. The address of the next instruction to be executed by the current process is provided by the _____
- a) CPU registers
 - b) Program counter

c) Process stack

d) Pipe

[View Answer](#)

Answer: b

Explanation: The address of the next instruction to be executed by the current process is provided by the Program Counter. After every instruction is executed, the Program Counter is incremented by 1 i.e. address of the next instruction to be executed. CPU fetches instruction from the address denoted by Program Counter and execute it.

1. A Process Control Block(PCB) does not contain which of the following?

a) Code

b) Stack

c) Bootstrap program

d) Data

[View Answer](#)

Answer: c

Explanation: Process Control Block (PCB) contains information related to a process such as Process State, Program Counter, CPU Register, etc. Process Control Block is also known as Task Control Block. Bootstrap program is a program which runs initially when the system or computer is booted or rebooted.

2. The number of processes completed per unit time is known as _____

a) Output

b) Throughput

c) Efficiency

d) Capacity

[View Answer](#)

Answer: b

Explanation: The number of processes completed per unit time is known as Throughput. Suppose there are 4 processes A, B, C & D they are taking 1, 3, 4 & 7 units of time respectively for their executions. For 10 units of time, throughput is high if process A, B & C are running first as 3 processes can execute. If process C runs first then throughput is low as maximum only 2 processes can execute. Throughput is low for processes which take a long time for execution. Throughput is high for processes which take a short time for execution.

3. The state of a process is defined by _____

- a) the final activity of the process
- b) the activity just executed by the process
- c) the activity to next be executed by the process
- d) the current activity of the process

[View Answer](#)

Answer: d

Explanation: The state of a process is defined by the current activity of the process. A process state changes when the process executes. The process states are as New, Ready, Running, Wait, Terminated.

4. Which of the following is not the state of a process?

- a) New
- b) Old
- c) Waiting
- d) Running

[View Answer](#)

Answer: b

Explanation: There is no process state such as old. When a process is created then the process is in New state. When the process gets the CPU for its execution then the process is in Running state. When the process is waiting for an external event then the process is in a Waiting state.

5. What is a Process Control Block?

- a) Process type variable
- b) Data Structure
- c) A secondary storage section
- d) A Block in memory

[View Answer](#)

Answer: b

Explanation: A Process Control Block (PCB) is a data structure. It contains information related to a process such as Process State, Program Counter, CPU Register, etc. Process Control Block is also known as Task Control Block.

6. The entry of all the PCBs of the current processes is in _____

- a) Process Register
- b) Program Counter
- c) Process Table
- d) Process Unit

[View Answer](#)

Answer: c

Explanation: The entry of all the PCBs of the current processes is in Process Table. The Process Table has the status of each and every process that is created in OS along with their PIDs.

7. What is the degree of multiprogramming?

- a) the number of processes executed per unit time
- b) the number of processes in the ready queue
- c) the number of processes in the I/O queue
- d) the number of processes in memory

[View Answer](#)

Answer: d

Explanation: Multiprogramming means the number of processes are in the ready states. To increase utilization of CPU, Multiprogramming is one of the most important abilities of OS. Generally, a single process cannot use CPU or I/O at all time, whenever CPU or I/O is available another process can use it. By doing this CPU utilization is increased.

8. A single thread of control allows the process to perform _____

- a) only one task at a time
- b) multiple tasks at a time
- c) only two tasks at a time
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: A single thread of control allows the process to perform only one task at a time. In the case of multi-core, multiple threads can be run simultaneously and can perform multiple tasks at a time.

9. What is the objective of multiprogramming?

- a) Have a process running at all time
- b) Have multiple programs waiting in a queue ready to run
- c) To increase CPU utilization
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: The objective of multiprogramming is to increase CPU utilization. Generally, a single process cannot use CPU or I/O at all time, whenever CPU or I/O is available another process can use it. Multiprogramming offers this ability to OS by keeping multiple programs in a ready queue.

1. Which of the following do not belong to queues for processes?

- a) Job Queue
- b) PCB queue
- c) Device Queue
- d) Ready Queue

[View Answer](#)

Answer: b

Explanation: PCB queue does not belong to queues for processes. PCB is a process control block which contains information related to process. Each process is represented by PCB.

2. When the process issues an I/O request _____

- a) It is placed in an I/O queue
- b) It is placed in a waiting queue
- c) It is placed in the ready queue
- d) It is placed in the Job queue

[View Answer](#)

Answer: a

Explanation: When the process issues an I/O request it is placed in an I/O queue. I/O is a resource and it should be used effectively and every process should get access to it. There might be multiple processes which requested for I/O. Depending on scheduling algorithm I/O is allocated to any particular process and after completing I/O operation, I/O access is returned to the OS.

3. What will happen when a process terminates?

- a) It is removed from all queues
- b) It is removed from all, but the job queue
- c) Its process control block is de-allocated
- d) Its process control block is never de-allocated

[View Answer](#)

Answer: a

Explanation: When a process terminates, it removes from all queues. All allocated resources to that particular process are deallocated and all those resources are returned back to OS.

4. What is a long-term scheduler?

- a) It selects processes which have to be brought into the ready queue
- b) It selects processes which have to be executed next and allocates CPU
- c) It selects processes which have to remove from memory by swapping

d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: A long-term scheduler selects processes which have to be brought into the ready queue. When processes enter the system, they are put in the job queue. Long-term scheduler selects processes from the job queue and puts them in the ready queue. It is also known as Job Scheduler.

5. If all processes I/O bound, the ready queue will almost always be _____ and the Short term Scheduler will have a _____ to do.

- a) full, little
- b) full, lot
- c) empty, little
- d) empty, lot

[View Answer](#)

Answer: c

Explanation: If all processes are I/O bound, the ready queue will almost empty and the short-term scheduler will have a little to do. I/O bound processes spend more time doing I/O than computation.

6. What is a medium-term scheduler?

- a) It selects which process has to be brought into the ready queue
- b) It selects which process has to be executed next and allocates CPU
- c) It selects which process to remove from memory by swapping
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: A medium-term scheduler selects which process to remove from memory by swapping. The medium-term scheduler swapped out the process and later swapped in. Swapping helps to free up memory.

7. What is a short-term scheduler?

- a) It selects which process has to be brought into the ready queue
- b) It selects which process has to be executed next and allocates CPU
- c) It selects which process to remove from memory by swapping
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: A short-term scheduler selects a process which has to be executed next and allocates CPU. Short-term scheduler selects a process from the ready queue. It selects processes frequently.

8. The primary distinction between the short term scheduler and the long term scheduler is _____

- a) The length of their queues
- b) The type of processes they schedule
- c) The frequency of their execution
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: The primary distinction between the short-term scheduler and the long-term scheduler is the frequency of their execution. The short-term scheduler executes frequently while the long-term scheduler executes much less frequently.

9. The only state transition that is initiated by the user process itself is _____

- a) block
- b) wakeup
- c) dispatch
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: The only state transition that is initiated by the user process itself is block. Whenever a user process initiates an I/O request it goes into block state unless and until the I/O request is not completed.

10. In a time-sharing operating system, when the time slot given to a process is completed, the process goes from the running state to the _____

- a) Blocked state
- b) Ready state
- c) Suspended state
- d) Terminated state

[View Answer](#)

Answer: b

Explanation: In a time-sharing operating system, when the time slot given to a process is completed, the process goes from the running state to the Ready

State. In a time-sharing operating system unit time is defined for sharing CPU, it is called a time quantum or time slice. If a process takes less than 1 time quantum, then the process itself releases the CPU.

11. In a multiprogramming environment _____
- a) the processor executes more than one process at a time
 - b) the programs are developed by more than one person
 - c) more than one process resides in the memory
 - d) a single user can execute many programs at the same time

[View Answer](#)

Answer: c

Explanation: In a multiprogramming environment more than one process resides in the memory. Whenever a CPU is available, one process amongst all present in memory gets the CPU for execution. Multiprogramming increases CPU utilization.

12. Suppose that a process is in "Blocked" state waiting for some I/O service. When the service is completed, it goes to the _____
- a) Running state
 - b) Ready state
 - c) Suspended state
 - d) Terminated state

[View Answer](#)

Answer: b

Explanation: Suppose that a process is in "Blocked" state waiting for some I/O service. When the service is completed, it goes to the ready state. Process never goes directly to the running state from the waiting state. Only processes which are in ready state go to the running state whenever CPU allocated by operating system.

13. The context of a process in the PCB of a process does not contain _____
- a) the value of the CPU registers
 - b) the process state
 - c) memory-management information
 - d) context switch time

[View Answer](#)

Answer: d

Explanation: The context of a process in the PCB of a process does not contain context switch time. When switching CPU from one process to another,

the current context of the process needs to be saved. It includes values of the CPU registers, process states, memory-management information.

14. Which of the following need not necessarily be saved on a context switch between processes?

- a) General purpose registers
- b) Translation lookaside buffer
- c) Program counter
- d) All of the mentioned

[View Answer](#)

Answer: b

Explanation: Translation Look-aside Buffer (TLB) need not necessarily be saved on a context switch between processes. A special, small, fast-lookup hardware cache is called Translation Look-aside Buffer. TLB used to reduce memory access time.

15. Which of the following does not interrupt a running process?

- a) A device
- b) Timer
- c) Scheduler process
- d) Power failure

[View Answer](#)

Answer: c

Explanation: Scheduler process does not interrupt a running process. Scheduler process selects an available process from a pool of available processes and allocates CPU to it.

1. Which process can be affected by other processes executing in the system?

- a) cooperating process
- b) child process
- c) parent process
- d) init process

[View Answer](#)

Answer: a

Explanation: A cooperating process can be affected by other processes executing in the system. Also it can affect other processes executing in the system. A process shares data with other processes, such a process is known as a cooperating process.

2. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place is called _____

- a) dynamic condition
- b) race condition
- c) essential condition
- d) critical condition

[View Answer](#)

Answer: b

Explanation: When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which access takes place is called race condition.

3. If a process is executing in its critical section, then no other processes can be executing in their critical section. What is this condition called?

- a) mutual exclusion
- b) critical exclusion
- c) synchronous exclusion
- d) asynchronous exclusion

[View Answer](#)

Answer: a

Explanation: If a process is executing in its critical section, then no other processes can be executed in their critical section. This condition is called Mutual Exclusion. Critical section of the process is shared between multiple processes. If this section is executed by more than one or all of them concurrently then the outcome of this is not as per desired outcome. For this reason the critical section of the process should not be executed concurrently.

4. Which one of the following is a synchronization tool?

- a) thread
- b) pipe
- c) semaphore
- d) socket

[View Answer](#)

Answer: c

Explanation: Semaphore is a synchronization tool. Semaphore is a mechanism which synchronizes or controls access of threads on critical resources. There are two types of semaphores i) Binary Semaphore ii) Counting Semaphore.

5. A semaphore is a shared integer variable _____

- a) that can not drop below zero
- b) that can not be more than zero
- c) that can not drop below one
- d) that can not be more than one

[View Answer](#)

Answer: a

Explanation: A semaphore is a shared integer variable that can not drop below zero. In binary semaphore, if the value of the semaphore variable is zero that means there is a process that uses a critical resource and no other process can access the same critical resource until it is released. In Counting semaphore, if the value of the semaphore variable is zero that means there is no resource available.

6. Mutual exclusion can be provided by the _____

- a) mutex locks
- b) binary semaphores
- c) both mutex locks and binary semaphores
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: Mutual exclusion can be provided by both mutex locks and binary semaphore. Mutex is a short form of **Mutual Exclusion**. Binary semaphore also provides a mechanism for mutual exclusion. Binary semaphore behaves similar to mutex locks.

7. When high priority task is indirectly preempted by medium priority task effectively inverting the relative priority of the two tasks, the scenario is called

- a) priority inversion
- b) priority removal
- c) priority exchange
- d) priority modification

[View Answer](#)

Answer: a

Explanation: When a high priority task is indirectly preempted by a medium priority task effectively inverting the relative priority of the two tasks, the scenario is called priority inversion.

8. Process synchronization can be done on _____

- a) hardware level
- b) software level
- c) both hardware and software level
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: Process synchronization can be done on both hardware and software level. Critical section problems can be resolved using hardware synchronisation. But this method is not simple for implementation so software synchronization is mostly used.

9. A monitor is a module that encapsulates _____

- a) shared data structures
- b) procedures that operate on shared data structure
- c) synchronization between concurrent procedure invocation
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: A monitor is a module that encapsulates shared data structures, procedures that operate on shared data structure, synchronization between concurrent procedure invocation.

10. To enable a process to wait within the monitor _____

- a) a condition variable must be declared as condition
- b) condition variables must be used as boolean objects
- c) semaphore must be used
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: To enable a process to wait within the monitor a condition variable must be declared as condition.

1. Restricting the child process to a subset of the parent's resources prevents any process from _____

- a) overloading the system by using a lot of secondary storage
- b) under-loading the system by very less CPU utilization
- c) overloading the system by creating a lot of sub-processes
- d) crashing the system by utilizing multiple resources

[View Answer](#)

Answer: c

Explanation: Restricting the child process to a subset of the parent's resources prevents any process from overloading the system by creating a lot of sub-processes. A process creates a child process, child process requires certain resources to complete its task. A child process can demand required resources directly from the system, but by doing this system will be overloaded. So to avoid overloading of the system, the parent process shares its resources among children.

2. A parent process calling _____ system call will be suspended until children processes terminate.

- a) wait
- b) fork
- c) exit
- d) exec

[View Answer](#)

Answer: a

Explanation: A parent process calling wait system call will be suspended until children processes terminate. A parameter is passed to wait system call which will obtain exit status of child as well as wait system call returns PID of terminated process.

3. Cascading termination refers to termination of all child processes if the parent process terminates _____

- a) Normally
- b) Abnormally
- c) Normally or abnormally
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: Cascading termination refers to termination of all child processes if the parent process terminates Normally or Abnormally. Some systems don't allow child processes to exist if the parent process has terminated. Cascading termination is normally initiated by the operating system.

4. With _____ only one process can execute at a time; meanwhile all other process are waiting for the processor. With _____ more than one process can be running simultaneously each on a different processor.

- a) Multiprocessing, Multiprogramming
- b) Multiprogramming, Uniprocessing

- c) Multiprogramming, Multiprocessing
- d) Uniprogramming, Multiprocessing

[View Answer](#)

Answer: d

Explanation: With Uniprogramming only one process can execute at a time; meanwhile all other processes are waiting for the processor. With Multiprocessing more than one process can run simultaneously each on different processors. The Uniprogramming system has only one program inside the core while the Multiprocessing system has multiple processes inside multiple cores. The core is one which executes instructions and stores data locally into registers.

5. In UNIX, each process is identified by its _____

- a) Process Control Block
- b) Device Queue
- c) Process Identifier
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: In Unix, each process is identified by its Process Identifier or PID. The PID provides unique value to each process in the system so that each process can be identified uniquely.

6. In UNIX, the return value for the fork system call is _____ for the child process and _____ for the parent process.

- a) A Negative integer, Zero
- b) Zero, A Negative integer
- c) Zero, A nonzero integer
- d) A nonzero integer, Zero

[View Answer](#)

Answer: c

Explanation: In Unix, the return value of the fork system call is Zero for the child process and Non-zero value for parent process. A fork system call returns the PID of a newly created (child) process to the parent and returns Zero to that newly created (child) process.

7. The child process can _____

- a) be a duplicate of the parent process
- b) never be a duplicate of the parent process
- c) cannot have another program loaded into it

d) never have another program loaded into it

[View Answer](#)

Answer: a

Explanation: The child process can be a duplicate of the parent process. The child process created by fork consists of a copy of the address space of the parent process.

8. The child process completes execution, but the parent keeps executing, then the child process is known as _____

- a) Orphan
- b) Zombie
- c) Body
- d) Dead

[View Answer](#)

Answer: b

Explanation: The child process completes execution, but the parent keeps executing, then the child process is known as Zombie. When a child process terminates, its resources get deallocated but its entry in the Process Control Block (PCB) remains there until its parent calls wait system call.

1. What is Interprocess communication?

- a) allows processes to communicate and synchronize their actions when using the same address space
- b) allows processes to communicate and synchronize their actions
- c) allows the processes to only synchronize their actions without communication
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: Interprocess Communication allows processes to communicate and synchronize their actions. Interprocess Communication (IPC) mechanism is used by cooperating processes to exchange data and information.

There are two models of IPC:

- Shared Memory
- Message Passing

2. Message passing system allows processes to _____

- a) communicate with each other without sharing the same address space
- b) communicate with one another by resorting to shared data
- c) share data

d) name the recipient or sender of the message

[View Answer](#)

Answer: a

Explanation: Message Passing system allows processes to communicate with each other without sharing the same address space.

3. Which of the following two operations are provided by the IPC facility?

- a) write & delete message
- b) delete & receive message
- c) send & delete message
- d) receive & send message

[View Answer](#)

Answer: d

Explanation: Two operations provided by the IPC facility are receive and send messages. Exchange of data takes place in cooperating processes.

4. Messages sent by a process _____

- a) have to be of a fixed size
- b) have to be a variable size
- c) can be fixed or variable sized
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: Messages sent by a process can be fixed or variable size. If the message size of the process is fixed then system level implementation is straightforward but it makes the task of programming more difficult. If the message size of the process is variable then system level implementation is more complex but it makes the task of programming simpler.

5. The link between two processes P and Q to send and receive messages is called _____

- a) communication link
- b) message-passing link
- c) synchronization link
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: The link between two processes P and Q to send and receive messages is called communication link. Two processes P and Q want to

communicate with each other; there should be a communication link that must exist between these two processes so that both processes can able to send and receive messages using that link.

6. Which of the following are TRUE for direct communication?

- a) A communication link can be associated with N number of process(N = max. number of processes supported by system)
- b) A communication link is associated with exactly two processes
- c) Exactly $N/2$ links exist between each pair of processes(N = max. number of processes supported by system)
- d) Exactly two link exists between each pair of processes

[View Answer](#)

Answer: b

Explanation: For direct communication, a communication link is associated with exactly two processes. One communication link must exist between a pair of processes.

7. In indirect communication between processes P and Q _____

- a) there is another process R to handle and pass on the messages between P and Q
- b) there is another machine between the two processes to help communication
- c) there is a mailbox to help communication between P and Q
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: In indirect communication between processes P and Q there is a mailbox to help communication between P and Q. A mailbox can be viewed abstractly as an object into which messages can be placed by processes and from which messages can be removed.

8. In the non blocking send _____

- a) the sending process keeps sending until the message is received
- b) the sending process sends the message and resumes operation
- c) the sending process keeps sending until it receives a message
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: In the non blocking send, the sending process sends the message and resumes operation. Sending process doesn't care about reception. It is also known as asynchronous send.

9. In the Zero capacity queue _____

- a) the queue can store at least one message
- b) the sender blocks until the receiver receives the message
- c) the sender keeps sending and the messages don't wait in the queue
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: In the Zero capacity queue the sender blocks until the receiver receives the message. Zero capacity queue has maximum capacity of Zero; thus message queue does not have any waiting message in it.

10. The Zero Capacity queue _____

- a) is referred to as a message system with buffering
- b) is referred to as a message system with no buffering
- c) is referred to as a link
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: The Zero capacity queue is referred to as a message system with no buffering. Zero capacity queue has maximum capacity of Zero; thus message queue does not have any waiting message in it.

11. Bounded capacity and Unbounded capacity queues are referred to as _____

- a) Programmed buffering
- b) Automatic buffering
- c) User defined buffering
- d) No buffering

[View Answer](#)

Answer: b

Explanation: Bounded capacity and Unbounded capacity queues are referred to as Automatic buffering. Buffer capacity of the Bounded capacity queue is finite length and buffer capacity of the Unbounded queue is infinite.

1. Remote Procedure Calls are used _____

- a) for communication between two processes remotely different from each other on the same system
- b) for communication between two processes on the same system
- c) for communication between two processes on separate systems

d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. To differentiate the many network services a system supports _____ are used.

a) Variables

b) Sockets

c) Ports

d) Service names

[View Answer](#)

Answer: c

Explanation: None.

3. RPC provides a(an) _____ on the client-side, a separate one for each remote procedure.

a) stub

b) identifier

c) name

d) process identifier

[View Answer](#)

Answer: a

Explanation: None.

4. What is stub?

a) transmits the message to the server where the server side stub receives the message and invokes procedure on the server side

b) packs the parameters into a form transmittable over the network

c) locates the port on the server

d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. To resolve the problem of data representation on different systems RPCs define _____

a) machine dependent representation of data

- b) machine representation of data
- c) machine-independent representation of data
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. What is the full form of RMI?

- a) Remote Memory Installation
- b) Remote Memory Invocation
- c) Remote Method Installation
- d) Remote Method Invocation

[View Answer](#)

Answer: d

Explanation: None.

7. The remote method invocation _____

- a) allows a process to invoke memory on a remote object
- b) allows a thread to invoke a method on a remote object
- c) allows a thread to invoke memory on a remote object
- d) allows a process to invoke a method on a remote object

[View Answer](#)

Answer: b

Explanation: None.

8. A process that is based on IPC mechanism which executes on different systems and can communicate with other processes using message based communication, is called _____

- a) Local Procedure Call
- b) Inter Process Communication
- c) Remote Procedure Call
- d) Remote Machine Invocation

[View Answer](#)

Answer: c

Explanation: None.

1. The initial program that is run when the computer is powered up is called _____

- a) boot program

- b) bootloader
- c) initializer
- d) bootstrap program

[View Answer](#)

Answer: d

Explanation: None.

2. How does the software trigger an interrupt?

- a) Sending signals to CPU through bus
- b) Executing a special operation called system call
- c) Executing a special program called system program
- d) Executing a special program called interrupt trigger program

[View Answer](#)

Answer: b

Explanation: None.

3. What is a trap/exception?

- a) hardware generated interrupt caused by an error
- b) software generated interrupt caused by an error
- c) user generated interrupt caused by an error
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. What is an ISR?

- a) Information Service Request
- b) Interrupt Service Request
- c) Interrupt Service Routine
- d) Information Service Routine

[View Answer](#)

Answer: c

Explanation: None.

5. What is an interrupt vector?

- a) It is an address that is indexed to an interrupt handler
- b) It is a unique device number that is indexed by an address
- c) It is a unique identity given to an interrupt

d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. DMA is used for _____

- a) High speed devices(disks and communications network)
- b) Low speed devices
- c) Utilizing CPU cycles
- d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. In a memory mapped input/output _____

- a) the CPU uses polling to watch the control bit constantly, looping to see if a device is ready
- b) the CPU writes one data byte to the data register and sets a bit in control register to show that a byte is available
- c) the CPU receives an interrupt when the device is ready for the next byte
- d) the CPU runs a user written code and does accordingly

[View Answer](#)

Answer: b

Explanation: None.

8. In a programmed input/output(PIO) _____

- a) the CPU uses polling to watch the control bit constantly, looping to see if a device is ready
- b) the CPU writes one data byte to the data register and sets a bit in control register to show that a byte is available
- c) the CPU receives an interrupt when the device is ready for the next byte
- d) the CPU runs a user written code and does accordingly

[View Answer](#)

Answer: a

Explanation: None.

9. In an interrupt driven input/output _____

- a) the CPU uses polling to watch the control bit constantly, looping to see if a device is ready

- b) the CPU writes one data byte to the data register and sets a bit in control register to show that a byte is available
- c) the CPU receives an interrupt when the device is ready for the next byte
- d) the CPU runs a user written code and does accordingly

[View Answer](#)

Answer: c

Explanation: None.

10. In the layered approach of Operating Systems _____

- a) Bottom Layer(0) is the User interface
- b) Highest Layer(N) is the User interface
- c) Bottom Layer(N) is the hardware
- d) Highest Layer(N) is the hardware

[View Answer](#)

Answer: b

Explanation: None.

11. How does the Hardware trigger an interrupt?

- a) Sending signals to CPU through a system bus
- b) Executing a special program called interrupt program
- c) Executing a special program called system program
- d) Executing a special operation called system call

[View Answer](#)

Answer: a

Explanation: None.

12. Which operation is performed by an interrupt handler?

- a) Saving the current state of the system
- b) Loading the interrupt handling code and executing it
- c) Once done handling, bringing back the system to the original state it was before the interrupt occurred
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

1. Which module gives control of the CPU to the process selected by the short-term scheduler?

- a) dispatcher

- b) interrupt
- c) scheduler
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called _____

- a) job queue
- b) ready queue
- c) execution queue
- d) process queue

[View Answer](#)

Answer: b

Explanation: None.

3. The interval from the time of submission of a process to the time of completion is termed as _____

- a) waiting time
- b) turnaround time
- c) response time
- d) throughput

[View Answer](#)

Answer: b

Explanation: None.

4. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?

- a) first-come, first-served scheduling
- b) shortest job scheduling
- c) priority scheduling
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. In priority scheduling algorithm _____

- a) CPU is allocated to the process with highest priority

- b) CPU is allocated to the process with lowest priority
- c) Equal priority processes can not be scheduled
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. In priority scheduling algorithm, when a process arrives at the ready queue, its priority is compared with the priority of _____

- a) all process
- b) currently running process
- c) parent process
- d) init process

[View Answer](#)

Answer: b

Explanation: None.

7. Which algorithm is defined in Time quantum?

- a) shortest job scheduling algorithm
- b) round robin scheduling algorithm
- c) priority scheduling algorithm
- d) multilevel queue scheduling algorithm

[View Answer](#)

Answer: b

Explanation: None.

8. Process are classified into different groups in _____

- a) shortest job scheduling algorithm
- b) round robin scheduling algorithm
- c) priority scheduling algorithm
- d) multilevel queue scheduling algorithm

[View Answer](#)

Answer: d

Explanation: None.

9. In multilevel feedback scheduling algorithm _____

- a) a process can move to a different classified ready queue
- b) classification of ready queue is permanent

- c) processes are not classified into groups
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. Which one of the following can not be scheduled by the kernel?

- a) kernel level thread
- b) user level thread
- c) process
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: User level threads are managed by thread library and the kernel is unaware of them.

1. CPU scheduling is the basis of _____

- a) multiprocessor systems
- b) multiprogramming operating systems
- c) larger memory sized systems
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. With multiprogramming _____ is used productively.

- a) time
- b) space
- c) money
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. What are the two steps of a process execution?

- a) I/O & OS Burst
- b) CPU & I/O Burst
- c) Memory & I/O Burst

d) OS & Memory Burst

[View Answer](#)

Answer: b

Explanation: None.

4. An I/O bound program will typically have _____

- a) a few very short CPU bursts
- b) many very short I/O bursts
- c) many very short CPU bursts
- d) a few very short I/O bursts

[View Answer](#)

Answer: c

Explanation: None.

5. A process is selected from the _____ queue by the _____ scheduler, to be executed.

- a) blocked, short term
- b) wait, long term
- c) ready, short term
- d) ready, long term

[View Answer](#)

Answer: c

Explanation: None.

6. In the following cases non - preemptive scheduling occurs?

- a) When a process switches from the running state to the ready state
- b) When a process goes from the running state to the waiting state
- c) When a process switches from the waiting state to the ready state
- d) All of the mentioned

[View Answer](#)

Answer: b

Explanation: There is no other choice.

7. The switching of the CPU from one process or thread to another is called _____

- a) process switch
- b) task switch
- c) context switch

d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. What is Dispatch latency?

- a) the speed of dispatching a process from running to the ready state
- b) the time of dispatching a process from running to ready state and keeping the CPU idle
- c) the time to stop one process and start running another one
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. Scheduling is done so as to _____

- a) increase CPU utilization
- b) decrease CPU utilization
- c) keep the CPU more idle
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. Scheduling is done so as to _____

- a) increase the throughput
- b) decrease the throughput
- c) increase the duration of a specific amount of work
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. What is Turnaround time?

- a) the total waiting time for a process to finish execution
- b) the total time spent in the ready queue
- c) the total time spent in the running queue
- d) the total time from the completion till the submission of a process

[View Answer](#)

Answer: d

Explanation: None.

12. Scheduling is done so as to _____

- a) increase the turnaround time
- b) decrease the turnaround time
- c) keep the turnaround time same
- d) there is no relation between scheduling and turnaround time

[View Answer](#)

Answer: b

Explanation: None.

13. What is Waiting time?

- a) the total time in the blocked and waiting queues
- b) the total time spent in the ready queue
- c) the total time spent in the running queue
- d) the total time from the completion till the submission of a process

[View Answer](#)

Answer: b

Explanation: None.

14. Scheduling is done so as to _____

- a) increase the waiting time
- b) keep the waiting time the same
- c) decrease the waiting time
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

15. What is Response time?

- a) the total time taken from the submission time till the completion time
- b) the total time taken from the submission time till the first response is produced
- c) the total time taken from submission time till the response is output
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. Round robin scheduling falls under the category of _____

- a) Non-preemptive scheduling
- b) Preemptive scheduling
- c) All of the mentioned
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. With round robin scheduling algorithm in a time shared system

- a) using very large time slices converts it into First come First served scheduling algorithm
- b) using very small time slices converts it into First come First served scheduling algorithm
- c) using extremely small time slices increases performance
- d) using very small time slices converts it into Shortest Job First algorithm

[View Answer](#)

Answer: a

Explanation: All the processes will be able to get completed.

3. The portion of the process scheduler in an operating system that dispatches processes is concerned with _____

- a) assigning ready processes to CPU
- b) assigning ready processes to waiting queue
- c) assigning running processes to blocked queue
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. Complex scheduling algorithms _____

- a) are very appropriate for very large computers
- b) use minimal resources
- c) use many resources
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: Large computers are overloaded with a greater number of processes.

5. What is FIFO algorithm?

- a) first executes the job that came in last in the queue
- b) first executes the job that came in first in the queue
- c) first executes the job that needs minimal processor
- d) first executes the job that has maximum processor needs

[View Answer](#)

Answer: b

Explanation: None.

6. The strategy of making processes that are logically runnable to be temporarily suspended is called _____

- a) Non preemptive scheduling
- b) Preemptive scheduling
- c) Shortest job first
- d) First come First served

[View Answer](#)

Answer: b

Explanation: None.

7. What is Scheduling?

- a) allowing a job to use the processor
- b) making proper use of processor
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. There are 10 different processes running on a workstation. Idle processes are waiting for an input event in the input queue. Busy processes are scheduled with the Round-Robin time sharing method. Which out of the following quantum times is the best value for small response times, if the processes have a short runtime, e.g. less than 10ms?

- a) $tQ = 15\text{ms}$
- b) $tQ = 40\text{ms}$

c) $tQ = 45\text{ms}$

d) $tQ = 50\text{ms}$

[View Answer](#)

Answer: a

Explanation: None.

9. Orders are processed in the sequence they arrive if _____ rule sequences the jobs.

- a) earliest due date
- b) slack time remaining
- c) first come, first served
- d) critical ratio

[View Answer](#)

Answer: c

Explanation: None.

10. Which of the following algorithms tends to minimize the process flow time?

- a) First come First served
- b) Shortest Job First
- c) Earliest Deadline First
- d) Longest Job First

[View Answer](#)

Answer: b

Explanation: None.

11. Under multiprogramming, turnaround time for short jobs is usually _____ and that for long jobs is slightly _____

- a) Lengthened; Shortened
- b) Shortened; Lengthened
- c) Shortened; Shortened
- d) Shortened; Unchanged

[View Answer](#)

Answer: b

Explanation: None.

12. Which of the following statements are true? (GATE 2010)

- I. Shortest remaining time first scheduling may cause starvation

II. Preemptive scheduling may cause starvation

III. Round robin is better than FCFS in terms of response time

- a) I only
- b) I and III only
- c) II and III only
- d) I, II and III

[View Answer](#)

Answer: d

Explanation: I) Shortest remaining time first scheduling is a preemptive version of shortest job scheduling. It may cause starvation as shorter processes may keep coming and a long CPU burst process never gets CPU.

II) Preemption may cause starvation. If priority based scheduling with preemption is used, then a low priority process may never get CPU.

III) Round Robin Scheduling improves response time as all processes get CPU after a specified time.

1. Which is the most optimal scheduling algorithm?

- a) FCFS - First come First served
- b) SJF - Shortest Job First
- c) RR - Round Robin
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. The real difficulty with SJF in short term scheduling is _____

- a) it is too good an algorithm
- b) knowing the length of the next CPU request
- c) it is too complex to understand
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. The FCFS algorithm is particularly troublesome for _____

- a) time sharing systems
- b) multiprogramming systems
- c) multiprocessor systems

d) operating systems

[View Answer](#)

Answer: b

Explanation: In a time sharing system, each user needs to get a share of the CPU at regular intervals.

4. Consider the following set of processes, the length of the CPU burst time given in milliseconds.

Process	Burst time
P1	6
P2	8
P3	7
P4	3

Assuming the above process being scheduled with the SJF scheduling algorithm.

- a) The waiting time for process P1 is 3ms
- b) The waiting time for process P1 is 0ms
- c) The waiting time for process P1 is 16ms
- d) The waiting time for process P1 is 9ms

[View Answer](#)

Answer: a

Explanation: None.

5. Preemptive Shortest Job First scheduling is sometimes called

- a) Fast SJF scheduling
- b) EDF scheduling - Earliest Deadline First
- c) HRRN scheduling - Highest Response Ratio Next
- d) SRTN scheduling - Shortest Remaining Time Next

[View Answer](#)

Answer: d

Explanation: None.

6. An SJF algorithm is simply a priority algorithm where the priority is

- a) the predicted next CPU burst
- b) the inverse of the predicted next CPU burst
- c) the current CPU burst

d) anything the user wants

[View Answer](#)

Answer: a

Explanation: The larger the CPU burst, the lower the priority.

7. Choose one of the disadvantages of the priority scheduling algorithm?

- a) it schedules in a very complex manner
- b) its scheduling takes up a lot of time
- c) it can lead to some low priority process waiting indefinitely for the CPU
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

8. What is 'Aging'?

- a) keeping track of cache contents
- b) keeping track of what pages are currently residing in memory
- c) keeping track of how many times a given page is referenced
- d) increasing the priority of jobs to ensure termination in a finite time

[View Answer](#)

Answer: d

Explanation: None.

9. A solution to the problem of indefinite blockage of low - priority processes is

-
- a) Starvation
 - b) Wait queue
 - c) Ready queue
 - d) Aging

[View Answer](#)

Answer: d

Explanation: None.

10. Which of the following statements are true? (GATE 2010)

- i) Shortest remaining time first scheduling may cause starvation
- ii) Preemptive scheduling may cause starvation

iii) Round robin is better than FCFS in terms of response time

- a) i only
- b) i and iii only
- c) ii and iii only
- d) i, ii and iii

[View Answer](#)

Answer: d

Explanation: None.

11. Which of the following scheduling algorithms gives minimum average waiting time?

- a) FCFS
- b) SJF
- c) Round - robin
- d) Priority

[View Answer](#)

Answer: b

Explanation: None.

1. Concurrent access to shared data may result in _____

- a) data consistency
- b) data insecurity
- c) data inconsistency
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. A situation where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which access takes place is called _____

- a) data consistency
- b) race condition
- c) aging
- d) starvation

[View Answer](#)

Answer: b

Explanation: None.

3. The segment of code in which the process may change common variables, update tables, write into files is known as _____

- a) program
- b) critical section
- c) non - critical section
- d) synchronizing

[View Answer](#)

Answer: b

Explanation: None.

4. Which of the following conditions must be satisfied to solve the critical section problem?

- a) Mutual Exclusion
- b) Progress
- c) Bounded Waiting
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. Mutual exclusion implies that _____

- a) if a process is executing in its critical section, then no other process must be executing in their critical sections
- b) if a process is executing in its critical section, then other processes must be executing in their critical sections
- c) if a process is executing in its critical section, then all the resources of the system must be blocked until it finishes execution
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. Bounded waiting implies that there exists a bound on the number of times a process is allowed to enter its critical section _____

- a) after a process has made a request to enter its critical section and before the request is granted
- b) when another process is in its critical section
- c) before a process has made a request to enter its critical section
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. A minimum of _____ variable(s) is/are required to be shared between processes to solve the critical section problem.

- a) one
- b) two
- c) three
- d) four

[View Answer](#)

Answer: b

Explanation: None.

8. In the bakery algorithm to solve the critical section problem _____

- a) each process is put into a queue and picked up in an ordered manner
- b) each process receives a number (may or may not be unique) and the one with the lowest number is served next
- c) each process gets a unique number and the one with the highest number is served next
- d) each process gets a unique number and the one with the lowest number is served next

[View Answer](#)

Answer: b

Explanation: None.

1. An un-interruptible unit is known as _____

- a) single
- b) atomic
- c) static
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. TestAndSet instruction is executed _____

- a) after a particular process
- b) periodically
- c) atomically
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. Semaphore is a/an _____ to solve the critical section problem.

- a) hardware for a system
- b) special program for a system
- c) integer variable
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

4. What are the two atomic operations permissible on semaphores?

- a) wait
- b) stop
- c) hold
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. What are Spinlocks?

- a) CPU cycles wasting locks over critical sections of programs
- b) Locks that avoid time wastage in context switches
- c) Locks that work better on multiprocessor systems
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

6. What is the main disadvantage of spinlocks?

- a) they are not sufficient for many process
- b) they require busy waiting
- c) they are unreliable sometimes
- d) they are too complex for programmers

[View Answer](#)

Answer: b

Explanation: None.

7. The wait operation of the semaphore basically works on the basic _____ system call.

- a) stop()
- b) block()
- c) hold()
- d) wait()

[View Answer](#)

Answer: b

Explanation: None.

8. The signal operation of the semaphore basically works on the basic _____ system call.

- a) continue()
- b) wakeup()
- c) getup()
- d) start()

[View Answer](#)

Answer: b

Explanation: None.

9. If the semaphore value is negative _____

- a) its magnitude is the number of processes waiting on that semaphore
- b) it is invalid
- c) no operation can be further performed on it until the signal operation is performed on it
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. The code that changes the value of the semaphore is _____

- a) remainder section code
- b) non - critical section code
- c) critical section code
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

11. The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialized as $S0 = 1$, $S1 = 0$, $S2 = 0$.

Process P0

```
while(true)
{
    wait(S0);
    print '0';
    release(S1);
    release(S2);
}
```

Process P1

```
wait(S1);
release(S0);
```

Process P2

```
wait(S2);
release(S0);
```

How many times will P0 print '0'?

- a) At least twice
- b) Exactly twice
- c) Exactly thrice
- d) Exactly once

[View Answer](#)

Answer: a

Explanation: None.

12. Each process P_i , $i = 0, 1, 2, 3, \dots, 9$ is coded as follows.

```
repeat
P(mutex)
{CriticalSection}
V(mutex)
```

```
forever
```

The code for P10 is identical except that it uses V(mutex) instead of P(mutex). What is the largest number of processes that can be inside the critical section at any moment (the mutex being initialized to 1)?

- a) 1
- b) 2
- c) 3
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: Any one of the 9 processes can get into critical section after executing P(mutex) which decrements the mutex value to 0. At this time P10 can enter critical section by incrementing the value to 1. Now any of the 9 processes can enter the critical section by again decrementing the mutex value to 0. None of the remaining processes can get into their critical sections.

13. Two processes, P1 and P2, need to access a critical section of code. Consider the following synchronization construct used by the processes.

Process P1 :

```
while(true)
{
    w1 = true;
    while(w2 == true);
    Critical section
    w1 = false;
}
```

Remainder Section

Process P2 :

```
while(true)
{
    w2 = true;
    while(w1 == true);
    Critical section
```

```
w2 = false;
```

```
}
```

Remainder Section

Here, w1 and w2 have shared variables, which are initialized to false. Which one of the following statements is TRUE about the above construct?

- a) It does not ensure mutual exclusion
- b) It does not ensure bounded waiting
- c) It requires that processes enter the critical section in strict alternation
- d) It does not prevent deadlocks but ensures mutual exclusion

[View Answer](#)

Answer: d

Explanation: None.

1. What will happen if a non-recursive mutex is locked more than once?

- a) Starvation
- b) Deadlock
- c) Aging
- d) Signaling

[View Answer](#)

Answer: b

Explanation: If a thread which had already locked a mutex, tries to lock the mutex again, it will enter into the waiting list of that mutex, which results in a deadlock. It is because no other thread can unlock the mutex.

2. What is a semaphore?

- a) is a binary mutex
- b) must be accessed from only one process
- c) can be accessed from multiple processes
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. What are the two kinds of semaphores?

- a) mutex & counting
- b) binary & counting
- c) counting & decimal
- d) decimal & binary

[View Answer](#)

Answer: b

Explanation: None.

4. What is a mutex?

- a) is a binary mutex
- b) must be accessed from only one process
- c) can be accessed from multiple processes
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. At a particular time of computation the value of a counting semaphore is 7. Then 20 P operations and 15 V operations were completed on this semaphore. The resulting value of the semaphore is? (GATE 1987)

- a) 42
- b) 2
- c) 7
- d) 12

[View Answer](#)

Answer: b

Explanation: P represents Wait and V represents Signal. P operation will decrease the value by 1 every time and V operation will increase the value by 1 every time.

6. A binary semaphore is a semaphore with integer values _____

- a) 1
- b) -1
- c) 0.8
- d) 0.5

[View Answer](#)

Answer: a

Explanation: None.

7. The following pair of processes share a common variable X.

Process A

```
int Y;  
A1: Y = X*2;
```

```
A2: X = Y;
```

Process B

```
int Z;  
B1: Z = X+1;  
B2: X = Z;
```

X is set to 5 before either process begins execution. As usual, statements within a process are executed sequentially, but statements in process A may execute in any order with respect to statements in process B.

How many different values of X are possible after both processes finish executing?

- a) two
- b) three
- c) four
- d) eight

[View Answer](#)

Answer: c

Explanation: Here are the possible ways in which statements from A and B can be interleaved.

```
A1 A2 B1 B2: X = 11  
A1 B1 A2 B2: X = 6  
A1 B1 B2 A2: X = 10  
B1 A1 B2 A2: X = 10  
B1 A1 A2 B2: X = 6  
B1 B2 A1 A2: X = 12.
```

8. The program follows to use a shared binary semaphore T.

Process A

```
int Y;  
A1: Y = X*2;  
A2: X = Y;  
signal(T);
```

Process B

```
int Z;  
B1: wait(T);
```

B2: $Z = X+1;$

$X = Z;$

T is set to 0 before either process begins execution and, as before, X is set to 5.

Now, how many different values of X are possible after both processes finish executing?

- a) one
- b) two
- c) three
- d) four

[View Answer](#)

Answer: a

Explanation: The semaphore T ensures that all the statements from A finish execution before B begins. So now there is only one way in which statements from A and B can be interleaved:

A1 A2 B1 B2: $X = 11.$

9. Semaphores are mostly used to implement _____

- a) System calls
- b) IPC mechanisms
- c) System protection
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

10. Spinlocks are intended to provide _____ only.

- a) Mutual Exclusion
- b) Bounded Waiting
- c) Aging
- d) Progress

[View Answer](#)

Answer: b

Explanation: None.

1. The bounded buffer problem is also known as _____

- a) Readers - Writers problem
- b) Dining - Philosophers problem
- c) Producer - Consumer problem

d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. In the bounded buffer problem, there are the empty and full semaphores that _____

- a) count the number of empty and full buffers
- b) count the number of empty and full memory spaces
- c) count the number of empty and full queues
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. In the bounded buffer problem _____

- a) there is only one buffer
- b) there are n buffers (n being greater than one but finite)
- c) there are infinite buffers
- d) the buffer size is bounded

[View Answer](#)

Answer: b

Explanation: None.

4. To ensure difficulties do not arise in the readers - writers problem _____ are given exclusive access to the shared object.

- a) readers
- b) writers
- c) readers and writers
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. The dining - philosophers problem will occur in case of _____

- a) 5 philosophers and 5 chopsticks
- b) 4 philosophers and 5 chopsticks
- c) 3 philosophers and 5 chopsticks

d) 6 philosophers and 5 chopsticks

[View Answer](#)

Answer: a

Explanation: None.

6. A deadlock free solution to the dining philosophers problem _____

- a) necessarily eliminates the possibility of starvation
- b) does not necessarily eliminate the possibility of starvation
- c) eliminates any possibility of any kind of problem further
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. All processes share a semaphore variable **mutex**, initialized to 1. Each process must execute `wait(mutex)` before entering the critical section and `signal(mutex)` afterward.

Suppose a process executes in the following manner.

```
signal(mutex);
```

```
.....
```

```
critical section
```

```
.....
```

```
wait(mutex);
```

In this situation :

- a) a deadlock will occur
- b) processes will starve to enter critical section
- c) several processes maybe executing in their critical section
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

8. All processes share a semaphore variable **mutex**, initialized to 1. Each process must execute `wait(mutex)` before entering the critical section and `signal(mutex)` afterward.

Suppose a process executes in the following manner.

```
wait(mutex);
```

.....

critical section

.....

```
wait(mutex);
```

- a) a deadlock will occur
- b) processes will starve to enter critical section
- c) several processes maybe executing in their critical section
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared boolean variables S1 and S2 are randomly assigned. (GATE 2010)

Method used by P1 :

```
while(S1==S2);
```

Critical section

```
S1 = S2;
```

Method used by P2 :

```
while(S1!=S2);
```

Critical section

```
S2 = not(S1);
```

Which of the following statements describes properties achieved?

- a) Mutual exclusion but not progress
- b) Progress but not mutual exclusion
- c) Neither mutual exclusion nor progress
- d) Both mutual exclusion and progress

[View Answer](#)

Answer: d

Explanation: None.

1. A monitor is a type of _____

- a) semaphore
- b) low level synchronization construct
- c) high level synchronization construct
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. A monitor is characterized by _____

- a) a set of programmer defined operators
- b) an identifier
- c) the number of variables in it
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. A procedure defined within a _____ can access only those variables declared locally within the _____ and its formal parameters.

- a) process, semaphore
- b) process, monitor
- c) semaphore, semaphore
- d) monitor, monitor

[View Answer](#)

Answer: d

Explanation: None.

4. The monitor construct ensures that _____

- a) only one process can be active at a time within the monitor
- b) n number of processes can be active at a time within the monitor (n being greater than 1)
- c) the queue has only one process in it at a time
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. What are the operations that can be invoked on a condition variable?

- a) wait & signal
- b) hold & wait
- c) signal & hold
- d) continue & signal

[View Answer](#)

Answer: a

Explanation: None.

6. Which is the process of invoking the wait operation?

- a) suspended until another process invokes the signal operation
- b) waiting for another process to complete before it can itself call the signal operation
- c) stopped until the next process in the queue finishes execution
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. If no process is suspended, the signal operation _____

- a) puts the system into a deadlock state
- b) suspends some default process execution
- c) nothing happens
- d) the output is unpredictable

[View Answer](#)

Answer: c

Explanation: None.

1. A collection of instructions that performs a single logical function is called

-
- a) transaction
 - b) operation
 - c) function
 - d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. A terminated transaction that has completed its execution successfully is _____ otherwise it is _____

- a) committed, destroyed
- b) aborted, destroyed
- c) committed, aborted
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. The state of the data accessed by an aborted transaction must be restored to what it was just before the transaction started executing. This restoration is known as _____ of transaction.

- a) safety
- b) protection
- c) roll - back
- d) revert - back

[View Answer](#)

Answer: c

Explanation: None.

4. Write ahead logging is a way _____

- a) to ensure atomicity
- b) to keep data consistent
- c) that records data on stable storage
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. In the write ahead logging a _____ is maintained.

- a) a memory
- b) a system
- c) a disk
- d) a log record

[View Answer](#)

Answer: d

Explanation: None.

6. An actual update is not allowed to a data item _____
- a) before the corresponding log record is written out to stable storage
 - b) after the corresponding log record is written out to stable storage
 - c) until the whole log record has been checked for inconsistencies
 - d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. The undo and redo operations must be _____ to guarantee correct behaviour, even if a failure occurs during recovery process.

- a) idempotent
- b) easy
- c) protected
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: Idempotent - Multiple executions of an operation have the same result as does one execution.

8. The system periodically performs checkpoints that consists of the following operation(s) _____

- a) Putting all the log records currently in main memory onto stable storage
- b) putting all modified data residing in main memory onto stable storage
- c) putting a log record onto stable storage
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. Consider a transaction T1 that committed prior to checkpoint. The <T1 commits> record appears in the log before the <checkpoint> record. Any modifications made by T1 must have been written to the stable storage either with the checkpoint or prior to it. Thus at recovery time _____

- a) There is a need to perform an undo operation on T1
- b) There is a need to perform a redo operation on T1
- c) There is no need to perform an undo and redo operation on T1
- d) All of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. Serializable schedules are ones where _____

- a) concurrent execution of transactions is equivalent to the transactions executed serially
- b) the transactions can be carried out one after the other
- c) a valid result occurs after execution transactions
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. A locking protocol is one that _____

- a) governs how locks are acquired
- b) governs how locks are released
- c) governs how locks are acquired and released
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

12. The two phase locking protocol consists of _____

- a) growing & shrinking phase
- b) shrinking & creation phase
- c) creation & growing phase
- d) destruction & creation phase

[View Answer](#)

Answer: a

Explanation: None.

13. The growing phase is a phase in which?

- a) A transaction may obtain locks, but does not release any
- b) A transaction may obtain locks, and releases a few or all of them
- c) A transaction may release locks, but does not obtain any new locks
- d) A transaction may release locks, and does obtain new locks

[View Answer](#)

Answer: a

Explanation: None.

14. The shrinking phase is a phase in which?

- a) A transaction may obtain locks, but does not release any
- b) A transaction may obtain locks, and releases a few or all of them
- c) A transaction may release locks, but does not obtain any new locks
- d) A transaction may release locks, and does obtain new locks

[View Answer](#)

Answer: c

Explanation: None.

15. Which of the following concurrency control protocols ensure both conflict serializability and freedom from deadlock?

- I) 2-phase locking
- II) Timestamp ordering
- a) I only
- b) II only
- c) Both I and II
- d) Neither I nor II

[View Answer](#)

Answer: b

Explanation: None.

1. What is a reusable resource?

- a) that can be used by one process at a time and is not depleted by that use
- b) that can be used by more than one process at a time
- c) that can be shared between various threads
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. Which of the following condition is required for a deadlock to be possible?

- a) mutual exclusion
- b) a process may hold allocated resources while awaiting assignment of other resources
- c) no resource can be forcibly removed from a process holding it
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. A system is in the safe state if _____

- a) the system can allocate resources to each process in some order and still avoid a deadlock
- b) there exist a safe sequence
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. The circular wait condition can be prevented by _____

- a) defining a linear ordering of resource types
- b) using thread
- c) using pipes
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. Which one of the following is the deadlock avoidance algorithm?

- a) banker's algorithm
- b) round-robin algorithm
- c) elevator algorithm
- d) karn's algorithm

[View Answer](#)

Answer: a

Explanation: None.

6. What is the drawback of banker's algorithm?

- a) in advance processes rarely know how much resource they will need
- b) the number of processes changes as time progresses
- c) resource once available can disappear
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. For an effective operating system, when to check for deadlock?

- a) every time a resource request is made

- b) at fixed time intervals
- c) every time a resource request is made at fixed time intervals
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

8. A problem encountered in multitasking when a process is perpetually denied necessary resources is called _____

- a) deadlock
- b) starvation
- c) inversion
- d) aging

[View Answer](#)

Answer: b

Explanation: None.

9. Which one of the following is a visual (mathematical) way to determine the deadlock occurrence?

- a) resource allocation graph
- b) starvation graph
- c) inversion graph
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. To avoid deadlock _____

- a) there must be a fixed number of resources to allocate
- b) resource allocation must be done only once
- c) all deadlocked processes must be aborted
- d) inversion technique can be used

[View Answer](#)

Answer: a

Explanation: None.

1. The number of resources requested by a process _____

- a) must always be less than the total number of resources available in the system

- b) must always be equal to the total number of resources available in the system
- c) must not exceed the total number of resources available in the system
- d) must exceed the total number of resources available in the system

[View Answer](#)

Answer: c

Explanation: None.

2. The request and release of resources are _____

- a) command line statements
- b) interrupts
- c) system calls
- d) special programs

[View Answer](#)

Answer: c

Explanation: None.

3. What are Multithreaded programs?

- a) lesser prone to deadlocks
- b) more prone to deadlocks
- c) not at all prone to deadlocks
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: Multiple threads can compete for shared resources.

4. For a deadlock to arise, which of the following conditions must hold simultaneously?

- a) Mutual exclusion
- b) No preemption
- c) Hold and wait
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. For Mutual exclusion to prevail in the system _____

- a) at least one resource must be held in a non sharable mode
- b) the processor must be a uniprocessor rather than a multiprocessor
- c) there must be at least one resource in a sharable mode

d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: If another process requests that resource (non - shareable resource), the requesting process must be delayed until the resource has been released.

6. For a Hold and wait condition to prevail _____

- a) A process must be not be holding a resource, but waiting for one to be freed, and then request to acquire it
- b) A process must be holding at least one resource and waiting to acquire additional resources that are being held by other processes
- c) A process must hold at least one resource and not be waiting to acquire additional resources
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. Deadlock prevention is a set of methods _____

- a) to ensure that at least one of the necessary conditions cannot hold
- b) to ensure that all of the necessary conditions do not hold
- c) to decide if the requested resources for a process have to be given or not
- d) to recover from a deadlock

[View Answer](#)

Answer: a

Explanation: None.

8. For non sharable resources like a printer, mutual exclusion _____

- a) must exist
- b) must not exist
- c) may exist
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: A printer cannot be simultaneously shared by several processes.

9. For sharable resources, mutual exclusion _____

- a) is required

- b) is not required
- c) may be or may not be required
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: They do not require mutually exclusive access, and hence cannot be involved in a deadlock.

10. To ensure that the hold and wait condition never occurs in the system, it must be ensured that _____

- a) whenever a resource is requested by a process, it is not holding any other resources
- b) each process must request and be allocated all its resources before it begins its execution
- c) a process can request resources only when it has none
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: c - A process may request some resources and use them. Before it can request any additional resources, however it must release all the resources that it is currently allocated.

11. The disadvantage of a process being allocated all its resources before beginning its execution is _____

- a) Low CPU utilization
- b) Low resource utilization
- c) Very high resource utilization
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

12. To ensure no preemption, if a process is holding some resources and requests another resource that cannot be immediately allocated to it _____

- a) then the process waits for the resources be allocated to it
- b) the process keeps sending requests until the resource is allocated to it
- c) the process resumes execution without the resource being allocated to it
- d) then all resources currently being held are preempted

[View Answer](#)

Answer: d

Explanation: None.

13. One way to ensure that the circular wait condition never holds is to

- a) impose a total ordering of all resource types and to determine whether one precedes another in the ordering
- b) to never let a process acquire resources that are held by other processes
- c) to let a process wait for only one resource at a time
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. Each request requires that the system consider the _____ to decide whether the current request can be satisfied or must wait to avoid a future possible deadlock.

- a) resources currently available
- b) processes that have previously been in the system
- c) resources currently allocated to each process
- d) future requests and releases of each process

[View Answer](#)

Answer: a

Explanation: None.

2. Given a priori information about the _____ number of resources of each type that maybe requested for each process, it is possible to construct an algorithm that ensures that the system will never enter a deadlock state.

- a) minimum
- b) average
- c) maximum
- d) approximate

[View Answer](#)

Answer: c

Explanation: None.

3. A deadlock avoidance algorithm dynamically examines the _____ to ensure that a circular wait condition can never exist.

- a) resource allocation state
- b) system storage state

c) operating system

d) resources

View Answer

Answer: a

Explanation: Resource allocation states are used to maintain the availability of the already and current available resources.

4. A state is safe, if _____

a) the system does not crash due to deadlock occurrence

b) the system can allocate resources to each process in some order and still avoid a deadlock

c) the state keeps the system protected and safe

d) all of the mentioned

View Answer

Answer: b

Explanation: None.

5. A system is in a safe state only if there exists a _____

a) safe allocation

b) safe resource

c) safe sequence

d) all of the mentioned

View Answer

Answer: c

Explanation: None.

6. All unsafe states are _____

a) deadlocks

b) not deadlocks

c) fatal

d) none of the mentioned

View Answer

Answer: b

Explanation: None.

7. A system has 12 magnetic tape drives and 3 processes : P0, P1, and P2. Process P0 requires 10 tape drives, P1 requires 4 and P2 requires 9 tape drives.

Process

P0

P1

P2

Maximum needs (process-wise: P0 through P2 top to bottom)

10

4

9

Currently allocated (process-wise)

5

2

2

Which of the following sequence is a safe sequence?

- a) P0, P1, P2
- b) P1, P2, P0
- c) P2, P0, P1
- d) P1, P0, P2

[View Answer](#)

Answer: d

Explanation: None.

8. If no cycle exists in the resource allocation graph _____

- a) then the system will not be in a safe state
- b) then the system will be in a safe state
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

9. The resource allocation graph is not applicable to a resource allocation system

- a) with multiple instances of each resource type
- b) with a single instance of each resource type

- c) single & multiple instances of each resource type
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. The Banker's algorithm is _____ than the resource allocation graph algorithm.

- a) less efficient
- b) more efficient
- c) equal
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. The data structures available in the Banker's algorithm are _____

- a) Available
- b) Need
- c) Allocation
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

12. The content of the matrix Need is _____

- a) Allocation - Available
- b) Max - Available
- c) Max - Allocation
- d) Allocation - Max

[View Answer](#)

Answer: c

Explanation: None.

13. A system with 5 processes P0 through P4 and three resource types A, B, C have A with 10 instances, B with 5 instances, and C with 7 instances. At time t0, the following snapshot has been taken:

Process

P0

P1

P2

P3

P4

Allocation (process-wise : P0 through P4 top TO bottom)

A B C0 1 02 0 03 0 22 1 10 0 2

MAX (process-wise: P0 through P4 top TO bottom)

A B C7 5 33 2 29 0 22 2 24 3 3

Available

A B C3 3 2

The sequence <P1, P3, P4, P2, P0> leads the system to _____

- a) an unsafe state
- b) a safe state
- c) a protected state
- d) a deadlock

[View Answer](#)

Answer: b

Explanation: None.

1. The wait-for graph is a deadlock detection algorithm that is applicable when

- a) all resources have a single instance
- b) all resources have multiple instances
- c) all resources have a single 7 multiple instances
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. An edge from process Pi to Pj in a wait for graph indicates that

- a) Pi is waiting for Pj to release a resource that Pi needs

- b) Pj is waiting for Pi to release a resource that Pj needs
- c) Pi is waiting for Pj to leave the system
- d) Pj is waiting for Pi to leave the system

[View Answer](#)

Answer: a

Explanation: None.

3. If the wait for graph contains a cycle _____
- a) then a deadlock does not exist
 - b) then a deadlock exists
 - c) then the system is in a safe state
 - d) either deadlock exists or system is in a safe state

[View Answer](#)

Answer: b

Explanation: None.

4. If deadlocks occur frequently, the detection algorithm must be invoked _____
- a) rarely
 - b) frequently
 - c) rarely & frequently
 - d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. What is the disadvantage of invoking the detection algorithm for every request?
- a) overhead of the detection algorithm due to consumption of memory
 - b) excessive time consumed in the request to be allocated memory
 - c) considerable overhead in computation time
 - d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. A deadlock eventually cripples system throughput and will cause the CPU utilization to _____
- a) increase

- b) drop
- c) stay still
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. Every time a request for allocation cannot be granted immediately, the detection algorithm is invoked. This will help identify _____

- a) the set of processes that have been deadlocked
- b) the set of processes in the deadlock queue
- c) the specific process that caused the deadlock
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. A computer system has 6 tape drives, with 'n' processes competing for them. Each process may need 3 tape drives. The maximum value of 'n' for which the system is guaranteed to be deadlock free is?

- a) 2
- b) 3
- c) 4
- d) 1

[View Answer](#)

Answer: a

Explanation: None.

9. A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units then, deadlock _____

- a) can never occur
- b) may occur
- c) has to occur
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. 'm' processes share 'n' resources of the same type. The maximum need of each process doesn't exceed 'n' and the sum of all their maximum needs is always less than $m+n$. In this setup, deadlock _____

- a) can never occur
- b) may occur
- c) has to occur
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. A deadlock can be broken by _____

- a) abort one or more processes to break the circular wait
- b) abort all the process in the system
- c) preempt all resources from all processes
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. The two ways of aborting processes and eliminating deadlocks are _____

- a) Abort all deadlocked processes
- b) Abort all processes
- c) Abort one process at a time until the deadlock cycle is eliminated
- d) All of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. Those processes should be aborted on occurrence of a deadlock, the termination of which?

- a) is more time consuming
- b) incurs minimum cost
- c) safety is not hampered
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. The process to be aborted is chosen on the basis of the following factors?

- a) priority of the process
- b) process is interactive or batch
- c) how long the process has computed
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. Cost factors for process termination include _____

- a) Number of resources the deadlock process is not holding
- b) CPU utilization at the time of deadlock
- c) Amount of time a deadlocked process has thus far consumed during its execution
- d) All of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. If we preempt a resource from a process, the process cannot continue with its normal execution and it must be _____

- a) aborted
- b) rolled back
- c) terminated
- d) queued

[View Answer](#)

Answer: b

Explanation: None.

7. To _____ to a safe state, the system needs to keep more information about the states of processes.

- a) abort the process
- b) roll back the process
- c) queue the process
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. If the resources are always preempted from the same process _____ can occur.

- a) deadlock
- b) system crash
- c) aging
- d) starvation

[View Answer](#)

Answer: d

Explanation: None.

9. What is the solution to starvation?

- a) the number of rollbacks must be included in the cost factor
- b) the number of resources must be included in resource preemption
- c) resource preemption be done instead
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. What is Address Binding?

- a) going to an address in memory
- b) locating an address with the help of another address
- c) binding two addresses together to form a new address in a different memory space
- d) a mapping from one address space to another

[View Answer](#)

Answer: d

Explanation: None.

2. Binding of instructions and data to memory addresses can be done at _____

- a) Compile time
- b) Load time
- c) Execution time
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. If the process can be moved during its execution from one memory segment to another, then binding must be _____

- a) delayed until run time
- b) preponed to compile time
- c) preponed to load time
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. What is Dynamic loading?

- a) loading multiple routines dynamically
- b) loading a routine only when it is called
- c) loading multiple routines randomly
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. What is the advantage of dynamic loading?

- a) A used routine is used multiple times
- b) An unused routine is never loaded
- c) CPU utilization increases
- d) All of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

6. The idea of overlays is to _____

- a) data that are needed at any given time
- b) enable a process to be larger than the amount of memory allocated to it
- c) keep in memory only those instructions
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. The _____ must design and program the overlay structure.

- a) programmer

- b) system architect
- c) system designer
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. The _____ swaps processes in and out of the memory.

- a) Memory manager
- b) CPU
- c) CPU manager
- d) User

[View Answer](#)

Answer: a

Explanation: None.

9. If a higher priority process arrives and wants service, the memory manager can swap out the lower priority process to execute the higher priority process. When the higher priority process finishes, the lower priority process is swapped back in and continues execution. This variant of swapping is sometimes called?

- a) priority swapping
- b) pull out, push in
- c) roll out, roll in
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. If binding is done at assembly or load time, then the process _____ be moved to different locations after being swapped out and in again.

- a) can
- b) must
- c) can never
- d) may

[View Answer](#)

Answer: c

Explanation: None.

11. In a system that does not support swapping _____
- a) the compiler normally binds symbolic addresses (variables) to relocatable addresses
 - b) the compiler normally binds symbolic addresses to physical addresses
 - c) the loader binds relocatable addresses to physical addresses
 - d) binding of symbolic addresses to physical addresses normally takes place during execution

[View Answer](#)

Answer: a

Explanation: None.

12. Which of the following is TRUE?
- a) Overlays are used to increase the size of physical memory
 - b) Overlays are used to increase the logical address space
 - c) When overlays are used, the size of a process is not limited to the size of the physical memory
 - d) Overlays are used whenever the physical address space is smaller than the logical address space

[View Answer](#)

Answer: c

Explanation: None.

1. The address generated by the CPU is referred to as _____
- a) Physical address
 - b) Logical address
 - c) Neither physical nor logical
 - d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. The address loaded into the memory address register of the memory is referred to as _____
- a) Physical address
 - b) Logical address
 - c) Neither physical nor logical
 - d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. The run time mapping from virtual to physical addresses is done by a hardware device called the _____

- a) Virtual to physical mapper
- b) Memory management unit
- c) Memory mapping unit
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. The base register is also known as the _____

- a) basic register
- b) regular register
- c) relocation register
- d) delocation register

[View Answer](#)

Answer: c

Explanation: None.

5. The size of a process is limited to the size of _____

- a) physical memory
- b) external storage
- c) secondary storage
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. If execution time binding is being used, then a process _____ be swapped to a different memory space.

- a) has to be
- b) can never
- c) must
- d) may

[View Answer](#)

Answer: d

Explanation: None.

7. Swapping requires a _____

- a) motherboard
- b) keyboard
- c) monitor
- d) backing store

[View Answer](#)

Answer: d

Explanation: None.

8. The backing store is generally a _____

- a) fast disk
- b) disk large enough to accommodate copies of all memory images for all users
- c) disk to provide direct access to the memory images
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. The _____ consists of all processes whose memory images are in the backing store or in memory and are ready to run.

- a) wait queue
- b) ready queue
- c) cpu
- d) secondary storage

[View Answer](#)

Answer: b

Explanation: None.

10. The _____ time in a swap out of a running process and swap in of a new process into the memory is very high.

- a) context - switch
- b) waiting
- c) execution
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. The major part of swap time is _____ time.

- a) waiting
- b) transfer
- c) execution
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

12. Swapping _____ be done when a process has pending I/O, or has to execute I/O operations only into operating system buffers.

- a) must
- b) can
- c) must never
- d) maybe

[View Answer](#)

Answer: c

Explanation: None.

13. Swap space is allocated _____

- a) as a chunk of disk
- b) separate from a file system
- c) into a file system
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. CPU fetches the instruction from memory according to the value of

-
- a) program counter
 - b) status register
 - c) instruction register
 - d) program status word

[View Answer](#)

Answer: a

Explanation: None.

2. A memory buffer used to accommodate a speed differential is called _____

- a) stack pointer
- b) cache
- c) accumulator
- d) disk buffer

View Answer

Answer: b

Explanation: None.

3. Which one of the following is the address generated by CPU?

- a) physical address
- b) absolute address
- c) logical address
- d) none of the mentioned

View Answer

Answer: c

Explanation: None.

4. Run time mapping from virtual to physical address is done by _____

- a) Memory management unit
- b) CPU
- c) PCI
- d) None of the mentioned

View Answer

Answer: a

Explanation: None.

5. Memory management technique in which system stores and retrieves data from secondary storage for use in main memory is called?

- a) fragmentation
- b) paging
- c) mapping
- d) none of the mentioned

View Answer

Answer: b

Explanation: None.

6. The address of a page table in memory is pointed by _____

- a) stack pointer
- b) page table base register
- c) page register
- d) program counter

[View Answer](#)

Answer: b

Explanation: None.

7. Program always deals with _____

- a) logical address
- b) absolute address
- c) physical address
- d) relative address

[View Answer](#)

Answer: a

Explanation: None.

8. The page table contains _____

- a) base address of each page in physical memory
- b) page offset
- c) page size
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. What is compaction?

- a) a technique for overcoming internal fragmentation
- b) a paging technique
- c) a technique for overcoming external fragmentation
- d) a technique for overcoming fatal error

[View Answer](#)

Answer: c

Explanation: None.

10. Operating System maintains the page table for _____

- a) each process
- b) each thread
- c) each instruction
- d) each address

[View Answer](#)

Answer: a

Explanation: None.

1. The main memory accommodates _____

- a) operating system
- b) cpu
- c) user processes
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. What is the operating system?

- a) in the low memory
- b) in the high memory
- c) either low or high memory (depending on the location of interrupt vector)
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. In contiguous memory allocation _____

- a) each process is contained in a single contiguous section of memory
- b) all processes are contained in a single contiguous section of memory
- c) the memory space is contiguous
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. The relocation register helps in _____

- a) providing more address space to processes
- b) a different address space to processes

- c) to protect the address spaces of processes
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

5. With relocation and limit registers, each logical address must be _____ the limit register.

- a) less than
- b) equal to
- c) greater than
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. The operating system and the other processes are protected from being modified by an already running process because _____

- a) they are in different memory spaces
- b) they are in different logical addresses
- c) they have a protection algorithm
- d) every address generated by the CPU is being checked against the relocation and limit registers

[View Answer](#)

Answer: d

Explanation: None.

7. Transient operating system code is code that _____

- a) is not easily accessible
- b) comes and goes as needed
- c) stays in the memory always
- d) never enters the memory space

[View Answer](#)

Answer: b

Explanation: None.

8. Using transient code, _____ the size of the operating system during program execution.

- a) increases

b) decreases

c) changes

d) maintains

[View Answer](#)

Answer: c

Explanation: None.

9. When memory is divided into several fixed sized partitions, each partition may contain _____

a) exactly one process

b) at least one process

c) multiple processes at once

d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. In fixed size partition, the degree of multiprogramming is bounded by _____

a) the number of partitions

b) the CPU utilization

c) the memory size

d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None

11. The first fit, best fit and worst fit are strategies to select a _____

a) process from a queue to put in memory

b) processor to run the next process

c) free hole from a set of available holes

d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

1. In internal fragmentation, memory is internal to a partition and _____

a) is being used

- b) is not being used
- c) is always used
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. A solution to the problem of external fragmentation is _____

- a) compaction
- b) larger memory space
- c) smaller memory space
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. Another solution to the problem of external fragmentation problem is to

- a) permit the logical address space of a process to be noncontiguous
- b) permit smaller processes to be allocated memory at last
- c) permit larger processes to be allocated memory at last
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. If relocation is static and is done at assembly or load time, compaction

- a) cannot be done
- b) must be done
- c) must not be done
- d) can be done

[View Answer](#)

Answer: a

Explanation: None.

5. The disadvantage of moving all process to one end of memory and all holes to the other direction, producing one large hole of available memory is

- a) the cost incurred
- b) the memory used
- c) the CPU used
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. _____ is generally faster than _____ and _____

- a) first fit, best fit, worst fit
- b) best fit, first fit, worst fit
- c) worst fit, best fit, first fit
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. External fragmentation exists when?

- a) enough total memory exists to satisfy a request but it is not contiguous
- b) the total memory is insufficient to satisfy a request
- c) a request cannot be satisfied even when the total memory is free
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. External fragmentation will not occur when?

- a) first fit is used
- b) best fit is used
- c) worst fit is used
- d) no matter which algorithm is used, it will always occur

[View Answer](#)

Answer: d

Explanation: None.

9. Sometimes the overhead of keeping track of a hole might be _____

- a) larger than the memory
- b) larger than the hole itself
- c) very small

d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

10. When the memory allocated to a process is slightly larger than the process, then _____

- a) internal fragmentation occurs
- b) external fragmentation occurs
- c) both internal and external fragmentation occurs
- d) neither internal nor external fragmentation occurs

[View Answer](#)

Answer: a

Explanation: None.

1. Physical memory is broken into fixed-sized blocks called _____

- a) frames
- b) pages
- c) backing store
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. Logical memory is broken into blocks of the same size called _____

- a) frames
- b) pages
- c) backing store
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. Every address generated by the CPU is divided into two parts. They are _____

- a) frame bit & page number
- b) page number & page offset
- c) page offset & frame bit

d) frame offset & page offset

[View Answer](#)

Answer: b

Explanation: None.

4. The _____ is used as an index into the page table.

- a) frame bit
- b) page number
- c) page offset
- d) frame offset

[View Answer](#)

Answer: b

Explanation: None.

5. The _____ table contains the base address of each page in physical memory.

- a) process
- b) memory
- c) page
- d) frame

[View Answer](#)

Answer: c

Explanation: None.

6. The size of a page is typically _____

- a) varied
- b) power of 2
- c) power of 4
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. If the size of logical address space is 2^m , and a page size is 2^n addressing units, then the high order _____ bits of a logical address designate the page number, and the _____ low order bits designate the page offset.

- a) m, n
- b) n, m
- c) m - n, m

d) $m - n$, n

[View Answer](#)

Answer: d

Explanation: None.

8. With paging there is no _____ fragmentation.

- a) internal
- b) external
- c) either type of
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

9. The operating system maintains a _____ table that keeps track of how many frames have been allocated, how many are there, and how many are available.

- a) page
- b) mapping
- c) frame
- d) memory

[View Answer](#)

Answer: c

Explanation: None.

10. Paging increases the _____ time.

- a) waiting
- b) execution
- c) context - switch
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

11. Smaller page tables are implemented as a set of _____

- a) queues
- b) stacks
- c) counters
- d) registers

[View Answer](#)

Answer: d

Explanation: None.

12. The page table registers should be built with _____

- a) very low speed logic
- b) very high speed logic
- c) a large memory space
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

13. For larger page tables, they are kept in main memory and a _____ points to the page table.

- a) page table base register
- b) page table base pointer
- c) page table register pointer
- d) page table base

[View Answer](#)

Answer: a

Explanation: None.

14. For every process there is a _____

- a) page table
- b) copy of page table
- c) pointer to page table
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

15. Time taken in memory access through PTBR is _____

- a) extended by a factor of 3
- b) extended by a factor of 2
- c) slowed by a factor of 3
- d) slowed by a factor of 2

[View Answer](#)

Answer: d

Explanation: None.

1. Each entry in a translation lookaside buffer (TLB) consists of _____

- a) key
- b) value
- c) bit value
- d) constant

[View Answer](#)

Answer: a

Explanation: None.

2. If a page number is not found in the TLB, then it is known as a

- a) TLB miss
- b) Buffer miss
- c) TLB hit
- d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. An _____ uniquely identifies processes and is used to provide address space protection for that process.

- a) address space locator
- b) address space identifier
- c) address process identifier
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. The percentage of times a page number is found in the TLB is known as

- a) miss ratio
- b) hit ratio
- c) miss percent
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. Memory protection in a paged environment is accomplished by _____
- a) protection algorithm with each page
 - b) restricted access rights to users
 - c) restriction on page visibility
 - d) protection bit with each page

[View Answer](#)

Answer: d

Explanation: None.

6. When the valid - invalid bit is set to valid, it means that the associated page

- a) is in the TLB
- b) has data in it
- c) is in the process's logical address space
- d) is the system's physical address space

[View Answer](#)

Answer: c

Explanation: None.

7. Illegal addresses are trapped using the _____ bit.

- a) error
- b) protection
- c) valid - invalid
- d) access

[View Answer](#)

Answer: c

Explanation: None.

8. When there is a large logical address space, the best way of paging would be

- a) not to page
- b) a two level paging algorithm
- c) the page table itself
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

9. In a paged memory, the page hit ratio is 0.35. The time required to access a page in secondary memory is equal to 100 ns. The time required to access a page in primary memory is 10 ns. The average time required to access a page is?

- a) 3.0 ns
- b) 68.0 ns
- c) 68.5 ns
- d) 78.5 ns

[View Answer](#)

Answer: c

Explanation: None.

10. To obtain better memory utilization, dynamic loading is used. With dynamic loading, a routine is not loaded until it is called. For implementing dynamic loading

-
- a) special support from hardware is required
 - b) special support from operating system is essential
 - c) special support from both hardware and operating system is essential
 - d) user programs can implement dynamic loading without any special support from hardware or operating system

[View Answer](#)

Answer: d

Explanation: None.

11. In paged memory systems, if the page size is increased, then the internal fragmentation generally _____

- a) becomes less
- b) becomes more
- c) remains constant
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. In segmentation, each address is specified by _____

- a) a segment number & offset
- b) an offset & value
- c) a value & segment number
- d) a key & value

[View Answer](#)

Answer: a

Explanation: None.

2. In paging the user provides only _____ which is partitioned by the hardware into _____ and _____
- a) one address, page number, offset
 - b) one offset, page number, address
 - c) page number, offset, address
 - d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. Each entry in a segment table has a _____
- a) segment base
 - b) segment peak
 - c) segment value
 - d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. The segment base contains the _____
- a) starting logical address of the process
 - b) starting physical address of the segment in memory
 - c) segment length
 - d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. The segment limit contains the _____
- a) starting logical address of the process
 - b) starting physical address of the segment in memory
 - c) segment length
 - d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. The offset 'd' of the logical address must be _____

- a) greater than segment limit
- b) between 0 and segment limit
- c) between 0 and the segment number
- d) greater than the segment number

[View Answer](#)

Answer: b

Explanation: None.

7. If the offset is legal _____

- a) it is used as a physical memory address itself
- b) it is subtracted from the segment base to produce the physical memory address
- c) it is added to the segment base to produce the physical memory address
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. When the entries in the segment tables of two different processes point to the same physical location _____

- a) the segments are invalid
- b) the processes get blocked
- c) segments are shared
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. The protection bit is 0/1 based on _____

- a) write only
- b) read only
- c) read - write
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. If there are 32 segments, each of size 1Kb, then the logical address should have _____

- a) 13 bits
- b) 14 bits
- c) 15 bits
- d) 16 bits

[View Answer](#)

Answer: a

Explanation: To specify a particular segment, 5 bits are required. To select a particular byte after selecting a page, 10 more bits are required. Hence 15 bits are required.

11. Consider a computer with 8 Mbytes of main memory and a 128K cache. The cache block size is 4 K. It uses a direct mapping scheme for cache management. How many different main memory blocks can map onto a given physical cache block?

- a) 2048
- b) 256
- c) 64
- d) 8

[View Answer](#)

Answer: c

Explanation: None.

12. A multilevel page table is preferred in comparison to a single level page table for translating virtual address to physical address because _____

- a) it reduces the memory access time to read or write a memory location
- b) it helps to reduce the size of page table needed to implement the virtual address space of a process
- c) it is required by the translation lookaside buffer
- d) it helps to reduce the number of page faults in page replacement algorithms

[View Answer](#)

Answer: b

Explanation: None.

1. If one or more devices use a common set of wires to communicate with the computer system, the connection is called _____

- a) CPU
- b) Monitor
- c) Wirefull

d) Bus

[View Answer](#)

Answer: d

Explanation: None.

2. A _____ a set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires.

- a) port
- b) node
- c) bus
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. When device A has a cable that plugs into device B, and device B has a cable that plugs into device C and device C plugs into a port on the computer, this arrangement is called a _____

- a) port
- b) daisy chain
- c) bus
- d) cable

[View Answer](#)

Answer: b

Explanation: None.

4. The _____ present a uniform device-access interface to the I/O subsystem, much as system calls provide a standard interface between the application and the operating system.

- a) Devices
- b) Buses
- c) Device drivers
- d) I/O systems

[View Answer](#)

Answer: c

Explanation: None.

5. A _____ is a collection of electronics that can operate a port, a bus, or a device.

- a) controller
- b) driver
- c) host
- d) bus

[View Answer](#)

Answer: a

Explanation: None.

6. An I/O port typically consists of four registers status, control, _____ and _____ registers.

- a) system in, system out
- b) data in, data out
- c) flow in, flow out
- d) input, output

[View Answer](#)

Answer: b

Explanation: None.

7. The _____ register is read by the host to get input.

- a) flow in
- b) flow out
- c) data in
- d) data out

[View Answer](#)

Answer: c

Explanation: None.

8. The _____ register is written by the host to send output.

- a) status
- b) control
- c) data in
- d) data out

[View Answer](#)

Answer: d

Explanation: None.

9. The hardware mechanism that allows a device to notify the CPU is called _____

- a) polling

- b) interrupt
- c) driver
- d) controlling

[View Answer](#)

Answer: b

Explanation: None.

10. The CPU hardware has a wire called _____ that the CPU senses after executing every instruction.

- a) interrupt request line
- b) interrupt bus
- c) interrupt receive line
- d) interrupt sense line

[View Answer](#)

Answer: a

Explanation: None.

11. The _____ determines the cause of the interrupt, performs the necessary processing and executes a return from the interrupt instruction to return the CPU to the execution state prior to the interrupt.

- a) interrupt request line
- b) device driver
- c) interrupt handler
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

12. In general the two interrupt request lines are _____

- a) maskable & non maskable interrupts
- b) blocked & non maskable interrupts
- c) maskable & blocked interrupts
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

13. The _____ are reserved for events such as unrecoverable memory errors.

- a) non maskable interrupts
- b) blocked interrupts
- c) maskable interrupts
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. The _____ can be turned off by the CPU before the execution of critical instruction sequences that must not be interrupted.

- a) nonmaskable interrupt
- b) blocked interrupt
- c) maskable interrupt
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. The _____ is used by device controllers to request service.

- a) nonmaskable interrupt
- b) blocked interrupt
- c) maskable interrupt
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. The interrupt vector contains _____

- a) the interrupts
- b) the memory addresses of specialized interrupt handlers
- c) the identifiers of interrupts
- d) the device addresses

[View Answer](#)

Answer: b

Explanation: None.

4. Division by zero, accessing a protected or non existent memory address, or attempting to execute a privileged instruction from user mode are all categorized as _____

- a) errors
- b) exceptions
- c) interrupt handlers
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. For large data transfers, _____ is used.

- a) dma
- b) programmed I/O
- c) controller register
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. A character stream device transfers _____

- a) bytes one by one
- b) block of bytes as a unit
- c) with unpredictable response times
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. A block device transfers _____

- a) bytes one by one
- b) block of bytes as a unit
- c) with unpredictable response times
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. What is a dedicated device?

- a) opposite to a sharable device
- b) same as a sharable device

- c) can be used concurrently by several processes
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. A keyboard is an example of a device that is accessed through a _____ interface.

- a) block stream
- b) set of blocks
- c) character stream
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. In polling _____

- a) busy - wait cycles wait for I/O from device
- b) interrupt handler receives interrupts
- c) interrupt-request line is triggered by I/O device
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. A non blocking system call _____

- a) halts the execution of the application for an extended time
- b) does not halt the execution of the application
- c) does not block the interrupts
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

12. An asynchronous call _____

- a) returns immediately, without waiting for the I/O to complete
- b) does not return immediately and waits for the I/O to complete
- c) consumes a lot of time

d) is too slow

[View Answer](#)

Answer: a

Explanation: None.

1. Buffering is done to _____

- a) cope with device speed mismatch
- b) cope with device transfer size mismatch
- c) maintain copy semantics
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

2. Caching is _____ spooling.

- a) same as
- b) not the same as
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. Caching _____

- a) holds a copy of the data
- b) is fast memory
- c) holds the only copy of the data
- d) holds output for a device

[View Answer](#)

Answer: a

Explanation: None.

4. Spooling _____

- a) holds a copy of the data
- b) is fast memory
- c) holds the only copy of the data
- d) holds output for a device

[View Answer](#)

Answer: c

Explanation: None.

5. The _____ keeps state information about the use of I/O components.

- a) CPU
- b) OS
- c) kernel
- d) shell

[View Answer](#)

Answer: c

Explanation: None.

6. The kernel data structures include _____

- a) process table
- b) open file table
- c) close file table
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. Windows NT uses a _____ implementation for I/O.

- a) message - passing
- b) draft - passing
- c) secondary memory
- d) cache

[View Answer](#)

Answer: a

Explanation: None.

8. A _____ is a full duplex connection between a device driver and a user level process.

- a) Bus
- b) I/O operation
- c) Stream
- d) Flow

[View Answer](#)

Answer: c

Explanation: None.

9. I/O is a _____ in system performance.

- a) major factor
- b) minor factor
- c) does not matter
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. If the number of cycles spent busy - waiting is not excessive, then

- a) interrupt driven I/O is more efficient than programmed I/O
- b) programmed I/O is more efficient than interrupt driven I/O
- c) both programmed and interrupt driven I/O are equally efficient
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. In real time operating system _____

- a) all processes have the same priority
- b) a task must be serviced by its deadline period
- c) process scheduling can be done only once
- d) kernel is not required

[View Answer](#)

Answer: b

Explanation: None.

2. Hard real time operating system has _____ jitter than a soft real time operating system.

- a) less
- b) more
- c) equal
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: Jitter is the undesired deviation from the true periodicity.

3. For real time operating systems, interrupt latency should be _____

- a) minimal
- b) maximum
- c) zero
- d) dependent on the scheduling

[View Answer](#)

Answer: a

Explanation: Interrupt latency is the time duration between the generation of interrupt and execution of its service.

4. In rate monotonic scheduling _____

- a) shorter duration job has higher priority
- b) longer duration job has higher priority
- c) priority does not depend on the duration of the job
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. In which scheduling certain amount of CPU time is allocated to each process?

- a) earliest deadline first scheduling
- b) proportional share scheduling
- c) equal share scheduling
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

6. The problem of priority inversion can be solved by _____

- a) priority inheritance protocol
- b) priority inversion protocol
- c) both priority inheritance and inversion protocol
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. Time duration required for scheduling dispatcher to stop one process and start another is known as _____

- a) process latency
- b) dispatch latency
- c) execution latency
- d) interrupt latency

[View Answer](#)

Answer: b

Explanation: None.

8. Time required to synchronous switch from the context of one thread to the context of another thread is called?

- a) threads fly-back time
- b) jitter
- c) context switch time
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. Which one of the following is a real time operating system?

- a) RTLinux
- b) VxWorks
- c) Windows CE
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

10. VxWorks is centered around _____

- a) wind microkernel
- b) linux kernel
- c) unix kernel
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. What is the disadvantage of real addressing mode?

- a) there is a lot of cost involved
- b) time consumption overhead

- c) absence of memory protection between processes
- d) restricted access to memory locations by processes

[View Answer](#)

Answer: c

Explanation: None.

2. Preemptive, priority based scheduling guarantees _____

- a) hard real time functionality
- b) soft real time functionality
- c) protection of memory
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. Real time systems must have _____

- a) preemptive kernels
- b) non preemptive kernels
- c) preemptive kernels or non preemptive kernels
- d) neither preemptive nor non preemptive kernels

[View Answer](#)

Answer: a

Explanation: None.

4. What is Event latency?

- a) the amount of time an event takes to occur from when the system started
- b) the amount of time from the event occurrence till the system stops
- c) the amount of time from event occurrence till the event crashes
- d) the amount of time that elapses from when an event occurs to when it is serviced.

[View Answer](#)

Answer: d

Explanation: None.

5. Interrupt latency refers to the period of time _____

- a) from the occurrence of an event to the arrival of an interrupt
- b) from the occurrence of an event to the servicing of an interrupt
- c) from arrival of an interrupt to the start of the interrupt service routine

d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. Real time systems need to _____ the interrupt latency.

- a) minimize
- b) maximize
- c) not bother about
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. The amount of time required for the scheduling dispatcher to stop one process and start another is known as _____

- a) event latency
- b) interrupt latency
- c) dispatch latency
- d) context switch

[View Answer](#)

Answer: c

Explanation: None.

8. The most effective technique to keep dispatch latency low is to _____

- a) provide non preemptive kernels
- b) provide preemptive kernels
- c) make it user programmed
- d) run less number of processes at a time

[View Answer](#)

Answer: b

Explanation: None.

9. Priority inversion is solved by use of _____

- a) priority inheritance protocol
- b) two phase lock protocol
- c) time protocol

d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. In a real time system the computer results _____

- a) must be produced within a specific deadline period
- b) may be produced at any time
- c) may be correct
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. In a safety critical system, incorrect operation _____

- a) does not affect much
- b) causes minor problems
- c) causes major and serious problems
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. Antilock brake systems, flight management systems, pacemakers are examples of _____

- a) safety critical system
- b) hard real time system
- c) soft real time system
- d) safety critical system and hard real time system

[View Answer](#)

Answer: d

Explanation: None.

4. In a _____ real time system, it is guaranteed that critical real time tasks will be completed within their deadlines.

- a) soft
- b) hard
- c) critical

d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. Some of the properties of real time systems include _____

- a) single purpose
- b) inexpensively mass produced
- c) small size
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

6. The amount of memory in a real time system is generally _____

- a) less compared to PCs
- b) high compared to PCs
- c) same as in PCs
- d) they do not have any memory

[View Answer](#)

Answer: a

Explanation: None.

7. What is the priority of a real time task?

- a) must degrade over time
- b) must not degrade over time
- c) may degrade over time
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. Memory management units _____

- a) increase the cost of the system
- b) increase the power consumption of the system
- c) increase the time required to complete an operation
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. The technique in which the CPU generates physical addresses directly is known as _____

- a) relocation register method
- b) real addressing
- c) virtual addressing
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. Earliest deadline first algorithm assigns priorities according to _____

- a) periods
- b) deadlines
- c) burst times
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. A process P1 has a period of 50 and a CPU burst of $t_1 = 25$, P2 has a period of 80 and a CPU burst of 35. The total CPU utilization is _____

- a) 0.90
- b) 0.74
- c) 0.94
- d) 0.80

[View Answer](#)

Answer: c

Explanation: None.

3. A process P1 has a period of 50 and a CPU burst of $t_1 = 25$, P2 has a period of 80 and a CPU burst of 35., the priorities of P1 and P2 are?

- a) remain the same throughout
- b) keep varying from time to time
- c) may or may not be change
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. A process P1 has a period of 50 and a CPU burst of $t_1 = 25$, P2 has a period of 80 and a CPU burst of 35., can the two processes be scheduled using the EDF algorithm without missing their respective deadlines?

- a) Yes
- b) No
- c) Maybe
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. Using EDF algorithm practically, it is impossible to achieve 100 percent utilization due to _____

- a) the cost of context switching
- b) interrupt handling
- c) power consumption
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. T shares of time are allocated among all processes out of N shares in _____ scheduling algorithm.

- a) rate monotonic
- b) proportional share
- c) earliest deadline first
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. If there are a total of $T = 100$ shares to be divided among three processes, A, B and C. A is assigned 50 shares, B is assigned 15 shares and C is assigned 20 shares.

A will have _____ percent of the total processor time.

- a) 20
- b) 15

- c) 50
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

8. If there are a total of $T = 100$ shares to be divided among three processes, A, B and C. A is assigned 50 shares, B is assigned 15 shares and C is assigned 20 shares.

B will have _____ percent of the total processor time.

- a) 20
- b) 15
- c) 50
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

9. If there are a total of $T = 100$ shares to be divided among three processes, A, B and C. A is assigned 50 shares, B is assigned 15 shares and C is assigned 20 shares.

C will have _____ percent of the total processor time.

- a) 20
- b) 15
- c) 50
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. If there are a total of $T = 100$ shares to be divided among three processes, A, B and C. A is assigned 50 shares, B is assigned 15 shares and C is assigned 20 shares.

If a new process D requested 30 shares, the admission controller would

- a) allocate 30 shares to it
- b) deny entry to D in the system
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. To schedule the processes, they are considered _____

- a) infinitely long
- b) periodic
- c) heavy weight
- d) light weight

[View Answer](#)

Answer: b

Explanation: None.

2. If the period of a process is 'p', then what is the rate of the task?

- a) p^2
- b) 2^*p
- c) $1/p$
- d) p

[View Answer](#)

Answer: c

Explanation: None.

3. The scheduler admits a process using _____

- a) two phase locking protocol
- b) admission control algorithm
- c) busy wait polling
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

4. The _____ scheduling algorithm schedules periodic tasks using a static priority policy with preemption.

- a) earliest deadline first
- b) rate monotonic
- c) first cum first served
- d) priority

[View Answer](#)

Answer: b

Explanation: None.

5. Rate monotonic scheduling assumes that the _____

- a) processing time of a periodic process is same for each CPU burst
- b) processing time of a periodic process is different for each CPU burst
- c) periods of all processes is the same
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. In rate monotonic scheduling, a process with a shorter period is assigned _____

- a) a higher priority
- b) a lower priority
- c) higher & lower priority
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. There are two processes P1 and P2, whose periods are 50 and 100 respectively. P1 is assigned higher priority than P2. The processing times are $t_1 = 20$ for P1 and $t_2 = 35$ for P2. Is it possible to schedule these tasks so that each meets its deadline using Rate monotonic scheduling?

- a) yes
- b) no
- c) maybe
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. If a set of processes cannot be scheduled by rate monotonic scheduling algorithm, then _____

- a) they can be scheduled by EDF algorithm
- b) they cannot be scheduled by EDF algorithm
- c) they cannot be scheduled by any other algorithm
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. A process P1 has a period of 50 and a CPU burst of $t_1 = 25$, P2 has a period of 80 and a CPU burst of 35. The total CPU utilization is?

- a) 0.90
- b) 0.74
- c) 0.94
- d) 0.80

[View Answer](#)

Answer: c

Explanation: None.

10. A process P1 has a period of 50 and a CPU burst of $t_1 = 25$, P2 has a period of 80 and a CPU burst of 35. Can the processes be scheduled without missing the deadlines?

- a) Yes
- b) No
- c) Maybe
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. What is multimedia file?

- a) is same as any other regular file
- b) must be accessed at specific rate
- c) stored on remote server can not be delivered to its client
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. In which type of streaming multimedia file is delivered to the client, but not shared?

- a) real-time streaming
- b) progressive download
- c) compression
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. Which one of the following is the characteristic of a multimedia system?

- a) high storage
- b) high data rates
- c) both high storage and high data rates
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

4. The delay that occur during the playback of a stream is called _____

- a) stream delay
- b) playback delay
- c) jitter
- d) event delay

[View Answer](#)

Answer: c

Explanation: None.

5. Which algorithm can be optimized to meet the timing deadlines and rate requirements of continuous media?

- a) Earliest-Deadline-First scheduling
- b) SCAN-EDF scheduling
- c) Both Earliest-Deadline-First scheduling & SCAN-EDF scheduling
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. Real time streaming protocol is used _____

- a) to control streaming media servers
- b) for establishing and controlling media sessions between endpoints
- c) to provide real time control of playback of media files from the server
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. In teardown state of real time streaming protocol is _____

- a) the server resources for client
- b) server delivers the stream to client
- c) server suspends delivery of stream
- d) server breaks down the connection

[View Answer](#)

Answer: d

Explanation: None.

8. CineBlitz multimedia server supports _____

- a) real time clients
- b) non-real time clients
- c) both real time & non-real time clients
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. Multimedia system require hard real time scheduling _____

- a) to ensure critical tasks will be serviced within timing deadlines
- b) to deliver the media file to the client
- c) to minimize the delay
- d) for security

[View Answer](#)

Answer: a

Explanation: None.

10. Which one of the following resource is not necessarily required on a file server?

- a) secondary storage
- b) processor
- c) network
- d) monitor

[View Answer](#)

Answer: d

Explanation: None.

1. The major difference between a multimedia file and a regular file is _____

- a) the size
- b) the attributes
- c) the ownership
- d) the rate at which the file must be accessed

[View Answer](#)

Answer: d

Explanation: Multimedia files must be accessed at a specific rate whereas accessing regular files requires no special timings.

2. Video is represented as a series of images formally known as _____

- a) pics
- b) shots
- c) frames
- d) snaps

[View Answer](#)

Answer: c

Explanation: None.

3. The faster the frames are displayed, _____

- a) the rougher the video appears
- b) the smoother the video appears
- c) it gets blurry
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. The characteristic of the eye to retain the image for a short time after it has been presented is known as _____

- a) persistence of vision
- b) learning power
- c) memory mapped input
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. When will Local playback be used?

- a) the multimedia data are delivered from a local file system

- b) a computer next to you is playing something
- c) a multimedia file is being played on a system in the local network
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. Multimedia files stored on a remote server are delivered to a client across the network using a technique known as _____

- a) download
- b) streaming
- c) flowing
- d) leaking

[View Answer](#)

Answer: b

Explanation: None.

7. What are the two types of streaming techniques?

- a) progressive download & real time streaming
- b) regular download & real time streaming
- c) real time & virtual time streaming
- d) virtual time streaming

[View Answer](#)

Answer: a

Explanation: None.

8. A media file containing audio or video is downloaded and stored on the client's local file system in _____

- a) progressive download
- b) regular download
- c) real time streaming
- d) virtual time streaming

[View Answer](#)

Answer: a

Explanation: As the file is being downloaded, the client is able to play back the media file without having to wait for the file to be downloaded in its entirety.

9. Progressive download is most useful for _____

- a) short video clips

- b) long video clips
- c) extremely long and high quality videos
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. The media file is streamed to the client but is only played and not stored by the client in _____

- a) progressive download
- b) regular download
- c) real time streaming
- d) virtual time streaming

[View Answer](#)

Answer: c

Explanation: None.

11. Real time streaming is most useful for _____

- a) short video clips
- b) long video clips
- c) extremely short and low quality videos
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. The ability to move around within a media stream is known as _____

- a) buffering
- b) random access
- c) access
- d) sequential access

[View Answer](#)

Answer: b

Explanation: None.

2. What are the two types of real time streaming?

- a) live & on demand streaming
- b) dead & static streaming
- c) static & on demand streaming

d) on demand streaming

[View Answer](#)

Answer: a

Explanation: None.

3. Random access is not allowed in _____

- a) live streaming
- b) dead streaming
- c) static streaming
- d) on demand streaming

[View Answer](#)

Answer: a

Explanation: None.

4. The streaming that takes place as the event is occurring is _____

- a) live streaming
- b) dead streaming
- c) static streaming
- d) on demand streaming

[View Answer](#)

Answer: d

Explanation: None.

5. For a computer to deliver continuous media it must guarantee the specific rate and timing requirements, also known as _____

- a) deadline
- b) quality of service
- c) period
- d) burst time

[View Answer](#)

Answer: b

Explanation: None.

6. For QOS to be implemented properly _____

- a) file systems must be efficient to meet the rate requirements of continuous media
- b) network protocols must support bandwidth requirements while minimizing delay and jitter

- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

7. What will happen once a file is compressed?

- a) it has a better quality
- b) it takes up less space for storage
- c) it cannot be delivered to the client more quickly
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. Compression ratio is the ratio of _____

- a) the original file size to the size of the compressed file
- b) the number of pixels in a frame of the original size to those in a frame of the compressed file
- c) compressed file size to the original file size
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. Lossy and lossless are classifications of _____

- a) multimedia storage systems
- b) files
- c) compression algorithms
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. Lossy techniques provide _____ when compared to lossless techniques.

- a) lower compression ratios
- b) much higher compression ratios
- c) similar compression ratios

d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. What is the full form of MPEG?

- a) Motion Pictures Engineering Group
- b) Motion Picture Engineers Group
- c) Motion Picture Experts Group
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. What is MPEG compression?

- a) stores the compression values of each frame
- b) stores the differences between successive frames
- c) stores multiple frames' values together
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. What are the levels in QoS?

- a) Best effort service
- b) Soft QoS
- c) Hard QoS
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

4. The level that treats different types of traffics in different ways, giving certain traffic streams higher priority than other streams and with best efforts, but no guarantees are made _____

- a) Best effort service
- b) Soft QoS
- c) Worst effort service

d) Hard QoS

[View Answer](#)

Answer: b

Explanation: None.

5. The quality of service requirements are guaranteed in _____

- a) Best effort service
- b) Soft QoS
- c) Worst effort service
- d) Hard QoS

[View Answer](#)

Answer: d

Explanation: None.

6. What are the factors that define QoS?

- a) Throughput
- b) Jitter
- c) Delay
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. Delay and Jitter _____

- a) mean the same thing
- b) are two completely different things
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. What is the Delay?

- a) the time from when a request is first submitted to when the desired result is produced
- b) the delay that occurs during playback of the stream
- c) how the errors are handled during the transmission and processing of continuous media

d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. What is Admission control?

- a) the delay that occurs during playback of the stream
- b) the practice of admitting a request for service only if the server has sufficient resources to satisfy the request
- c) how the errors are handled during the transmission and processing of continuous media
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

10. An admission control scheme assigns a _____ to each type of resource.

- a) processor
- b) memory location
- c) resource manager
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

1. A scheduling algorithm can use either _____ priority or _____ priority.

- a) static, still
- b) static, dynamic
- c) live, dead
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. The priority of a process will _____ if the scheduler assigns it a static priority.

- a) change

- b) remain unchanged
- c) depends on the operating system
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. As disks have relatively low transfer rates and relatively high latency rates, disk schedulers must reduce latency times to _____

- a) ensure high bandwidth
- b) ensure low bandwidth
- c) make sure data is transferred
- d) reduce data transfer speeds

[View Answer](#)

Answer: a

Explanation: None.

4. Servicing requests strictly according to deadline using EDF may result in _____

- a) lower seek times
- b) lower bandwidth
- c) higher seek time
- d) higher bandwidth

[View Answer](#)

Answer: c

Explanation: None.

5. The hybrid algorithm that combines EDF with SCAN algorithm is known as _____

- a) EDS
- b) SDF
- c) SCAN-EDF
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. If several requests have different deadlines that are relatively close together, then using the SCAN - EDF algorithm _____

- a) the SCAN ordering will service the requests in that batch
- b) the EDF ordering will service the requests in that batch
- c) the FCFS ordering will service the requests in that batch
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. Multimedia systems require _____ scheduling to ensure critical tasks will be serviced within timing deadlines.

- a) soft real time
- b) hard real time
- c) normal
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. The EDF scheduler uses _____ to order requests according to their deadlines.

- a) stack
- b) disks
- c) queue
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. In SCAN - EDF, requests with the same deadlines are ordered according to

- a) SCAN policy
- b) EDF policy
- c) FCFS policy
- d) FIFO policy

[View Answer](#)

Answer: a

Explanation: None.

1. The three general methods for delivering content from a server to a client across a network are _____

- a) unicasting
- b) multicasting
- c) broadcasting
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

2. Unicasting delivers the content to _____

- a) a single client
- b) all clients, regardless whether they want the content or not
- c) a group of receivers who indicate they wish to receive the content
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. Broadcasting delivers the content to _____

- a) a single client
- b) all clients, regardless whether they want the content or not
- c) a group of receivers who indicate they wish to receive the content
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. Multicasting delivers the content to _____

- a) a single client
- b) all clients, regardless whether they want the content or not
- c) a group of receivers who indicate they wish to receive the content
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

5. RTSP stands for _____

- a) Real Time Streaming Policy

- b) Real Time Streaming Protocol
- c) Real Time Systems Protocol
- d) Read Time Streaming Policy

[View Answer](#)

Answer: b

Explanation: None.

6. HTTP is _____

- a) a stateful protocol
- b) a stateless protocol
- c) a protocol that maintains the status of its connection with the client
- d) a stateless protocol that does not maintain the status of its connection with the client

[View Answer](#)

Answer: d

Explanation: None.

7. RTSP includes which of the following states?

- a) SETUP
- b) PLAY
- c) PAUSE
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. In the SETUP state _____

- a) the server is setup
- b) the client is setup
- c) the server allocates resources for the client session
- d) the client sends requests to the server

[View Answer](#)

Answer: c

Explanation: None.

9. In the TEARDOWN state _____

- a) the server breaks down the connection and releases the resources allocated for the session
- b) the client breaks down the connection and releases the resources allocated

for the session

- c) the system crashes
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. RTP stands for _____

- a) real time protocol
- b) real time transmission control protocol
- c) real time transmission protocol
- d) real time transport protocol

[View Answer](#)

Answer: d

Explanation: None.

11. The problem with unicast delivery is that the _____

- a) memory allocation is difficult
- b) server must establish a separate unicast session for each client
- c) the routers must support unicasting
- d) the clients must be close to the server

[View Answer](#)

Answer: b

Explanation: None.

12. The difficulty with multicasting from a practical point of view is _____

- a) memory allocation is difficult
- b) server must establish a separate unicast session for each client
- c) the routers must support multicasting
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

13. To let a client have random access to a media stream with _____

- a) the protocol used must not be stateless
- b) the server must support download
- c) the stream should give access rights to the client

d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. Which of the following are forms of malicious attack?

- a) Theft of information
- b) Modification of data
- c) Wiping of information
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

2. What are the common security threats?

- a) File Shredding
- b) File sharing and permission
- c) File corrupting
- d) File integrity

[View Answer](#)

Answer: b

Explanation: Sharing and associated permissions are usual exploits which can compromise the system.

3. From the following, which is not a common file permission?

- a) Write
- b) Execute
- c) Stop
- d) Read

[View Answer](#)

Answer: c

Explanation: None.

4. Which of the following is a good practice?

- a) Give full permission for remote transferring
- b) Grant read only permission
- c) Grant limited permission to specified account
- d) Give both read and write permission but not execute

[View Answer](#)

Answer: c

Explanation: Limited access is a key method to circumvent unauthorized access and exploits.

5. What is not a good practice for user administration?

- a) Isolating a system after a compromise
- b) Perform random auditing procedures
- c) Granting privileges on a per host basis
- d) Using telnet and FTP for remote access

[View Answer](#)

Answer: d

Explanation: Telnet and FTP are not encrypted and can be compromised.

6. Which of the following is the least secure method of authentication?

- a) Key card
- b) fingerprint
- c) retina pattern
- d) Password

[View Answer](#)

Answer: d

Explanation: Passwords can be compromised more easily than to replicate a physical thing like key card, fingerprint or retina.

7. Which of the following is a strong password?

- a) 19thAugust88
- b) Delhi88
- c) P@ssw0rd
- d) laugustdelhi

[View Answer](#)

Answer: c

Explanation: It has a combination of Alphabet both capital and small along with number and special character. Thus always use complex password with a combination of all these.

8. Why is one time password safe?

- a) It is easy to generated
- b) It cannot be shared
- c) It is different for every access
- d) It is a complex encrypted password

[View Answer](#)

Answer: c

Explanation: One time password is safe since it is generated per access and thus cannot be brute forced or deduced.

9. What does Light Directory Access Protocol (LDAP) doesn't store?

- a) Users
- b) Address
- c) Passwords
- d) Security Keys

[View Answer](#)

Answer: b

Explanation: None.

10. What is characteristic of RADIUS system?

- a) It is essential for centralized encryption and authentication
- b) It works on Network layer to deny access to unauthorized people
- c) It provides centralized authentication mechanism via network devices
- d) It's a strong File access system

[View Answer](#)

Answer: c

Explanation: None.

11. Which happens first authorization or authentication?

- a) Authorization
- b) Authentication
- c) Authorization & Authentication are same
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

12. What are the characteristics of Authorization?

- a) RADIUS and RSA
- b) 3 way handshaking with syn and fin
- c) Multilayered protection for securing resources
- d) Deals with privileges and rights

[View Answer](#)

Answer: d

Explanation: None.

13. What forces the user to change password at first login?

- a) Default behavior of OS
- b) Part of AES encryption practice
- c) Devices being accessed forces the user
- d) Account administrator

[View Answer](#)

Answer: d

Explanation: Its administrator's job to ensure that password of the user remains private and is known only to user. But while making a new user account he assigns a random general password to give it to user. Thus even administrator cannot access a particular users account.

14. What is not a best practice for password policy?

- a) Deciding maximum age of password
- b) Restriction on password reuse and history
- c) Password encryption
- d) Having change password every 2 years

[View Answer](#)

Answer: d

Explanation: Old passwords are more vulnerable to being misplaced or compromised. Passwords should be changed periodically to enhance security.

1. What is the breach of integrity?

- a) This type of violation involves unauthorized reading of data
- b) This violation involves unauthorized modification of data
- c) This violation involves unauthorized destruction of data
- d) This violation involves unauthorized use of resources

[View Answer](#)

Answer: b

Explanation: None.

2. What is breach of confidentiality?

- a) This type of violation involves unauthorized reading of data
- b) This violation involves unauthorized modification of data
- c) This violation involves unauthorized destruction of data
- d) This violation involves unauthorized use of resources

[View Answer](#)

Answer: a

Explanation: None.

3. What is theft of service?

- a) This type of violation involves unauthorized reading of data
- b) This violation involves unauthorized modification of data
- c) This violation involves unauthorized destruction of data
- d) This violation involves unauthorized use of resources

[View Answer](#)

Answer: d

Explanation: None.

4. What is breach of availability?

- a) This type of violation involves unauthorized reading of data
- b) This violation involves unauthorized modification of data
- c) This violation involves unauthorized destruction of data
- d) This violation involves unauthorized use of resources

[View Answer](#)

Answer: c

Explanation: None.

5. What is Trojan horse?

- a) It is a useful way to encrypt password
- b) It is a user which steals valuable information
- c) It is a rogue program which tricks users
- d) It's a brute force attack algorithm

[View Answer](#)

Answer: c

Explanation: None.

6. What is trap door?

- a) IT is trap door in WarGames
- b) It is a hole in software left by designer
- c) It is a Trojan horse
- d) It is a virus which traps and locks user terminal

[View Answer](#)

Answer: b

Explanation: None.

7. Which mechanism is used by worm process?

- a) Trap door
- b) Fake process

c) Spawn Process

d) VAX process

[View Answer](#)

Answer: c

Explanation: None.

8. Which of the following is not a characteristic of a virus?

a) Virus destroy and modify user data

b) Virus is a standalone program

c) Virus is a code embedded in a legitimate program

d) Virus cannot be detected

[View Answer](#)

Answer: d

Explanation: Virus can be detected by having an antivirus program.

9. What is known as masquerading?

a) When one participant in communication pretends to be someone else

b) When attacker modifies data in communication

c) When attack is of fraudulent repeat of a valid data

d) When attack gains access to remote systems

[View Answer](#)

Answer: a

Explanation: None.

10. Who unleashed famous worm attack in 1988 which effected UNIX systems and caused losses in millions?

a) Robert Morris

b) Bob Milano

c) Mark zuckerberg

d) Bill Gates

[View Answer](#)

Answer: a

Explanation: None.

11. What is port scanning?

a) It is a software used to scan system for attack

b) It is a software application designed to probe a server or host for open ports

c) It is software used to scan system for introducing attacks by brute force

d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

12. Which is not a port scan type?

- a) TCP scanning
- b) SYN scanning
- c) UDP scanning
- d) SYSTEM Scanning

[View Answer](#)

Answer: d

Explanation: None.

13. Which is not a valid port scan type?

- a) ACK scanning
- b) Window scanning
- c) IGMP scan
- d) FIN scanning

[View Answer](#)

Answer: c

Explanation: None.

14. What are zombie systems?

- a) Are specific system which are designed to attack by manufacturer
- b) They are network of known hacking group
- c) These systems are previously compromised independent systems
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

15. What is known as a DOS attack?

- a) It is attacked to block traffic of network
- b) It is attacked to harm contents stored in HDD by worm spawn processes
- c) It is an attempt to make a machine or network resource unavailable
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

16. With regard to DOS attack what is not true from below options?

- a) We can stop DOS attack completely
- b) By upgrading OS vulnerability we can stop DOS attack to some extent
- c) DOS attack has to be stopped at network level
- d) Such attack can last for hours

[View Answer](#)

Answer: a

Explanation: None.

1. What is not an important part of security protection?

- a) Large amount of RAM to support antivirus
- b) Strong passwords
- c) Audit log periodically
- d) Scan for unauthorized programs in system directories

[View Answer](#)

Answer: a

Explanation: RAM has no effect on security of a system. System's protection remains unchanged in increasing or decreasing amount of RAM.

2. What is used to protect network from outside internet access?

- a) A trusted antivirus
- b) 24 hours scanning for virus
- c) Firewall to separate trusted and untrusted network
- d) Deny users access to websites which can potentially cause security leak

[View Answer](#)

Answer: c

Explanation: Firewall create a protective barrier to secure internal network. An antivirus can only detect harmful viruses but cannot stop illegal access by remote attacker.

3. What is the best practice in the firewall domain environment?

- a) Create two domain trusted and untrusted domain
- b) Create strong policy in firewall to support different types of users
- c) Create a Demilitarized zone
- d) Create two DMZ zones with one untrusted domain

[View Answer](#)

Answer: c

Explanation: All live servers or workstations are kept in a separate zone than inside and outside to enhance protection.

4. Which direction access cannot happen using DMZ zone by default?

- a) Company computer to DMZ
- b) Internet to DMZ
- c) Internet to company computer
- d) Company computer to internet

[View Answer](#)

Answer: c

Explanation: Connection from internet is never allowed to directly access internal PCs but is routed through DMZ zone to prevent attacks.

5. What are the two features of a tripwire file system?

- a) It is a tool to monitor file systems
- b) It is used to automatically take corrective action
- c) It is used to secure UNIX system
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. How do viruses avoid basic pattern match of antivirus?

- a) They are encrypted
- b) They act with special permissions
- c) They modify themselves
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

7. How does an antivirus of today identify viruses?

- a) Previously known patterns
- b) It can detect unknown patterns
- c) It can take high priority to increase scanning speed
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. What is known as a sandbox?

- a) It is a program which can be molded to do the desired task
- b) It is a program that is controlled or emulated section of OS
- c) It is a special mode of antivirus
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

9. What are two safe computing practices?

- a) Not to open software from unknown vendors
- b) Open and execute programs in admin level/root
- c) Open and execute programs in presence of antivirus
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: Disgruntled employees have in past infected the master copies of software programs to do economic harm to the company.

1. What are the different ways to intrude?

- a) Buffer overflows
- b) Unexpected combinations and unhandled input
- c) Race conditions
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

2. What are the major components of the intrusion detection system?

- a) Analysis Engine
- b) Event provider
- c) Alert Database
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. What are the different ways to classify an IDS?

- a) anomaly detection
- b) signature based misuse
- c) stack based
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

4. What are the different ways to classify an IDS?

- a) Zone based
- b) Host & Network based
- c) Network & Zone based
- d) Level based

[View Answer](#)

Answer: b

Explanation: None.

5. What are the characteristics of anomaly based IDS?

- a) It models the normal usage of network as a noise characterization
- b) It doesn't detect novel attacks
- c) Anything distinct from the noise is not assumed to be intrusion activity
- d) It detects based on signature

[View Answer](#)

Answer: a

Explanation: None.

6. What is the major drawback of anomaly detection IDS?

- a) These are very slow at detection
- b) It generates many false alarms
- c) It doesn't detect novel attacks
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. What are the characteristics of signature based IDS?

- a) Most are based on simple pattern matching algorithms
- b) It is programmed to interpret a certain series of packets

- c) It models the normal usage of network as a noise characterization
- d) Anything distinct from the noise is assumed to be intrusion activity

[View Answer](#)

Answer: a

Explanation: None.

8. What are the drawbacks of signature based IDS?

- a) They are unable to detect novel attacks
- b) They suffer from false alarms
- c) They have to be programmed again for every new pattern to be detected
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. What are the characteristics of Host based IDS?

- a) The host operating system logs in the audit information
- b) Logs includes logins,file opens and program executions
- c) Logs are analysed to detect tails of intrusion
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

10. What are the drawbacks of the host based IDS?

- a) Unselective logging of messages may increase the audit burdens
- b) Selective logging runs the risk of missed attacks
- c) They are very fast to detect
- d) They have to be programmed for new patterns

[View Answer](#)

Answer: a

Explanation: None.

11. What are the strengths of the host based IDS?

- a) Attack verification
- b) System specific activity
- c) No additional hardware required
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

12. What are characteristics of stack based IDS?

- a) They are integrated closely with the TCP/IP stack and watch packets
- b) The host operating system logs in the audit information
- c) It is programmed to interpret a certain series of packets
- d) It models the normal usage of network as a noise characterization

[View Answer](#)

Answer: a

Explanation: None.

13. What are characteristics of Network based IDS?

- a) They look for attack signatures in network traffic
- b) Filter decides which traffic will not be discarded or passed
- c) It is programmed to interpret a certain series of packet
- d) It models the normal usage of network as a noise characterization

[View Answer](#)

Answer: a

Explanation: None.

14. What are strengths of Network based IDS?

- a) Cost of ownership reduced
- b) Malicious intent detection
- c) Real time detection and response
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

1. What is the preferred way of encryption?

- a) pre shared secret key
- b) using key distribution center (KDC)
- c) public key-encryption
- d) symmetric key

[View Answer](#)

Answer: c

Explanation: Pre Shared key can be compromised and either party can be

suspected. Likewise KDC or symmetric key can have breach which are undesirable. Public and private key encryption is a known industry standard.

2. What is not a role of encryption?

- a) It is used to protect data from unauthorized access during transmission
- b) It is used to ensure user authentication
- c) It is used to ensure data integrity
- d) It is used to ensure data corruption doesn't happen

[View Answer](#)

Answer: d

Explanation: Encryption doesn't have error correction or detection facility thus cannot be used to safeguard from data corruption.

3. What is cipher-block chaining?

- a) Data is logically 'ANDed' with previous block
- b) Data is logically 'ORed' with previous block
- c) Data is logically 'XORed' with previous block
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

4. What is not an encryption standard?

- a) AES
- b) TES
- c) Triple DES
- d) DES

[View Answer](#)

Answer: b

Explanation: None.

5. Which of the following is not a stream cipher?

- a) Two fish
- b) RC5
- c) RC4
- d) TBONE

[View Answer](#)

Answer: d

Explanation: None.

6. What is a Hash Function?

- a) It creates a small flexible block of data
- b) It creates a small,fixed block of data
- c) It creates a encrypted block of data
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. MD5 produces _____ bits hash data.

- a) 128
- b) 150
- c) 160
- d) 112

[View Answer](#)

Answer: a

Explanation: None.

8. SHA-1 produces _____ bit of hash.

- a) 128
- b) 160
- c) 150
- d) 112

[View Answer](#)

Answer: b

Explanation: None.

9. Which two of the following are authentication algorithms?

- a) MAC
- b) AES
- c) DAS
- d) Digital-signature

[View Answer](#)

Answer: a

Explanation: None.

10. What is the role of Key Distribution Center?

- a) It is used to distribute keys to everyone in world
- b) It intended to reduce the risks inherent in exchanging keys

- c) All of the mentioned
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. Which one of the following is not a secondary storage?

- a) Magnetic disks
- b) Magnetic tapes
- c) RAM
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. Which private network uses storage protocol rather than networking protocol?

- a) storage area network
- b) local area network
- c) wide area network
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. The time for the disk arm to move the heads to the cylinder containing the desired sector is called _____

- a) disk time
- b) seek time
- c) arm time
- d) sector time

[View Answer](#)

Answer: b

Explanation: None.

4. Which algorithm of disk scheduling selects the request with the least seek time from the current head positions?

- a) SSTF scheduling
- b) FCFS scheduling
- c) SCAN scheduling

d) LOOK scheduling

[View Answer](#)

Answer: a

Explanation: None.

5. The operating system is responsible for?

- a) disk initialization
- b) booting from disk
- c) bad-block recovery
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

6. A swap space can reside in _____

- a) Separate disk partition
- b) RAM
- c) Cache
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. RAID level 1 refers to _____

- a) disk arrays with striping
- b) disk mirroring
- c) both disk arrays with striping and disk mirroring
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: A variety of disk-organization techniques is called "redundant arrays of independent disks (RAID)".

8. When we write something on the disk, which one of the following can not happen?

- a) successful completion
- b) partial failure
- c) total failure

d) none of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. What will happen during the recovery from a failure?

- a) each pair of physical block is examined
- b) specified pair of physical block is examined
- c) first pair of physical block is examined
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. The replacement of a bad block generally is not totally automatic because

-
- a) data in bad block can not be replaced
 - b) data in bad block is usually lost
 - c) bad block does not contain any data
 - d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. Linux uses a time-sharing algorithm _____

- a) to pair preemptive scheduling between multiple processes
- b) for tasks where absolute priorities are more important than fairness
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. The first linux kernel which supports the SMP hardware?

- a) linux 0.1
- b) linux 1.0
- c) linux 1.2
- d) linux 2.0

[View Answer](#)

Answer: d

Explanation: None.

3. Which one of the following linux file system does not support journaling feature?

- a) ext2
- b) ext3
- c) ext4
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. Which binary format is supported by linux?

- a) a.out
- b) elf
- c) both a.out and ELF
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

5. Which one of the following bootloader is not used by linux?

- a) GRUB
- b) LILO
- c) NTLDR
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. The first process launched by the linux kernel is _____

- a) init process
- b) zombie process
- c) batch process
- d) boot process

[View Answer](#)

Answer: a

Explanation: None.

7. Which desktop environment is not used in any linux distribution?

- a) gnome
- b) kde
- c) unity
- d) none of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. Standard set of functions through which interacts with kernel is defined by

-
- a) system libraries
 - b) kernel code
 - c) compilers
 - d) utility programs

[View Answer](#)

Answer: a

Explanation: None.

9. What is Linux?

- a) single user, single tasking
- b) single user, multitasking
- c) multi user, single tasking
- d) multi user, multitasking

[View Answer](#)

Answer: d

Explanation: None.

10. Which one of the following is not a linux distribution?

- a) debian
- b) gentoo
- c) open SUSE
- d) multics

[View Answer](#)

Answer: d

Explanation: None.

1. Which one of the following is not shared by threads?

- a) program counter
- b) stack
- c) both program counter and stack
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. A process can be _____

- a) single threaded
- b) multithreaded
- c) both single threaded and multithreaded
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. If one thread opens a file with read privileges then _____

- a) other threads in the another process can also read from that file
- b) other threads in the same process can also read from that file
- c) any other thread can not read from that file
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. The time required to create a new thread in an existing process is _____

- a) greater than the time required to create a new process
- b) less than the time required to create a new process
- c) equal to the time required to create a new process
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. When the event for which a thread is blocked occurs?

- a) thread moves to the ready queue

- b) thread remains blocked
- c) thread completes
- d) a new thread is provided

[View Answer](#)

Answer: a

Explanation: None.

6. The jacketing technique is used to _____
- a) convert a blocking system call into non blocking system call
 - b) create a new thread
 - c) communicate between threads
 - d) terminate a thread

[View Answer](#)

Answer: a

Explanation: None.

7. Termination of the process terminates _____
- a) first thread of the process
 - b) first two threads of the process
 - c) all threads within the process
 - d) no thread within the process

[View Answer](#)

Answer: c

Explanation: None.

8. Which one of the following is not a valid state of a thread?
- a) running
 - b) parsing
 - c) ready
 - d) blocked

[View Answer](#)

Answer: b

Explanation: None.

9. The register context and stacks of a thread are deallocated when the thread?
- a) terminates
 - b) blocks
 - c) unblocks

d) spawns

[View Answer](#)

Answer: a

Explanation: None.

10. Thread synchronization is required because _____

- a) all threads of a process share the same address space
- b) all threads of a process share the same global variables
- c) all threads of a process can share the same files
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

1. A thread is also called _____

- a) Light Weight Process(LWP)
- b) Heavy Weight Process(HWP)
- c) Process
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. A thread shares its resources(like data section, code section, open files, signals) with _____

- a) other process similar to the one that the thread belongs to
- b) other threads that belong to similar processes
- c) other threads that belong to the same process
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. A heavy weight process _____

- a) has multiple threads of execution
- b) has a single thread of execution
- c) can have multiple or a single thread for execution
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. A process having multiple threads of control implies _____

- a) it can do more than one task at a time
- b) it can do only one task at a time, but much faster
- c) it has to use only one thread per process
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. Multithreading an interactive program will increase responsiveness to the user by _____

- a) continuing to run even if a part of it is blocked
- b) waiting for one part to finish before the other begins
- c) asking the user to decide the order of multithreading
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. Resource sharing helps _____

- a) share the memory and resources of the process to which the threads belong
- b) an application have several different threads of activity all within the same address space
- c) reduce the address space that a process could potentially use
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. Multithreading on a multi - CPU machine _____

- a) decreases concurrency
- b) increases concurrency
- c) doesn't affect the concurrency
- d) can increase or decrease the concurrency

[View Answer](#)

Answer: b

Explanation: None.

8. The kernel is _____ of user threads.

- a) a part of
- b) the creator of
- c) unaware of
- d) aware of

[View Answer](#)

Answer: c

Explanation: None.

9. If the kernel is single threaded, then any user level thread performing a blocking system call will _____

- a) cause the entire process to run along with the other threads
- b) cause the thread to block with the other threads running
- c) cause the entire process to block even if the other threads are available to run
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. Because the kernel thread management is done by the Operating System itself _____

- a) kernel threads are faster to create than user threads
- b) kernel threads are slower to create than user threads
- c) kernel threads are easier to manage as well as create than user threads
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

11. If a kernel thread performs a blocking system call, _____

- a) the kernel can schedule another thread in the application for execution
- b) the kernel cannot schedule another thread in the same application for execution
- c) the kernel must schedule another thread of a different application for execution
- d) the kernel must schedule another thread of the same application on a

different processor

[View Answer](#)

Answer: a

Explanation: None.

12. Which of the following is FALSE?

- a) Context switch time is longer for kernel level threads than for user level threads
- b) User level threads do not need any hardware support
- c) Related kernel level threads can be scheduled on different processors in a multiprocessor system
- d) Blocking one kernel level thread blocks all other related threads

[View Answer](#)

Answer: d

Explanation: None.

1. The model in which one kernel thread is mapped to many user-level threads is called _____

- a) Many to One model
- b) One to Many model
- c) Many to Many model
- d) One to One model

[View Answer](#)

Answer: a

Explanation: None.

2. The model in which one user-level thread is mapped to many kernel level threads is called _____

- a) Many to One model
- b) One to Many model
- c) Many to Many model
- d) One to One model

[View Answer](#)

Answer: b

Explanation: None.

3. In the Many to One model, if a thread makes a blocking system call _____

- a) the entire process will be blocked

- b) a part of the process will stay blocked, with the rest running
- c) the entire process will run
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. In the Many to One model, multiple threads are unable to run in parallel on multiprocessors because of _____

- a) only one thread can access the kernel at a time
- b) many user threads have access to just one kernel thread
- c) there is only one kernel thread
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. The One to One model allows _____

- a) increased concurrency
- b) decreased concurrency
- c) increased or decreased concurrency
- d) concurrency equivalent to other models

[View Answer](#)

Answer: a

Explanation: None.

6. In the One to One model when a thread makes a blocking system call _____

- a) other threads are strictly prohibited from running
- b) other threads are allowed to run
- c) other threads only from other processes are allowed to run
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. Which of the following is the drawback of the One to One Model?

- a) increased concurrency provided by this model
- b) decreased concurrency provided by this model

- c) creating so many threads at once can crash the system
- d) creating a user thread requires creating the corresponding kernel thread

[View Answer](#)

Answer: d

Explanation: None.

8. When is the Many to One model at an advantage?

- a) When the program does not need multithreading
- b) When the program has to be multi-threaded
- c) When there is a single processor
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. In the Many to Many model true concurrency cannot be gained because

-
- a) the kernel can schedule only one thread at a time
 - b) there are too many threads to handle
 - c) it is hard to map threads with each other
 - d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. In the Many to Many models when a thread performs a blocking system call

-
- a) other threads are strictly prohibited from running
 - b) other threads are allowed to run
 - c) other threads only from other processes are allowed to run
 - d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. Which of the following system calls does not return control to the calling point, on termination?

- a) fork
- b) exec

- c) ioctl
- d) longjmp

[View Answer](#)

Answer: b

Explanation: None.

2. The following program results in the creation of?

```
main()
{
    if(fork()>0)
        sleep(100);
}
```

- a) an orphan process
- b) a zombie process
- c) a process that executes forever
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. Which of the following system calls transforms executable binary file into a process?

- a) fork
- b) exec
- c) ioctl
- d) longjmp

[View Answer](#)

Answer: b

Explanation: None.

4. How many times the following C program prints yes?

```
main()
{
    fork();fork();printf("yes");
```

}

- a) only once
- b) twice
- c) four times
- d) eight times

[View Answer](#)

Answer: c

Explanation: None.

5. Which of the following calls never returns an error?

- a) getpid
- b) fork
- c) ioctl
- d) open

[View Answer](#)

Answer: a

Explanation: None.

6. A fork system call will fail if _____

- a) the previously executed statement is also a fork call
- b) the limit on the maximum number of processes in the system would be executed
- c) the limit on the minimum number of processes that can be under execution by a single user would be executed
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. If a thread invokes the exec system call _____

- a) only the exec executes as a separate process
- b) the program specified in the parameter to exec will replace the entire process
- c) the exec is ignored as it is invoked by a thread
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. If exec is called immediately after forking _____

- a) the program specified in the parameter to exec will replace the entire process
- b) all the threads will be duplicated
- c) all the threads may be duplicated
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. If a process does not call exec after forking _____

- a) the program specified in the parameter to exec will replace the entire process
- b) all the threads should be duplicated
- c) all the threads should not be duplicated
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: The new process is purely based on fork, due to no exec command, duplication will be done.

1. What is Thread cancellation?

- a) the task of destroying the thread once its work is done
- b) the task of removing a thread once its work is done
- c) the task of terminating a thread before it has completed
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. When a web page is loading, and the user presses a button on the browser to stop loading the page?

- a) the thread loading the page continues with the loading
- b) the thread loading the page does not stop but continues with another task
- c) the thread loading the page is paused
- d) the thread loading the page is cancelled

[View Answer](#)

Answer: d

Explanation: None.

3. When one thread immediately terminates the target thread, it is called

- a) Asynchronous cancellation
- b) Systematic cancellation
- c) Sudden Termination
- d) Deferred cancellation

[View Answer](#)

Answer: a

Explanation: None.

4. When the target thread periodically checks if it should terminate and terminates itself in an orderly manner, it is called?

- a) Asynchronous cancellation
- b) Systematic cancellation
- c) Sudden Termination
- d) Deferred cancellation

[View Answer](#)

Answer: d

Explanation: None.

5. Cancelling a thread asynchronously _____

- a) frees all the resources properly
- b) may not free each resource
- c) spoils the process execution
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

6. Cancellation point is the point where _____

- a) the thread can be cancelled - safely or otherwise doesn't matter
- b) the thread can be cancelled safely
- c) the whole process can be cancelled safely
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. If multiple threads are concurrently searching through a database and one thread returns the result then the remaining threads must be _____

- a) continued
- b) cancelled
- c) protected
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. Signals that occur at the same time, are presented to the process

- a) one at a time, in a particular order
- b) one at a time, in no particular order
- c) all at a time
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. Which of the following is not TRUE?

- a) Processes may send each other signals
- b) Kernel may send signals internally
- c) A field is updated in the signal table when the signal is sent
- d) Each signal is maintained by a single bit

[View Answer](#)

Answer: c

Explanation: A field is updated in the **process table** when the signal is sent.

3. Signals of a given type _____

- a) are queued
- b) are all sent as one
- c) cannot be queued
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: The signal handler will be invoked only once.

4. The three ways in which a process responds to a signal are _____

- a) ignoring the signal
- b) handling the signal
- c) performing some default action
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. Signals are identified by _____

- a) signal identifiers
- b) signal handlers
- c) signal actions
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. When a process blocks the receipt of certain signals?

- a) The signals are delivered
- b) The signals are not delivered
- c) The signals are received until they are unblocked
- d) The signals are received by the process once they are delivered

[View Answer](#)

Answer: a

Explanation: None.

7. The _____ maintains pending and blocked bit vectors in the context of each process.

- a) CPU
- b) Memory
- c) Process
- d) Kernel

[View Answer](#)

Answer: d

Explanation: None.

8. In UNIX, the set of masked signals can be set or cleared using the _____ function.

- a) sigmask
- b) sigmaskproc
- c) sigprocmask
- d) sigproc

[View Answer](#)

Answer: c

Explanation: None.

9. The usefulness of signals as a general inter process communication mechanism is limited because _____

- a) they do not work between processes
- b) they are user generated
- c) they cannot carry information directly
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. The usual effect of abnormal termination of a program is _____

- a) core dump file generation
- b) system crash
- c) program switch
- d) signal destruction

[View Answer](#)

Answer: a

Explanation: None.

11. In UNIX, the abort() function sends the _____ signal to the calling process, causing abnormal termination.

- a) SIGTERM
- b) SIGSTOP
- c) SIGABORT
- d) SIGABRT

[View Answer](#)

Answer: d

Explanation: None.

12. In most cases, if a process is sent a signal while it is executing a system call _____

- a) the system call will continue execution and the signal will be ignored completely
- b) the system call is interrupted by the signal, and the signal handler comes in
- c) the signal has no effect until the system call completes
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

13. A process can never be sure that a signal it has sent _____

- a) has which identifier
- b) has not been lost
- c) has been sent
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

14. In UNIX, the _____ system call is used to send a signal.

- a) sig
- b) send
- c) kill
- d) sigsend

[View Answer](#)

Answer: c

Explanation: None.

1. Thread pools are useful when _____

- a) when we need to limit the number of threads running in the application at the same time
- b) when we need to limit the number of threads running in the application as a whole
- c) when we need to arrange the ordering of threads
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. Instead of starting a new thread for every task to execute concurrently, the task can be passed to a _____

- a) process
- b) thread pool
- c) thread queue
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. Each connection arriving at multi threaded servers via network is generally _____

- a) is directly put into the blocking queue
- b) is wrapped as a task and passed on to a thread pool
- c) is kept in a normal queue and then sent to the blocking queue from where it is dequeued
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. What is the idea behind thread pools?

- a) a number of threads are created at process startup and placed in a pool where they sit and wait for work
- b) when a process begins, a pool of threads is chosen from the many existing and each thread is allotted equal amount of work
- c) all threads in a pool distribute the task equally among themselves
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. If the thread pool contains no available thread _____

- a) the server runs a new process
- b) the server goes to another thread pool
- c) the server demands for a new pool creation
- d) the server waits until one becomes free

[View Answer](#)

Answer: d

Explanation: None.

6. Thread pools help in _____

- a) servicing multiple requests using one thread
- b) servicing a single request using multiple threads from the pool
- c) faster servicing of requests with an existing thread rather than waiting to create a new thread
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

7. Thread pools limit the number of threads that exist at any one point, hence _____

- a) not letting the system resources like CPU time and memory exhaust
- b) helping a limited number of processes at a time
- c) not serving all requests and ignoring many
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. The number of the threads in the pool can be decided on factors such as _____

- a) number of CPUs in the system
- b) amount of physical memory
- c) expected number of concurrent client requests
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

1. Because of virtual memory, the memory can be shared among _____

- a) processes
- b) threads
- c) instructions
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. _____ is the concept in which a process is copied into the main memory from the secondary memory according to the requirement.

- a) Paging
- b) Demand paging
- c) Segmentation
- d) Swapping

[View Answer](#)

Answer: b

Explanation: None.

3. The pager concerns with the _____

- a) individual page of a process
- b) entire process
- c) entire thread
- d) first page of a process

[View Answer](#)

Answer: a

Explanation: None.

4. Swap space exists in _____

- a) primary memory
- b) secondary memory
- c) cpu
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. When a program tries to access a page that is mapped in address space but not loaded in physical memory, then _____

- a) segmentation fault occurs
- b) fatal error occurs
- c) page fault occurs
- d) no error occurs

[View Answer](#)

Answer: c

Explanation: None.

6. Effective access time is directly proportional to _____

- a) page-fault rate
- b) hit ratio
- c) memory access time
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. In FIFO page replacement algorithm, when a page must be replaced

- a) oldest page is chosen
- b) newest page is chosen
- c) random page is chosen
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. Which algorithm chooses the page that has not been used for the longest period of time whenever the page required to be replaced?

- a) first in first out algorithm
- b) additional reference bit algorithm
- c) least recently used algorithm
- d) counting based page replacement algorithm

[View Answer](#)

Answer: c

Explanation: None.

9. A process is thrashing if _____

- a) it is spending more time paging than executing
- b) it is spending less time paging than executing
- c) page fault occurs
- d) swapping can not take place

[View Answer](#)

Answer: a

Explanation: None.

10. Working set model for page replacement is based on the assumption of _____

- a) modularity
- b) locality
- c) globalization
- d) random access

[View Answer](#)

Answer: b

Explanation: None.

1. Virtual memory allows _____

- a) execution of a process that may not be completely in memory
- b) a program to be smaller than the physical memory
- c) a program to be larger than the secondary storage
- d) execution of a process without being in physical memory

[View Answer](#)

Answer: a

Explanation: None.

2. The instruction being executed, must be in _____

- a) physical memory
- b) logical memory
- c) physical & logical memory
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. Error handler codes, to handle **unusual** errors are _____

- a) almost never executed
- b) executed very often
- c) executed periodically
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. The ability to execute a program that is only partially in memory has benefits like _____

- a) The amount of physical memory cannot put a constraint on the program
- b) Programs for an extremely large virtual space can be created
- c) Throughput increases
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. In virtual memory, the programmer _____ of overlays.

- a) has to take care
- b) does not have to take care
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

6. Virtual memory is normally implemented by _____

- a) demand paging
- b) buses
- c) virtualization
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. Segment replacement algorithms are more complex than page replacement algorithms because _____

- a) Segments are better than pages
- b) Pages are better than segments
- c) Segments have variable sizes
- d) Segments have fixed sizes

[View Answer](#)

Answer: c

Explanation: None.

8. A swapper manipulates _____ whereas the pager is concerned with individual _____ of a process.

- a) the entire process, parts
- b) all the pages of a process, segments
- c) the entire process, pages
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. Using a pager _____

- a) increases the swap time
- b) decreases the swap time
- c) decreases the swap time & amount of physical memory needed
- d) increases the amount of physical memory needed

[View Answer](#)

Answer: c

Explanation: None.

10. The valid - invalid bit, in this case, when valid indicates?

- a) the page is not legal
- b) the page is illegal
- c) the page is in memory
- d) the page is not in memory

[View Answer](#)

Answer: c

Explanation: None.

11. A page fault occurs when?

- a) a page gives inconsistent data
- b) a page cannot be accessed due to its absence from memory
- c) a page is invisible
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

12. When a page fault occurs, the state of the interrupted process is

- a) disrupted
- b) invalid
- c) saved
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

13. When a process begins execution with no pages in memory?

- a) process execution becomes impossible
- b) a page fault occurs for every page brought into memory
- c) process causes system crash
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

14. If the memory access time is denoted by 'ma' and 'p' is the probability of a page fault ($0 \leq p \leq 1$). Then the effective access time for a demand paged memory is _____

- a) $p \times ma + (1-p) \times \text{page fault time}$
- b) $ma + \text{page fault time}$
- c) $(1-p) \times ma + p \times \text{page fault time}$
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

15. When the page fault rate is low _____

- a) the turnaround time increases
- b) the effective access time increases
- c) the effective access time decreases
- d) turnaround time & effective access time increases

[View Answer](#)

Answer: c

Explanation: None.

16. Locality of reference implies that the page reference being made by a process _____

- a) will always be to the page used in the previous page reference
- b) is likely to be one of the pages used in the last few page references
- c) will always be one of the pages existing in memory
- d) will always lead to page faults

[View Answer](#)

Answer: b

Explanation: None.

1. Which of the following page replacement algorithms suffers from Belady's Anomaly?

- a) Optimal replacement
- b) LRU
- c) FIFO
- d) Both optimal replacement and FIFO

[View Answer](#)

Answer: c

Explanation: None.

2. A process refers to 5 pages, A, B, C, D, E in the order : A, B, C, D, A, B, E, A, B, C, D, E. If the page replacement algorithm is FIFO, the number of page transfers with an empty internal store of 3 frames is?

- a) 8
- b) 10
- c) 9
- d) 7

[View Answer](#)

Answer: c

Explanation: None.

3. In question 2, if the number of page frames is increased to 4, then the number of page transfers _____

- a) decreases
- b) increases
- c) remains the same
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. A memory page containing a heavily used variable that was initialized very early and is in constant use is removed, then the page replacement algorithm used is _____

- a) LRU
- b) LFU
- c) FIFO
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

5. A virtual memory system uses First In First Out (FIFO) page replacement policy and allocates a fixed number of frames to a process. Consider the following statements.

P : Increasing the number of page frames allocated to a process sometimes increases the page fault rate

Q : Some programs do not exhibit locality of reference

Which of the following is TRUE?

- a) Both P and Q are true, and Q is the reason for P
- b) Both P and Q are true, but Q is not the reason for P
- c) P is false but Q is true
- d) Both P and Q are false

[View Answer](#)

Answer: c

Explanation: None.

6. Users _____ that their processes are running on a paged system.

- a) are aware
- b) are unaware
- c) may unaware
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. If no frames are free, _____ page transfer(s) is/are required.

- a) one
- b) two
- c) three

d) four

[View Answer](#)

Answer: b

Explanation: None.

8. When a page is selected for replacement, and its modify bit is set _____

- a) the page is clean
- b) the page has been modified since it was read in from the disk
- c) the page is dirty
- d) the page has been modified since it was read in from the disk & page is dirty

[View Answer](#)

Answer: d

Explanation: None.

9. The aim of creating page replacement algorithms is to _____

- a) replace pages faster
- b) increase the page fault rate
- c) decrease the page fault rate
- d) to allocate multiple pages to processes

[View Answer](#)

Answer: c

Explanation: None.

10. A FIFO replacement algorithm associates with each page the _____

- a) time it was brought into memory
- b) size of the page in memory
- c) page after and before it
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. What is the Optimal page - replacement algorithm?

- a) Replace the page that has not been used for a long time
- b) Replace the page that has been used for a long time
- c) Replace the page that will not be used for a long time
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

12. Optimal page - replacement algorithm is difficult to implement, because _____

- a) it requires a lot of information
- b) it requires future knowledge of the reference string
- c) it is too complex
- d) it is extremely expensive

[View Answer](#)

Answer: b

Explanation: None.

13. LRU page - replacement algorithm associates with each page the _____

- a) time it was brought into memory
- b) the time of that page's last use
- c) page after and before it
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

14. For 3 page frames, the following is the reference string:

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

How many page faults does the LRU page replacement algorithm produce?

- a) 10
- b) 15
- c) 11
- d) 12

[View Answer](#)

Answer: d

Explanation: None.

15. What are the two methods of the LRU page replacement policy that can be implemented in hardware?

- a) Counters
- b) RAM & Registers
- c) Stack & Counters
- d) Registers

[View Answer](#)

Answer: c

Explanation: None.

1. When using counters to implement LRU, we replace the page with the _____

- a) smallest time value
- b) largest time value
- c) greatest size
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: Whenever a reference to a page is made, the contents of the clock register are copied into the time-of-use field in the page-table entry for that page. In this way, we always have the time of the last reference to each page.

2. In the stack implementation of the LRU algorithm, a stack can be maintained in a manner _____

- a) whenever a page is used, it is removed from the stack and put on bottom
- b) the bottom of the stack is the LRU page
- c) the top of the stack contains the LRU page and all new pages are added to the top
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. There is a set of page replacement algorithms that can never exhibit Belady's Anomaly, called _____

- a) queue algorithms
- b) stack algorithms
- c) string algorithms
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. Applying the LRU page replacement to the following reference string.

1 2 4 5 2 1 2 4

The main memory can accommodate 3 pages and it already has pages 1 and 2.

Page 1 came in before page 2.

How many page faults will occur?

- a) 2
- b) 3
- c) 4
- d) 5

[View Answer](#)

Answer: c

Explanation: None.

5. Increasing the RAM of a computer typically improves performance because _____

- a) Virtual memory increases
- b) Larger RAMs are faster
- c) Fewer page faults occur
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. The essential content(s) in each entry of a page table is/are _____

- a) Virtual page number
- b) Page frame number
- c) Both virtual page number and page frame number
- d) Access right information

[View Answer](#)

Answer: b

Explanation: None.

7. The minimum number of page frames that must be allocated to a running process in a virtual memory environment is determined by _____

- a) the instruction set architecture
- b) page size
- c) physical memory size
- d) number of processes in memory

[View Answer](#)

Answer: a

Explanation: None.

8. What is the reason for using the LFU page replacement algorithm?

- a) an actively used page should have a large reference count
- b) a less used page has more chances to be used again
- c) it is extremely efficient and optimal
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. What is the reason for using the MFU page replacement algorithm?

- a) an actively used page should have a large reference count
- b) a less used page has more chances to be used again
- c) it is extremely efficient and optimal
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

10. The implementation of the LFU and the MFU algorithm is very uncommon because _____

- a) they are too complicated
- b) they are optimal
- c) they are expensive
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

1. The minimum number of frames to be allocated to a process is decided by the

- a) the amount of available physical memory
- b) operating System
- c) instruction set architecture
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. When a page fault occurs before an executing instruction is complete if _____

- a) the instruction must be restarted
- b) the instruction must be ignored
- c) the instruction must be completed ignoring the page fault
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. Consider a machine in which all memory reference instructions have only one memory address, for them we need at least _____ frame(s).

- a) one
- b) two
- c) three
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: At least one frame for the instruction and one for the memory reference.

4. The maximum number of frames per process is defined by _____

- a) the amount of available physical memory
- b) operating System
- c) instruction set architecture
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. The algorithm in which we split m frames among n processes, to give everyone an equal share, m/n frames is known as _____

- a) proportional allocation algorithm
- b) equal allocation algorithm
- c) split allocation algorithm
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

6. The algorithm in which we allocate memory to each process according to its size is known as _____

- a) proportional allocation algorithm
- b) equal allocation algorithm
- c) split allocation algorithm
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. With either equal or proportional algorithm, a high priority process is treated _____ a low priority process.

- a) greater than
- b) same as
- c) lesser than
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. _____ replacement allows a process to select a replacement frame from the set of all frames, even if the frame is currently allocated to some other process.

- a) Local
- b) Universal
- c) Global
- d) Public

[View Answer](#)

Answer: c

Explanation: None.

9. _____ replacement allows each process to only select from its own set of allocated frames.

- a) Local
- b) Universal
- c) Global

d) Public

[View Answer](#)

Answer: a

Explanation: None.

10. One problem with the global replacement algorithm is that _____

- a) it is very expensive
- b) many frames can be allocated to a process
- c) only a few frames can be allocated to a process
- d) a process cannot control its own page - fault rate

[View Answer](#)

Answer: d

Explanation: None.

11. _____ replacement generally results in greater system throughput.

- a) Local
- b) Global
- c) Universal
- d) Public

[View Answer](#)

Answer: b

Explanation: None.

1. A process is thrashing if _____

- a) it spends a lot of time executing, rather than paging
- b) it spends a lot of time paging than executing
- c) it has no memory allocated to it
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. Thrashing _____ the CPU utilization.

- a) increases
- b) keeps constant
- c) decreases
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. What is a locality?

- a) a set of pages that are actively used together
- b) a space in memory
- c) an area near a set of processes
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. When a subroutine is called _____

- a) it defines a new locality
- b) it is in the same locality from where it was called
- c) it does not define a new locality
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. A program is generally composed of several different localities, which _____ overlap.

- a) may
- b) must
- c) do not
- d) must not

[View Answer](#)

Answer: a

Explanation: None.

6. In the working set model, for:

2 6 1 5 7 7 7 7 5 1 6 2 3 4 1 2 3 4 4 4 3 4 3 4 4 4 1 3 2 3

if $\text{DELTA} = 10$, then the working set at time $t1$ (....7 5 1) is?

- a) {1, 2, 4, 5, 6}
- b) {2, 1, 6, 7, 3}
- c) {1, 6, 5, 7, 2}
- d) {1, 2, 3, 4, 5}

[View Answer](#)

Answer: c

Explanation: None.

7. The accuracy of the working set depends on the selection of _____

- a) working set model
- b) working set size
- c) memory size
- d) number of pages in memory

[View Answer](#)

Answer: b

Explanation: None.

8. If working set window is too small _____

- a) it will not encompass entire locality
- b) it may overlap several localities
- c) it will cause memory problems
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. If working set window is too large _____

- a) it will not encompass entire locality
- b) it may overlap several localities
- c) it will cause memory problems
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

10. If the sum of the working - set sizes increases, exceeding the total number of available frames _____

- a) then the process crashes
- b) the memory overflows
- c) the system crashes
- d) the operating system selects a process to suspend

[View Answer](#)

Answer: d

Explanation: None.

11. Consider the following page reference string.

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For LRU page replacement algorithm with 4 frames, the number of page faults is?

- a) 10
- b) 14
- c) 8
- d) 11

[View Answer](#)

Answer: a

Explanation: None.

12. Consider the following page reference string.

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For LRU page replacement algorithm with 5 frames, the number of page faults is?

- a) 10
- b) 14
- c) 8
- d) 11

[View Answer](#)

Answer: c

Explanation: None.

13. Consider the following page reference string.

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For FIFO page replacement algorithms with 3 frames, the number of page faults is?

- a) 16
- b) 15
- c) 14
- d) 11

[View Answer](#)

Answer: a

Explanation: None.

14. Consider the following page reference string.

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For FIFO page replacement algorithms with 4 frames, the number of page faults is?

- a) 16
- b) 15

c) 14

d) 11

View Answer

Answer: c

Explanation: None.

15. Consider the following page reference string.

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

For Optimal page replacement algorithms with 3 frames, the number of page faults is?

a) 16

b) 15

c) 14

d) 11

View Answer

Answer: d

Explanation: None.

1. _____ is a unique tag, usually a number identifies the file within the file system.

a) File identifier

b) File name

c) File type

d) None of the mentioned

View Answer

Answer: a

Explanation: None.

2. To create a file _____

a) allocate the space in file system

b) make an entry for new file in directory

c) allocate the space in file system & make an entry for new file in directory

d) none of the mentioned

View Answer

Answer: c

Explanation: None.

3. By using the specific system call, we can _____

a) open the file

- b) read the file
- c) write into the file
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

4. File type can be represented by _____

- a) file name
- b) file extension
- c) file identifier
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. Which file is a sequence of bytes organized into blocks understandable by the system's linker?

- a) object file
- b) source file
- c) executable file
- d) text file

[View Answer](#)

Answer: a

Explanation: None.

6. What is the mounting of file system?

- a) crating of a filesystem
- b) deleting a filesystem
- c) attaching portion of the file system into a directory structure
- d) removing the portion of the file system into a directory structure

[View Answer](#)

Answer: c

Explanation: None.

7. Mapping of file is managed by _____

- a) file metadata
- b) page table
- c) virtual memory

d) file system

[View Answer](#)

Answer: a

Explanation: None.

8. Mapping of network file system protocol to local file system is done by _____

- a) network file system
- b) local file system
- c) volume manager
- d) remote mirror

[View Answer](#)

Answer: a

Explanation: None.

9. Which one of the following explains the sequential file access method?

- a) random access according to the given byte number
- b) read bytes one at a time, in order
- c) read/write sequentially by record
- d) read/write randomly by record

[View Answer](#)

Answer: b

Explanation: None.

10. When will file system fragmentation occur?

- a) unused space or single file are not contiguous
- b) used space is not contiguous
- c) unused space is non-contiguous
- d) multiple files are non-contiguous

[View Answer](#)

Answer: a

Explanation: None.

1. Management of metadata information is done by _____

- a) file-organisation module
- b) logical file system
- c) basic file system
- d) application programs

[View Answer](#)

Answer: b

Explanation: None.

2. A file control block contains the information about _____

- a) file ownership
- b) file permissions
- c) location of file contents
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. Which table contains the information about each mounted volume?

- a) mount table
- b) system-wide open-file table
- c) per-process open-file table
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

4. To create a new file application program calls _____

- a) basic file system
- b) logical file system
- c) file-organisation module
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. What will happens when a process closes the file?

- a) per-process table entry is not removed
- b) system wide entry's open count is decremented
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

6. What is raw disk?

- a) disk without file system
- b) empty disk
- c) disk lacking logical file system
- d) disk having file system

[View Answer](#)

Answer: a

Explanation: None.

7. The data structure used for file directory is called _____

- a) mount table
- b) hash table
- c) file table
- d) process table

[View Answer](#)

Answer: b

Explanation: None.

8. In which type of allocation method each file occupy a set of contiguous block on the disk?

- a) contiguous allocation
- b) dynamic-storage allocation
- c) linked allocation
- d) indexed allocation

[View Answer](#)

Answer: a

Explanation: None.

9. If the block of free-space list is free then bit will _____

- a) 1
- b) 0
- c) any of 0 or 1
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. Which protocol establishes the initial logical connection between a server and a client?

- a) transmission control protocol
- b) user datagram protocol
- c) mount protocol
- d) datagram congestion control protocol

[View Answer](#)

Answer: c

Explanation: None.

1. Data cannot be written to secondary storage unless written within a _____

- a) file
- b) swap space
- c) directory
- d) text format

[View Answer](#)

Answer: a

Explanation: None.

2. File attributes consist of _____

- a) name
- b) type
- c) identifier
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. The information about all files is kept in _____

- a) swap space
- b) operating system
- c) separate directory structure
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

4. A file is a/an _____ data type.

- a) abstract
- b) primitive
- c) public

d) private

[View Answer](#)

Answer: a

Explanation: None.

5. The operating system keeps a small table containing information about all open files called _____

- a) system table
- b) open-file table
- c) file table
- d) directory table

[View Answer](#)

Answer: b

Explanation: None.

6. In UNIX, what will the open system call return?

- a) pointer to the entry in the open file table
- b) pointer to the entry in the system wide table
- c) a file to the process calling it
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. System wide table in UNIX contains process independent information such as

- a) location of file on disk
- b) access dates
- c) file size
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. The open file table has a/an _____ associated with each file.

- a) file content
- b) file permission
- c) open count

d) close count

[View Answer](#)

Answer: c

Explanation: open count indicates the number of processes that have the file open.

9. Which of the following are the two parts of the file name?

- a) name & identifier
- b) identifier & type
- c) extension & name
- d) type & extension

[View Answer](#)

Answer: c

Explanation: None.

1. The UNIX system uses a/an _____ stored at the beginning of a some files to indicate roughly the type of file.

- a) identifier
- b) extension
- c) virtual number
- d) magic number

[View Answer](#)

Answer: d

Explanation: None.

2. The larger the block size, the _____ the internal fragmentation.

- a) greater
- b) lesser
- c) same
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. In the sequential access method, information in the file is processed

- a) one disk after the other, record access doesnt matter
- b) one record after the other
- c) one text document after the other

d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. Sequential access method _____ on random access devices.

- a) works well
- b) doesn't work well
- c) maybe works well and doesn't work well
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. The direct access method is based on a _____ model of a file, as _____ allow random access to any file block.

- a) magnetic tape, magnetic tapes
- b) tape, tapes
- c) disk, disks
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. For a direct access file _____

- a) there are restrictions on the order of reading and writing
- b) there are no restrictions on the order of reading and writing
- c) access is restricted permission wise
- d) access is not restricted permission wise

[View Answer](#)

Answer: b

Explanation: None.

7. A relative block number is an index relative to _____

- a) the beginning of the file
- b) the end of the file
- c) the last written position in file
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. The index contains _____

- a) names of all contents of file
- b) pointers to each page
- c) pointers to the various blocks
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. For large files, when the index itself becomes too large to be kept in memory?

- a) index is called
- b) an index is created for the index file
- c) secondary index files are created
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. To organise file systems on disk _____

- a) they are split into one or more partitions
- b) information about files is added to each partition
- c) they are made on different storage spaces
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. The directory can be viewed as a _____ that translates file names into their directory entries.

- a) symbol table
- b) partition
- c) swap space
- d) cache

[View Answer](#)

Answer: a

Explanation: None.

3. What will happen in the single level directory?

- a) All files are contained in different directories all at the same level
- b) All files are contained in the same directory
- c) Depends on the operating system
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. What will happen in the single level directory?

- a) all directories must have unique names
- b) all files must have unique names
- c) all files must have unique owners
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. What will happen in the two level directory structure?

- a) each user has his/her own user file directory
- b) the system doesn't its own master file directory
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. When a user job starts in a two level directory system, or a user logs in

-
- a) the users user file directory is searched
 - b) the system's master file directory is not searched
 - c) the master file directory is indexed by user name or account number, and each entry points to the UFD for that user
 - d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

7. When a user refers to a particular file?

- a) system MFD is searched
- b) his own UFD is not searched
- c) both MFD and UFD are searched
- d) every directory is searched

[View Answer](#)

Answer: c

Explanation: None.

8. What is the disadvantage of the two level directory structure?

- a) it does not solve the name collision problem
- b) it solves the name collision problem
- c) it does not isolate users from one another
- d) it isolates users from one another

[View Answer](#)

Answer: d

Explanation: None.

9. In the tree structured directories _____

- a) the tree has the stem directory
- b) the tree has the leaf directory
- c) the tree has the root directory
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. The current directory contains, most of the files that are _____

- a) of current interest to the user
- b) stored currently in the system
- c) not used in the system
- d) not of current interest to the system

[View Answer](#)

Answer: a

Explanation: None.

11. Which of the following are the types of Path names?

- a) absolute & relative
- b) local & global

- c) global & relative
- d) relative & local

[View Answer](#)

Answer: a

Explanation: None.

1. An absolute path name begins at the _____

- a) leaf
- b) stem
- c) current directory
- d) root

[View Answer](#)

Answer: d

Explanation: None.

2. A relative path name begins at the _____

- a) leaf
- b) stem
- c) current directory
- d) root

[View Answer](#)

Answer: c

Explanation: None.

3. In a tree structure, when deleting a directory that is not empty?

- a) The contents of the directory are safe
- b) The contents of the directory are also deleted
- c) contents of the directory are not deleted
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. When two users keep a subdirectory in their own directories, the structure being referred to is _____

- a) tree structure
- b) cyclic graph directory structure
- c) two level directory structure

d) acyclic graph directory

[View Answer](#)

Answer: d

Explanation: None.

5. A tree structure _____ the sharing of files and directories.

- a) allows
- b) may restrict
- c) restricts
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. With a shared file _____

- a) actual file exists
- b) there are two copies of the file
- c) the changes made by one person are not reflected to the other
- d) the changes made by one person are reflected to the other

[View Answer](#)

Answer: d

Explanation: None.

7. In UNIX, what is a link?

- a) a directory entry
- b) a pointer to another file or subdirectory
- c) implemented as an absolute or relative path name
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. The operating system _____ the links when traversing directory trees, to preserve the acyclic structure of the system.

- a) considers
- b) ignores
- c) deletes
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

9. The deletion of a link _____ the original file.

- a) deletes
- b) affects
- c) does not affect
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. When keeping a list of all the links/references to a file, and the list is empty, implies that _____

- a) the file has no copies
- b) the file is deleted
- c) the file is hidden
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

11. When a cycle exists, the reference count maybe non zero, even when it is no longer possible to refer to a directory or file, due to _____

- a) the possibility of one hidden reference
- b) the possibility of two hidden references
- c) the possibility of self referencing
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

1. What is the mount point?

- a) an empty directory at which the mounted file system will be attached
- b) a location where every time file systems are mounted
- c) is the time when the mounting is done
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. When a file system is mounted over a directory that is not empty then _____

- a) the system may not allow the mount
- b) the system must allow the mount
- c) the system may allow the mount and the directory's existing files will then be made obscure
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. In UNIX, exactly which operations can be executed by group members and other users is definable by _____

- a) the group's head
- b) the file's owner
- c) the file's permissions
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. A process _____ lower the priority of another process if both are owned by the same owner.

- a) must
- b) can
- c) cannot
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. In distributed file system _____ directories are visible from the local machine.

- a) protected
- b) local
- c) private

d) remote

[View Answer](#)

Answer: d

Explanation: None.

6. In the world wide web, a _____ is needed to gain access to the remote files, and separate operations are used to transfer files.

- a) laptop
- b) plugin
- c) browser
- d) player

[View Answer](#)

Answer: c

Explanation: None.

7. Anonymous access allows a user to transfer files _____

- a) without having an account on the remote system
- b) only if he accesses the system with a guest account
- c) only if he has an account on the remote system
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: The world wide web uses anonymous file exchange almost exclusively.

8. The machine containing the files is the _____ and the machine wanting to access the files is the _____

- a) master, slave
- b) memory, user
- c) server, client
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. Distributed naming services/Distributed information systems have been devised to _____

- a) provide information about all the systems
- b) provide unified access to the information needed for remote computing

- c) provide unique names to all systems in a network
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

10. Domain name system provides _____

- a) host-name-to-network-address translations for the entire internet
- b) network-address-to-host-name translations for the entire internet
- c) binary to hex translations for the entire internet
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. To recover from failures in the network operations _____ information may be maintained.

- a) ip address
- b) state
- c) stateless
- d) operating system

[View Answer](#)

Answer: b

Explanation: None.

12. The series of accesses between the open and close operations is a _____

- a) transaction
- b) procedure
- c) program
- d) file session

[View Answer](#)

Answer: d

Explanation: None.

1. Reliability of files can be increased by _____

- a) keeping the files safely in the memory
- b) making a different partition for the files
- c) by keeping them in external storage

d) by keeping duplicate copies of the file

[View Answer](#)

Answer: d

Explanation: None.

2. Protection is only provided at the _____ level.

- a) lower
- b) central
- c) higher
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. What is the main problem with access control lists?

- a) their maintenance
- b) their length
- c) their permissions
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. Many systems recognize three classifications of users in connection with each file (to condense the access control list).

- a) Owner
- b) Group
- c) Universe
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. All users in a group get _____ access to a file.

- a) different
- b) similar
- c) same
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

6. Universe consists of _____

- a) all users that aren't included in the group or owners
- b) all users that are not owners
- c) all users in the system
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

7. In UNIX, groups can be created and modified by?

- a) superuser
- b) any user
- c) a programmer only
- d) the people in the group only

[View Answer](#)

Answer: a

Explanation: None.

8. To control access the three bits used in UNIX are represented by

-
- a) r
 - b) w
 - c) x
 - d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. If each access to a file is controlled by a password, then what is the disadvantage?

- a) user will need to remember a lot of passwords
- b) it is not reliable
- c) it is not efficient
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. What will happen in a multi level directory structure?

- a) the same previous techniques will be used as in the other structures
- b) a mechanism for directory protection will have to be applied
- c) the subdirectories do not need protection once the directory is protected
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

11. In UNIX, the directory protection is handled _____ to the file protection.

- a) different
- b) similar
- c) it is not handled at all
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

12. Disks are segmented into one or more partitions, each containing a file system or _____

- a) left 'raw'
- b) made into swap space
- c) made into backup space
- d) left 'ripe'

[View Answer](#)

Answer: a

Explanation: None.

1. The three major methods of allocating disk space that are in wide use are _____

- a) contiguous
- b) linked
- c) indexed
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

2. In contiguous allocation _____

- a) each file must occupy a set of contiguous blocks on the disk
- b) each file is a linked list of disk blocks
- c) all the pointers to scattered blocks are placed together in one location
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. In linked allocation _____

- a) each file must occupy a set of contiguous blocks on the disk
- b) each file is a linked list of disk blocks
- c) all the pointers to scattered blocks are placed together in one location
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. In indexed allocation _____

- a) each file must occupy a set of contiguous blocks on the disk
- b) each file is a linked list of disk blocks
- c) all the pointers to scattered blocks are placed together in one location
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

5. On systems where there are multiple operating system, the decision to load a particular one is done by _____

- a) boot loader
- b) bootstrap
- c) process control block
- d) file control block

[View Answer](#)

Answer: a

Explanation: None.

6. The VFS (virtual file system) activates file system specific operations to handle local requests according to their _____

- a) size
- b) commands
- c) timings
- d) file system types

[View Answer](#)

Answer: d

Explanation: None.

7. What is the real disadvantage of a linear list of directory entries?

- a) size of the linear list in memory
- b) linear search to find a file
- c) it is not reliable
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. Contiguous allocation of a file is defined by _____

- a) disk address of the first block & length
- b) length & size of the block
- c) size of the block
- d) total size of the file

[View Answer](#)

Answer: a

Explanation: None.

9. One difficulty of contiguous allocation is _____

- a) finding space for a new file
- b) inefficient
- c) costly
- d) time taking

[View Answer](#)

Answer: a

Explanation: None.

10. _____ and _____ are the most common strategies used to select a free hole from the set of available holes.

- a) First fit, Best fit
- b) Worst fit, First fit
- c) Best fit, Worst fit
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. The first fit and best fit algorithms suffer from _____

- a) internal fragmentation
- b) external fragmentation
- c) starvation
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

12. To solve the problem of external fragmentation _____ needs to be done periodically.

- a) compaction
- b) check
- c) formatting
- d) replacing memory

[View Answer](#)

Answer: a

Explanation: None.

13. If too little space is allocated to a file _____

- a) the file will not work
- b) there will not be any space for the data, as the FCB takes it all
- c) the file cannot be extended
- d) the file cannot be opened

[View Answer](#)

Answer: c

Explanation: None.

1. A device driver can be thought of like a translator. Its input consists of _____ commands and output consists of _____ instructions.

- a) high level, low level

- b) low level, high level
- c) complex, simple
- d) low level, complex

[View Answer](#)

Answer: a

Explanation: None.

2. The file organization module knows about _____

- a) files
- b) logical blocks of files
- c) physical blocks of files
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. Metadata includes _____

- a) all of the file system structure
- b) contents of files
- c) both file system structure and contents of files
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

4. For each file there exists a _____ that contains information about the file, including ownership, permissions and location of the file contents.

- a) metadata
- b) file control block
- c) process control block
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. For processes to request access to file contents, they need _____

- a) to run a separate program
- b) special interrupts
- c) to implement the open and close system calls

d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. During compaction time, other normal system operations _____ be permitted.

- a) can
- b) cannot
- c) is
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. When in contiguous allocation the space cannot be extended easily?

- a) the contents of the file have to be copied to a new space, a larger hole
- b) the file gets destroyed
- c) the file will get formatted and lost all its data
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. In the linked allocation, the directory contains a pointer to which block?

- I. first block
- II. last block
- a) I only
- b) II only
- c) Both I and II
- d) Neither I nor II

[View Answer](#)

Answer: c

Explanation: None.

9. There is no _____ with linked allocation.

- a) internal fragmentation
- b) external fragmentation
- c) starvation

d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

10. What is the major disadvantage with a linked allocation?

- a) internal fragmentation
- b) external fragmentation
- c) there is no sequential access
- d) there is only sequential access

[View Answer](#)

Answer: d

Explanation: None.

11. What if a pointer is lost or damaged in a linked allocation?

- a) the entire file could get damaged
- b) only a part of the file would be affected
- c) there would not be any problems
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

12. FAT stands for _____

- a) File Attribute Transport
- b) File Allocation Table
- c) Fork At Time
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

13. By using FAT, random access time is _____

- a) the same
- b) increased
- c) decreased
- d) not affected

[View Answer](#)

Answer: c

Explanation: None.

1. A better way of contiguous allocation to extend the file size is

- a) adding an extent (another chunk of contiguous space)
- b) adding an index table to the first contiguous block
- c) adding pointers into the first contiguous block
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. If the extents are too large, then what is the problem that comes in?

- a) internal fragmentation
- b) external fragmentation
- c) starvation
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. The FAT is used much as a _____

- a) stack
- b) linked list
- c) data
- d) pointer

[View Answer](#)

Answer: b

Explanation: None.

4. A section of disk at the beginning of each partition is set aside to contain the table in _____

- a) fat
- b) linked allocation
- c) hashed allocation
- d) indexed allocation

[View Answer](#)

Answer: a

Explanation: None.

5. Contiguous allocation has two problems _____ and _____ that linked allocation solves.

- a) external - fragmentation & size - declaration
- b) internal - fragmentation & external - fragmentation
- c) size - declaration & internal - fragmentation
- d) memory - allocation & size - declaration

[View Answer](#)

Answer: a

Explanation: None.

6. Each _____ has its own index block.

- a) partition
- b) address
- c) file
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

7. Indexed allocation _____ direct access.

- a) supports
- b) does not support
- c) is not related to
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. The pointer overhead of indexed allocation is generally _____ the pointer overhead of linked allocation.

- a) less than
- b) equal to
- c) greater than
- d) keeps varying with

[View Answer](#)

Answer: c

Explanation: None.

9. For any type of access, contiguous allocation requires _____ access to get a disk block.

- a) only one
- b) at least two
- c) exactly two
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: We can easily keep the initial address of the file in memory and calculate immediately the disk address of the ith block and read it directly.

10. Consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26 and 27 are free and the rest of the blocks are allocated. Then the free space bitmap would be _____

- a) 100001100000011001111100011111...
- b) 1100001100000011001111100011111...
- c) 011110011111000110000001100000...
- d) 00111100111110001100000011100000...

[View Answer](#)

Answer: d

Explanation: None.

1. _____ tend to represent a major bottleneck in system performance.

- a) CPUs
- b) Disks
- c) Programs
- d) I/O

[View Answer](#)

Answer: b

Explanation: None.

2. In UNIX, even an 'empty' disk has a percentage of its space lost to _____

- a) programs
- b) inodes
- c) virtual memory
- d) stacks

[View Answer](#)

Answer: b

Explanation: None.

3. By preallocating the inodes and spreading them across the volume, we _____ the system performance.

- a) improve
- b) decrease
- c) maintain
- d) do not affect

[View Answer](#)

Answer: a

Explanation: None.

4. _____ writes occur in the order in which the disk subsystem receives them, and the writes are not buffered.

- a) Asynchronous
- b) Regular
- c) Synchronous
- d) Irregular

[View Answer](#)

Answer: c

Explanation: None.

5. In _____ writes, the data is stored in the cache.

- a) Asynchronous
- b) Regular
- c) Synchronous
- d) Irregular

[View Answer](#)

Answer: a

Explanation: None.

6. A file being read or written sequentially should not have its pages replaced in LRU order, because _____

- a) it is very costly
- b) the most recently used page will be used last
- c) it is not efficient
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. In the optimized technique for sequential access _____ removes a page from the buffer as soon as the next page is requested.

- a) write ahead
- b) read ahead
- c) free-behind
- d) add-front

[View Answer](#)

Answer: c

Explanation: None.

8. With _____ a requested page and several subsequent pages are read and cached.

- a) write ahead
- b) read ahead
- c) free-behind
- d) add-front

[View Answer](#)

Answer: b

Explanation: None.

1. Some directory information is kept in main memory or cache to _____

- a) fill up the cache
- b) increase free space in secondary storage
- c) decrease free space in secondary storage
- d) speed up access

[View Answer](#)

Answer: d

Explanation: None.

2. A systems program such as fsck in _____ is a consistency checker.

- a) UNIX
- b) Windows
- c) Macintosh
- d) Solaris

[View Answer](#)

Answer: a

Explanation: None.

3. A consistency checker _____ and tries to fix any inconsistencies it finds.

- a) compares the data in the secondary storage with the data in the cache
- b) compares the data in the directory structure with the data blocks on disk
- c) compares the system generated output and user required output
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. Each set of operations for performing a specific task is a _____

- a) program
- b) code
- c) transaction
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

5. Once the changes are written to the log, they are considered to be _____

- a) committed
- b) aborted
- c) completed
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. When an entire committed transaction is completed, _____

- a) it is stored in the memory
- b) it is removed from the log file
- c) it is redone
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. What is a circular buffer?

- a) writes to the end of its space and then continues at the beginning
- b) overwrites older values as it goes
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. All the changes that were done from a transaction that did not commit before the system crashed, have to be _____

- a) saved
- b) saved and the transaction redone
- c) undone
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

1. A machine in Network file system (NFS) can be _____

- a) client
- b) server
- c) both client and server
- d) neither client nor server

[View Answer](#)

Answer: c

Explanation: None.

2. A _____ directory is mounted over a directory of a _____ file system.

- a) local, remote
- b) remote, local
- c) local, local
- d) none of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. The _____ becomes the name of the root of the newly mounted directory.

- a) root of the previous directory
- b) local directory
- c) remote directory itself
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. _____ mounts, is when a file system can be mounted over another file system, that is remotely mounted, not local.

- a) recursive
- b) cascading
- c) trivial
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

5. The mount mechanism _____ a transitive property.

- a) exhibits
- b) does not exhibit
- c) may exhibit
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: Mounting a remote file system does not give the client access to other file systems that were, by chance, mounted over the former file system.

6. A mount operation includes the _____

- a) name of the network
- b) name of the remote directory to be mounted
- c) name of the server machine storing it
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. The mount request is mapped to the corresponding _____ and is forwarded to the mount server running on the specific server machine.

- a) IPC
- b) System
- c) CPU
- d) RPC

[View Answer](#)

Answer: b

Explanation: None.

8. The server maintains a/an _____ that specifies local file systems that it exports for mounting, along with names of machines that are permitted to mount them.

- a) export list
- b) import list
- c) sending list
- d) receiving list

[View Answer](#)

Answer: a

Explanation: None.

9. In UNIX, the file handle consists of a _____ and _____

- a) file-system identifier & an inode number
- b) an inode number & FAT
- c) a FAT & an inode number
- d) a file pointer & FAT

[View Answer](#)

Answer: a

Explanation: None.

1. The NFS servers _____

- a) are stateless
- b) save the current state of the request
- c) maybe stateless
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. Every NFS request has a _____ allowing the server to determine if a request is duplicated or if any are missing.

- a) name
- b) transaction
- c) sequence number
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. A server crash and recovery will _____ to a client.

- a) be visible
- b) affect
- c) be invisible
- d) harm

[View Answer](#)

Answer: c

Explanation: All blocks that the server is managing for the client will be intact.

4. The server must write all NFS data _____

- a) synchronously
- b) asynchronously
- c) index-wise
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. A single NFS write procedure _____

- a) can be atomic
- b) is atomic
- c) is non atomic
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

6. The NFS protocol _____ concurrency control mechanisms.

- a) provides
- b) does not provide
- c) may provide
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. _____ in NFS involves the parsing of a path name into separate directory entries - or components.

- a) Path parse
- b) Path name parse
- c) Path name translation
- d) Path name parsing

[View Answer](#)

Answer: c

Explanation: None.

8. For every pair of component and directory vnode after path name translation

- a) a single NFS lookup call is used sequentially
- b) a single NFS lookup call is used beginning from the last component
- c) at least two NFS lookup calls per component are performed
- d) a separate NFS lookup call is performed

[View Answer](#)

Answer: d

Explanation: None.

9. When a client has a cascading mount _____ server(s) is/are involved in a path name traversal.

- a) at least one
- b) more than one
- c) more than two
- d) more than three

[View Answer](#)

Answer: b

Explanation: None.

1. I/O hardware contains _____

- a) Bus
- b) Controller
- c) I/O port and its registers
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

2. The data-in register of I/O port is _____

- a) Read by host to get input
- b) Read by controller to get input
- c) Written by host to send output
- d) Written by host to start a command

[View Answer](#)

Answer: a

Explanation: None.

3. The host sets _____ bit when a command is available for the controller to execute.

- a) write
- b) status
- c) command-ready
- d) control

[View Answer](#)

Answer: c

Explanation: None.

4. When hardware is accessed by reading and writing to the specific memory locations, then it is called _____

- a) port-mapped I/O
- b) controller-mapped I/O
- c) bus-mapped I/O
- d) none of the mentioned

[View Answer](#)

Answer: d

Explanation: It is called memory-mapped I/O.

5. Device drivers are implemented to interface _____

- a) character devices
- b) block devices
- c) network devices
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

6. Which hardware triggers some operation after certain programmed count?

- a) programmable interval timer
- b) interrupt timer
- c) programmable timer
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. The device-status table contains _____

- a) each I/O device type
- b) each I/O device address
- c) each I/O device state
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. Which buffer holds the output for a device?

- a) spool
- b) output
- c) status
- d) magic

[View Answer](#)

Answer: a

Explanation: None.

9. Which one of the following connects high-speed high-bandwidth device to memory subsystem and CPU.

- a) Expansion bus

- b) PCI bus
- c) SCSI bus
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. A process is moved to wait queue when I/O request is made with

- a) non-blocking I/O
- b) blocking I/O
- c) asynchronous I/O
- d) synchronous I/O

[View Answer](#)

Answer: b

Explanation: None.

1. In _____ information is recorded magnetically on platters.

- a) magnetic disks
- b) electrical disks
- c) assemblies
- d) cylinders

[View Answer](#)

Answer: a

Explanation: None.

2. The heads of the magnetic disk are attached to a _____ that moves all the heads as a unit.

- a) spindle
- b) disk arm
- c) track
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. The set of tracks that are at one arm position make up a _____

- a) magnetic disks
- b) electrical disks

c) assemblies

d) cylinders

View Answer

Answer: d

Explanation: None.

4. The time taken to move the disk arm to the desired cylinder is called the

a) positioning time

b) random access time

c) seek time

d) rotational latency

View Answer

Answer: c

Explanation: None.

5. The time taken for the desired sector to rotate to the disk head is called

a) positioning time

b) random access time

c) seek time

d) rotational latency

View Answer

Answer: d

Explanation: None.

6. When the head damages the magnetic surface, it is known as _____

a) disk crash

b) head crash

c) magnetic damage

d) all of the mentioned

View Answer

Answer: b

Explanation: None.

7. A floppy disk is designed to rotate _____ as compared to a hard disk drive.

a) faster

b) slower

- c) at the same speed
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. What is the host controller?

- a) controller built at the end of each disk
- b) controller at the computer end of the bus
- c) all of the mentioned
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

9. _____ controller sends the command placed into it, via messages to the _____ controller.

- a) host, host
- b) disk, disk
- c) host, disk
- d) disk, host

[View Answer](#)

Answer: c

Explanation: None.

10. What is the disk bandwidth?

- a) the total number of bytes transferred
- b) total time between the first request for service and the completion on the last transfer
- c) the total number of bytes transferred divided by the total time between the first request for service and the completion on the last transfer
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

1. Whenever a process needs I/O to or from a disk it issues a

- a) system call to the CPU

- b) system call to the operating system
- c) a special procedure
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. If a process needs I/O to or from a disk, and if the drive or controller is busy then _____

- a) the request will be placed in the queue of pending requests for that drive
- b) the request will not be processed and will be ignored completely
- c) the request will be not be placed
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. Consider a disk queue with requests for I/O to blocks on cylinders.

98 183 37 122 14 124 65 67

Considering FCFS (first cum first served) scheduling, the total number of head movements is, if the disk head is initially at 53 is?

- a) 600
- b) 620
- c) 630
- d) 640

[View Answer](#)

Answer: d

Explanation: None.

4. Consider a disk queue with requests for I/O to blocks on cylinders.

98 183 37 122 14 124 65 67

Considering SSTF (shortest seek time first) scheduling, the total number of head movements is, if the disk head is initially at 53 is?

- a) 224
- b) 236
- c) 245
- d) 240

[View Answer](#)

Answer: b

Explanation: None.

5. Random access in magnetic tapes is _____ compared to magnetic disks.

- a) fast
- b) very fast
- c) slow
- d) very slow

[View Answer](#)

Answer: d

Explanation: None.

6. Magnetic tape drives can write data at a speed _____ disk drives.

- a) much lesser than
- b) comparable to
- c) much faster than
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

7. On media that use constant linear velocity (CLV), the _____ is uniform.

- a) density of bits on the disk
- b) density of bits per sector
- c) the density of bits per track
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: The farther a track is from the center of the disk.

8. SSTF algorithm, like SJF _____ of some requests.

- a) may cause starvation
- b) will cause starvation
- c) does not cause starvation
- d) causes aging

[View Answer](#)

Answer: a

Explanation: None.

9. In the _____ algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests till the other end of the disk. At the other end, the direction is reversed and servicing continues.

- a) LOOK
- b) SCAN
- c) C-SCAN
- d) C-LOOK

[View Answer](#)

Answer: b

Explanation: None.

10. In the _____ algorithm, the disk head moves from one end to the other, servicing requests along the way. When the head reaches the other end, it immediately returns to the beginning of the disk without servicing any requests on the return trip.

- a) LOOK
- b) SCAN
- c) C-SCAN
- d) C-LOOK

[View Answer](#)

Answer: c

Explanation: None.

11. In the _____ algorithm, the disk arm goes as far as the final request in each direction, then reverses direction immediately without going to the end of the disk.

- a) LOOK
- b) SCAN
- c) C-SCAN
- d) C-LOOK

[View Answer](#)

Answer: a

Explanation: None.

1. The process of dividing a disk into sectors that the disk controller can read and write, before a disk can store data is known as _____

- a) partitioning
- b) swap space creation
- c) low-level formatting

d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. The data structure for a sector typically contains _____

- a) header
- b) data area
- c) trailer
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. The header and trailer of a sector contain information used by the disk controller such as _____ and _____

- a) main section & disk identifier
- b) error correcting codes (ECC) & sector number
- c) sector number & main section
- d) disk identifier & sector number

[View Answer](#)

Answer: b

Explanation: None.

4. The two steps the operating system takes to use a disk to hold its files are _____ and _____

- a) partitioning & logical formatting
- b) swap space creation & caching
- c) caching & logical formatting
- d) logical formatting & swap space creation

[View Answer](#)

Answer: a

Explanation: None.

5. The _____ program initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory, and then starts the operating system.

- a) main
- b) bootloader

c) bootstrap

d) rom

[View Answer](#)

Answer: c

Explanation: None.

6. For most computers, the bootstrap is stored in _____

a) RAM

b) ROM

c) Cache

d) Tertiary storage

[View Answer](#)

Answer: b

Explanation: None.

7. A disk that has a boot partition is called a _____

a) start disk

b) end disk

c) boot disk

d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

8. Defective sectors on disks are often known as _____

a) good blocks

b) destroyed blocks

c) bad blocks

d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. In SCSI disks used in high end PCs, the controller maintains a list of _____ on the disk. The disk is initialized during _____ formatting which sets aside spare sectors not visible to the operating system.

a) destroyed blocks, high level formatting

b) bad blocks, partitioning

- c) bad blocks, low level formatting
- d) destroyed blocks, partitioning

[View Answer](#)

Answer: c

Explanation: None.

10. The scheme used in the above question is known as _____ or _____

- a) sector sparing & forwarding
- b) forwarding & sector utilization
- c) backwarding & forwarding
- d) sector utilization & backwarding

[View Answer](#)

Answer: a

Explanation: None.

11. An unrecoverable error is known as _____

- a) hard error
- b) tough error
- c) soft error
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. Virtual memory uses disk space as an extension of _____

- a) secondary storage
- b) main memory
- c) tertiary storage
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. Using swap space significantly _____ system performance.

- a) increases
- b) decreases
- c) maintains
- d) does not affect

[View Answer](#)

Answer: b

Explanation: Disk access is much slower than memory access.

3. Linux _____ the use of multiple swap spaces.

- a) allows
- b) does not allow
- c) may allow
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: Putting these swap spaces on separate disks reduces the load places on the I/O system.

4. A single swap space _____ reside in two places.

- a) can
- b) cannot
- c) must not
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. If the swap space is simply a large file, within the file system, _____ used to create it, name it and allocate its space.

- a) special routines must be
- b) normal file system routines can be
- c) normal file system routines cannot be
- d) swap space storage manager is

[View Answer](#)

Answer: b

Explanation: None.

6. For swap space created in a separate disk partition where no file system or directory structure is placed, _____ used to allocate and deallocate the blocks.

- a) special routines must be
- b) normal file system routines can be
- c) normal file system routines cannot be
- d) swap space storage manager is

[View Answer](#)

Answer: d

Explanation: None.

7. When a fixed amount of swap space is created during disk partitioning, more swap space can be added only by?

- I) repartitioning of the disk
- II) adding another swap space elsewhere
- a) only I
- b) only II
- c) both I and II
- d) neither I nor II

[View Answer](#)

Answer: c

Explanation: None.

8. In UNIX, two per process _____ are used by the kernel to track swap space use.

- a) process tables
- b) swap maps
- c) memory maps
- d) partition maps

[View Answer](#)

Answer: b

Explanation: None.

9. It is _____ to reread a page from the file system than to write it to swap space and then to reread it from there.

- a) useless
- b) less efficient
- c) more efficient
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

1. RAID level 3 supports a lower number of I/Os per second, because

-
- a) Every disk has to participate in every I/O request
 - b) Only one disk participates per I/O request
 - c) I/O cycle consumes a lot of CPU time

d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. RAID level _____ is also known as block interleaved parity organisation and uses block level striping and keeps a parity block on a separate disk.

- a) 1
- b) 2
- c) 3
- d) 4

[View Answer](#)

Answer: d

Explanation: None.

3. A performance problem with _____ is the expense of computing and writing parity.

- a) non-parity based RAID levels
- b) parity based RAID levels
- c) all RAID levels
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. In RAID level 4, one block read, accesses _____

- a) only one disk
- b) all disks simultaneously
- c) all disks sequentially
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: Other requests are allowed to be processed by other disks.

5. The overall I/O rate in RAID level 4 is _____

- a) low
- b) very low
- c) high

d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: All disks can be read in parallel.

6. A write of a block has to access _____

- a) the disk on which the block is stored
- b) parity disk
- c) a parity block
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. RAID level 5 is also known as _____

- a) bit-interleaved parity organization
- b) block-interleaved parity organization
- c) block-interleaved distributed parity
- d) memory-style ECC organization

[View Answer](#)

Answer: c

Explanation: None.

8. RAID level _____ spreads parity and data among all N+1 disks rather than storing data in N disks and parity in 1.

- a) 3
- b) 4
- c) 5
- d) 6

[View Answer](#)

Answer: c

Explanation: None.

9. The potential overuse of a single parity disk is avoided in RAID level _____

- a) 3
- b) 4
- c) 5
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. RAID level 0+1 is used because, RAID level 0 provides _____ whereas RAID level 1 provides _____

- a) performance, redundancy
- b) performance, reliability
- c) redundancy, performance
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

11. If a disk fails in RAID level _____ rebuilding lost data is easiest.

- a) 1
- b) 2
- c) 3
- d) 4

[View Answer](#)

Answer: a

Explanation: Data can be copied from another disk in raid level 1, for other raid levels all other disks have to be accessed.

12. Where performance and reliability are both important, RAID level _____ is used.

- a) 0
- b) 1
- c) 2
- d) 0+1

[View Answer](#)

Answer: d

Explanation: None.

1. A large number of disks in a system improves the rate at which data can be read or written if _____

- a) the disks are operated on sequentially
- b) the disks are operated on selectively
- c) the disks are operated in parallel
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. RAID stands for _____

- a) Redundant Allocation of Inexpensive Disks
- b) Redundant Array of Important Disks
- c) Redundant Allocation of Independent Disks
- d) Redundant Array of Independent Disks

[View Answer](#)

Answer: d

Explanation: None.

3. If the mean time to failure of a single disk is 100,000 hours, then the mean time to failure of some disk in an array of 100 disks will be _____

- a) 100 hours
- b) 10 days
- c) 10 hours
- d) 1000 hours

[View Answer](#)

Answer: d

Explanation: None.

4. The solution to the problem of reliability is the introduction of _____

- a) aging
- b) scheduling
- c) redundancy
- d) disks

[View Answer](#)

Answer: c

Explanation: None.

5. The technique of duplicating every disk is known as _____

- a) mirroring
- b) shadowing
- c) redundancy
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. The mean time to failure of a mirrored disk depends on _____

- I) the mean time to failure of individual disks
- II) the mean time to repair
- a) Only I
- b) Only II
- c) Both I and II
- d) Neither I nor II

[View Answer](#)

Answer: c

Explanation: None.

7. RAID level _____ refers to disk arrays with striping at the level of blocks, but without any redundancy.

- a) 0
- b) 1
- c) 2
- d) 3

[View Answer](#)

Answer: a

Explanation: None.

8. RAID level _____ refers to disk mirroring.

- a) 0
- b) 1
- c) 2
- d) 3

[View Answer](#)

Answer: b

Explanation: None.

9. RAID level _____ is also known as bit interleaved parity organisation.

- a) 0
- b) 1
- c) 2
- d) 3

[View Answer](#)

Answer: d

Explanation: None.

10. A single parity bit can be used for _____

- a) detection
- b) multiple error corrections
- c) few error corrections
- d) all of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. RAID level _____ is also known as memory style error correcting code(ECC) organization.

- a) 1
- b) 2
- c) 3
- d) 4

[View Answer](#)

Answer: b

Explanation: None.

12. RAID level 3 does not have _____ as in RAID level 1.

- a) efficiency
- b) enough storage space for data
- c) storage overhead
- d) time consumption overhead

[View Answer](#)

Answer: c

Explanation: There is one mirror disk for every disk in level 1.

1. Tertiary storage is built with _____

- a) a lot of money
- b) unremovable media
- c) removable media
- d) secondary storage

[View Answer](#)

Answer: c

Explanation: None.

2. Floppy disks are examples of _____

- a) primary storage

- b) secondary storage
- c) tertiary storage
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. What is a magneto-optic disk?

- a) primary storage
- b) secondary storage
- c) removable disk
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

4. The magneto-optic head flies _____ the disk surface than a magnetic disk head does.

- a) much farther from
- b) much closer to
- c) at the same distance as
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

5. Optical disks _____ magnetism.

- a) use
- b) do not use
- c) may use
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

6. The phase change disk is coated with a material that can freeze into either _____ or _____ state.

- a) crystalline, solid
- b) ice, amorphous

- c) crystalline, liquid
- d) crystalline, amorphous

[View Answer](#)

Answer: d

Explanation: None.

7. WORM stands for _____

- a) write only, read mandatory
- b) write once, read many times
- c) write only once, read multiple
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

8. A tape holds _____ data than optical or magnetic disk cartridge.

- a) lesser
- b) more
- c) much lesser
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

9. Random access to tape is _____ a disk seek.

- a) much slower than
- b) much faster than
- c) comparable to
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. A typical tape drive is _____ a typical disk drive.

- a) more expensive than
- b) cheaper than
- c) of the same cost as
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. The surface area of tape is _____ the surface area of a disk.

- a) much lesser than
- b) much larger than
- c) equal to
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. In domain structure what is Access-right equal to?

- a) Access-right = object-name, rights-set
- b) Access-right = read-name, write-set
- c) Access-right = read-name, execute-set
- d) Access-right = object-name, execute-set

[View Answer](#)

Answer: a

Explanation: None.

2. What is meaning of right-set?

- a) It is a subset consist of read and write
- b) It is a subset of all valid operations that can be performed on the object
- c) It is a subset consist of read, write and execute
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. What is Domain?

- a) Domain = Set of all objects
- b) It is a collection of protection policies
- c) Domain= set of access-rights
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

4. What does the access matrix represent?

- a) Rows-Domains, Columns-Objects
- b) Rows-Objects, Columns-Domains
- c) Rows-Access List, Columns-Domains
- d) Rows-Domains, Columns-Access list

[View Answer](#)

Answer: a

Explanation: None.

5. What are the three additional operations to change the contents of the access-matrix?

- a) copy
- b) Owner
- c) control
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None

6. Who can add new rights and remove some rights?

- a) copy
- b) transfer
- c) limited copy
- d) owner

[View Answer](#)

Answer: d

Explanation: None.

7. What are the three copyrights?

- a) copy
- b) transfer
- c) limited copy
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. Which two rights allow a process to change the entries in a column?

- a) copy and transfer

- b) copy and owner
- c) owner and transfer
- d) deny and copy

[View Answer](#)

Answer: a

Explanation: None.

9. Which is an unsolvable problem in access-matrix?

- a) Owner override
- b) Brute force
- c) Access denied
- d) Confinement

[View Answer](#)

Answer: d

Explanation: None.

10. Which of the following objects require protection?

- a) CPU
- b) Printers
- c) Motherboard
- d) All of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

11. What is 'separation' in security of Operating systems?

- a) To have separate login for different users
- b) To have separate Hard disk drive/partition for different users
- c) It means keeping one user's objects separate from other users
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

12. Which of the following statements are correct?

- i) Physical separation - in which process use different physical objects like separate printers
- ii) Physical separation - in which process having different security requirement at different times

- iii) Logical separation - In which users operate under illusion that no other processes exist
 - iv) Logical separation - In which processes conceal their data and computations
- a) i
 - b) i and iii
 - c) ii and iii
 - d) iii and iv

[View Answer](#)

Answer: b

Explanation: None.

13. Which of the following statements are correct?

- i) Physical separation - in which process use different physical objects like separate printers
 - ii) Temporal separation - in which process having different security requirement at different times
 - iii) Physical separation - In which users operate under illusion that no other processes exist
 - iv) Temporal separation - In which processes conceal their data and computations
- a) i
 - b) i and ii
 - c) ii and iii
 - d) iii and iv

[View Answer](#)

Answer: b

Explanation: None.

14. Which of the following statements are correct?

- i) logical separation - in which process use different physical objects like separate printers
 - ii) cryptographic separation - in which process having different security requirement at different times
 - iii) Logical separation - In which users operate under illusion that no other processes exist
 - iv) cryptographic separation - In which processes conceal their data and computations
- a) i
 - b) i and ii
 - c) ii and iii

d) iii and iv

View Answer

Answer: d

Explanation: None.

15. What are the various roles of protection?

- a) It is used to detect errors which can prevent contamination of system
- b) It is used used to accelerate a process
- c) It is used to optimize system downtime
- d) None of the mentioned

View Answer

Answer: a

Explanation: None.

16. Which of the following objects require protection?

- a) Memory
- b) Monitor
- c) Power supply unit
- d) All of the mentioned

View Answer

Answer: a

Explanation: None.

1. Which principle states that programs, users and even the systems be given just enough privileges to perform their task?

- a) principle of operating system
- b) principle of least privilege
- c) principle of process scheduling
- d) none of the mentioned

View Answer

Answer: b

Explanation: None.

2. _____ is an approach to restricting system access to authorized users.

- a) Role-based access control
- b) Process-based access control
- c) Job-based access control
- d) None of the mentioned

View Answer

Answer: a

Explanation: None.

3. For system protection, a process should access _____

- a) all the resources
- b) only those resources for which it has authorization
- c) few resources but authorization is not required
- d) all of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. The protection domain of a process contains _____

- a) object name
- b) rights-set
- c) both object name and rights-set
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

5. If the set of resources available to the process is fixed throughout the process's lifetime then its domain is _____

- a) static
- b) dynamic
- c) neither static nor dynamic
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. Access matrix model for user authentication contains _____

- a) a list of objects
- b) a list of domains
- c) a function which returns an object's type
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. Global table implementation of the matrix table contains _____

- a) domain
- b) object
- c) right-set
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. For a domain _____ is a list of objects together with the operation allowed on these objects.

- a) capability list
- b) access list
- c) both capability and access list
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. Which one of the following is capability based protection system?

- a) hydra
- b) cambridge CAP system
- c) both hydra and cambridge CAP system
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

10. In UNIX, domain switch is accomplished via _____

- a) file system
- b) user
- c) superuser
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

1. When an attempt is to make a machine or network resource unavailable to its intended users, the attack is called _____

- a) denial-of-service attack
- b) slow read attack
- c) spoofed attack
- d) starvation attack

[View Answer](#)

Answer: a

Explanation: None.

2. The code segment that misuses its environment is called a _____

- a) internal thief
- b) trojan horse
- c) code stacker
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

3. The internal code of any software that will set off a malicious function when specified conditions are met, is called _____

- a) logic bomb
- b) trap door
- c) code stacker
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. The pattern that can be used to identify a virus is known as _____

- a) stealth
- b) virus signature
- c) armoured
- d) multipartite

[View Answer](#)

Answer: b

Explanation: None.

5. Which one of the following is a process that uses the spawn mechanism to ravage the system performance?

- a) worm

- b) trojan
- c) threat
- d) virus

[View Answer](#)

Answer: a

Explanation: None.

6. What is a trap door in a program?

- a) a security hole, inserted at programming time in the system for later use
- b) a type of antivirus
- c) security hole in a network
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. Which one of the following is not an attack, but a search for vulnerabilities to attack?

- a) denial of service
- b) port scanning
- c) memory access violation
- d) dumpster diving

[View Answer](#)

Answer: b

Explanation: None.

8. File virus attaches itself to the _____

- a) source file
- b) object file
- c) executable file
- d) all of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. Multipartite viruses attack on _____

- a) files
- b) boot sector
- c) memory

d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

10. In asymmetric encryption _____

- a) same key is used for encryption and decryption
- b) different keys are used encryption and decryption
- c) no key is required for encryption and decryption
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

1. What is true regarding 'Fence'?

- a) Its a method to confine users to one side of a boundary
- b) It can protect Operating system from one user
- c) It cannot protect users from each other
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

2. What is not true regarding 'Fence'?

- a) It is implemented via hardware register
- b) It doesn't protect users from each other
- c) It good to protect OS from abusive users
- d) Its implementation is unrestricted and can take any amount of space in Operating system.

[View Answer](#)

Answer: d

Explanation: None.

3. What is correct regarding 'relocation' w.r.t protecting memory?

- a) It is a process of taking a program as if it began at address 0
- b) It is a process of taking a program as if it began at address OA
- c) Fence cannot be used within relocation process
- d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. How can fence and relocation be used together?

- a) To each program address, the contents of fence register are added
- b) To contents of fence register is subtracted from actual address of program
- c) To each program address, the contents of fence register are not added
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: This both relocates the address and guarantees that no one can access a location lower than a fence address.

5. What is the basic need in protecting memory in multi-user environment?

- a) We need two registers one 'start' and other 'end'
- b) We need a variable register
- c) A fence register has to be used known as base register.
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. What is the role of base/bound registers?

- a) They give starting address to a program
- b) Program's addresses are neatly confined to space between the base and the bound registers
- c) They provide encrypted environment
- d) This technique doesn't protect a program's address from modification by another user

[View Answer](#)

Answer: b

Explanation: None.

7. What is all-or-nothing situation for sharing in memory?

- a) Program makes all its data available to be accessed
- b) It prohibits access to some
- c) It creates rules who can access program memory
- d) It separates program memory and data memory

[View Answer](#)

Answer: a

Explanation: None.

8. How is disadvantage of all-or-nothing approach overcome?

- a) Base/Bound
- b) Relocation technique
- c) Fence method
- d) Tagged architecture

[View Answer](#)

Answer: d

Explanation: None.

9. What is true regarding tagged architecture?

- a) Every word of machine memory has one or more extra bits
- b) Extra bits are used to do padding
- c) Extra bits are not used to identify rights to that word
- d) It is very compatible to code upgrades

[View Answer](#)

Answer: a

Explanation: None.

10. What is best solution to have effect of unbounded number if base/bound registers?

- a) Tagged architecture
- b) Segmentation
- c) Fence method
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

11. What is a major feature of segmentation?

- a) Program is divided in data memory and program memory
- b) Program is executed in segments
- c) Program is divided into pieces having different access rights
- d) It has effect of an unbounded architecture

[View Answer](#)

Answer: c

Explanation: None.

12. What is the correct way the segmentation program address is stored?

- a) name, offset
- b) start, stop
- c) access, rights
- d) offset, rights

[View Answer](#)

Answer: a

Explanation: OS can retrieve the real address via looking for the table then making a simple calculation: address of the name + offset.

13. What is the main objective of protection?

- a) Ensure all objects are protected individually
- b) Objects have different priority and thus different levels of protection
- c) Ensure that each object is accessed correctly and only by allowed processes
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

14. What is the principle of least privilege?

- a) Less privileges provide difficulty in executing admin tasks
- b) Users can get temporary high privilege access
- c) Users should be given just enough privileges to perform their tasks
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

15. What is the need of protection?

- a) Prevent mischievous violation
- b) Prevent and intentional
- c) Ensure that each program component uses resources allotted to it only
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

1. What are the incorrect methods of revocation of access rights?

- a) Immediate/Delayed

- b) Selective/General
- c) Partial/total
- d) Crucial

[View Answer](#)

Answer: d

Explanation: None.

2. Why is it difficult to revoke capabilities?

- a) They are too many
- b) They are not defined precisely
- c) They are distributed throughout the system
- d) None of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

3. What is the reacquisition scheme to revoke capability?

- a) When a process capability is revoked then it won't be able to reacquire it
- b) Pointers are maintained for each object which can be used to revoke
- c) Indirect pointing is done to revoke object's capabilities
- d) Master key can be used compare and revoke.

[View Answer](#)

Answer: a

Explanation: None.

4. What is false regarding Back-Pointers scheme to revoke capability?

- a) List of pointers is maintained with each object
- b) When revocation is required these pointers are followed
- c) This scheme is not adopted in MULTICS system
- d) These point to all capabilities associated with that object

[View Answer](#)

Answer: c

Explanation: None.

5. What is true about Indirection to revoke capability?

- a) Capabilities point indirectly to the objects
- b) Each capability will not have a unique entry in global
- c) Table entries cannot be reused for other capabilities

d) This system was adopted in MULTICS system

[View Answer](#)

Answer: a

Explanation: None.

advertisement

6. How can Keys be defined or replaced?

- a) create [keyname] [bits]
- b) set-key
- c) Key
- d) MAKE [Key Name]

[View Answer](#)

Answer: b

Explanation: None.

7. What are the characteristics of the Hydra system?

- a) It consists of known access rights and interpreted by the system
- b) A user can of protection system can declare other rights
- c) Hydra system is not flexible
- d) Hydra doesn't provide rights amplification

[View Answer](#)

Answer: a

Explanation: None.

8. What are the characteristics of rights amplification in Hydra?

- a) This scheme allows a procedure to be certified as trustworthy
- b) Amplification of rights cannot be stated explicitly in declaration
- c) It includes kernel rights such as read
- d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. What is the problem of mutually suspicious subsystem?

- a) Service program can steal users data
- b) Service program can malfunction and retain some rights of data provided by user
- c) Calling program can get access to restricted portion from service program

d) Calling program gets unrestricted access

[View Answer](#)

Answer: b

Explanation: Both calling program and service program are vulnerable to access each others private data/rights.

10. What are the characteristics of Cambridge CAP system as compared to Hydra system?

- a) It is simpler and less powerful than hydra system
- b) It is more powerful than hydra system
- c) It is powerful than hydra system
- d) It is not as secure as Hydra system

[View Answer](#)

Answer: a

Explanation: None.

11. What are the two capabilities defined in CAP system?

- a) data & software capability
- b) address & data capability
- c) hardware & software capability
- d) software capability

[View Answer](#)

Answer: a

Explanation: None.

1. In distributed system, each processor has its own _____

- a) local memory
- b) clock
- c) both local memory and clock
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

2. If one site fails in distributed system then _____

- a) the remaining sites can continue operating
- b) all the sites will stop working
- c) directly connected sites will stop working

d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

3. Network operating system runs on _____

- a) server
- b) every system in the network
- c) both server and every system in the network
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

4. Which technique is based on compile-time program transformation for accessing remote data in a distributed-memory parallel system?

- a) cache coherence scheme
- b) computation migration
- c) remote procedure call
- d) message passing

[View Answer](#)

Answer: b

Explanation: None.

5. Logical extension of computation migration is _____

- a) process migration
- b) system migration
- c) thread migration
- d) data migration

[View Answer](#)

Answer: a

Explanation: None.

6. Processes on the remote systems are identified by _____

- a) host ID
- b) host name and identifier
- c) identifier

d) process ID

[View Answer](#)

Answer: b

Explanation: None.

7. Which routing technique is used in a distributed system?

- a) fixed routing
- b) virtual routing
- c) dynamic routing
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. In distributed systems, link and site failure is detected by _____

- a) polling
- b) handshaking
- c) token passing
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

9. The capability of a system to adapt the increased service load is called _____

- a) scalability
- b) tolerance
- c) capacity
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. Internet provides _____ for remote login.

- a) telnet
- b) http
- c) ftp
- d) rpc

[View Answer](#)

Answer: a

Explanation: None.

1. What is not true about a distributed system?

- a) It is a collection of processor
- b) All processors are synchronized
- c) They do not share memory
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. What are the characteristics of processor in distributed system?

- a) They vary in size and function
- b) They are same in size and function
- c) They are manufactured with single purpose
- d) They are real-time devices

[View Answer](#)

Answer: a

Explanation: None.

3. What are the characteristics of a distributed file system?

- a) Its users, servers and storage devices are dispersed
- b) Service activity is not carried out across the network
- c) They have single centralized data repository
- d) There are multiple dependent storage devices

[View Answer](#)

Answer: a

Explanation: None.

4. What is not a major reason for building distributed systems?

- a) Resource sharing
- b) Computation speedup
- c) Reliability
- d) Simplicity

[View Answer](#)

Answer: d

Explanation: None.

5. What are the types of distributed operating system?

- a) Network Operating system
- b) Zone based Operating system
- c) Level based Operating system
- d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

6. What are characteristic of Network Operating Systems?

- a) Users are aware of multiplicity of machines
- b) They are transparent
- c) They are simple to use
- d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. How is access to resources of various machines is done?

- a) Remote logging using ssh or telnet
- b) Zone are configured for automatic access
- c) FTP is not used
- d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. What are the characteristics of Distributed Operating system?

- a) Users are aware of multiplicity of machines
- b) Access is done like local resources
- c) Users are aware of multiplicity of machines
- d) They have multiple zones to access files

[View Answer](#)

Answer: b

Explanation: None.

9. What are the characteristics of data migration?

- a) transfer data by entire file or immediate portion required
- b) transfer the computation rather than the data

- c) execute an entire process or parts of it at different sites
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. What are the characteristics of computation migration?
- a) transfer data by entire file or immediate portion required
 - b) transfer the computation rather than the data
 - c) execute an entire process or parts of it at different sites
 - d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

11. What are the characteristics of process migration?
- a) transfer data by entire file or immediate portion required
 - b) transfer the computation rather than the data
 - c) execute an entire process or parts of it at different sites
 - d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

1. What are the parts of network structure?

- a) Workstation
- b) Gateway
- c) Laptop
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

2. What is a valid network topology?

- a) Multiaccess bus
- b) Ring
- c) Star
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. What are sites in network topology compared?

- a) Basic cost
- b) Communication cost
- c) Reliability
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

4. Which design features of a communication network are important?

- a) Naming and name resolution
- b) Routing strategies
- c) Connection strategies
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. What are the characteristics of Naming and Name resolution?

- a) name systems in the network
- b) address messages with the process-id
- c) virtual circuit
- d) message switching

[View Answer](#)

Answer: b

Explanation: None.

6. What are routing strategies which is not used in distributed systems?

- a) Fixed routing
- b) Token routing
- c) Virtual circuit
- d) Dynamic routing

[View Answer](#)

Answer: c

Explanation: None.

7. What are the connection strategies not used in distributed systems?

- a) Circuit switching
- b) Message switching
- c) Token switching
- d) Packet switching

[View Answer](#)

Answer: c

Explanation: None.

8. How are collisions avoided in network?

- a) Carrier sense with multiple access (CSMA); collision detection (CD)
- b) Carrier sense multiple access with collision avoidance
- c) Message slots
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. What is a common problem found in distributed system?

- a) Process Synchronization
- b) Communication synchronization
- c) Deadlock problem
- d) Power failure

[View Answer](#)

Answer: c

Explanation: None.

1. How many layers does the Internet model ISO consist of?

- a) Three
- b) Five
- c) Seven
- d) Eight

[View Answer](#)

Answer: c

Explanation: None.

2. Which layer is responsible for The process-to-process delivery?

- a) Network
- b) Transport

c) Application

d) Physical

[View Answer](#)

Answer: b

Explanation: None.

3. Which layer is the layer closest to the transmission medium?

a) Physical

b) Data link

c) Network

d) Transport

[View Answer](#)

Answer: a

Explanation: None.

4. Header are _____ when data packet moves from upper to the lower layers?

a) Modified

b) Removed

c) Added

d) All of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

5. Which layer lies between the transport layer and data link layer?

a) Physical

b) Network

c) Application

d) Session

[View Answer](#)

Answer: b

Explanation: None.

6. Which of the following is an application layer service?

a) Mail service

b) File transfer

c) Remote access

d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. What are the different ways distributed may suffer?

- a) Failure of a link
- b) Failure of a site
- c) Loss of message
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. What are design issues in distributed system structure?

- a) Scalability
- b) Fault-tolerance
- c) Clustering
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. In which OSI layer encryption and decryption happens?

- a) Application
- b) Presentation
- c) Transport
- d) Data Link

[View Answer](#)

Answer: b

Explanation: None.

10. What are the important steps followed when recovery from failure happens?

- a) Post repairing integration with main system should happen smoothly and gracefully
- b) Upon link failure both parties at end must not be notified
- c) Fault recovery system must me adjusted
- d) Failures are logged systematically

[View Answer](#)

Answer: a

Explanation: None.

1. What are the different ways in which clients and servers are dispersed across machines?

- a) Servers may not run on dedicated machines
- b) Servers and clients can be on same machines
- c) Distribution cannot be interposed between a OS and the file system
- d) OS cannot be distributed with the file system a part of that distribution

[View Answer](#)

Answer: b

Explanation: None.

2. What are not the characteristics of a DFS?

- a) login transparency and access transparency
- b) Files need not contain information about their physical location
- c) No Multiplicity of users
- d) No Multiplicity if files

[View Answer](#)

Answer: c

Explanation: None.

3. What are characteristic of a DFS?

- a) Fault tolerance
- b) Scalability
- c) Heterogeneity of the system
- d) Upgradation

[View Answer](#)

Answer: d

Explanation: None.

4. What are the different ways file accesses take place?

- a) sequential access
- b) direct access
- c) indexed sequential access
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

5. Which is not a major component of a file system?

- a) Directory service

- b) Authorization service
- c) Shadow service
- d) System service

[View Answer](#)

Answer: c

Explanation: None.

6. What are the different ways mounting of the file system?

- a) boot mounting
- b) auto mounting
- c) explicit mounting
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. What is the advantage of caching in remote file access?

- a) Reduced network traffic by retaining recently accessed disk blocks
- b) Faster network access
- c) Copies of data creates backup automatically
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

8. What is networked virtual memory?

- a) Caching
- b) Segmentation
- c) RAM disk
- d) None of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. What are the examples of state information?

- a) opened files and their clients
- b) file descriptors and file handles
- c) current file position pointers

d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

10. Which is not an example of state information?

- a) Mounting information
- b) Description of HDD space
- c) Session keys
- d) Lock status

[View Answer](#)

Answer: b

Explanation: None.

1. What is a stateless file server?

- a) It keeps tracks of states of different objects
- b) It maintains internally no state information at all
- c) It maintains some information in them
- d) None of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. What are the characteristics of the stateless server?

- a) Easier to implement
- b) They are not fault-tolerant upon client or server failures
- c) They store all information file server
- d) They are redundant to keep data safe

[View Answer](#)

Answer: a

Explanation: None.

3. Implementation of a stateless file server must not follow?

- a) Idempotency requirement
- b) Encryption of keys
- c) File locking mechanism
- d) Cache consistency

[View Answer](#)

Answer: b

Explanation: None.

4. What are the advantages of file replication?

- a) Improves availability & performance
- b) Decreases performance
- c) They are consistent
- d) Improves speed

[View Answer](#)

Answer: a

Explanation: None.

5. What are characteristic of NFS protocol?

- a) Search for file within directory
- b) Read a set of directory entries
- c) Manipulate links and directories
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

6. What is the coherency of replicated data?

- a) All replicas are identical at all times
- b) Replicas are perceived as identical only at some points in time
- c) Users always read the most recent data in the replicas
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

7. What are the three popular semantic modes?

- a) Unix, Coherent & Session semantics
- b) Unix, Transaction & Session semantics
- c) Coherent, Transaction & Session semantics
- d) Session, Coherent semantics

[View Answer](#)

Answer: b

Explanation: None.

8. What are the characteristics of Unix semantics?

- a) Easy to implement in a single processor system
- b) Data cached on a per process basis using write through case control
- c) Write-back enhances access performance
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. What are the characteristics of transaction semantics?

- a) Suitable for applications that are concerned about coherence of data
- b) The users of this model are interested in the atomicity property for their transaction
- c) Easy to implement in a single processor system
- d) Write-back enhances access performance

[View Answer](#)

Answer: b

Explanation: None.

10. What are non characteristics of session semantics?

- a) Each client obtains a working copy from the server
- b) When file is closed, the modified file is copied to the file server
- c) The burden of coordinating file sharing is ignored by the system
- d) Easy to implement in a single processor system

[View Answer](#)

Answer: d

Explanation: None.

1. The file once created can not be changed is called _____

- a) immutable file
- b) mutex file
- c) mutable file
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

2. _____ of the distributed file system are dispersed among various machines of distributed system.

- a) Clients
- b) Servers
- c) Storage devices
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

3. _____ is not possible in distributed file system.

- a) File replication
- b) Migration
- c) Client interface
- d) Remote access

[View Answer](#)

Answer: b

Explanation: None.

4. Which one of the following hides the location where in the network the file is stored?

- a) transparent distributed file system
- b) hidden distributed file system
- c) escaped distribution file system
- d) spy distributed file system

[View Answer](#)

Answer: a

Explanation: None.

5. In a distributed file system, when a file's physical storage location changes

-
- a) file name need to be changed
 - b) file name need not to be changed
 - c) file's host name need to be changed
 - d) file's local name need to be changed

[View Answer](#)

Answer: b

Explanation: None.

6. In a distributed file system, _____ is mapping between logical and physical objects.

- a) client interfacing
- b) naming
- c) migration
- d) heterogeneity

[View Answer](#)

Answer: b

Explanation: None.

7. In a distributed file system, a file is uniquely identified by _____

- a) host name
- b) local name
- c) the combination of host name and local name
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

8. There is no need to establish and terminate a connection through open and close operation in _____

- a) stateless file service
- b) stateful file service
- c) both stateless and stateful file service
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

9. In distributed file system, file name does not reveal the file's _____

- a) local name
- b) physical storage location
- c) both local name and physical storage location
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

10. Which one of the following is a distributed file system?

- a) andrew file system
- b) network file system

- c) novel network
- d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

1. What are the characteristics of tightly coupled system?

- i) Same clock, usually shared memory
 - ii) Communication is via this shared memory
 - iii) Multiprocessors
 - iv) Different clock
- a) i
 - b) i, ii and iii
 - c) ii and iii
 - d) i, iii and iv

[View Answer](#)

Answer: b

Explanation: None.

2. What are the characteristics of tightly coupled system?

- i) Different clock
 - ii) Use communication links
 - iii) Same clock
 - iv) Distributed systems
- a) i
 - b) i and iv
 - c) i, ii and iii
 - d) ii, iii and iv

[View Answer](#)

Answer: d

Explanation: None.

3. What are the characteristics of mutual exclusion using centralized approach?

- a) One processor as coordinator which handles all requests
- b) It requires request, reply and release per critical section entry
- c) The method is free from starvation
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

4. What are the characteristics of fully distributed approach?

- i) When responses are received from all processes, then process can enter its Critical Section
 - ii) When process exits its critical section, the process sends reply messages to all its deferred requests.
 - iii) It requires request, reply and release per critical section entry
 - iv) One processor as coordinator which handles all requests
- a) i
 - b) i and ii
 - c) ii and iii
 - d) iii and iv

[View Answer](#)

Answer: b

Explanation: None.

5. What are the advantages of token(with rings) passing approach?

- i) One processor as coordinator which handles all requests
 - ii) No starvation if the ring is unidirectional
 - iii) There are many messages passed per section entered if few users want to get in section
 - iv) One processor as coordinator which handles all requests
 - v) Only one message/entry if everyone wants to get in
- a) i
 - b) ii and iii
 - c) i, ii and iii
 - d) i, ii and iv

[View Answer](#)

Answer: d

Explanation: None.

6. What are the characteristics of atomicity?

- a) All operations associated are executed to completion or none are performed
- b) One processor as coordinator which handles all requests
- c) When responses are received from all processes, then the process can enter its Critical Section
- d) Use communication links

[View Answer](#)

Answer: a

Explanation: None.

7. What things are the transaction coordinator responsible for?

- a) Starting the execution of the transaction
- b) Breaking transaction into a number of subtransactions
- c) Coordinating the termination of the transaction
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. Which of the following advantages follows the single coordinator approach?

- a) Simple implementation
- b) Simple deadlock handling
- c) bottleneck
- d) All of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

9. Which of the following disadvantages follows the single coordinator approach?

- a) Bottleneck
- b) Slow response
- c) Deadlock
- d) One request per second

[View Answer](#)

Answer: a

Explanation: None.

10. What are the disadvantages of majority protocol?

- a) Complicated implementation
- b) Deadlock cannot occur easily
- c) Bottleneck
- d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

11. What are the parts of a global unique identifier?

- a) Local unique timestamp
- b) Remote timestamp
- c) Clock number
- d) All of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

12. Which are the two complementary deadlock-prevention schemes using timestamps?

- a) The wait-die & wound-wait scheme
- b) The wait-n-watch scheme
- c) The wound-wait scheme
- d) The wait-wound & wound-wait scheme

[View Answer](#)

Answer: a

Explanation: None.

1. In distributed systems, a logical clock is associated with _____

- a) each instruction
- b) each process
- c) each register
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

2. If timestamps of two events are same, then the events are _____

- a) concurrent
- b) non-concurrent
- c) monotonic
- d) non-monotonic

[View Answer](#)

Answer: a

Explanation: None.

3. If a process is executing in its critical section _____

- a) any other process can also execute in its critical section

- b) no other process can execute in its critical section
- c) one more process can execute in its critical section
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

4. A process can enter into its critical section _____

- a) anytime
- b) when it receives a reply message from its parent process
- c) when it receives a reply message from all other processes in the system
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

5. For proper synchronization in distributed systems _____

- a) prevention from the deadlock is must
- b) prevention from the starvation is must
- c) prevention from the deadlock & starvation is must
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

6. In the token passing approach of distributed systems, processes are organized in a ring structure _____

- a) logically
- b) physically
- c) both logically and physically
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

7. In distributed systems, what will the transaction coordinator do?

- a) starts the execution of transaction
- b) breaks the transaction into number of sub transactions
- c) coordinates the termination of the transaction

d) all of the mentioned

[View Answer](#)

Answer: d

Explanation: None.

8. In case of failure, a new transaction coordinator can be elected by _____

- a) bully algorithm
- b) ring algorithm
- c) both bully and ring algorithm
- d) none of the mentioned

[View Answer](#)

Answer: c

Explanation: None.

9. In distributed systems, election algorithms assumes that _____

- a) a unique priority number is associated with each active process in system
- b) there is no priority number associated with any process
- c) priority of the processes is not required
- d) none of the mentioned

[View Answer](#)

Answer: a

Explanation: None.

10. According to the ring algorithm, links between processes are _____

- a) bidirectional
- b) unidirectional
- c) both bidirectional and unidirectional
- d) none of the mentioned

[View Answer](#)

Answer: b

Explanation: None.

UNIT 1
INTRODUCTION
PART A

MULTIPLE CHOICE QUESTIONS:

1. What is operating system?(TB-2 p.no 48) [L1]
 - a) collection of programs that manages hardware resources
 - b) System service provider to the application programs
 - c) link to interface the hardware and application programs
 - d) Link to interface the hardware only

Answer: b
2. To access the services of operating system, interface is provided by the (TB-2 p.no 50) [L1]
 - a) system calls
 - b) API
 - c) library
 - d) assembly instructions

Answer: a
3. Which one of the following is not true?(TB-2 p.no 71) [L1]
 - a) kernel is the program that constitutes the central core of the operating system
 - b) kernel is the first part of operating system to load into memory during booting
 - c) kernel is made of various modules which cannot be loaded in running operating system
 - d) kernel remains in the memory during the entire computer session

Answer: c
4. Which one of the following error will be handle by the operating system?(TB-2 p.no 49) [L1]
 - a) power failure, connection failure in the network
 - b) Protocol problem
 - c) Memory access
 - d) ISA

Answer: d
5. The ISA Define the repertoire of ___ Language instruction that a computer can allows(TB-2 p.no:51) [L1]
 - a) Machine
 - b) Binary
 - c) Python
 - d) C

Answer: a
6. By operating system, the resource management can be done via(TB-2 p.no:50) [L1]
 - a) time division multiplexing
 - b) space division multiplexing
 - c) both (a) and (b)
 - d) Time Sharing

Answer: c

7. If a process fails, most operating system write the error information to a(TB-2 p.no 52) [L1]

- a) log file
- b) Another running process
- c) New file
- d) Batch file

Answer: a

8. What are the major problems in Serial Processing (TB-2 p.no:52) [L1]

- a) Scheduling
- b) Dlocate
- c) Dmap
- d) Dadd

Answer: a

9. Which one of the following is not a real time operating system?(TB-2 p.no 80) [L1]

- a) VxWorks
- b) Windows CE
- c) RTLinux
- d) Palm OS

Answer: d

10. The OS X has(PgNo:71) [L1]

- a) monolithic kernel
- b) hybrid kernel
- c) microkernel
- d) monolithic kernel with modules

Answer: b

11. The systems which allows only one process execution at a time, are called(TB-2 p.no 57) [L1]

- a) uniprogramming systems
- b) Uniprocessing systems
- c) unit processing systems
- d) unique programming systems

Answer: a

12. In operating system, a user program execute in a ___ mode(TB-2 p.no:56) [L1]

- a) Kernel Mode
- b) System Mode
- c) Server Mode
- d) User Mode

Answer: d

13. A single thread of control allows the process to perform(TB-2 p.no:71) [L1]

- a) only one task at a time
- b) multiple tasks at a time
- c) Two Task at a time
- d) No Process

Answer: a

14. The objective of multi-programming is to : (TB-2 p.no 58) [L1]

- a) Have some process running at all times
 - b) Have multiple programs waiting in a queue ready to run

 - c) To minimize CPU utilization
 - d) To maximize CPU utilization
- Answer: a and d
15. The Symmetric Multiprocessor is in one of the part of ____?(TB-2 p.no 77) [L1]
- a) Multiprocessor
 - b) PCB queue
 - c) Multi-Core
 - d) Multiprogramming
- Answer: a
16. Which one is managing thread Scheduling :(PgNo:83) [L1]
- a) Processor
 - b) Kernel
 - c) Process
 - d) Thread
- Answer: b
17. In operating system, a user program execute in a ___ mode?(TB-2 p.no 56) [L1]
- a) User Mode
 - b) Kernel Mode
 - c) Server Mode
 - d) Program mode
- Answer: b
18. What is an ISA ?(PgNo:50) [L1]
- a) Information Service Architecture
 - b) Interrupt Service Architecture
 - c) Instruction Set Architecture
 - d) Information Service Architecture
- Answer: c
19. Protecting of data from unauthorized modification(TB-2 p.no 19) [L1]
- a) Data Integrity
 - b) Confidentiality
 - c) Interrupt
 - d) Availability
- Answer: a
20. Multi-Core is used for : (TB-2 p.no 78) [L1]
- a) Shared Memory Access
 - b) Individual Memory Access
 - c) USB
 - d) Utilizing CPU cycles
- Answer: b
21. CPU scheduling is the basis of _____.(TB-2 p.no 58) [L1]
- a) multiprocessor systems

- b) Multiprogramming operating systems
- c) larger memory sized systems
- d) smaller memory sized systems

Answer: b

22. With multiprogramming, _____ is used productively.(TB-2 p.no 57) [L1]
- a) time
 - b) space
 - c) money
 - d) memory

Answer: a

23. The two steps of a process execution are ____ (TB-2 p.no 62) [L1]
- a) Input Burst
 - b) I/O & CPU Burst
 - c) Memory Burst
 - d) OS Burst

Answer: b

24. In which evolution stages we are not Using any other software (TB-2 p.no 52) [L1]
- a) Serial Processing
 - b) Simple Batch Processing
 - c) Multiprocessing
 - d) Timesharing

Answer: a

25. A process is selected from the _____ queue by the _____ scheduler, to be executed.(TB-2 p.no 62) [L1]
- a) blocked, short term
 - b) wait, long term
 - c) ready, short term
 - d) ready, long term

Answer: c

26. A process can be _____ (PgNo:163) [L1]
- a) single threaded
 - b) multithreaded
 - c) both single threaded and multithreaded
 - d) Simple threaded

Answer: c

27. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called(PgNo:111) [L1]
- a) job queue
 - b) ready queue
 - c) execution queue
 - d) process queue

Answer: b

28. The time required to create a new thread in an existing process is _____
(PgNo:164) [L1]
- a) greater than the time required to create a new process

- b) less than the time required to create a new process
- c) equal to the time required to create a new process
- d) Less than or equal to the time required to create a new process
- e)

Answer: b

29. A parent process calling _____ system call will be suspended until children processes terminate. (PgNo:115) [L1]

- a) Wait
- b) Fork
- c) Exit
- d) exec

Answer: a

30. In UNIX, each process is identified by its _____. (PgNo:107) [L1]

- a) Process Control Block
- b) Device Queue
- c) Process Identifier
- d) Program identifier

Answer: c

31. The child processes can _____. (PgNo:118) [L1]

- a) be a duplicate of the parent process
- b) never be a duplicate of the parent process
- c) cannot have another program loaded into it
- d) never have another program loaded into it

Answer: c

32. The child process completes execution, but the parent keeps executing, then the child process is known as _____. (PgNo:118) [L1]

- a) Orphan
- b) Zombie
- c) Body
- d) Dead

Answer: b

33. A process is selected from the _____ queue by the _____ scheduler, to be executed. (PgNo:107) [L1]

- a) blocked, short term
- b) wait, long term
- c) ready, short term
- d) ready, long term

Answer: c

34. In the following cases non – preemptive scheduling occurs : (Choose two) (PgNo:114) [L1]

- a) When a process switches from the running state to the ready state
- b) When a process goes from the running state to the waiting state
- c) When a process switches from the waiting state to the ready state
- d) When a process terminates

Answer: b and d

35. The switching of the CPU from one process or thread to another is called (PgNo:163)

[L1]

- a) process switch
- b) task switch
- c) context switch
- d) All of these

Answer: d

36. Dispatch latency is (PgNo:265) [L1]

- a) the speed of dispatching a process from running to the ready state
- b) the time of dispatching a process from running to ready state and keeping the CPU idle
- c) the time to stop one process and start running another one
- d) the time of processing a process from running to ready state and keeping the CPU idle

Answer: c

37. The Process Control Block is (PgNo:110) [L1]

- a) Process type variable
- b) Data Structure
- c) a secondary storage section
- d) a Block in memory

Answer: b

38. The entry of all the PCBs of the current processes is in (PgNo:111) [L1]

- a) Process Register
- b) Program Counter
- c) Process Table
- d) Process Unit

Answer: c

39. If all processes I/O bound, the ready queue will almost always be _____, and the Short term Scheduler will have a _____ to do.(PgNo:112) [L1]

- a) full, little
- b) full, lot
- c) empty, little
- d) empty, lot

Answer: c

40. Concurrent access to shared data may result in _____ (PgNo:203) [L1]

- a) data consistency
- b) data insecurity
- c) data inconsistency
- d) data security

Answer: c

41. Mutual exclusion implies that _____ (PgNo:206) [L1]

- a) if a process is executing in its critical section, then no other process must be executing in their critical sections
- b) if a process is executing in its critical section, then other processes must be executing in their critical sections

- c) if a process is executing in its critical section, then all the resources of the system must be blocked until it finishes execution
- d) if a process is not executing in its critical section, then other processes must be executing in their critical sections

Answer: a

42. A minimum of _____ variable(s) is/are required to be shared between processes to solve the critical section problem. (PgNo:204) [L1]

- a) One
- b) Two
- c) Three
- d) Four

Answer: b

43. Which one of the following is a synchronization tool? (PgNo:203) [L1]

- a) Thread
- b) Pipe
- c) Semaphore
- d) Socket

Answer: c

44. A semaphore is a shared integer variable _____(PgNo:201) [L1]

- a) that cannot drop below zero
- b) that cannot be more than zero
- c) that cannot drop below one
- d) that cannot be more than one

Answer: a

45. Which process can be affected by other processes executing in the system? (PgNo:112) [L1]

- a) cooperating process
- b) child process
- c) parent process
- d) init process

Answer: a

46. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called _____(PgNo:204) [L1]

- a) dynamic condition
- b) race condition
- c) essential condition
- d) critical condition

Answer: b

47. If a process is executing in its critical section, then no other processes can be executing in their critical section. This condition is called _____(PgNo:205) [L1]

- a) mutual exclusion
- b) critical exclusion
- c) synchronous exclusion
- d) asynchronous exclusion

Answer: a

48. Messages sent by a process _____ (PgNo:122) [L1]

- a) have to be of a fixed size
- b) have to be a variable size
- c) can be fixed or variable sized
- d) cannot be fixed

Answer: c

49. Message passing system allows processes to _____ (PgNo:124) [L1]

- a) communicate with one another without resorting to shared data
- b) communicate with one another by resorting to shared data
- c) share data
- d) name the recipient or sender of the message

Answer: a

50. Which of the following two operations are provided by the IPC facility? (PgNo:123) [L1]

- a) write & delete message
- b) delete & receive message
- c) send & delete message
- d) receive & send message

Answer: d

PART B

4 MARKS:

1. Define operating system and its role. (TB-2 p.no:48) [L1]

- An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.
- Role of Operating System:
- A computer is a set of resources for moving, storing, & processing data
- The OS is responsible for managing these resources
- The OS exercises its control through software

2. Write the Objective and functions of an OS. (TB-2 p.no 49) [L1]

Objective:

- Convenience
- Efficiency
- Ability to Evolve

Functions:

- Memory Management.
- Processor management.
- Interrupt Handling.
- Accounting.
- Automatic job sequencing.
- Management and control of I/O devices

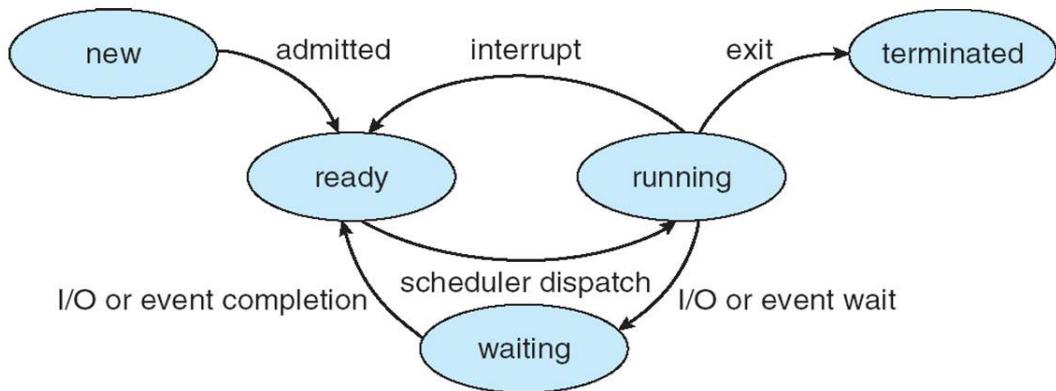
3. What is the need for an OS and its characteristics and services? (TB-2 p.no:49) [L1]

- A medium is needed to communicate between the user and the hardware.
 - An OS acts as a medium of interface. The characteristics of an OS are
 - User friendly.
 - Allows sharing of resources (H/W and S/W).
 - Provides adequate security.
 - Protection.
4. What are the methods we are following in Evolution of OS? (TB-2 p.no 52) [L1]
- Serial Processing
 - Simple Batch Systems
 - Multi programmed Batch Systems
 - Time-Sharing Systems
5. Give the importance problems about Serial processing. (TB-2 p.no:52) [L1]
1. **Scheduling:**
 - Most installations used a hardcopy sign-up sheet to reserve computer time.
 - Time allocations could run short or long, resulting in wasted computer time
 2. **Setup time**
 - A considerable amount of time was spent just on setting up the program to run
6. What is the Simple Batch System? (TB-2 p.no 53) [L1]
- Early computers were very expensive
 - Important to maximize processor utilization
- Monitor Point of View**
- User no longer has direct access to processor
 - Job is submitted to computer operator who batches them together and places them on an input device
 - Program branches back to the monitor when finished
- Processor Point of View**
- Processor executes instruction from the memory containing the monitor
 - Executes the instructions in the user program until it encounters an ending or error condition
 - “control is passed to a job” means processor is fetching and executing instructions in a user program
 - “control is returned to the monitor” means that the processor is fetching and executing instructions from the monitor program
7. Define Multi Processing with its types. (TB-2 p.no:77) [L1]
- The Simultaneous Processing of a number of Processes by a number of Processors Simultaneously at the same time is Multi Processing.
 - A Multi-Processing System is one in which there are more than one CPU, interleaved with each other.
 - So it helps in improving the amount of work done. i) Symmetric Multi-Processing
ii) Asymmetric Multi Processing. It is one in which each processor runs an

identical copy of the OS and these copies communicate with one another as needed.

- It is one in which each processor is assigned a specific task. A Master Processor controls the system and the other Processors are allocated work by the Master Processor

8. Draw the diagrammatic representation of various process states. (PgNo:105) [L1]



9. How does a process differ from a job? (PgNo:107) [L1]

- Program is **passive** entity stored on disk (**executable file**), process is **active**
- Program becomes process when executable file loaded into memory
- Execution of program started via GUI mouse clicks, command line entry of its name, etc
- One program can be several processes
- Consider multiple users executing the same program

10. Define Process Control Block. (PgNo:109) [L2]

Information associated with each process (also called **task control block**)

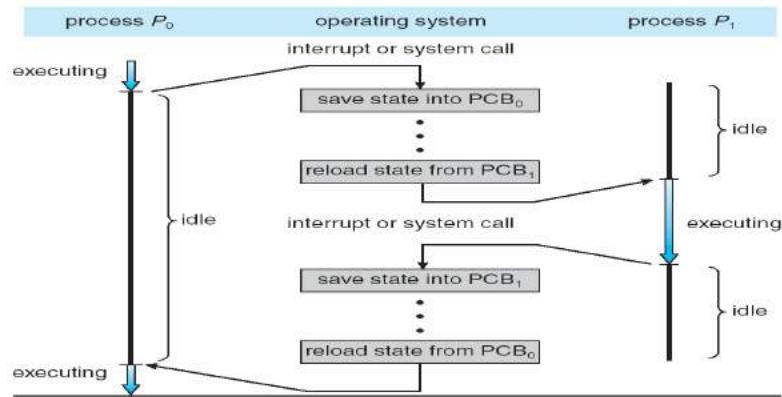
- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files

11. Formulate the relationship between threads and processes.(PgNo:163) [L2]

- So far, process has a single thread of execution
- Consider having multiple program counters per process
- Multiple locations can execute at once
- Multiple threads of control -> **threads**

- Must then have storage for thread details, multiple program counters in PCB

12. Argue how one process is switching to another? (PgNo:111) [L3]



13. List the operations involved in creating a process. (PgNo:115) [L2]

- Parent** process create **children** processes, which, in turn create other processes, forming a **tree of processes**
- Generally, process identified and managed via a **process identifier (pid)**
- Resource sharing options
- Parent and children share all resources
- Children share subset of parent's resources
- Parent and child share no resources
- Execution options
- Parent and children execute concurrently
- Parent waits until children terminate

14. Mention and define various types of processor scheduler? (PgNo:112)[L1]

- Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU. Sometimes the only scheduler in a system
Short-term scheduler is invoked frequently (milliseconds) \Rightarrow (must be fast)
- Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue. Long-term scheduler is invoked infrequently (seconds, minutes) \Rightarrow (may be slow). The long-term scheduler controls the **degree of multiprogramming**
- Medium-term scheduler** can be added if degree of multiple programming needs to decrease Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**

15. Differentiate the Interprocess communication that includes Shared memory and Message passing?(PgNo:113) [L2]

- An area of memory shared among the processes that wish to communicate
- The communication is under the control of the user processes not the operating system.
- Major issues is to provide mechanism that will allow the user processes to synchronize their actions when they access shared memory.
- Mechanism for processes to communicate and to synchronize their actions
- Message system – processes communicate with each other without resorting to shared variables
- IPC facility provides two operations:
- **send(message)**
- **receive(message)**
- The *message* size is either fixed or variable

16. Summarize about Race condition. (PgNo:204) [L1]

- Occurs when multiple processes or threads read and write shared data items
- The final result depends on the order of execution
- the “loser” of the race is the process that updates last and will determine the final value of the variable
- Assume P1 and P2 are executing this code and share the variable **a**
- Processes can be preempted at any time.
- Assume P1 is preempted after the input statement, and P2 then executes entirely
- The character echoed by P1 will be the one read by P2

17. State Critical Section Problem with example (PgNo:206) [L1]

- When a process executes code that manipulates shared data (or resources), we say that the process is in its critical section (CS) for that shared data
- We must enforce mutual exclusion on the execution of critical sections.
- Only one process at a time can be in its CS (for that shared data or resource).
- Each process must ask permission to enter critical section in **entry section**, may follow critical section with **exit section**, then **remainder section**

```
do {  
    entry section  
    critical section  
    exit section  
    remainder section  
} while (true);
```

18. Quote the solution for Critical Section Problem (PgNo:207) [L1]

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

- **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
- **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. Assume that each process executes at a nonzero speed. No assumption concerning **relative speed** of the n processes

19. How the critical section handled in Operating system? (PgNo:208) [L1]

Two approaches depending on if kernel is preemptive or non-preemptive

- **Preemptive** – allows preemption of process when running in kernel mode
- **Non-preemptive** – runs until exits kernel mode, blocks, or voluntarily yields CPU

Essentially free of race conditions in kernel mode

20. Classify Pipes and explain in detail (PgNo:136) [L2]

- Acts as a conduit allowing two processes to communicate
- Issues:
- Is communication unidirectional or bidirectional?
 - In the case of two-way communication, is it half or full-duplex?
 - Must there exist a relationship (i.e., **parent-child**) between the communicating processes?
 - Can the pipes be used over a network?
 - **Ordinary pipes** – cannot be accessed from outside the process that created it. Typically, a parent process creates a pipe and uses it to communicate with a child process that it created.
 - **Named pipes** – can be accessed without a parent-child relationship.

PART C

12 MARKS:

1. Explain the following
 - i. MultiProgrammed Batch System(TB-2 p.no 56)
 - ii. Time sharing system(TB-2 p.no 58)
2. Describe about Memory Management with example.(TB-2 p.no 66)
3. Elaborate in detail Scheduling and Resource management.(TB-2 p.no 69)
4. Discuss the evolution of operating system.(TB-2 p.no 52)
5. Elaborate multiprocessor and multicore architecture.(TB-2 p.no 77)
6. Elucidate the objectives and functions of operating system as a i) Interface ii) resource manager.(TB-2 p.no 48)

7. Examine the scheduler that how the process takes place and kept in queue with the help of context switch. (PgNo:105) [L2]
8. Interpret the following concepts (PgNo:117) [L2]
 - Process concept
 - PCB
 - Thread
9. Construct a process tree which consists of three parent nodes and three child nodes.
Perform process creation and termination among them. (PgNo:112) [L3]
10. Summarize process scheduling with various concepts. (PgNo:110) [L3]
11. Compare various system call with proper example program. (PgNo:118) [L3]
12. Design a server A and a client B that communicates with each other. Also how a shared memory working on it through the Message passing. (PgNo:122) [L3]
13. Justify how the concurrent access to a shared data can be handled with the help of process synchronization and validate through producer consumer problem and critical section problem.(PgNo:203) [L3]
14. Organize various Process synchronization problem with examples. (PgNo:203) [L3]

UNIT 2

PROCESS SYNCHRONIZATION,CPU SCHEDULING AND DEADLOCK

PART A

MULTIPLE CHOICE QUESTIONS:

1. Which process can be affected by other processes executing in the system?(Pgno:209)
[L1]

- a) cooperating process
- b) child process
- c) parent process
- d) init process

Answer : a

2. Peterson solution is restricted to ----- process that alternate execution between their critical section and remainder sections.(Pgno:206) [L2]

- a)one
- b) two
- c) three
- d) four

Answer : b

3. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called?(Pgno:211) [L1]

- a) dynamic condition
- b) race condition
- c) essential condition
- d) critical condition

Answer : b

4. Mutex is short for ----- (Pgno:206) [L1]

- a)Mutual Excluded
- b) mutually explained
- c) mutual exclusion
- d) mutual exception

Answer : c

5. If a process is executing in its critical section, then no other processes can be executing in their critical section. This condition is called?(Pgno:207) [L1]

- a) mutual exclusion
- b) critical exclusion

- c) synchronous exclusion
- d) asynchronous exclusion

Answer : a

6. Which one of the following is a synchronization tool?(Pgno:214) [1]
- a) thread
 - b) pipe
 - c) semaphore
 - d) socket

Answer : c

7. A semaphore is a shared integer variable _____(Pgno:214) [L1]
- a) that can not drop below zero
 - b) that can not be more than zero
 - c) that can not drop below one
 - d) that can not be more than one

Answer : a

8. Spinlocks are intended to provide----- (Pgno:213) [L1]
- a)mutual exclusion
 - b) bounded waiting
 - c) Aging
 - d) Progress

Answer : b

9. Mutual exclusion can be provided by the _____(Pgno:213) [L1]
- a) mutex locks
 - b) binary semaphores
 - c) both mutex locks and binary semaphores
 - d) binary lock

Answer : c

10. When high priority task is indirectly preempted by medium priority task effectively inverting the relative priority of the two tasks, the scenario is called
_____ (pg-no:216) [L1]
- a) priority inversion
 - b) priority removal
 - c) priority exchange
 - d) priority modification

Answer : a

11. Process synchronization can be done on _____(Pgno:209) [1]
- a) hardware level
 - b) software level

- c) both hardware and software level
- d) Process Level

Answer : c

12. Semaphore is a/an _____ to solve the critical section problem.(Pgno:215) [L1]

- a) hardware for a system
- b) special program for a system
- c) integer variable
- d) Software for a system

Answer : c

13. The wait operation of the semaphore basically works on the basic _____ system call.(Pgno:216) [L2]

- a) stop()
- b) block()
- c) hold()
- d) wait()

Answer : b

14. The signal operation of the semaphore basically works on the basic _____ system call.(Pgno:215) [L2]

- a) continue()
- b) wakeup()
- c) getup()
- d) start()

Answer : b

15. What are the two kinds of semaphores?(Pgno:214) [L1]

- a) mutex & counting
- b) binary & counting
- c) counting & decimal
- d) decimal & binary

Answer : b

16. At a particular time of computation the value of a counting semaphore is 7.Then 20 P operations and 15 V operations were completed on this semaphore. The resulting value of the semaphore is? [L3]

- a) 42
- b) 2
- c) 7
- d) 12

Answer : b

17. A binary semaphore is a semaphore with integer values _____(Pgno:214) [L2]

- a) 1
- b) -1
- c) 0.8
- d) 0.5

Answer : a

18. Spinlocks are intended to provide _____ only.(Pgno:213) [L2]

- a) Mutual Exclusion
- b) Bounded Waiting
- c) Aging
- d) Progress

Answer : b

19. The bounded buffer problem is also known as _____(pg-219) [L1]

- a) Readers – Writers problem
- b) Dining – Philosophers problem
- c) Producer – Consumer problem
- d) Critical Section Problem

Answer : c

20. In the bounded buffer problem _____(Pgno:219) [L2]

- a) there is only one buffer
- b) there are n buffers (n being greater than one but finite)
- c) there are infinite buffers
- d) the buffer size is bounded

Answer : b

21. The dining – philosophers problem will occur in case of _____(Pgno:222) [L2]

- a) 5 philosophers and 5 chopsticks
- b) 4 philosophers and 5 chopsticks
- c) 3 philosophers and 5 chopsticks
- d) 6 philosophers and 5 chopsticks

Answer : a

22. To ensure difficulties do not arise in the readers – writers problem _____ are given exclusive access to the shared object.(Pgno:220) [L1]

- a) readers
- b) writers
- c) readers and writers
- d) either reader or writers

Answer : b

23. A deadlock free solution to the dining philosophers problem _____(Pgno:223) [L2]

- a) necessarily eliminates the possibility of starvation
- b) does not necessarily eliminate the possibility of starvation
- c) eliminates any possibility of any kind of problem further
- d) does not eliminates any kind of problem

Answer : b

24. All processes share a semaphore variable mutex, initialized to 1. Each process must execute wait(mutex) before entering the critical section and signal(mutex) afterward. Suppose a process executes in the following manner.(Pgno:224) [L2]

Wait(mutex);

.....

Critical section

.....

Wait(mutex);

- a) deadlock will occur
- b) processes will starve to enter critical section
- c) several processes maybe executing in their critical section
- d) Critical section probem may occur

Answer : a

25. In the bounded buffer problem, there are the empty and full semaphores that _____(Pgno:219) [L2]

- a) count the number of empty and full buffers
- b) count the number of empty and full memory spaces
- c) count the number of empty and full queues
- d) count the number of empty and full process

Answer : a

26. Which module gives control of the CPU to the process selected by the short-term scheduler? (Pgno: 265) [L1]

- a) dispatcher
- b) interrupt
- c) scheduler
- d) context switch

Answer: a

27. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called _____ (Pgno: 267) [L1]

- a) job queue
- b) ready queue
- c) execution queue

d) process queue

Answer: b

28. The interval from the time of submission of a process to the time of completion is termed as _____ (Pgno: 265) [L1]

- a) waiting time
- b) turnaround time
- c) response time
- d) throughput

Answer: b

29. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first? (Pgno: 266) [L2]

- a) first-come, first-served scheduling
- b) shortest job scheduling
- c) priority scheduling
- d) Round robin scheduling

Answer: a

30. In priority scheduling algorithm _____ (Pgno: 270) [L1]

- a) CPU is allocated to the process with highest priority
- b) CPU is allocated to the process with lowest priority
- c) Equal priority processes can not be scheduled
- d) processes are allocated by order

Answer: a

31. In priority scheduling algorithm, when a process arrives at the ready queue, its priority is compared with the priority of _____ (Pgno: 270) [L2]

- a) all process
- b) currently running process
- c) parent process
- d) init process

Answer: b

32. Which algorithm is defined in Time quantum? (Pgno: 271) [L1]

- a) shortest job scheduling algorithm
- b) round robin scheduling algorithm
- c) priority scheduling algorithm
- d) multilevel queue scheduling algorithm

Answer: b

33. Process are classified into different groups in _____ (Pgno: 273) [L1]

- a) shortest job scheduling algorithm
- b) round robin scheduling algorithm
- c) priority scheduling algorithm
- d) multilevel queue scheduling algorithm

Answer: d

34. In multilevel feedback scheduling algorithm _____ (Pgno: 275) [L1]

- a) a process can move to a different classified ready queue
- b) classification of ready queue is permanent
- c) processes are not classified into groups

d) processes are classified into groups

Answer: a

35. Which one of the following cannot be scheduled by the kernel? (Pgno: 277) [L2]

- a) kernel level thread
- b) User level thread
- c) process
- d) process as well as kernel

Answer: b

36. With round robin scheduling algorithm in a time shared system _____ (Pgno: 272) [L2]

- a) using very large time slices converts it into First come First served scheduling algorithm
- b) using very small time slices converts it into First come First served scheduling algorithm
- c) using extremely small time slices increases performance
- d) using very small time slices converts it into Shortest Job First algorithm

Answer: a

37. What is FIFO algorithm? (Pgno: 266) [L1]

- a) first executes the job that came in last in the queue
- b) first executes the job that came in first in the queue
- c) first executes the job that needs minimal processor
- d) first executes the job that has maximum processor needs

Answer: b

38. There are 10 different processes running on a workstation. Idle processes are waiting for an input event in the input queue. Busy processes are scheduled with the Round-Robin time sharing method. Which out of the following quantum times is the best value for small response times, if the processes have a short runtime, e.g. less than 10ms? (Pgno: 272) [L2]

- a) $tQ = 15\text{ms}$
- b) $tQ = 40\text{ms}$
- c) $tQ = 45\text{ms}$
- d) $tQ = 50\text{ms}$

Answer: a

39. Which of the following algorithms tends to minimize the process flow time? (Pgno: 268)

[L1]

- a) First come First served
- b) Shortest Job First
- c) Earliest Deadline First
- d) Longest Job First

Answer: b

40. Which of the following statements are true? (Pgno: 267,271) [L2]

- I. Shortest remaining time first scheduling may cause starvation
- II. Preemptive scheduling may cause starvation
- III. Round robin is better than FCFS in terms of response time

- a) I only
- b) I and III only
- c) II and III only
- d) I, II and III

Answer: d

41. A system is in the safe state if _____ (Pgno: 328) [L1]
- a) the system can allocate resources to each process in some order and still avoid a deadlock
 - b) there exist a safe sequence
 - c) the system cannot allocate resources to each process in some order and still avoid a deadlock
 - d) enters into deadlock condition

Answer: a

42. The circular wait condition can be prevented by _____ (Pgno: 325) [L2]
- a) defining a linear ordering of resource types
 - b) using thread
 - c) using pipes
 - d) using both thread and pipes.

Answer: a

43. Which one of the following is the deadlock avoidance algorithm? (Pgno: 330) [L2]
- a) banker's algorithm
 - b) round-robin algorithm
 - c) elevator algorithm
 - d) karn's algorithm

Answer: a

44. A problem encountered in multitasking when a process is perpetually denied necessary resources is called _____ (Pgno: 338) [L2]
- a) deadlock
 - b) starvation
 - c) inversion
 - d) aging

Answer: b

45. To avoid deadlock _____ (Pgno: 327) [L2]
- a) there must be a fixed number of resources to allocate
 - b) resource allocation must be done only once
 - c) all deadlocked processes must be aborted
 - d) inversion technique can be used

Answer: a

46. The number of resources requested by a process _____ (Pgno: 327) [L2]
- a) must always be less than the total number of resources available in the system
 - b) must always be equal to the total number of resources available in the system
 - c) must not exceed the total number of resources available in the system
 - d) must exceed the total number of resources available in the system

Answer: c

47. The request and release of resources are _____ (Pgno: 328) [L2]

- a) command line statements
- b) interrupts
- c) system calls
- d) special programs

Answer: c

48. Deadlock prevention is a set of methods _____ (Pgno: 323) [L2]

- a) to ensure that at least one of the necessary conditions cannot hold
- b) to ensure that all of the necessary conditions do not hold
- c) to decide if the requested resources for a process have to be given or not
- d) to recover from a deadlock

Answer: a

49. If deadlocks occur frequently, the detection algorithm must be invoked _____ (Pgno: 336) [L1]

- a) rarely
- b) frequently
- c) rarely & frequently
- d) casually

Answer: b

50. A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units then, deadlock _____ (PgNo:CH-7) [L2]

- a) can never occur
- b) may occur
- c) has to occur
- d) cannot tell

Answer: a

51. Which among the following is helpful in CPU scheduling decisions?

- a. When a process switches from the waiting state to the running state
- b. When a process switches from the ready state to the running state
- c. When a process switches from the waiting state to the ready state
- d. When a process begins

(Pgno: 203) [L2]

Answer: c

PART B

4 MARKS:

1. What is critical section problem? (PgNo:263) [L1]

- Consider a system consists of ‘n’ processes. Each process has segment of Code called a critical section, in which the process may be changing common variables, updating a table, writing a file.
- When one process is executing in its critical section, no other process can allowed to execute in its critical section.

2. Define busy waiting and spinlock. (Pgno:213) [L1]

- When a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code.
 - This is called as busy waiting and this type of semaphore is also called a spinlock, because the process while waiting for the lock.
3. What are the requirements that a solution to the critical section problem must satisfy?
(Pgno:207) [L1]
- The three requirements are,
- Mutual Exclusion
 - Progress
 - Bounded Problem
4. Define entry section and exit section. (Pgno:207) [L1]
- The critical section problem is to design a protocol that the processes can use to cooperate.
 - Each process must request permission to enter its critical section. The section of the code implementing this request is the entry section.
 - The critical section is followed by an exit section. The remaining code is the remainder section.
5. Define semaphores. (Pgno:214) [L1]
- Semaphore is a synchronization toll. A semaphore S is an integer variable that apart from initialization is accessed only through 2 standard atomic operations.
 - i. Wait
 - ii. Signal
6. Name some classic problems of synchronization? (Pgno:219) [L1]
- The Bounded – Buffer Problem
 - The Reader – Writer Problem
 - The Dining – Philosophers Problem
7. What is the concept behind strong semaphore and spinlock?(Pgno:214) [L2]
- A semaphore is a generalization of a lock (or, the other way around, a lock is a special case of a semaphore).
 - Spinlocks are only valid within one process whereas semaphores can be used to synchronize between different processes, too
8. Elucidate mutex locks with its procedure.(Pgno:206) [L2]
- A mutual exclusion object (mutex) is a program object that allows multiple program threads to share the same resource, such as file access, but not simultaneously.

- After this stage, any thread that needs the resource must lock the mutex from other threads while it is using the resource.
9. Can a mutex be locked more than once ? (Pgno:224) [L2]
- a recursive mutex can be locked more than once (POSIX complaint systems), in which a count is associated with it, yet retains only one state (locked/unlocked).
 - The programmer must unlock the mutex as many number times as it was locked.
10. What are the different synchronization mechanisms? (Pgno:209) [L1]
- There are two types of synchronization: data synchronization and process synchronization:
 - The simultaneous execution of multiple threads or processes to reach a handshake such that they commit a certain sequence of actions.
 - Lock, mutex, and semaphores are examples of process synchronization.
11. Define classical problem of synchronization(Pgno:207) [L1]
- Bounded buffer problem or producer-consumer problem is a classical synchronization problem
 - we have a buffer with n cells or n slots and there are 2 process producers and consumers can produce and consume one article at a time.
12. Why is process synchronization required? (Pgno:210) [L2]
- The need for synchronization originates when processes need to execute concurrently.
 - The main purpose of synchronization is the sharing of resources without interference using mutual exclusion.
 - The other purpose is the coordination of the process interactions in an operating system.
13. List out the circumstances under which CPU scheduling decisions takes place. (PgNo:263) [L1]
- When a process switches from the running state to the waiting state (for example, as the result of an I/O request or an invocation of wait() for the termination of a child process)
 - When a process switches from the running state to the ready state (for example, when an interrupt occurs)
 - When a process switches from the waiting state to the ready state (for example, at completion of I/O)
 - When a process terminates
14. Differentiate preemptive and non-preemptive scheduling. (PgNo:264) [L2]
- In non-preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.
 - Otherwise it is preemptive scheduling.

15. Define dispatcher and state its functions. (PgNo:265) [L1]

- The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler.
- This function involves the following:
 - Switching context
 - Switching to user mode
 - Jumping to the proper location in the user program to restart that program

16. Suggest the criteria's for comparing CPU scheduling algorithms. (PgNo:265) [L1]

- CPU utilization.
- Throughput
- Turnaround time
- Waiting time
- Response time.

17. Is there any advantage in having different time – quantum sizes on different levels of a multi-level queuing system? Justify. (PgNo:275) [L2]

- The idea is to separate processes according to the characteristics of their CPU bursts.
- Processes that need more frequent servicing, for instance, interactive processes such as editors, can be in a queue with a small time quantum.
- Processes with no need for frequent servicing can be in a queue with a larger quantum, requiring fewer context switches to complete the processing, and thus making more efficient use of the computer.

18. What are the parameters present in multilevel feedback queue scheduler? (PgNo:276) [L1]

- The number of queues
- The scheduling algorithm for each queue
- The method used to determine when to upgrade a process to a higher-priority queue
- The method used to determine when to demote a process to a lower-priority queue
- The method used to determine which queue a process will enter when that process needs service

19. Define Deadlock. Specify the sequence in which process utilize a resource. (PgNo:315,316)[L1]

- waiting process is never again able to change state, because the resources it has requested are held by other waiting processes. This situation is called a deadlock.
- Sequence
 - Request
 - Use.
 - Release.

20. List the necessary conditions for deadlock situation to occur. (PgNo:319) [L1]

- Mutual exclusion.
 - Hold and wait
 - No preemption
 - Circular wait.
21. When a state is said to be safe? (PgNo:328) [L2]
- A state is safe if the system can allocate resources to each process (up to its maximum) in some order and still avoid a deadlock.
 - System is in a safe state only if there exists a safe sequence.
22. List three examples of deadlock in day today life. (PgNo:325) [L2]
- (Answer from website)
- Two cars crossing a single lane bridge from opposite directions.
 - A person going down a ladder while another person is climbing up the ladder.
 - Two trains traveling toward each other on the same track.
23. Is it possible to have deadlock involving only one process? Justify. (PgNo:341)
- It is not possible to have a deadlock involving only one single process.
 - The deadlock involves a circular “hold-and-wait” condition between two or more processes.
 - “one” process cannot hold a resource yet be waiting for another resource that it is holding. So for one process, deadlock cannot be possible
24. State the issues need to be addressed if preemption is required to deal with deadlocks. (PgNo:339) [L1]
- selecting a victim,
 - rollback,
 - starvation.
 - In a system that selects victims for rollback primarily on the basis of cost factors, starvation may occur, and the selected process can never complete its designated task
25. Why deadlock avoidance is necessary? (PgNo:339) [L2]
- A method for avoiding deadlocks, rather than preventing them, requires that the operating system have a priori information about how each process will utilize system resources.
 - The banker’s algorithm(for example)

PART C

12 MARKS:

1. Outline a solution using semaphores to solve dinning philosopher problem.(Pgno:215) [L3]
2. Show how wait () and signal() semaphore operations could be implemented in multiprocessor environments, using Test and Set instructions. The solution should exhibit minimal busy waiting. Develop pseudo code for implementing operations. (Pgno:226) [L2]
3. Briefly discuss about Peterson's solution.(Pgno:207)[L1]
4. (i)Explain the dining philosopher's critical section problem solution using monitor. (8) (Pgno:222)[L1]
(ii)Write the algorithm using test-and –set () instruction that satisfy all the critical section requirements. (4)(Pgno:222)[L2]
5. What is the important feature of critical section? State the dining philosopher's problem and show how to allocate the several resources among several processes in a deadlock and starvation free manner.(12) (pgno:221)[L3]
6. With an example elaborate Classical Problems of synchronization.(12) (pgno:219)[L1]
7. Compute non-preemptive SJF scheduling algorithm (PgNo:267) [L2]

Process	Arrival time	Burst time
P1	0	7
P2	2	4
P3	4	1
P4	5	4
P5	3	4

8. Consider the following set of processes with the length of the CPU-burst time in given ms:

Process	Arrival time	Burst time
P1	8	0
P2	4	1
P3	9	2
P4	5	3
P5	3	4

Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, priority and RR (quantum=2) scheduling. Also calculate waiting time and turnaround time for each scheduling algorithms. (PgNo: 266) [L2]

9. Discuss how the following pairs of scheduling criteria conflict in certain settings
 - i) CPU utilization and response time
 - ii) Average turnaround time and maximum waiting time.

- iii) I/O device utilization and CPU utilization (PgNo:265) [L1]
- 10. Explain Deadlock detection with suitable example. (PgNo:333) [L2]
- 11. Illustrate Bankers algorithm with an example (PgNo:330) [L2]
- 12. Describe Deadlock prevention in detail. (PgNo:323) [L2]
- 13. Explain the methods for handling deadlocks. (PgNo:322) [L1]

UNIT 3

MEMORY MANAGEMENT

PART A

MULTIPLE CHOICE QUESTIONS:

1. The processes on the disk that are waiting to be brought into memory for execution form the _____ (PgNo: 354) [L1]
 - a) Input queue
 - b) Output queue
 - c) Ready queue
 - d) Waiting queue

Answer: a

2. If it is not known at compile time that where the process will reside in memory, then the compiler must generate(PgNo: 355)[L1]
 - a) Relocatable code
 - b) Source code
 - c) Absolute code
 - d) Error

Answer: a

3. An address generated by the CPU is commonly referred to as (PgNo :355)[L1]
 - a) Logical address
 - b) Physical address
 - c) CPU address
 - d) Disk address

Answer:a

4. The set of all logical addresses generated by a program is (PgNo: 356)[L1]
 - a) Logical address space
 - b) Logical unit
 - c) Logical address set
 - d) Logical Space

Answer: a

5. The run-time mapping from virtual to physical addresses is done by a hardware device called(PgNo :356) [L1]
 - a) Memory management unit
 - b) Address space
 - c) Swap space
 - d) Compiler

Answer:a

6. All routines are kept on disk in a relocatable load format. The main program is loaded into memory and is executed whenever needed. This is called ____ (PgNo :357)[L1]
- a) Dynamic loading
 - b) Dynamic linking
 - c) Static loading
 - d) Static linking

Answer:a

7. A process can be taken temporarily out of memory to a backing store and then brought back into memory for continued execution. The process is called as ____ (PgNo :358)[L1]
- a) Swapping
 - b) Loading
 - c) Linking
 - d) Backing

Answer:a

8. consists of all processes whose memory images are on the backing store /memory and are ready to run (Pgno:359)[L1]
- a) Ready queue
 - b) Process queue
 - c) Wait queue
 - d) Static queue

Answer:a

9. In allocation, each process is contained in a single section of memory that is continuous to the section containing the next process. (Pgno:359)[L1]
- a) Contiguous memory allocation
 - b) Dynamic memory allocation
 - c) Partition
 - d) Fragmentation

Answer:a

10. When a partition is free, a process is selected from the input queue and is loaded into the free partition. This is called (Pgno:362)[L1]
- a) contiguous memory allocation
 - b) Dynamic memory allocation
 - c) Multiple partition method
 - d) Variable partition method

Answer:c

11. In which partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied. (Pgno:362)[L1]
- a) contiguous memory allocation
 - b) Dynamic memory allocation

- c) Multiple partition method
- d) Variable partition method

Answer:d

12. All memory available for user processes are considered as one large block of available memory is referred as hole.

13. _____ strategy allocates the first hole that is large enough to accommodate the process (Pgno:363)[L1]

- a) First fit
- b) Best fit
- c) Worst fit
- d) Least fit

Answer:a,

14. _____ strategy searches the entire ordered list and allocates the smallest hole that is large enough to accommodate the process (Pgno:363)[L1]

- a) First fit
- b) Best fit
- c) Worst fit
- d) Least fit

Answer:b

15. _____ strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole (Pgno:363)[L1]

- a) First fit
- b) Best fit
- c) Worst fit
- d) Least fit

Answer:c

16. _____ is a condition, where when there is enough total memory space to satisfy a request but the available spaces are not contiguous (Pgno:363)[L1]

- a) First fit
- b) External fragmentation
- c) Internal fragmentation
- d) Segmentation

Answer:b

17. The process where the memory contents are shuffled, so as to place all free memory together in one large block is called as (Pgno:364)[L1]

- a) Compaction
- b) External fragmentation
- c) Internal fragmentation
- d) Segmentation

Answer:a

18. The memory allocated to a process may be slightly larger than the requested memory. The difference between these two numbers is called (Pgno:363)[L1]

- a) Compaction
- b) External fragmentation
- c) Internal fragmentation
- d) Segmentation

Answer:c

19. are the two techniques that permit the logical address space of the processes to be noncontiguous, thus allowing a process to be allocated with physical memory wherever such memory is available.(Pgno:364)[L1]

- a) Segmentation and Paging
- b) Fragmentation and Paging
- c) Segmentation and Compaction
- d) Paging and Compaction

Answer:a

20. Which of the following segments are constructed by a compiler, whenever a program is compiled (Pgno:365)[L1]

- a) Code
- b) Global variables
- c) Stacks used by each thread
- d) Code ,global variables and stack used by each thread.

Answer:d

21. In segmentation, each address is specified by _____(Pgno:364)[L1]

- a) Segment number & offset
- b) Offset & value
- c) Value & segment number
- d) Key & value

Answer:a

22. In Paging, physical memory is broken into fixed-sized blocks called (Pgno:367)[L1]

- a) Frames
- b) Pages
- c) Segments
- d) Thread

Answer:a

23. In paging, logical memory is broken into blocks of the same size called _____(Pgno:367)[L1]

- a) Frames
- b) Pages

- c) Segments
- d) Thread

Answer:b

24. Every address generated by the CPU is divided into two parts. They are _____(Pgno:368)[L1]

- a) frame bit & page number
- b) page number & page offset
- c) page offset & frame bit
- d) frame offset & page offset

Answer:b

25. The size of a page is typically of size _____(Pgno:368)[L1]

- a) varied
- b) power of 2
- c) power of 4
- d) power of 6

Answer:b

26. Modern computer systems support a large logical address space of size _____(Pgno:378)[L1]

- a) 2^{32} to 2^{64}
- b) 2^{64} to 2^{128}
- c) 2^{16}
- d) 2^{68}

Answer: a

27. Address translation works from the outer page table inward, this scheme is also known as _____(Pgno:379)[L1]

- a) Forward Mapped page table
- b) Backward Mapped page table
- c) Forward and backward page table
- d) Page table

Answer: a

28. _____ Architecture supports a variation of one level paging.(Pgno:379)[L1]

- a) ARM
- b) VAX
- c) Intel 32
- d) SPARC

Answer: b

29. _____ Architecture supports a variation of four level paging. (Pgno:380) [L1]

- a) ARM
- b) VAX

- c) Intel 32
- d) ULTRASPARC

Answer: d

30. _____ is an approach for handling address space larger than 32 bits.(Pgno:380)[L1]
- a) Hashed page table
 - b) Inverted page table
 - c) Hierarchical page table
 - d) Forward mapped page table

Answer: a

31. _____ being the virtual page number available in hashed page table. (Pgno:380)[L1]
- a) Index value
 - b) Hash Value
 - c) Location of the page
 - d) Pointers to the page

Answer: b

32. Each entry in the hash table contains a _____ of elements that hash to the same location. (Pgno:380)[L1]
- a) Stack
 - b) Linked List
 - c) Arrays
 - d) Queue

Answer: b

33. Each entry in the hash table contains a linked list of elements that hash to the same location to handle _____.(Pgno:380)[L1]
- a) Cohesion
 - b) Collision
 - c) Coupling
 - d) Collaboration

Answer: b

34. 64 bit address space uses _____ page table. (Pgno:380)[L1]
- a) Hashed
 - b) Inverted
 - c) Clustered
 - d) Hierarchical

Answer :c

35. Clustered page table are useful for _____ address space. (Pgno:381)[L1]
- a) Sparse
 - b) Dense
 - c) Large

d) Small

Answer :a

36. _____ is the solution to keep track of how physical memory are used when page table consists of millions of entries. (Pgno:381)[L1]

- a) Hashed page table
- b) Inverted page table
- c) Hierarchical page table
- d) Forward mapped page table

Answer: b

37. _____ is the pair of inverted page table entry. (Pgno:382)[L1]

- a) < process id, page number>
- b) <page number, process id>
- c) <process id, offset>
- d) <offset, page number>

Answer:a

38. Inverted page table is sorted by _____ but look ups occur on _____.(Pgno:382)[L1]

- a) Virtual address , Physical address
- b) Physical address, virtual address
- c) Virtual address, logical address
- d) Physical address, logical address

Answer: b

39. _____ is a memory management scheme that supports user view of memory.(Pgno:364) [L1]

- a) Paging
- b) Segmentation
- c) Page replacement
- d) Virtual memory

Answer: b

40. The _____ register points to the page directory for the current process.(Pgno:385)[L1]

- a) CR3
- b) CR4
- c) CR
- d) Address

Answer: a

41. If Page_size flag is set, the page directory points directly to the _____ page frame, bypassing the inner page table. (Pgno: 385)[L1]

- a) 4KB

- b) 2KB
- c) 4MB
- d) 2MB

Answer: c

42. As software developers began to discover the 4-GB memory limitations of 32-bit architectures, Intel adopted a _____, which allows 32-bit processors to access a physical address space larger than 4 GB. (Pgno:386)[L1]

- a) Hash table
- b) page address extension
- c) page address
- d) page size

Answer: b

43. _____ paging is used for 4-KB and 16-KB pages. (Pgno:388)[L1]

- a) two-level
- b) One-Level
- c) Three-level
- d) Four-level

Answer: a

44. A _____ address space is a collection of segments.(Pgno:364)[L1]

- a) Physical
- b) Logical
- c) Virtual
- d) Hashed

Answer: b

45. Each entry in the segment table has a _____ and a _____.(Pgno:366)[L1]

- a) Segment base, segment Limit
- b) Segment id , offset
- c) Segment base, offset
- d) Offset, segment limit

Answer: a

46. The segment base contains the starting _____ where the segment resides in memory. (Pgno: 366)[L1]

- a) Logical address
- b) physical address
- c) virtual address
- d) hash address

Answer: b

PART B

4 MARKS:

1. Write short notes on hierarchical page table.(Pgno:378)[L1]
 - Most modern computer systems support a large logical address space (2³² to 2⁶⁴).
 - Clearly, we would not want to allocate the page table contiguously in main memory.
 - One simple solution to this problem is to divide the page table into smaller pieces.
 - Hierarchical Paging is a paging scheme which consists of two or more levels of page tables in a hierarchical manner.
 - The entries of the level 1 page table are pointers to a level 2 page table and entries of the level 2 page tables are pointers to a level 3 page table and so on.
 - The entries of the last level page table store actual frame information. Level 1 contains single page table and address of that table is stored in PTBR (Page Table Base Register).
2. Write short notes on inverted page table.(Pgno:381)[L1]
 - Inverted Page Table structure consists of one-page table entry for every frame of the main memory.
 - So the number of page table entries in the Inverted Page Table reduces to the number of frames in physical memory and a single page table is used to represent the paging information of all the processes.
 - Through the inverted page table, the overhead of storing an individual page table for every process gets eliminated and only a fixed portion of memory is required to store the paging information of all the processes together.
 - This technique is called as inverted paging as the indexing is done with respect to the frame number instead of the logical page number.
3. Write short notes on hashed page table.(Pgno:380)[L1]
 - A common approach for handling address spaces larger than 32 bits is to use a hashed page table, with the hash value being the virtual page number.
 - Each entry in the hash table contains a linked list of elements that hash to the same location (to handle collisions).
 - Each element consists of three fields:
 - o (1) the virtual page number,
 - o (2) the value of the mapped page frame, and
 - o (3) a pointer to the next element in the linked list.
 - The algorithm works as follows:
 - o The virtual page number in the virtual address is hashed into the hash table.
 - o The virtual page number is compared with field 1 in the first element in the linked list.

- o If there is a match, the corresponding page frame(field2)is used to form the desired physical address.
 - o If there is no match, subsequent entries in the linked list are searched for a matching virtual page number.
4. Discuss on X86-64 bit architecture.(Pgno:387)[L3]
- **x86-64** is the 64-bit version of the x86 instruction set.
 - It introduces two new modes of operation, 64-bit mode and compatibility mode, along with a new 4-level paging mode.
 - With 64-bit mode and the new paging mode, it supports vastly larger amounts of virtual memory and physical memory than is possible on its 32-bit predecessors, allowing programs to store larger amounts of data in memory.
 - In 64-bit mode, instructions are modified to support 64-bit operands and 64-bit addressing mode.
 - As the full x86 16-bit and 32-bit instruction sets remain implemented in hardware without any intervening emulation, these older executables can run with little or no performance penalty, while newer or modified applications can take advantage of new features of the processor design to achieve performance improvements.
5. Discuss about variations in intel32-bit and intel 64-bit architecture.(Pgno:383)[L3]

The difference between 32-bit and 64-bit processors are:

Parameter	32-bit processors	64-bit processors
Addressable space	It has 4 GB addressable space	64-bit processors have 16 GB addressable space
Application support	64-bit applications and programs won't work	32-bit applications and programs will work
OS SUPPORT	Needed 32bit OS	It can run on 32 and 64 bit OS
Support of Multitasking	Not an ideal option for stress testing and multi-tasking.	Works best for performing multi-tasking and stress testing.
Memory Limits	32-bit systems limited to 3.2 GB of RAM 32 bit Windows. It addresses limitation doesn't allow you to use full 4GB of Physical memory space.	64-bit systems will enable you to store up to 17. Billion GB of RAM

6. Explain briefly on Intel-32 bit Segmentation.(Pgno:384)[L3]
- The IA-32 architecture allows a segment to be as large as 4 GB, and the maximum number of segments per process is 16 K.

- The logical address space of a process is divided into two partitions. The first partition consists of up to 8 K segments that are private to that process.
- The second partition consists of up to 8 K segments that are shared among all the processes.
- Information about the first partition is kept in the local descriptor table (LDT); information about the second partition is kept in the global descriptor table (GDT).
- Each entry in the LDT and GDT consists of an 8-byte segment descriptor with detailed information about a particular segment, including the base location and limit of that segment.
- The logical address is a pair (selector, offset), where the selector is a 16-bit in which s designates the segment number, g indicates whether the segment is in the GDT or LDT, and p deals with protection.
- The offset is a 32-bit number specifying the location of the byte within the segment in question.

7. Explain briefly on Intel-32 bit paging.(Pgno:385)[L3]

- The IA-32 architecture allows a page size of either 4 KB or 4 MB. For 4-KB pages, IA-32 uses a two-level paging scheme in which the division of the 32-bit linear address
- The 10 high-order bits reference an entry in the outermost page table, which IA-32 terms the page directory.
- The page directory entry points to an inner page table that is indexed by the contents of the innermost 10 bits in the linear address.
- Finally, the low-order bits 0–11 refer to the offset in the 4-KB page pointed to in the page table.
- One entry in the page directory is the Page Size flag, which if set indicates that the size of the page frame is 4 MB and not the standard 4 KB.
- If this flag is set, the page directory points directly to the 4-MB page frame, bypassing the inner page table; and the 22 low-order bits in the linear address refer to the offset in the 4-MB page frame.

8. Brief on the concept of page address extension(PAE).(Pgno:386)[L2]

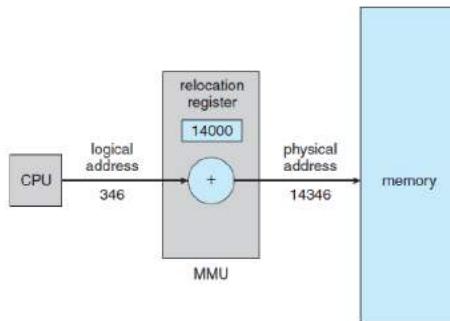
- Software developers discovered the 4-GB memory limitations of 32-bit architectures, Intel adopted a page address extension (PAE), which allows 32-bit processors to access a physical address space larger than 4 GB.
- The fundamental difference introduced by PAE support was that paging went from a two-level scheme to a three-level scheme, where the top two bits refer to a page directory pointer table.
- PAE also increased the page-directory and page-table entries from 32 to 64 bits in size, which allowed the base address of page tables and page frames to extend from 20 to 24 bits.

- Combined with the 12-bit offset, adding PAE support to IA-32 increased the address space to 36 bits, which supports up to 64 GB of physical memory.
 - It is important to note that operating system support is required to use PAE.
9. Explain compile time, load time, execution time with respect to address binding (Pgno:354)[L3]
- The binding of instructions and data to memory addresses can be done at any step along the way:
- Compile time. If you know at compile time where the process will reside in memory, then absolute code can be generated. For example, if you know that a user process will reside starting at location R , then the generated compiler code will start at that location and extend up from there. If, at some later time, the starting location changes, then it will be necessary to recompile this code. The MS-DOS .COM-format programs are bound at compile time.
 - Load time. If it is not known at compile time where the process will reside in memory, then the compiler must generate relocatable code. In this case, final binding is delayed until load time. If the starting address changes, we need only reload the user code to incorporate this changed value.
 - Execution time. If the process can be moved during its execution from one memory segment to another, then binding must be delayed until run time. Special hardware must be available for this scheme to work
10. List the differences between logical and physical address (Pgno: 356)[L2]

LOGICAL ADDRESS	PHYSICAL ADDRESS
It is the virtual address generated by CPU	The physical address is a location in a memory unit
Set of all logical addresses generated by CPU in reference to a program is referred as Logical Address Space.	Set of all physical addresses mapped to the corresponding logical addresses is referred as Physical Address.
The user uses the logical address to access the physical address.	The user can not directly access physical address.

11. Discuss the need for relocation register . (Pgno:356)[L3]
- The run-time mapping from virtual to physical addresses is done by a hardware device called the memory-management unit (MMU).
 - We can choose from many different methods to accomplish such mapping. For the time being, we illustrate this mapping with a simple MMU scheme that is a generalization of the base-register scheme.

- The base register is now called a relocation register. The value in the relocation register is added to every address generated by a user process at the time the address is sent to memory.
- For example, if the base is at 14000, then an attempt by the user to address location 0 is dynamically relocated to location 14000; an access to location 346 is mapped to location 14346.



12. Write the purpose of dynamic loading (Pgno:357)[L1]

- The size of a process has been limited to the size of physical memory. To obtain better memory-space utilization, we can use dynamic loading. With dynamic loading, a routine is not loaded until it is called.
- All routines are kept on disk in a relocatable load format. The main program is loaded into memory and is executed.
- When a routine needs to call another routine, the calling routine first checks to see whether the other routine has been loaded.
- If it has not, the relocatable linking loader is called to load the desired routine into memory and to update the program's address tables to reflect this change.
- Then control is passed to the newly loaded routine. The advantage of dynamic loading is that a routine is loaded only when it is needed.
- This method is particularly useful when large amounts of code are needed to handle infrequently occurring cases, such as error routines.

13. How are dynamic linked libraries used (Pgno:357)[L2]

- Dynamically linked libraries are system libraries that are linked to user programs when the programs are run.
- This feature is usually used with system libraries, such as language subroutine libraries. Without this facility, each program on a system must include a copy of its language library (or at least the routines referenced by the program) in the executable image. This requirement wastes both disk space and main memory.
- With dynamic linking, a stub is included in the image for each library routine reference. The stub is a small piece of code that indicates how to locate the appropriate memory-resident library routine or how to load the library if the routine is not already present. When the stub is executed, it checks to see whether the needed routine is already in memory.
- If it is not, the program loads the routine into memory. Either way, the stub replaces itself with the address of the routine and executes the routine.

14. What is the use of swapping (Pgno:358)[L2]
- A process must be in memory to be executed. A process, however, can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution.
 - Swapping makes it possible for the total physical address space of all processes to exceed the real physical memory of the system, thus increasing the degree of multiprogramming in a system.
15. Differentiate multiple partition method and variable partition method. (Pgno:362) [L2]
- In this multiple partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition.
 - When the process terminates, the partition becomes available for another process.
 - In the variable-partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.
 - Initially, all memory is available for user processes and is considered one large block of available memory, a hole.
 - As processes enter the system, they are put into an input queue. The operating system takes into account the memory requirements of each process and the amount of available memory space in determining which processes are allocated memory.
 - When a process is allocated space, it is loaded into memory, and it can then compete for CPU time.
 - When a process terminates, it releases its memory, which the operating system may then fill with another process from the input queue.
16. Write notes on internal fragmentation and external fragmentation (Pgno: 363)[L1]
- **Internal Fragmentation:**

Internal fragmentation happens when the memory is split into mounted sized blocks. Whenever a method request for the memory, the mounted sized block is allotted to the method. just in case the memory allotted to the method is somewhat larger than the memory requested, then the distinction between allotted and requested memory is that the Internal fragmentation.
 - **External Fragmentation:**

External fragmentation happens when there's a sufficient quantity of area within the memory to satisfy the memory request of a method. however the process's memory request cannot be fulfilled because the memory

offered is during a non-contiguous manner. Either you apply first-fit or best-fit memory allocation strategy it'll cause external fragmentation.

17. Provide the 3 strategies for dynamic storage allocation problem. (Pgno:362)[L1]

- **First fit.** Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or at the location where the previous first-fit search ended. We can stop searching as soon as we find a free hole that is large enough.
- **Best fit.** Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.
- **Worst fit.** Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.

18. Explain the fifty percent rule in fragmentation. (Pgno:363)[L3].

- Depending on the total amount of memory storage and the average process size, external fragmentation may be a minor or a major problem.
- Statistical analysis of first fit, for instance, reveals that, even with some optimization, given N allocated blocks, another $0.5 N$ blocks will be lost to fragmentation.
- That is, one-third of memory may be unusable! This property is known as the 50-percent rule.

19. Suggest the best solution for external fragmentation.(Pgno:364)[L3]

- External fragmentation exists when there is enough total memory to satisfy a request (from a process usually), but the total required memory is not available at a contiguous location i.e, its fragmented.

Solution to external fragmentation :

- 1) Compaction : shuffling the fragmented memory into one contiguous location.
- 2) Virtual memory addressing by using paging and segmentation.

20. What is the difference between segmentation and paging.(Pgno:365)[L2]

- The basic difference between paging and segmentation is that a page is always of fixed block size whereas, a segment is of variable size.
- Paging may lead to internal fragmentation as the page is of fixed block size, but it may happen that the process does not acquire the entire block size which will generate the internal fragment in memory.
- The segmentation may lead to external fragmentation as the memory is filled with the variable sized blocks.
- In paging the user only provides a single integer as the address which is divided by the hardware into a page number and Offset. On the other hands, in

segmentation the user specifies the address in two quantities i.e. segment number and offset.

- The size of the page is decided or specified by the hardware. On the other hands, the size of the segment is specified by the user.
 - In paging, the page table maps the logical address to the physical address, and it contains base address of each page stored in the frames of physical memory space. However, in segmentation, the segment table maps the logical address to the physical address, and it contains segment number and offset (segment limit).
21. Explain the structure of page table. (Pgno:368)[L3]
- Every address generated by the CPU is divided into two parts: a page number(p) and a page offset(d).The page number is used as an index into a page table.
 - The page table contains the base address of each page in physical memory. This base address is combined with the page offset to define the physical memory address that is sent to the memory unit.

PART C

12 MARKS:

1. Draw the diagram of segmentation memory management scheme and explain its principle. (Pgno: 364)[L3]
2. Describe in detail about the common techniques for structuring the page table. (Pgno:378)[L2]
3. Elaborate on INTEL 32-bit architecture. (Pgno:383)[L1]
4. Discuss about ARM architecture. (Pgno:388)[L3]
5. Describe in detail on Inverted page table with neat diagram. (Pgno:381) [L3]
6. Explain segmentation in detail. (Pgno:364)[L3]
7. Explain paging in detail. (Pgno:366)[L3]
8. Elucidate fragmentation and propose the solutions to avoid fragmentation. (PgNo:363) [L3]
9. Illustrate memory allocation schemes in detail. (Pgno: 362)[L3]

UNIT 4
VIRTUAL MEMORY
PART A

MULTIPLE CHOICE QUESTIONS:

1. Which of the following involves the separation of logical memory as perceived by the users from physical memory (PgNo:398) [L1]
 - a) Virtualmemory
 - b) Primary memory
 - c) Cache memory
 - d) Secondary memory Page

Answer : a

2. Which process refers to the logical view of a process stored in memory (PgNo:398) [L1]
 - a) Sparse Address Space
 - b) Virtual Address Space
 - c) Memory Address Space
 - d) Logical Address Space

Answer : b

3. Virtual address spaces that includes holes are called as (PgNo:399) [L1]
 - a) Sparse Address Space
 - b) Virtual Address Space**
 - c) Memory Address Space
 - d) Logical Address Space

Answer : a

4. Accessing a page marked as invalid causes a ----- (PgNo:403) [L1]
 - a) Page swap
 - b) Page fault
 - c) swap space
 - d) Bit space

Answer : b

5. which has the ability to mark an entry invalid through a valid-invalid bit or a special value of protection bits.(PgNo:404) [L1]
 - a) Page table
 - b) Demand Paging
 - c) swappingd.pure
 - d) demand paging

Answer : a

6. A page fault occurs (PgNo:403) [L1]
 - a) when the page is not in the memory

- b) when the page is in the memory
- c) when the process enters the blocked state
- d) when the process is in the ready state

Answer : a

7. _____ associates with each page the time when that page was brought into memory.(PgNo:413) [L1]

- a) FIFO page replacement algorithm
- b) Optimal Page replacement algorithm
- c) frame allocation algorithm
- d) page replacement algorithm

Answer : a

8. The -----page-replacement algorithm requires that the page with the smallest count be replaced (PgNo:420) [L1]

- a) Most frequently used
- b) frequently used
- c) Dynamically used
- d) least frequently used

Answer : d

9. The page-replacement algorithm is basedon the argument that the page with the smallest count was probably justbrought in and has yet to be used.(PgNo:420) [L1]

- a) least frequently used
- b) most frequently used
- c) Dynamically used
- d) frequently used

Answer : b

10. ----- is used to reduce the overhead during page replacement[PgNo 411] [L1]

- a) Bit
- b) Modify bit
- c) Frame
- d) Frequently used page

Answer : b

11. When the FIFO replacement algorithm mistakenly replaces a page that is still in active use,(PgNo:421) [L1]

- a) I/O is necessary
- b) I/O is not necessary
- c) That page is quickly retrieved from the free-framePool
- d) Both b and C

Answer : d

12. After allocating the frames to the process the leftover frames can be used as a free-frame buffer pool. This scheme is called as _____ (PgNo:423) [L1]

- a) Equal allocation
- b) Proportional allocation
- c) Dynamicallocation
- d) Staticallocation

Answer : a

13. Memory allocation based on Process size is called as (PgNo:423) [L1]

- a) Equal Allocation
- b) Dynamic allocation
- c) Proportional allocation
- d) Static allocation

Answer : c

14. ----- allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process (PgNo:424) [L1]

- a) Global replacement
- b) Local replacement
- c) Uniform replacement
- d) Non-Uniformreplacement

Answer : a

15. ----- requires that each process elects from only its own set of allocated frames. (PgNo:424) [L1]

- a) Global replacement
- b) Local replacement
- c) Uniform replacement
- d) Non-Uniformreplacement

Answer : b

16. A process cannot control its own page-fault rate is problem of (PgNo:424) [L1]

- a) Global replacement
- b) Local replacement
- c) Uniform replacement
- d) Non-Uniformreplacement

Answer : a

17. NUMA stands for (PgNo:425) [L1]

- a) Null unified memory access
- b) Neat unified memory access
- c) Non-uniform memory access
- d) Neat uniform memory access

Answer : c

18. A process is----- if it is spending more time paging than executing.

(PgNo:426) [L1]

- a) Thrashing
- b) Accumulating
- c) Abstracting
- d) Preparing TT

Answer : a

19. Local replacement algorithm is also called as ----- (PgNo:427) [L1]

- a) Time replacement algorithm
- b) priority replacement algorithm
- c) State replacement algorithm
- d) Scheduling algorithm

Answer : b

20. -----is based on the assumption of locality (PgNo:427) [L1]

- a) Working-set model
- b) State set model
- c) Priority model
- d) Replacement model

Answer : a

21. Working set window can be defined using (PgNo:427) [L1]

- a) Ω
- b) \mathfrak{L}
- c) π
- d) Δ

Answer : d

22. Raw disk is a ----- (PgNo:421) [L1]

- a) Sequential array of logical blocks, without any file-system data structures
- b) Non-Sequential array of logical blocks, without any file-system data structures
- c) Sequential array of logical blocks, with file-system data structures
- d) Non-Sequential array of logical blocks, with file-system data structures

Answer: a

23. When a process is swapped in, its pages are not swapped in all at once. Rather they are swapped in only when the process needs them .This particular method is called as (PgNo:401) [L1]

- a) Lazy swapper
- b) Busy Swapper**
- c) Swapper
- d) Smart Swapper

Answer: a

24. PFF stands for ----- (PgNo:429) [L1]

- a) Page Find Frequency
- b) Page Fault Frequency
- c) Peak Fault Frequency
- d) Peak Find Frequency

Answer : b

25. Operating system supports different page replacement policy. From the given below option which is not a valid page replacement policy?(PgNo:409) [L1]

- a) Least Recently Used
- b) First in first out
- c) Currently used policy
- d) Optimal page replacement policy

Answer : c

26. When a page fault occurs for a process that is below its working-set maximum(PgNo:446) [L1]

- a) Secondary memory allocates a page
- b) Virtual memory manager allocates a page from this list of free pages
- c) Main memory allocates a page
- d) Both primary and secondary memory allocates page

Answer : b

27. Copying a process from memory to disk to allow space for other processes is called ____(Pg No 408) [L1]

- a) Swapping
- b) Deadlock
- c) Demand Paging
- d) Page fault

Answer : a

28. .The type of memory that allows for very effective multiprogramming and relieves the user of memory size constraints is referred to as (PgNo:397) [L1]

- a) Real memory
- b) Virtual memory
- c) Main memory
- d) Secondary memory

Answer : b

29. The situation where the processor spends most of its time swapping process pieces rather than executing instructions is called: (PgNo:401) [L1]

- a) Paging
- b) The Principle of Locality
- c) Thrashing

- d) Swapping

Answer : c

30. Which one of the following is the simplest page replacement algorithm

? (PgNo:409) [L1]

- a) FIFO
- b) LRU
- c) Optimal
- d) SJF

Answer : a

31. _____ has the lowest fault rate of all the page replacement

algorithms. (PgNo:414) [L1]

- a) Optimal page replacement algorithm
- b) LRU replacement algorithm
- c) FIFO
- d) Counting based

Answer : a

32. Optimal page replacement algorithm is also called as _____ (PgNo:414)

[L1]

- a) LIFO
- b) NRU
- c) Clairvoyant replacement algorithm
- d) Page buffering

Answer: C

33. In a optimal page replacement algorithm, when a page is to be replaced, which of the following pages is chosen? (PgNo:414) [L1]

- a)** Oldest page
- b)** Newest page
- c)** Frequently occurred page in the future
- d)** Not frequently occurred page in the future

Answer : d

34. Optimal page replacement algorithm is implemented in _____ (PgNo:414)

[L1]

- a) General-purpose operating system
- b) Special-purpose operating system
- c) In any kind of operating system
- d) In Windows only

Answer : b

35. The two methods how LRU page replacement policy can be implemented in hardware are: (PgNo:416) [L1]

- a) Counters

- b) RAM & Registers
- c) Stack & Counters
- d) Registers

Answer : c

36. . Which is a technique to efficiently copy data resources in a computer system(PgNo:408) [L1]

- a) Copy-on-write
- b) Swapping
- c) Thrashing
- d) Paging

Answer : a

37. . _____ algorithm associates with each page the time when the page was brought into memory.(PgNo:413) [L1]

- a) Optimal page replacement
- b) FIFO
- c) LRU replacement algorithm
- d) Counting based replacement

Answer : b

38. Which of the following page replacement algorithms return the minimum number of page faults?(PgNo:416) [L1]

- a) LRU replacement algorithm
- b) Optimal page replacement algorithm
- c) FIFO
- d) Counting based replacement

Answer : b

39. Which of the following is the main drawback of FIFO page replacement algorithm? (PgNo:413) [L1]

- a) Requirement of large memory
- b) Frame allocation
- c) Reduction in multiprogramming
- d) Reduced optimality

Answer : c

40. . Which of the following is required to determine the number of page faults in FIFO? (PgNo:413) [L1]

- a) Page number
- b) Page frame number
- c) Memory capacity
- d) Segment number

Answer : b

41. In a FIFO algorithm, when a page is to be replaced, which of the following page is chosen?(PgNo:413) [L1]

- a) Oldest page
- b) Newest page
- c) Frequently occurred page in past
- d) Frequently occurred page in future

Answer : a

42. .FIFO algorithm is used by _____ operating system.(PgNo:413) [L1]

- a) Linux
- b) Mac
- c) Windows
- d) VAX/VMS

Answer : d

43. .The maximum number of frames per process is defined by

_____ (PgNo:403) [L1]

- a) the amount of available physical memory
- b) operating System
- c) instruction set architecture
- d) information set architecture

Answer : a

44. Pages used to satisfy copy-on-write duplications are typically allocated using

_____ (PgNo:408) [L1]

- a) zero-fill-on-demand
- b) demand paging
- c) page fault
- d) thrashing

Answer : a

45. Which one of the following is a valid statement (PgNo: 421) [L1]

a.Page is quickly retrieved from the free-frame

b.No I/O is necessary

- a) a is valid
- b) b is valid
- c) Both a and b are valid
- d) Both a and b are in valid

Answer : c

46. Thrashing occurs when(PgNo:425) [L1]

- a) When a page fault occurs
- b) Processes on system frequently access pages not memory
- c) Processes on system are in running state
- d) Processes on system are in waiting state

Answer : b

47. _____ is the concept in which a process is copied into the main memory from the secondary memory according to the requirement.(PgNo:401) [L1]

- a) Paging
- b) Demand paging
- c) Segmentation
- d) Swapping

Answer : b

48. Working set model for page replacement is based on the assumption of

_____ (PgNo:402) [L1]

- a) modularity
- b) locality
- c) globalization
- d) random access

Answer : b

49. When a program tries to access a page that is mapped in address space but not loaded in physical memory, then _____(PgNo:403) [L1]

- a) segmentation fault occurs
- b) fatal error occurs
- c) page fault occurs
- d) no error occurs

Answer : c

50. The aim of creating page replacement algorithms is to _____(PgNo:409) [L1]

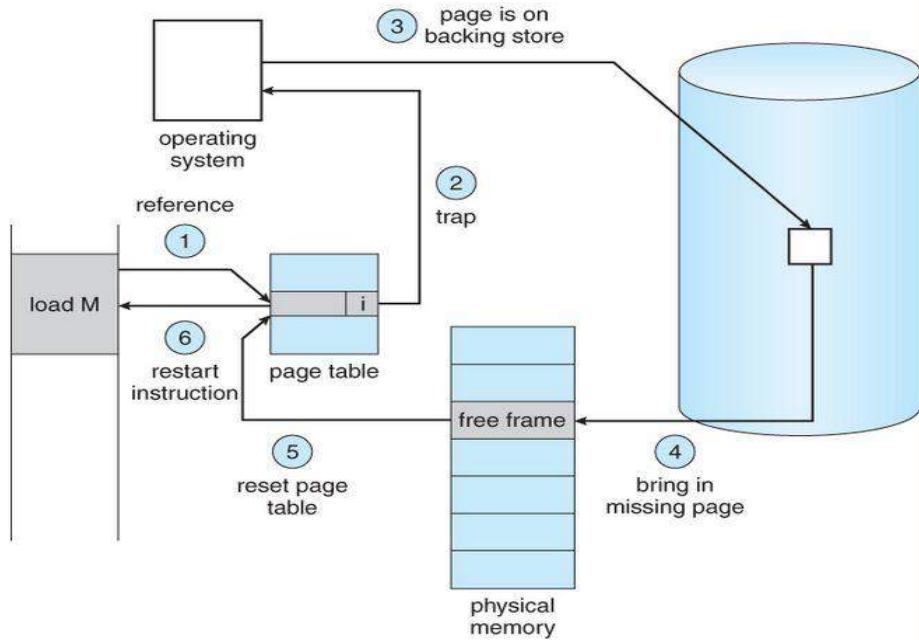
- a) Increase the page fault rate
- b) Decrease the page fault rate
- c) To allocate multiple pages to processes
- d) To deallocate pages to processors

Answer : c

PART B

4 MARKS:

1. Give the steps in handling a page fault in pictorial representation. (PgNo:403) [L2]



2. Define pure demand paging. (PgNo 404) [L2]

- There are cases when no pages are loaded into the memory initially, pages are only loaded when demanded by the process by generating page faults. This is called Pure Demand Paging.
- In pure demand paging, even a single page is not loaded into memory initially.
- Hence pure demand paging causes a page fault When starting execution of a process with no pages in memory, the operating system sets the instruction pointer to the first instruction of the process, which is on a non-memory resident page, the process immediately faults for the page.
- After this page is brought into memory, the process continues to execute, faulting as necessary until every page that it needs is in memory.
- At that point, it can execute with no more faults. This schema is pure demand paging.

3. What is locality model?(PgNo: 404) [L2]

- The locality model states that, as a process executes, it moves from locality to locality. A locality is a set of pages that are actively used together . A program is generally composed of several different localities, which may overlap.
 - For example, when a function is called, it defines a new locality. In this locality, memory references are made to the instructions of the function call, its local variables, and a subset of the global variables. When we exit the function, the process leaves this locality, since the local variables and instructions of the function are no longer in active use.
4. What is the role of secondary memory in demand paging?(PgNo:404) [L2]
- Memory that holds those pages that are not present in main memory.
 - The secondary memory is usually a high-speed disk.
 - It is known as the swap device, and the section of disk used for this purpose is known as swap space
5. List out the major components of the page-fault service time(PgNo:406)[L1]
- 1.Service the page-fault interrupt.
 - 2. Read in the page.
 - 3. Restart the process.
6. How do you limit the effect of thrashing? (PgNo:427)[L2]
- We can limit the effects of thrashing by using a local replacement algorithm (or priority replacement algorithm).
 - With local replacement, if one process starts thrashing, it cannot steal frames from another process and cause the latter to thrash as well. However, the problem is not entirely solved.
 - If processes are thrashing, they will be in the queue for the paging device most of the time.
 - The average service time for a page fault will increase because of the longer average queue for the paging device.
 - Thus, the effective access time will increase even for a process that is not thrashing.
 - To prevent thrashing, we must provide a process with as many frames as it needs.

7. Define page fault frequency.(PgNo:429) [L2]

- Thrashing has a high page-fault rate. Thus, we want to control the page-fault rate.
- When it is too high, we know that the process needs more frames.
- Conversely, if the page-fault rate is too low, then the process may have too many frames.
- We can establish upper and lower bounds on the desired page-fault rate. If the actual page-fault rate exceeds the upper limit, we allocate the process another frame.
- If the page-fault rate falls below the lower limit, we remove a frame from the process.
- Thus, we can directly measure and control the page-fault rate to prevent thrashing.
- As with the working-set strategy, we may have to swap out a process.
- If the page-fault rate increases and no free frames are available, we must select some process and swap it out to backing store.
- The freed frames are then distributed to processes with high page-fault rates.

8. How to compute effective access time?(PgNo:405)[L3]

- Let p be the probability of a page fault ($0 \leq p \leq 1$). We would expect p to
- be close to zero—that is, we would expect to have only a few page faults.
- The effective access time is then effective access time = $(1 - p) \times ma + p \times \text{page fault time}$.
- To compute the effective access time, we must know how much time is needed to service a page fault.

9. List out the advantages and disadvantages of demand paging?(PgNo:401)[L2]

The advantages of Demand Paging :

- Large virtual memory.
- More efficient use of memory.
- Unconstrained multiprogramming.
- There is no limit on degree of multiprogramming.

The disadvantages of Demand Paging:

- Number of tables and amount of processor overhead for handling
- page interrupts are greater than in the case of the simple paged management techniques.
- Due to the lack of an explicit constraints on a jobs address space size.

10. What is Virtual Memory?(PgNo:397)[L2]

- Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory.
- This is done by treating a part of secondary memory as the main memory.
- In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process. Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.
- By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

11. Define the schemes of counting –based page replacement (PgNo:420)[L2]

- The least frequently used (LFU) page-replacement algorithm requires that the page with the smallest count be replaced.
- The reason for this selection is that an actively used page should have a large reference count.
- A problem arises, however, when a page is used heavily during the initial phase of a process but then is never used again. Since it was used heavily, it has a large count and remains in memory even though it is no longer needed.
- One solution is to shift the counts right by 1 bit at regular intervals, forming an exponentially decaying average usage count. The most frequently used (MFU) page-replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

12. Write a short notes on equal allocation(PgNo:423)[L2]

- The easiest way to split m frames among n processes is to give everyone an equal share, m/n frames (ignoring frames needed by the operating system for the moment).
- For instance, if there are 93 frames and five processes, each process will get 18 frames.
- The three leftover frames can be used as a free-frame buffer pool. This scheme is called equal allocation.

13. Give a short notes on proportional allocation (PgNo:423)[L2]

- We allocate available memory to each process according to its size. Let the size of the virtual memory for process p_i be s_i , and define $S = \sum s_i$.
- Then, if the total number of available frames is m , we allocate a_i frames to process p_i , where $a_i = s_i/S \times m$.

- Of course, we must adjust each a_i to be an integer that is greater than the minimum number of frames required by the instruction set, with a sum not exceeding m .
- With proportional allocation, we would split 62 frames between two processes, one of 10 pages and one of 127 pages, by allocating 4 frames and 57 frames, respectively, since $10/137 \times 62 \approx 4$, and $127/137 \times 62 \approx 57$.

14. Define NUMA. (PgNo:424) [L2]

- In systems with multiple CPUs a given CPU can access some sections of main memory faster than it can access others.
- These performance differences are caused by how CPUs and memory are interconnected in the system.
- Frequently, such a system is made up of several system boards, each containing multiple CPUs and some memory.
- The system boards are interconnected in various ways, ranging from system buses to high-speed network connections like Infini Band.
- As you might expect, the CPUs on a particular board can access the memory on that board with less delay than they can access memory on other boards in the system.
- Systems in which memory access times vary significantly are known collectively as non-uniform memory access (NUMA) systems, and without exception, they are slower than systems in which memory and CPUs are located on the same board.

15. What is thrashing? (PgNo:425)[L2]

- Thrashing in computing is an issue caused when virtual memory is in use.
- It occurs when the virtual memory of a computer is rapidly exchanging data for data on hard disk, to the exclusion of most application-level processing.
- As the main memory gets filled, additional pages need to be swapped in and out of virtual memory.
- The swapping causes a very high rate of hard disk access.
- Thrashing can continue for a long duration until the underlying issue is addressed.
- Thrashing can potentially result in total collapse of the hard drive of the computer. Thrashing is also known as disk thrashing.

16. Write a short notes on Demand paging.(PgNo:401)[L2]

- Demand paging is referred when not all of a process's pages are in the RAM, then the OS brings the missing(and required) pages from the disk into the RAM.
- A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance.

- When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory. Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.

17. Give a short notes on Working-Set Model (PgNo:427)[L2]

- This model is based on the above-stated concept of the Locality Model. The basic principle states that if we allocate enough frames to a process to accommodate its current locality, it will only fault whenever it moves to some new locality. But if the allocated frames are lesser than the size of the current locality, the process is bound to thrash.
- According to this model, based on a parameter A, the working set is defined as the set of pages in the most recent 'A' page references. Hence, all the actively used pages would always end up being a part of the working set.
- The accuracy of the working set is dependant on the value of parameter A. If A is too large, then working sets may overlap. On the other hand, for smaller values of A, the locality might not be covered entirely.
- If D is the total demand for frames. Now, if 'm' is the number of frames available in the memory, there are 2 possibilities:
 - (i) $D > m$ i.e. total demand exceeds the number of frames, then thrashing will occur as some processes would not get enough frames.
 - (ii) $D \leq m$, then there would be no thrashing.
If there are enough extra frames, then some more processes can be loaded in the memory.
- On the other hand, if the summation of working set sizes exceeds the availability of frames, then some of the processes have to be suspended(swapped out of memory).

18. What is optimal Page replacement algorithm?(PgNo:419)[L2]

- **Optimal Page Replacement algorithm** → this algorithms replaces the page which will not be referred for so long in future.
- Although it can not be practically implementable but it can be used as a benchmark. Other algorithms are compared to this in terms of optimality.

19. Define the strategy of page fault frequency (PgNo:429)[L1]

- Page Fault Frequency is a replacement algorithm that can be used in systems with a local replacement strategy.

- A too high page fault frequency is an indication that the process may be thrashing and an unusually low page fault frequency indicates that too many frames are allocated to the process.
- The algorithm estimates the page fault frequency by measuring the time between page fault interrupts.
- If the page fault frequency is above an upper limit, the process is assigned an extra frame.
- If the page fault frequency is below a lower limit, a frame is removed from the process. A suitable frame to remove can be localized with help from the reference bits in the page table.

20. Mention few Page Replacement strategies.(PgNo:419) [L2]

- **Optimal Page Replacement algorithm** → this algorithms replaces the page which will not be referred for so long in future. Although it can not be practically implementable but it can be used as a benchmark. Other algorithms are compared to this in terms of optimality.
- **Least recent used (LRU) page replacement algorithm** → this algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.
- **FIFO** → in this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault..

21. When does thrashing Occurs?(PgNo:425) [L3]

- Thrashing in computing is an issue caused when virtual memory is in use. It occurs when the virtual memory of a computer is rapidly exchanging data for data on hard disk, to the exclusion of most application-level processing.
- As the main memory gets filled, additional pages need to be swapped in and out of virtual memory. The swapping causes a very high rate of hard disk access.
- Thrashing can continue for a long duration until the underlying issue is addressed. Thrashing can potentially result in total collapse of the hard drive of the computer.Thrashing is also known as disk thrashing.

22. Define Lazy Swapper. (PgNo: 401)[L2]

- A lazy swapper never swaps a page into memory unless that page will be needed.
- It is used in demand paging

23. Explain the major problems to implement demand paging (PgNo:401)[L2]

The two major problems to implement demand paging is developing

- a. Frame allocation algorithm
- b. Page replacement algorithm

24. Construct the steps in handling a page fault.(PgNo:403)[L3]

- We check an internal table (usually kept with the process control block) for this process to determine whether the reference was a valid or an invalid memory access.
- If the reference was invalid, we terminate the process. If it was valid but we have not yet brought in that page, we now page it in.
- We find a free frame (by taking one from the free-frame list, for example).
- We schedule a disk operation to read the desired page into the newly allocated frame.
- When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
- We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.

PART C

12 MARKS:

1.Illustrate Allocation of frames with a suitable example (PgNo: 421) [L3]

2.Elaborate in detail about NUMA? (PgNo:424) [L2]

3.Describe about Thrashing and causes of Thrashing (PgNo: 425) [L2]

4.Outline Working set model with a detailed description. (PgNo:427) [L3]

5.Explain about page replacement algorithms and discuss the difference between them? (PgNo:424) [L2]

6.Discuss about FIFO page replacement algorithms. (PgNo:420)[L2]

7.Elucidate Optimal Page Replacement algorithm. (PgNo:414)[L2]

8.Depict the workflow of LRU Page Replacement Algorithm in detail. (PgNo:416) [L3]

9.Outline the concept of demand paging with its performance. (PgNo:405)[L2]

10.Explain Page Buffering Algorithm in detail. (PgNo:420) [L2]

UNIT 5

STORAGE MANAGEMENT

PART A

MULTIPLE CHOICE QUESTIONS:

1. _____ has a flat circular shape, like a CD (PgNo: 467)[L1]

- a) Platter
- b) Disk
- c) Tracks
- d) Sector

Answer: a

2. The heads are attached to a _____ that moves all the heads as a unit (PgNo: 468) [L1]

- a) disk arm
- b) tracks
- c) cylinder
- d) sectors.

Answer: a

3. The forms of removable disks include CDs, DVDs, and Blu-ray discs as well as removable flash-memory devices known as _____ (PgNo: 469)[L1]

- a) Flash drives
- b) Disk arm
- c) ram
- d) rom

Answer: a

4. _____ was used as an early secondary-storage medium. (PgNo: 469)[L1]

- a) Magnetic tapes
- b) Host controller
- c) Disk controller
- d) Logical blocks

Answer: a

5. The _____ is the additional time for the disk to rotate the desired sector to the disk head (PgNo: 473)[L1]

- a) Bandwidth
- b) rotational latency
- c) seek time

d) Storage-area network.

Answer: b

6. The SCAN algorithm is sometimes called the _____ (PgNo: 476)[L1]

- a) shortest-seek-time-first (SSTF) algorithm.
- b) elevator algorithm
- c) Circular SCAN (C-SCAN) scheduling
- d) C-LOOK scheduling

Answer: b

7. The header and trailer contain information used by the disk controller, such as a sector number and an _____ (PgNo: 479)[L1]

- a) low-level formatting
- b) physical formatting
- c) error-correcting code (ECC)
- d) soft error

Answer: c

8. To increase efficiency, most file systems group blocks together into larger chunks, frequently called _____ (PgNo: 479)[L1]

- a) partition
- b) logical formatting
- c) soft error
- d) clusters

Answer: d

9. _____ is a low-level task of the operating system. (PgNo: 482)[L1]

- a) raw partition
- b) Swap-space management
- c) page slots
- d) swap map

Answer: a

10. A _____ is a sequence of characters organized into lines (PgNo: 503)[L1]

- a) executable file
- b) text file
- c) source file
- d) file

Answer: b

11. The system must keep a _____ to the location in the file where the next write is to take place (PgNo: 506) [L1]

- a) Write pointer
- b) Read pointer
- c) seek
- d) Current file position

Answer: a

12. The open-file table also has an _____ associated with each file (PgNo: 507) [L1]

- a) open-file table
- b) open count
- c) seek

- d) read pointer

Answer: b

13. The UNIX system uses a crude _____ stored at the beginning of some files (PgNo:511) [L1]

- a) Shell script
- b) Magic number
- c) exclusive lock
- d) shared lock

Answer: b

14. The block number provided by the user to the operating system is normally _____ (PgNo:
514) [L1]

- a) Allocation problem
- b) relative block number
- c) index
- d) allocation problem

Answer: b

15. Any entity containing a file system is generally known as _____ (PgNo: 516) [L1]

- a) volume
- b) device directory
- c) directory
- d) volume table of contents

Answer: a

16. The _____ provides host-name-to-network-address translations for the entire Internet. (PgNo: 530)[L1]

- a) distributed information systems
- b) distributed naming services
- c) domain name system
- d) network information service

Answer: c

17. An _____ begins at the root and follows a path down to the specified file. (PgNo: 522) [L1]

- a) relative pathname
- b) absolute path name
- c) current directory
- d) current directory

Answer: b

18. _____ scheme to determine when the last reference has been deleted and the disk space can be reallocated. (PgNo: 526) [L1]

- a) garbage collection
- b) hard links
- c) mount point
- d) collection

Answer:a

19. The industry is moving toward use of the _____ as a secure

distributed naming mechanism (PgNo: 531)[L1]

- a) common Internet filesystem
- b) active directory
- c) LDAP (lightweight directory-access protocol)
- d) distributed naming services

Answer: c

20. The series of accesses between the open () and close() operations makes up a ____ (PgNo:532) [L1]

- a) File session
- b) Consistency semantics state
- c) Information
- d) immutable shared files

Answer: c

21. ____ is used to Write new information at the end of the file. (PgNo: 534)[L1]

- a) Read
- b) Execute
- c) Delete
- d) Append

Answer: d

22. ____ is used to Load the file into memory and execute it (PgNo: 534)[L1]

- a) Read
- b) Execute
- c) Delete
- d) Append

Answer: b

23. _____ provide efficient and convenient access to the disk by allowing data to be stored, located, and retrieved easily (PgNo: 544)[L1]

- a) Filesystems
- b) I/O control
- c) basic filesystem
- d) logical file system

Answer: a

24. The _____ manages metadata information (PgNo: 545)[L1]

- a) logical filesystem
- b) file control block(FCB)
- c) UNIX filesystem
- d) extended filesystem.

Answer: a

25. _____ requires that each file occupy a set of contiguous blocks on the disk (PgNo: 553)[L1]

- a) Contiguous allocation
- b) Dynamic storage-allocation
- c) External fragmentation
- d) compacts

Answer: a

26. The contiguous-allocation problem can be seen as a particular application of the general _____ problem (PgNo: 554)[L1]

- a) Dynamic storage-allocation
- b) External fragmentation
- c) Contiguous allocation
- d) Compacts

Answer: a

27. Systems require that Contiguous allocation function be done _____ (PgNo: 555) [L1]

- a) off-line
- b) on-line
- c) sleeping mode
- d) in run time.

Answer: a

28. Most modern systems that need defragmentation can perform it _____ (PgNo: 555)

[L1]

- a) off-line
- b) on-line
- c) sleeping mode
- d) in run time.

Answer: a

29. If that amount proves not to be large enough, another chunk of contiguous space, known as an _____ (PgNo: 555)[L1]

- a) extent
- b) compacts
- c) sleeping mode
- d) external fragmentation

Answer: a

30. The usual solution to the problem is to collect blocks into multiples is called _____ (556) [L1]

- a) extent
- b) compacts
- c) clusters
- d) paging

Answer: c

31. An important variation on linked allocation is the use of a _____ (PgNo: 557)

[L1]

- a) Dynamic storage-allocation
- b) External fragmentation
- c) Contiguous allocation
- d) file-allocation table (FAT)

Answer: d

32. _____ solves the problem by bringing all the pointers together into one location the index block (PgNo: 557)[L1]

- a) Dynamic storage-allocation
- b) External fragmentation
- c) Contiguous allocation
- d) Indexed allocation

Answer: d

33. A block is obtained from the free-space manager, and its address is put in the _____ (PgNo:558)

- a) ith-indexed block entry
- b) index block
- c) entry list.
- d) Allocation

Answer: a

34. The first 12 of these pointers of combined scheme point to _____ (PgNo: 559)[L1]

- a) index block
- b) direct blocks
- c) indirect blocks
- d) single indirect

Answer: b

35. The next three pointers of combined scheme point to _____ (PgNo: 559)[L1]

- a) index block
- b) direct blocks
- c) indirect blocks
- d) single indirect

Answer: c

36. The first points of a combined schema , point to a _____ block, which is an index block containing not data but the addresses of blocks that do contain data.

(PgNo: 559)[L1]

- a) index block
- b) direct blocks
- c) indirect blocks
- d) single indirect

Answer: d

37. The second points to a _____ block, which contains the address of a block that contains the addresses of blocks that contain pointers to the actual data blocks

(PgNo: 559)[L1]

- a) Double indirect
- b) Direct blocks
- c) Indirect blocks
- d) single indirect

Answer: a

38. The last pointer of combine schema contains the address of a _____ block (PgNo:559) [L1]

- a) Double indirect
- b) Direct blocks
- c) Indirect blocks

- d) triple indirect

Answer: d

39. For any type of access, contiguous allocation requires only one access to get a _____ block
(PgNo: 560)[L1]

- a) disk
- b) directblocks
- c) indirect blocks
- d) triple indirect

Answer: a

40. The performance of indexed allocation depends on the _____, on the size of the file, and on the position of the block desired (PgNo: 561)[L1]

- a) Index structure.
- b) instruction
- c) two-level index
- d) two index-block

Answer: a

41. Track of free disk space, the system maintains a _____. (PgNo: 561)[L1]

- a) bit map
- b) bit vector
- c) free-space list
- d) vector

Answer: c

42. Frequently, the free-space list is implemented as a _____. (PgNo: 561)[L1]

- a) bit map
- b) bit vector
- c) free-space vector
- d) either bit map or bit vector

Answer: d

43. The systems maintain a separate section of main memory for a _____. (PgNo: 565)[L1]

- a) Page cache
- b) buffer cache
- c) virtual memory
- d) main memory

Answer: b

44. Systems cache file data using a _____. (PgNo: 565)[L1]

- a) page cache
- b) buffer cache
- c) virtual memory
- d) main memory

Answer: a

45. Including Solaris, Linux, and Windows —use page caching to cache both process pages and file data is known as ___. (PgNo: 566)[L1]

- a) Page cache
- b) buffer cache

- c) virtual memory
- d) unified virtual memory

Answer: d

46. Versions of UNIX and Linux provide a _____ cache (PgNo: 566)[L1]

- a) Unified buffer
- b) buffer cache
- c) page cache
- d) virtual memory

Answer: a

47. A memory mapping proceeds by reading in disk blocks from the file system and storing them in the buffer cache is known as _____(PgNo: 566)[L1]

- a) Unified buffer
- b) buffer cache
- c) page cache
- d) double caching

Answer: d

48. _____ occur in the order in which the disk subsystem receives them, and the writes are not buffered (PgNo: 567)[L1]

- a) Synchronous writes
- b) Asynchronous write
- c) read-ahead
- d) Free-behind

Answer:a

49. _____, the data are stored in the cache, and control returns to the caller. (Pg: 567)[L1]

- a) Synchronous writes
- b) Asynchronous write
- c) read-ahead
- d) Free-behind

Answer: b

50. _____ removes a page from the buffer as soon as the next page is requested (PgNo:567) [L1]

- a) Synchronous writes
- b) Asynchronous write
- c) read-ahead
- d) Free-behind

Answer: d

PART B

4 MARKS:

1. Give the diagrammatic representation of magnetic disks . (Pg No : 467-469)[L2]

- Each disk platter has a flat circular shape, like a CD.

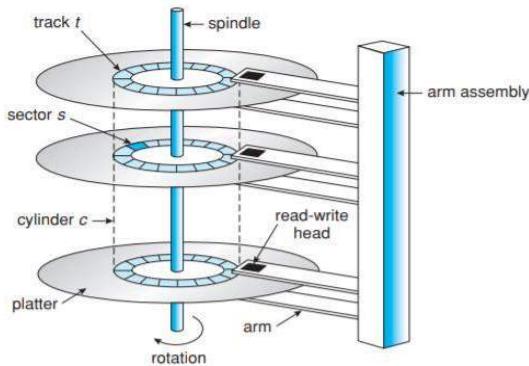
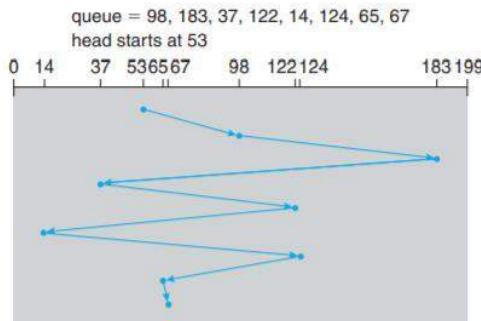


Figure 10.1 Moving-head disk mechanism.

- The two surfaces of a platter are covered with a magnetic material. We store information by recording it magnetically on the platters.
 - A read–write head “flies” just above each surface of every platter. The heads are attached to a disk arm that moves all the heads as a unit.
 - The surface of a platter is logically divided into circular tracks, which are subdivided into sectors. The set of tracks that are at one arm position makes up a cylinder.
2. Write a note on solid state disk (Pg No : 469,470)[L1]
 - The growing importance of solid-state disks, or SSDs. Simply described, an SSD is nonvolatile memory that is used like a hard drive.
 - SSDs have the same characteristics as traditional hard disks but can be more reliable because they have no moving parts and faster because they have no seek time or latency.
 3. Give a short note on FCFS (Pg No : 473-474)[L1]
 - The simplest form of disk scheduling is, of course, the first-come, first-served (FCFS) algorithm.
 - This algorithm is intrinsically fair, but it generally does not provide the fastest service. Example:



4. Give a difference between SCAN scheduling and CSCAN scheduling (Pg No : 475-476)(L2)
- SCAN SCHEDULING:**

- In the SCAN algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.
- The SCAN algorithm is sometimes called the elevator algorithm, since the disk arm behaves just like an elevator in a building, first servicing all the requests going up and then reversing to service requests the other way.

CSCAN SCHEDULING:

- Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time.
- Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way.

5. Compare LOOK scheduling with Disk- Scheduling Algorithm (Pg No :477-478)[L2]

LOOK SCHEDULING:

- The arm goes only as far as the final request in each direction. Then, it reverses direction immediately, without going all the way to the end of the disk. Versions of SCAN and C-SCAN that follow this pattern are called LOOK and C-LOOK scheduling, because they look for a request before continuing to move in a given direction.

SELECTION OF A DISK-SCHEDULING ALGORITHM :

- SSTF is common and has a natural appeal because it increases performance over FCFS. SCAN and C-SCAN perform better for systems that place a heavy load on the disk, because they are less likely to cause a starvation problem.



6. Distinguish file attributes and file operation. (Pg No : 504-510)[L2]

FILE ATTRIBUTES

- Name -The symbolic file name is the only information kept in human readable form.
- Identifier-
This unique tag, usually a number, identifies the file within the filesystem; it is the non-human-readable name for the file.
- Type - This information is needed for systems that support different types of files.

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

FILE OPERATION

- Creating a file.
- Writing a file.
- **Write.** Write or rewrite the file
- **Execute.** Load the file into memory and execute it.
- **Append.** Write new information at the end of the file.
- **Delete.** Delete the file and free its space for possible reuse.
- **List.** List the name and attributes of the file.

7.List the types of file (Pg No : 510-511)[L2]

- The system uses the extension to indicate the type of the file and the type of operations that can be done on that file. Only a file with a .com, .exe ,or .sh extension can be executed ,for instance.
- The .comand .exe files are two forms of binary executable files, whereas the .sh file is a shell script containing, in ASCII format, commands to the operating system.

11.1 File Concept

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

8.Compare and contrast sequential and direct access methods (Pg No : 513-514)(L2)

SEQUENTIAL ACCESS:

- The simplest access method is sequential access. Information in the file is processed in order one record after the other.
- Reads and writes make up the bulk of the operations on a file. A read operation—read next()—reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location.

DIRECT ACCESS :

- Another method is direct access (or relative access). Here, a file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
- For the direct-access method, the file operations must be modified to include the block number as a parameter.

9.State remote file system methods (Pg No :529-532)[L2]

REMOTE FILE SYSTEMS :

- Through the evolution of network and file technology, remote file-sharing methods have changed.
- The second major method uses a distributed file system (DFS) in which remote directories are visible from a local machine.

THE CLIENT–SERVER MODEL:

- Remote file systems allow a computer to mount one or more file systems from one or more remote machines. In this case, the machine containing the files is the server and the machine seeking access to the files is the client.
- A client can be specified by a network name or other identifier, such as an IP address, but these can be spoofed, or imitated. As a result of spoofing, an unauthorized client could be allowed access to the server.

DISTRIBUTED INFORMATION SYSTEMS:

- To make client–server systems easier to manage, distributed information systems, also known as distributed naming services, provide unified access to the information needed for remote computing.
- The domain name system (DNS) provides host-name-to-network-address translations for the entire Internet.
- It centralizes storage of user names, host names, printer information, and the like. File Sharing Unfortunately, it uses unsecure authentication methods, including sending user passwords unencrypted (in clear identifying hosts by IP address).

10. What is (a) consistency semantics , (b) UNIX semantics, (c) session semantics, (d)Immutable shared files semantics (Pg No : 532-533)[L2]

(a) consistency semantics:

- Consistency semantics represent an important criterion for evaluating any file system that supports file sharing. These semantics specify how multiple users of a system are to access a shared file simultaneously.

(b) UNIXsemantics:

- a. Writes to an open file by a user are visible immediately to other users who have this fileopen.
- b. One mode of sharing allows users to share the pointer of current location into the file. Thus, the advancing of the pointer by one user affects all sharing users. Here, a file has a single image that interleaves all accesses, regardless of theirorigin.

(c) session semantics:

- c. Writes to an open file by a user are not visible immediately to other users that have the same fileopen.
- d. Once a file is closed, the changes made to it are visible only in sessions starting later. Already open instances of the file do not reflect thesechange.

(d) Immutable shared files semantics:

- Unique approach is that of **immutable shared files**. Once a file is declared as shared by its creator, it cannot be modified.

Mention the different type of operation that is used to control the file system .(Pg No : 534) [L2]

11. Depict file system structure (Pg No : 543-545)[L2]

- To improve I/O efficiency, I/O transfers between memory and disk are performed in units of **blocks**. Each block has one or more sectors. Depending on the disk drive, sector size varies from 32 bytes to 4,096 bytes; the usual size is 512 bytes.
- **File systems** provide efficient and convenient access to the disk by allowing data to be stored, located, and retrieved easily. A file system poses two quite different design problems.
- The **I/O control** level consists of device drivers and interrupt handlers to transfer information between the main memory and the disk system. A device driver can be thought of as a translator.
- The **basic file system** needs only to issue generic commands to the appropriate device driver to read and write physical blocks on the disk. Each physical block is identified by its numeric disk address.

12. Give a note on following (a) Linear list (b) Hash table (Pg No : 552-553)[L2]

(a) **Linear list:**

- The simplest method of implementing a directory is to use a linear list of file names with pointers to the data blocks.
- This method is simple to program but time-consuming to execute. To create a new file, we must first search the directory to be sure that no existing file has the same name. Then, we add a new entry at the end of the directory.

(b) **Hashtable:**

- Another data structure used for a file directory is a hash table. Here, a linear list stores the directory entries, but a hash data structure is also used.
- The hash table takes a value computed from the file name and returns a pointer to the file name in the linear list.

13. Compare and contrast the pros and cons of all types of file allocation methods.
(Pg No : 560-561)(L2)

- The allocation methods that we have discussed vary in their storage efficiency and data-block access times. Both are important criteria in selecting the proper method or methods for an operating system to implement.
- A file created for sequential access will be linked and cannot be used for direct access. A file created for direct access will be contiguous and can support both direct access and sequential access, but its maximum length must be declared when it is created.
- Indexed allocation is more complex. If the index block is already in memory, then the access can be made directly. However, keeping the index block in memory requires considerable space.

14. List methods available in free space management (Pg No :561-564).[L1]

- Bit vector
- Linked list
- Grouping
- Counting
- Space maps

15. State the file operations that are carried in file system (Pg No : 505)[L2]

- a. Creating a file.
- b. Writing a file.
- c. **Write.** Write or rewrite the file
- d. **Execute.** Load the file into memory and execute it.
- e. **Append.** Write new information at the end of the file.
- f. **Delete.** Delete the file and free its space for possible reuse.
- g. **List.** List the name and attributes of the file.

16. Write a short note on file structure (Pg No : 511)(L3)

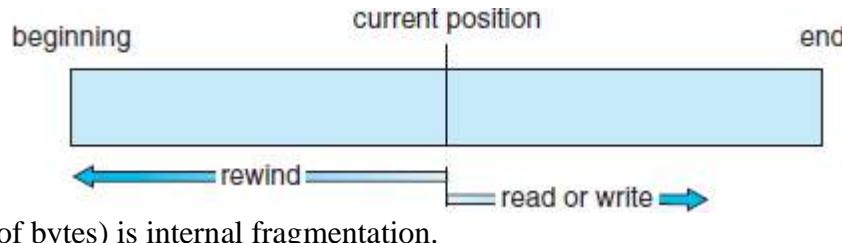
- File types also can be used to indicate the internal structure of the file. For example, the operating system requires that an executable file have a specific structure so that it can determine where in memory to load the file and what the location of the first instruction is.
 - For example, assume that a system supports two types of files: text files (composed of ASCII characters separated by a carriage return and line feed) and executable binary files.
 - Now, if we (as users) want to define an encrypted file to protect the contents from being read by unauthorized people, we may find neither file type to be appropriate. The encrypted file is not ASCII text lines but rather is (apparently) random bits.

17. How the internal file structure operate on file structuring(Pg No : 512)[L2]

- Internally, locating an offset within a file can be complicated for the operating system. Disk systems typically have a well-defined block size determined by the size of a sector.
- All disk I/O is performed in units of one block (physical record), and all

blocks are the same size. It is unlikely that the physical record size will exactly match the length of the desired logical record.

- If each block were 512 bytes, for example, then a file of 1,949 bytes would be allocated four blocks (2,048 bytes); the last 99 bytes would be wasted. The waste incurred to keep everything in units of blocks (instead



of bytes) is internal fragmentation.

18. How the multiple users perform in file sharing (Pg No : 528-529)[L2]

- When an operating system accommodates multiple users, the issues of file sharing, file naming, and file protection become preeminent. Given a directory structure that allows files to be shared by users, the system must mediate the file sharing.
- Although many approaches have been taken to meet this requirement, most systems have evolved to use the concepts of file (or directory) **owner** (or **user**) and **group**.
- The owner is the user who can change attributes and grant access and who has the most control over the file. The group attribute defines a subset of users who can share access to the file.
- In these cases, the ID checking and permission matching are straightforward, once the file systems are mounted.

19. What are the failure modes in Remote file sharing (Pg No :531-532)[L2]

- Local file systems can fail for a variety of reasons, including failure of the disk containing the file system, corruption of the directory structure or other disk-management information (collectively called **metadata**), disk-controller failure, cable failure, and host-adapter failure.
- To implement this kind of recovery from failure, some kind of **state information** may be maintained on both the client and the server. If both server and client maintain knowledge of their current activities and open files, then they can seamlessly recover from a failure.
- In the situation where the server crashes but must recognize that it has remotely mounted exported file systems and opened files, NFS takes a simple approach, implementing a **stateless DFS**.

20. Distinguish direct and sequential access methods.(PgNo:513) (L2)

SEQUENTIAL ACCESS :

- a. The simplest access method is sequential access. Information in the file is

processed in order, one record after the other. This mode of access is by far the most common; for example, editors and compilers usually access files in this fashion.

- b. Reads and writes make up the bulk of the operations on a file. A read operation—read next()—reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location. Similarly, the write operation—write next()—appends to the end of the file and advances to the end of the newly written material.

DIRECT ACCESS :

- Another method is direct access (or relative access). Here, a file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
- The direct-access method is based on a disk model of a file, since disks allow random access to any file block. For direct access, the file is viewed as a numbered sequence of blocks or records. Thus, we may read block 14, then read block 53, and then write block 7. There are no restrictions on the order of reading or writing for a direct-access file.
- A relative block number is an index relative to the beginning of the file. Thus, the first relative block of the file is 0, the next is 1, and so on, even though the absolute disk address may be 14703 for the first block and 3192 for the second. The use of relative block numbers allows the operating system to decide where the file should be placed.

21. Mention the purpose of indexed allocation (Pg No : 559)[L2]

- a. **Linked scheme.** An index block is normally one disk block. Thus, it can be read and written directly by itself. To allow for large files, we can link together several index blocks.
- b. **Multilevel index.** A variant of linked representation uses a first-level index block to point to a set of second-level index blocks, which in turn point to the file blocks.

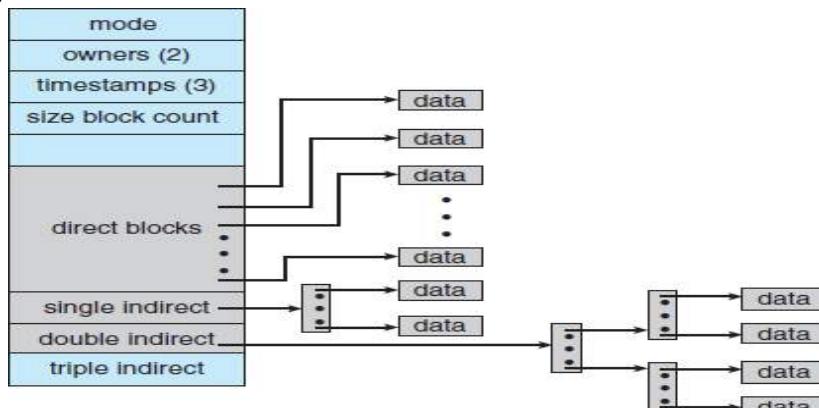
This approach could be continued to a third or fourth level, depending on the desired maximum file size. With 4,096-byte blocks, we could store 1,024 four-byte pointers in an index block. Two levels of indexes allow 1,048,576 data blocks and a file size of up to 4 GB.

- c. **Combined scheme.** Another alternative, used in UNIX-based file systems, is to keep the first, say, 15 pointers of the index block in the file's inode.
- d. The first 12 of these pointers point to **direct blocks**; that is, they contain addresses of blocks that contain data of the file. Thus, the data for small files (of no more than 12 blocks) do not need a separate index block.
- e. If the block size is 4 KB, then up to 48 KB of data can be accessed directly. The next three pointers point to **indirect blocks**. The first points to a **single indirect block**, which is an index block containing not data but the addresses of blocks that do contain data.
- f. The second points to a **double indirect block**, which contains the address

of a block that contains the addresses of blocks that contain pointers to the actual data blocks. The last pointer contains the address of a **triple in direct block**.

22. Draw the performance matrices of file system implementation in operating system.

(Pg No : 560-561)(L2)



23. State how access control in protection of file system works?.(Pg No : 534-538)[L2]

- a. Different users may need different types of access to a file or directory.
The most general scheme to implement identity dependent access is to associate with each file and directory an **access-control list (ACL)** specifying user names and the types of access allowed for each user.
- b. **Owner.** The user who created the file is the owner.
- c. **Group.** A set of users who are sharing the file and need similar access is a group, or work group.
- d. **Universe.** All other users in the system constitute the universe

PART C

12 MARKS:

1. What is magnetic disk and explain in brief about solid state disk and magnetic disk with pictorial representation. (Pg No : 467-470)[L1]
2. Elucidate in detail about all five types of disk scheduling (Pg No : 472-478)[L1]
3. Give a brief note on file concepts which says about file attribute , types and operation . scheduling (PgNo : 503-511) [L2]
4. Explain in detail about file structure (Pg No : 511-513)[L1]
5. What is file access method and give a note on all kinds of it. (Pg No : 513-515)[L1]
6. Describe in detail about file sharing (Pg No : 528-533)[L1]
7. Depict in detail about all three types of file allocation methods (Pg No : 553-560)[L1]
8. Elaborate the low level task on OS (swap space management) (Pg No : 482-484)[L2]
9. Give the brief note on storage management in operating system (Pg No : 467-470)[L2]
10. Compare and contrast the contiguous allocation and linked allocation (Pg No : 553-557)(L2)

*****END*****

Operating Systems Question Bank

1.What is operating system?

- a) collection of programs that manages hardware resources b) system service provider to the application programs
- c) link to interface the hardware and application programs d) all of the mentioned

Answer:d

2. To access the services of operating system, the interface is provided by the

- a) system calls b) API c) library d) assembly instructions

Answer:a

3. Which one of the following is not true?

- a) kernel is the program that constitutes the central core of the operating system
- b) kernel is the first part of operating system to load into memory during booting
- c) kernel is made of various modules which can not be loaded in running operating system
- d) kernel remains in the memory during the entire computer session

Answer:c

4. Which one of the following error will be handle by the operating system?

- a) power failure b) lack of paper in printer c) connection failure in the network d) all of the mentioned

Answer:d

5. The main function of the command interpreter is

- a) to get and execute the next user-specified command
- b) to provide the interface between the API and application program
- c) to handle the files in operating system
- d) none of the mentioned

Answer:a

6. By operating system, the resource management can be done via

- a) time division multiplexing b) space division multiplexing c) both (a) and (b)
- d) none of the mentioned

Answer:c

7. If a process fails, most operating system write the error information to a

- a) log file b) another running process c) new file d) none of the mentioned

Answer:a

8. Which facility dynamically adds probes to a running system, both in user processes and in the kernel?
a) Dtrace b) Dlocate c) Dmap d) Dadd

Answer:a

9. Which one of the following is not a real time operating system?

- a) VxWorks b) Windows CE c) RTLinux d) Palm OS

Answer:d

10. The OS X has

- a) monolithic kernel
b) hybrid kernel
c) microkernel
d) monolithic kernel with modules

Answer:b

11. The systems which allows only one process execution at a time, are called

- a) uniprogramming systems
b) uniprocessing systems
c) unitasking systems
d) none of the mentioned

Answer:a

12. In operating system, each process has its own

- a) address space and global variables
b) open files
c) pending alarms, signals and signal handlers
d) all of the mentioned

Answer:d

13) A single thread of control allows the process to perform :

- a) only one task at a time
b) multiple tasks at a time
c) All of these

Answer: a

14) The objective of multi-programming is to : (choose two)

- a) Have some process running at all times
b) Have multiple programs waiting in a queue ready to run
c) To minimize CPU utilization
d) To maximize CPU utilization

Answer: a and d

- 15) Which of the following do not belong to queues for processes ?
a) Job Queue b) PCB queue c) Device Queue d) Ready Queue
Answer: b

16) When the process issues an I/O request :

- a) It is placed in an I/O queue
- b) It is placed in a waiting queue
- c) It is placed in the ready queue
- d) It is placed in the Job queue

Answer: a

17) What is a trap/exception ?

- a) hardware generated interrupt caused by an error
- b) software generated interrupt caused by an error
- c) user generated interrupt caused by an error
- d) None of these

Answer: b

18) What is an ISR ?

- a) Information Service Request
- b) Interrupt Service Request
- c) Interrupt Service Routine
- d) Information Service Routine

Answer: c

19) An interrupt vector

- a) is an address that is indexed to an interrupt handler
- b) is a unique device number that is indexed by an address
- c) is a unique identity given to an interrupt
- d) None of these

Answer: a

20) DMA is used for : (choose two)

- a) High speed devices(disks and communications network)
- b) Low speed devices
- c) Saving CPU cycles
- d) Utilizing CPU cycles

Answer: a and c

21) In a memory mapped input/output :

- a) the CPU uses polling to watch the control bit constantly, looping to see if device is ready
- b) the CPU writes one data byte to the data register and sets a bit in control register to show that a byte is available
- c) the CPU receives an interrupt when the device is ready for the next byte

d) the CPU runs a user written code and does accordingly

Answer: b

22) In a programmed input/output(PIO) :

- a) the CPU uses polling to watch the control bit constantly, looping to see if device is ready
- b) the CPU writes one data byte to the data register and sets a bit in control register to show that a byte is available
- c) the CPU receives an interrupt when the device is ready for the next byte
- d) the CPU runs a user written code and does accordingly

Answer: a

23) In an interrupt driven input/output :

- a) the CPU uses polling to watch the control bit constantly, looping to see if device is ready
- b) the CPU writes one data byte to the data register and sets a bit in control register to show that a byte is available
- c) the CPU receives an interrupt when the device is ready for the next byte
- d) the CPU runs a user written code and does accordingly

Answer: c

24) In the layered approach of Operating Systems : (choose two)

- a) Bottom Layer(0) is the User interface
- b) Highest Layer(N) is the User interface
- c) Bottom Layer(0) is the hardware
- d) Highest Layer(N) is the hardware

Answer: b and c

25) How does the Hardware trigger an interrupt ?

- a) Sending signals to CPU through system bus
- b) Executing a special program called interrupt program
- c) Executing a special program called system program
- d) Executing a special operation called system call

Answer: a

26) Which operation is performed by an interrupt handler ?

- a) Saving the current state of the system
- b) Loading the interrupt handling code and executing it
- c) Once done handling, bringing back the system to the original state it was before the interrupt occurred
- d) All of these

Answer: d

27. Which module gives control of the CPU to the process selected by the short-term scheduler?

- a) dispatcher
- b) interrupt
- c) scheduler
- d) none of the mentioned

Answer:a

28. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called

- a) job queue
- b) ready queue
- c) execution queue
- d) process queue

Answer:b

29. The interval from the time of submission of a process to the time of completion is termed as
a) waiting time b) turnaround time c) response time d) throughput

Answer:b

30. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?

- a) first-come, first-served scheduling b) shortest job scheduling c) priority scheduling
d) none of the mentioned

Answer:a

31. In priority scheduling algorithm

- a) CPU is allocated to the process with highest priority b) CPU is allocated to the process with lowest priority
c) equal priority processes can not be scheduled d) none of the mentioned

Answer:a

32. In priority scheduling algorithm, when a process arrives at the ready queue, its priority is compared with the priority of

- a) all process b) currently running process c) parent process d) init process

Answer:b

33. Time quantum is defined in

- a) shortest job scheduling algorithm b) round robin scheduling algorithm
c) priority scheduling algorithm d) multilevel queue scheduling algorithm

Answer:b

34. Process are classified into different groups in

- a) shortest job scheduling algorithm b) round robin scheduling algorithm
c) priority scheduling algorithm d) multilevel queue scheduling algorithm

Answer:d

35. In multilevel feedback scheduling algorithm

- a) a process can move to a different classified ready queue
b) classification of ready queue is permanent
c) processes are not classified into groups
d) none of the mentioned

Answer:a

36. Which one of the following can not be scheduled by the kernel?

- a) kernel level thread
b) user level thread
c) process
d) none of the mentioned

Answer:b.

- 37) CPU scheduling is the basis of _____.

- a) multiprocessor systems b) multiprogramming operating systems
c) larger memory sized systems d) None of these

Answer: b

38) With multiprogramming, _____ is used productively.

- a) time
- b) space
- c) money
- d) All of these

Answer: a

39) The two steps of a process execution are : (choose two)

- a) I/O Burst
- b) CPU Burst
- c) Memory Burst
- d) OS Burst

Answer: a and b

40) An I/O bound program will typically have :

- a) a few very short CPU bursts
- b) many very short I/O bursts
- c) many very short CPU bursts
- d) a few very short I/O bursts

Answer: c

41) A process is selected from the _____ queue by the _____ scheduler, to be executed.

- a) blocked, short term
- b) wait, long term
- c) ready, short term
- d) ready, long term

Answer: c

42) In the following cases non – preemptive scheduling occurs : (Choose two)

- a) When a process switches from the running state to the ready state
- b) When a process goes from the running state to the waiting state
- c) When a process switches from the waiting state to the ready state
- d) When a process terminates

Answer: b and d

43) The switching of the CPU from one process or thread to another is called :

- a) process switch
- b) task switch
- c) context switch
- d) All of these

Answer: d

44) Dispatch latency is :

- a) the speed of dispatching a process from running to the ready state
- b) the time of dispatching a process from running to ready state and keeping the CPU idle
- c) the time to stop one process and start running another one
- d) None of these

Answer: c

45. In Unix, Which system call creates the new process?

- a) fork
- b) create
- c) new
- d) none of the mentioned

Answer:a

46. A process can be terminated due to

- a) normal exit
- b) fatal error

- c) killed by another process
- d) all of the mentioned

Answer:d

47. What is the ready state of a process?

- a) when process is scheduled to run after some execution
- b) when process is unable to run until some task has been completed
- c) when process is using the CPU
- d) none of the mentioned

Answer:a

Explanation:When process is unable to run until some task has been completed, the process is in blocked state and if process is using the CPU, it is in running state.

48. What is interprocess communication?

- a) communication within the process
- b) communication between two process
- c) communication between two threads of same process
- d) none of the mentioned

Answer:b

49. A set of processes is deadlock if

- a) each process is blocked and will remain so forever
- b) each process is terminated
- c) all processes are trying to kill each other
- d) none of the mentioned

Answer:a

50. A process stack does not contain

- a) function parameters
- b) local variables
- c) return addresses
- d) PID of child process

Answer:d

51. Which system call returns the process identifier of a terminated child?

- a) wait
- b) exit
- c) fork
- d) get

Answer:a

52) A Process Control Block(PCB) does not contain which of the following :

- a) Code
- b) Stack
- c) Heap
- d) Data

- e) Program Counter
- f) Process State
- g) I/O status information
- h) bootstrap program

Answer: h

53) The number of processes completed per unit time is known as _____.

- a) Output
- b) Throughput
- c) Efficiency
- d) Capacity

Answer: b

54) The state of a process is defined by :

- a) the final activity of the process
- b) the activity just executed by the process
- c) the activity to next be executed by the process
- d) the current activity of the process

Answer: d

55) Which of the following is not the state of a process ?

- a) New
- b) Old
- c) Waiting
- d) Running
- e) Ready
- f) Terminated

Answer: b

56) The Process Control Block is :

- a) Process type variable
- b) Data Structure
- c) a secondary storage section
- d) a Block in memory

Answer: b

57) The entry of all the PCBs of the current processes is in :

- a) Process Register
- b) Program Counter
- c) Process Table
- d) Process Unit

Answer: c

58) The degree of multi-programming is :

- a) the number of processes executed per unit time
- b) the number of processes in the ready queue
- c) the number of processes in the I/O queue
- d) the number of processes in memory

Answer: d

59) When a process terminates : (Choose Two)

- a) It is removed from all queues
- b) It is removed from all, but the job queue
- c) Its process control block is de-allocated
- d) Its process control block is never de-allocated

Answer: a and c

60) What is a long-term scheduler ?

- a) It selects which process has to be brought into the ready queue
- b) It selects which process has to be executed next and allocates CPU
- c) It selects which process to remove from memory by swapping
- d) None of these

Answer: a

61) If all processes I/O bound, the ready queue will almost always be _____, and the Short term Scheduler will have a _____ to do.

- a) full, little
- b) full, lot
- c) empty, little
- d) empty, lot

Answer: c

62) What is a medium-term scheduler ?

- a) It selects which process has to be brought into the ready queue
- b) It selects which process has to be executed next and allocates CPU
- c) It selects which process to remove from memory by swapping
- d) None of these

63) There are 10 different processes running on a workstation. Idle processes are waiting for an input event in the input queue. Busy processes are scheduled with the Round-Robin timesharing method.

Which out of the following quantum times is the best value for small response times, if the processes have a short runtime, e.g. less than 10ms ?

- a) $t_Q = 15\text{ms}$
- b) $t_Q = 40\text{ms}$
- c) $t_Q = 45\text{ms}$
- d) $t_Q = 50\text{ms}$

[View Answer](#)

Answer: a

Explanation: None.

64) Orders are processed in the sequence they arrive if _____ rule sequences the jobs.

- a) earliest due date
- b) slack time remaining
- c) first come, first served
- d) critical ratio

[View Answer](#)

Answer: c

Explanation: None.

65) Which of the following algorithms tends to minimize the process flow time ?

- a) First come First served
- b) Shortest Job First
- c) Earliest Deadline First
- d) Longest Job First

[View Answer](#)

Answer: b

Explanation: None.

66) Under multiprogramming, turnaround time for short jobs is usually _____ and that for long jobs is slightly _____.

- a) Lengthened; Shortened
- b) Shortened; Lengthened
- c) Shortened; Shortened
- d) Shortened; Unchanged

[View Answer](#)

Answer: b

Explanation: None.

67) Which of the following statements are true ? (GATE 2010)

- I. Shortest remaining time first scheduling may cause starvation
- II. Preemptive scheduling may cause starvation
- III. Round robin is better than FCFS in terms of response time

- a) I only
- b) I and III only
- c) II and III only
- d) I, II and III

[View Answer](#)

Answer: d

Explanation:

I) Shortest remaining time first scheduling is a preemptive version of shortest job scheduling. It may cause starvation as shorter processes may keep coming and a long CPU burst process never gets CPU.
II) Preemption may cause starvation. If priority based scheduling with preemption is used, then a low priority process may never get CPU.

III) Round Robin Scheduling improves response time as all processes get CPU after a specified time

ting System

68) The most optimal scheduling algorithm is :

- a) FCFS – First come First served
- b) SJF – Shortest Job First
- c) RR – Round Robin
- d) None of these

[View Answer](#)

Answer: b

Explanation: None.

69) The real difficulty with SJF in short term scheduling is :

- a) it is too good an algorithm
- b) knowing the length of the next CPU request
- c) it is too complex to understand
- d) None of these

[View Answer](#)

Answer: b

Explanation: None.

70) The FCFS algorithm is particularly troublesome for _____.

- a) time sharing systems
- b) multiprogramming systems
- c) multiprocessor systems
- d) Operating systems

[View Answer](#)

Answer: b

Explanation: In a time sharing system, each user needs to get a share of the CPU at regular intervals.

71) Consider the following set of processes, the length of the CPU burst time given in milliseconds :

Process Burst time

P1 6

P2 8

P3 7

P4 3

i) Assuming the above process being scheduled with the SJF scheduling algorithm :

- a) The waiting time for process P1 is 3ms.
- b) The waiting time for process P1 is 0ms.
- c) The waiting time for process P1 is 16ms.
- d) The waiting time for process P1 is 9ms.

[View Answer](#)

Answer: a

Explanation: None.

ii) Assuming the above process being scheduled with the SJF scheduling algorithm :

- a) The waiting time for process P2 is 3ms.
- b) The waiting time for process P2 is 0ms.
- c) The waiting time for process P2 is 16ms.
- d) The waiting time for process P2 is 9ms.

[View Answer](#)

Answer: c

Explanation: None.

iii) Assuming the above process being scheduled with the SJF scheduling algorithm :

- a) The waiting time for process P4 is 3ms.
- b) The waiting time for process P4 is 0ms.
- c) The waiting time for process P4 is 16ms.
- d) The waiting time for process P4 is 9ms.

[View Answer](#)

Answer: b

Explanation: None.

iv) Assuming the above process being scheduled with the SJF scheduling algorithm :

- a) The waiting time for process P3 is 3ms.
- b) The waiting time for process P3 is 0ms.
- c) The waiting time for process P3 is 16ms.
- d) The waiting time for process P3 is 9ms.

[View Answer](#)

Answer: d

Explanation: None.

72) Preemptive Shortest Job First scheduling is sometimes called :

- a) Fast SJF scheduling
- b) EDF scheduling – Earliest Deadline First
- c) HRRN scheduling – Highest Response Ratio Next
- d) SRTN scheduling – Shortest Remaining Time Next

[View Answer](#)

Answer: d

Explanation: None.

73) An SJF algorithm is simply a priority algorithm where the priority is :

- a) the predicted next CPU burst
- b) the inverse of the predicted next CPU burst
- c) the current CPU burst
- d) anything the user wants

[View Answer](#)

Answer: a

Explanation: The larger the CPU burst, the lower the priority.

74) One of the disadvantages of the priority scheduling algorithm is that :

- a) it schedules in a very complex manner
- b) its scheduling takes up a lot of time
- c) it can lead to some low priority process waiting indefinitely for the CPU
- d) None of these

[View Answer](#)

Answer: c

Explanation: None.

75) ‘Aging’ is :

- a) keeping track of cache contents
- b) keeping track of what pages are currently residing in memory
- c) keeping track of how many times a given page is referenced
- d) increasing the priority of jobs to ensure termination in a finite time

[View Answer](#)

Answer: d

Explanation: None.

76) A solution to the problem of indefinite blockage of low – priority processes is :

- a) Starvation
- b) Wait queue
- c) Ready queue
- d) Aging

[View Answer](#)

Answer: d

Explanation: None.

77) Which of the following statements are true ? (GATE 2010)

- i) Shortest remaining time first scheduling may cause starvation
 - ii) Preemptive scheduling may cause starvation
 - iii) Round robin is better than FCFS in terms of response time
- a) i only
 - b) i and iii only
 - c) ii and iii only
 - d) i, ii and iii

[View Answer](#)

Answer: d

Explanation: None.

78) Which of the following scheduling algorithms gives minimum average waiting time ?

- a) FCFS
- b) SJF
- c) Round – robin
- d) Priority

[View Answer](#)

Answer: b

Answer: c

79) What is a short-term scheduler ?

- a) It selects which process has to be brought into the ready queue
- b) It selects which process has to be executed next and allocates CPU
- c) It selects which process to remove from memory by swapping
- d) None of these

Answer: b

80) The primary distinction between the short term scheduler and the long term scheduler is :

- a) The length of their queues
- b) The type of processes they schedule
- c) The frequency of their execution
- d) None of these

Answer: c

81) The only state transition that is initiated by the user process itself is :

- a) block

- b) wakeup
- c) dispatch
- d) None of these

Answer: a

82) In a time-sharing operating system, when the time slot given to a process is completed, the process goes from the running state to the :

- a) Blocked state
- b) Ready state
- c) Suspended state
- d) Terminated state

Answer: b

83) In a multi-programming environment :

- a) the processor executes more than one process at a time
- b) the programs are developed by more than one person
- c) more than one process resides in the memory
- d) a single user can execute many programs at the same time

Answer: c

84) Suppose that a process is in “Blocked” state waiting for some I/O service. When the service is completed, it goes to the :

- a) Running state
- b) Ready state
- c) Suspended state
- d) Terminated state

Answer: b

85) The context of a process in the PCB of a process does not contain :

- a) the value of the CPU registers
- b) the process state
- c) memory-management information
- d) context switch time

Answer: d

86) Which of the following need not necessarily be saved on a context switch between processes ? (GATE CS 2000)

- a) General purpose registers
- b) Translation look-aside buffer
- c) Program counter
- d) All of these

Answer: b

87) Which of the following does not interrupt a running process ? (GATE CS 2001)

- a) A device
- b) Timer

- c) Scheduler process
- d) Power failure

Answer: c

88) Several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called a(n) ____.

- a) Shared Memory Segments
- b) Entry Section
- c) Race condition
- d) Process Synchronization

Answer: c

89) Which of the following state transitions is not possible ?

- a) blocked to running
- b) ready to running
- c) blocked to ready
- d) running to blocked

Answer: a

90. Which process can be affected by other processes executing in the system?

- a) cooperating process
- b) child process
- c) parent process
- d) init process

Answer:a

91. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called

- a) dynamic condition
- b) race condition
- c) essential condition
- d) critical condition

Answer:b

92. If a process is executing in its critical section, then no other processes can be executing in their critical section. This condition is called

- a) mutual exclusion
- b) critical exclusion
- c) synchronous exclusion
- d) asynchronous exclusion

Answer:a

93. Which one of the following is a synchronization tool?

- a) thread
- b) pipe
- c) semaphore
- d) socket

Answer:c

94. A semaphore is a shared integer variable
- a) that can not drop below zero
 - b) that can not be more than zero
 - c) that can not drop below one
 - d) that can not be more than one

Answer:a

95. A section of code within a process that requires access to shared resources and that may not be executed while another process in a corresponding section of code is called

- a) **Critical section** b) Deadlock c) Livelock d) Mutual Exclusion

96. A situation in which two or more processes are unable to proceed because each is waiting for one of the others to do something

- a) Critical section **b) Deadlock** c) Livelock d) Mutual Exclusion

97. A situation in which 2 or more processes continuously change their state in response to the changes their state in response to the changes in the other process without doing any useful work is called

- a) Critical section b) Deadlock **c) Livelock** d) Mutual Exclusion

98. The requirement that when one process is in the critical section that accesses shared resources , no other process may be in a critical section that accesses any of these shared resources is called

- a) Critical section b) Deadlock c) Livelock **d) Mutual Exclusion**

99. The condition in which multiple threads or processes read and write a shared data item and the final result depending on the relative timing of the execution is called

- a) Critical section b) Deadlock **c) Race condition** d) starvation

100. A situation in which a runnable process is overlooked indefinitely by the scheduler;although it is able to proceed,it is never chosen is called

- a) Critical section b) Deadlock c) Race condition **d) starvation**

101. There are _____ requirements for mutual exclusion.

- a)6**
- b)7
- c)5
- d) 8

102. The hardware approaches to mutual exclusion are

- a) Interrupt disabling b) Test and Set instruction c) Exchange instruction **d)All of these**

103. The various properties of machine -instruction approach are

- a) Busy waiting
- b) Starvation
- c) Deadlock
- d)All of these**

104. Semaphore is a variable that has an integer value upon which only _____ operations are defined.
a)1 b)2 c)**3** d)4
105. Binary Semaphore is also known as
a) **Mutex** b)counting semaphore c)general semaphore d)none of these
106. Binary Semaphore takes values 0 and 1 and can be defined by _____ operations.
a)1 b)2 c)**3** d)4
107. Non-binary semaphore is also known as
a) Mutex b)counting semaphore c)general semaphore d)**both**
108. The process that has been blocked the longest and is released first from the queue is a _____ semaphore.
a)weak b)**strong** c)counting d)general
109. A semaphore that does not specify order in which processes are removed from the queue is a _____ semaphore.
a)**weak** b)strong c)counting d)general
110. Which of these is not a combination implemented in synchronization?
a) Blocking send, blocking receive
b) Non-blocking send, blocking receive
c) **Non-blocking receive ,blocking send**
d) Non-blocking send, non-blocking receive
111. _____ Relationship allows private communications link to be set up between 2 processes.
a) **One to one** b)Many to one c)One to many d)Many to many
112. _____ Relationship is useful for client/server interaction.
a) One to one b)**Many to one** c)One to many d)Many to many
113. _____ Relationship allows for one sender and multiple receivers.
a) One to one b)Many to one c)**One to many** d)Many to many

114. _____ relationship allows multiple server processes to provide concurrent service to multiple clients.

- a) One to one b)Many to one c)One to many d)**Many to many**

115. There are _____conditions for readers-writers problem.

- a)1 b)2 c)**3** d)4

116. Which of the following is not a condition for deadlock?

- a)Mutual exclusion b)Hold and wait c)**preemption** d)circular wait

117. Mutual exclusion can be provided by the

- a) mutex locks
b) binary semaphores
c) both (a) and (b)
d) none of the mentioned

Answer:c

Explanation:Binary Semaphores are known as mutex locks.

118.. When high priority task is indirectly preempted by medium priority task effectively inverting the relative priority of the two tasks, the scenario is called

- a) priority inversion
b) priority removal
c) priority exchange
d) priority modification

Answer:a

119. Process synchronization can be done on

- a) hardware level
b) software level
c) both (a) and (b)
d) none of the mentioned

Answer:c

120. A monitor is a module that encapsulates

- a) shared data structures
b) procedures that operate on shared data structure
c) synchronization between concurrent procedure invocation
d) all of the mentioned

Answer:d

121. To enable a process to wait within the monitor,
- a) a condition variable must be declared as condition
 - b) condition variables must be used as boolean objects
 - c) semaphore must be used
 - d) all of the mentioned

Answer:a

- 122) Restricting the child process to a subset of the parent's resources prevents any process from :

- a) overloading the system by using a lot of secondary storage
- b) under-loading the system by very less CPU utilization
- c) overloading the system by creating a lot of sub-processes
- d) crashing the system by utilizing multiple resources

Answer: c

- 123) A parent process calling _____ system call will be suspended until children processes terminate.

- a) wait
- b) fork
- c) exit
- d) exec

Answer: a

- 124) Cascading termination refers to termination of all child processes before the parent terminates _____.

- a) Normally
- b) Abnormally
- c) Normally or abnormally
- d) None of these

Answer: a

- 125) With only one process can execute at a time; meanwhile all other process are waiting for the processor. With more than one process can be running simultaneously each on a different processor.

- a) Multiprocessing, Multiprogramming
- b) Multiprogramming, Uniprocessing
- c) Multiprogramming, Multiprocessing
- d) Uniprocessing, Multiprocessing

Answer: d

- 126) In UNIX, each process is identified by its :

- a) Process Control Block
- b) Device Queue
- c) Process Identifier
- d) None of these

Answer: c

- 127) In UNIX, the return value for the fork system call is _____ for the child process and _____ for the parent process.

- a) A Negative integer, Zero
- b) Zero, A Negative integer
- c) Zero, A nonzero integer
- d) A nonzero integer, Zero

Answer: c

- 128) The child process can : (choose two)

- a) be a duplicate of the parent process
- b) never be a duplicate of the parent process

c) have another program loaded into it d) never have another program loaded into it
Answer: a and c

129) The child process completes execution, but the parent keeps executing, then the child process is known as :

- a) Orphanb) Zombiec) Bodyd) DeadAnswer: b

130) Remote Procedure Calls are used :

- a) for communication between two processes remotely different from each other on the same system
b) for communication between two processes on the same system
c) for communication between two processes on separate systems
d) None of these Answer: c

131) To differentiate the many network services a system supports _____ are used.

- a) Variables b) Sockets c) Ports d) Service names

Answer: c

132) RPC provides a(an) _____ on the client side, a separate one for each remote procedure.

- a) stub b) identifier c) named) process identifier

Answer: a

133) The stub :

- a) transmits the message to the server where the server side stub receives the message and invokes procedure on the server side
b) packs the parameters into a form transmittable over the network
c) locates the port on the server
d) All of these

Answer: d

134) To resolve the problem of data representation on different systems RPCs define _____.

- a) machine dependent representation of data
b) machine representation of data
c) machine-independent representation of data
d) None of these

Answer: c

135) The full form of RMI :

- a) Remote Memory Installation
b) Remote Memory Invocation
c) Remote Method Installation
d) Remote Method Invocation

Answer: d

136) The remote method invocation :

- a) allows a process to invoke memory on a remote object

- b) allows a thread to invoke a method on a remote object
- c) allows a thread to invoke memory on a remote object
- d) allows a process to invoke a method on a remote object

Answer: b

137) A process that is based on IPC mechanism which executes on different systems and can communicate with other processes using message based communication, is called _____.

- a) Local Procedure Call
- b) Inter Process Communication
- c) Remote Procedure Call
- d) Remote Machine Invocation

Answer: c

138) The initial program that is run when the computer is powered up is called :

- a) boot program
- b) bootloader
- c) initializer
- d) bootstrap program

Answer: d

139) How does the software trigger an interrupt ?

- a) Sending signals to CPU through bus
- b) Executing a special operation called system call
- c) Executing a special program called system program
- d) Executing a special program calle interrupt trigger program

Answer: b

140) Scheduling is done so as to :

- a) increase CPU utilization b) decrease CPU utilization
- c) keep the CPU more idle d) None of these

Answer: a

141) Scheduling is done so as to :

- a) increase the throughput
- b) decrease the throughput
- c) increase the duration of a specific amount of work
- d) None of these

Answer: a

142) Turnaround time is :

- a) the total waiting time for a process to finish execution
- b) the total time spent in the ready queue

- c) the total time spent in the running queue
- d) the total time from the completion till the submission of a process

Answer: d

143) Scheduling is done so as to :

- a) increase the turnaround time
- b) decrease the turnaround time
- c) keep the turnaround time same
- d) there is no relation between scheduling and turnaround time

Answer: b

144) Waiting time is :

- a) the total time in the blocked and waiting queues
- b) the total time spent in the ready queue
- c) the total time spent in the running queue
- d) the total time from the completion till the submission of a process

Answer: b

145) Scheduling is done so as to :

- a) increase the waiting time
- b) keep the waiting time the same
- c) decrease the waiting time
- d) None of these

Answer: c

146) Response time is :

- a) the total time taken from the submission time till the completion time
- b) the total time taken from the submission time till the first response is produced
- c) the total time taken from submission time till the response is output
- d) None of these

Answer: b

147) Scheduling is done so as to :

- a) increase the response time
- b) keep the response time the same
- c) decrease the response time
- d) None of these

Answer: c

148) Round robin scheduling falls under the category of :

- a) Non preemptive scheduling

b) Preemptive scheduling

c) None of these

Answer: b

149) With round robin scheduling algorithm in a time shared system,

a) using very large time slices converts it into First come First served scheduling algorithm

b) using very small time slices converts it into First come First served scheduling algorithm

c) using extremely small time slices increases performance

d) using very small time slices converts it into Shortest Job First algorithm

Answer: a

150) The portion of the process scheduler in an operating system that dispatches processes is concerned with :

a) assigning ready processes to CPU

b) assigning ready processes to waiting queue

c) assigning running processes to blocked queue

d) All of these

Answer: a

151) Complex scheduling algorithms :

a) are very appropriate for very large computers

b) use minimal resources

c) use many resources

d) All of these

Answer: a

152) The FIFO algorithm :

a) first executes the job that came in last in the queue

b) first executes the job that came in first in the queue

c) first executes the job that needs minimal processor

d) first executes the job that has maximum processor needs

Answer: b

153) The strategy of making processes that are logically runnable to be temporarily suspended is called :

a) Non preemptive scheduling

b) Preemptive scheduling

c) Shortest job first

d) First come First served

Answer: b

154) What will happen if a non-recursive mutex is locked more than once ?

- a) Starvation
- b) Deadlock
- c) Aging
- d) Signaling

Answer: b

155) A semaphore :

- a) is a binary mutex
- b) must be accessed from only one process
- c) can be accessed from multiple processes
- d) None of these

Answer: c

156) The two kinds of semaphores are : (choose two)

- a) mutex
- b) binary
- c) counting
- d) decimal

Answer: b and c

157) A mutex :

- a) is a binary mutex
- b) must be accessed from only one process
- c) can be accessed from multiple processes
- d) None of these

Answer: b

158) At a particular time of computation the value of a counting semaphore is 7. Then 20 P operations and 15 V operations were completed on this semaphore. The resulting value of the semaphore is : (GATE 1987)

- a) 42
- b) 2
- c) 7
- d) 12

Answer: b

159) A binary semaphore is a semaphore with integer values : (choose two)

- a) 1
- b) -1
- c) 0
- d) 0.5

Answer: a and c

160) The following pair of processes share a common variable X :

Process A

```
int Y;  
A1: Y = X*2;  
A2: X = Y;  
Process B  
int Z;  
B1: Z = X+1;  
B2: X = Z;
```

X is set to 5 before either process begins execution. As usual, statements within a process are executed sequentially, but statements in process A may execute in any order with respect to statements in process B.

i) How many different values of X are possible after both processes finish executing ?

- a) two
- b) three
- c) four
- d) eight

Answer: c

161) T is set to 0 before either process begins execution and, as before, X is set to 5.

Now, how many different values of X are possible after both processes finish executing ?

- a) one
- b) two
- c) three
- d) four

Answer: a

162) Semaphores are mostly used to implement :

- a) System calls
- b) IPC mechanisms

- c) System protection
- d) None of these

Answer: b

163) Spinlocks are intended to provide _____ only.

- a) Mutual Exclusion
- b) Bounded Waiting
- c) Aging
- d) Progress

[View Answer](#)

Answer: b

UNIT-5

PART-B

1. In the scheme, two different parity calculations are carried out and stored in separate blocks on different disks.

- a) RAID Level 4
- b) RAID Level 5
- c) RAID Level 6
- d) RAID Level 3

Answer:c

2. Which of the following is/are the characteristics of RAID architecture.

- i) RAID is set of physical disk drives viewed by the operating system as a single logical drive
 - ii) Data are distributed across the physical drives of an array
 - iii) It is used to store parity information, which guarantees data recoverability in case of disk failure.
- a) i and ii only
 - b) ii and iii only
 - c) i and iii only
 - d) i, ii and iii

Answer:d

3. In policy, when the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again.

- a) Last in first out
- b) Shortest service time first

- c) SCAN
- d) Circular SCAN

Answer:d

4. When a user process issues an I/O request, the operating system assigns a buffer in the system portion of main memory to the operation is called

- a) Double buffer
- b) Single buffer
- c) Linear buffer
- d) Circular buffer

Answer:b

5. If a process is executing in its critical section, then no other processes can be executing in their critical section. This condition is called

- a)mutual exclusion
- b)critical exclusion
- c)synchronous exclusion
- d)asynchronous exclusion

Answer:a

6. When high priority task is indirectly preempted by medium priority task effectively inverting the relative priority of the two tasks, the scenario is called

- a)priority inversion
- b)priority removal
- c)priority exchange
- d)priority modification

Answer:a

7. A sender S sends a message m to receiver R, which is digitally signed by S with its private key. In this scenario, one or more of the following security violations can take place. (I) S can launch a birthday attack to replace m with a fraudulent message. (II) A third party attacker can launch a birthday attack to replace m with a fraudulent message. (III) R can launch a birthday attack to replace m with a fraudulent message.

a)(I) and (II) only

b)(I) only

c)(II) only

d)(II) and (III) only

Answer:b

8. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called?

a) dynamic condition

b) race condition

c) essential condition

d) critical condition

Answer:b

9. If a process is executing in its critical section, then no other processes can be executing in their critical section. This condition is called?

a) mutual exclusion

b) critical exclusion

c) synchronous exclusion

d) asynchronous exclusion

Answer:a

10. When high priority task is indirectly preempted by medium priority task effectively inverting the relative priority of the two tasks, the scenario is called _____

- a) priority inversion
- b) priority removal
- c) priority exchange
- d) priority modification

Answer:a

11. If the swap space is simply a large file, within the file system, _____ used to create it, name it and allocate its space.

- a) special routines must be
- b) normal file system routines can be
- c) normal file system routines cannot be
- d) swap space storage manager is

Answer:b

12..... are small fixed portions which provide greater flexibility which may require large tables or complex structures for their allocation.

- a) Blocks
- b) Columns
- c) Segments
- d) Partitions

Answer:a

13. Which of the following is/are the types of operations that may be performed on the directory.

i) Search ii) Create file iii) Create directory iv) List directory

- a) i, ii and iii only
- b) ii, iii and iv only
- c) i, ii and iv only
- d) i, ii, iii and iv

Answer:c

14. In file organization, a fixed format is used for records where all records are of the same length, consisting of the same number of fixed length fields in a particular order.

- a) pile
- b) sequential
- c) indexed sequential
- d) indexed

Answer:b

15. When a thread waits indefinitely for some resource, but other threads are actually using it is called

- a) Starvation
- b) Demand Paging
- c) Segmentation
- d) None of them

Answer:a

PART-C

1. Contiguous allocation has two problems ____ and ____ that linked allocation solves.

- a) external – fragmentation & size – declaration
- b) internal – fragmentation & external – fragmentation
- c) size – declaration & internal – fragmentation
- d) memory – allocation & size – declaration

Answer: a

2. Which of the following is false about File system manipulation?

- a) Computers can store files on the disk (Primary storage), for long-term storage purpose
- b) Program needs to read a file or write a file.
- c) Operating System provides an interface to the user to create/delete files.
- d) Operating System provides an interface to create the backup of file system.

Answer: a

3. The OS ensures that all access to system resources is controlled. The major activities of an operating system with respect to?

- a) Error handling
- b) Resource Management
- c) Protection
- d) Communication

Answer: c

4. Which of the following program threat, "Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources."

- a) Trojan Horse
- b) Trap Door
- c) Logic bomb
- d) Virus

Answer: a

5. To delete an user along with its home directory, the command is

- a) A userdel -d username
- b) B userdel -D username
- c) C userdel -r username
- d) D userdel -R username

Answer: c

6. Context switching between two threads of execution within the operating system is usually performed by a small assembly language function. In general terms, what does this small function do internally?

- i)Saves the current registers on the stack, then stores the stack pointer in the current thread's control block.
 - ii)Load the stack pointer for the new thread's control block and restore the registers it had saved, from its stack.
 - iii) The inode is a data structure which lists all attributes and points to all the addresses of the disk blocks
 - iv) Progress – The critical sections solution must not impede an entire thread's progress, halting the CPU.
- a)i,ii b) ii,iii c)i,iii d)iii,iv

Answer:a

7. What is the maximum file size supported by a file system with 16 direct blocks, single, double, and triple indirection? The block size is 512 bytes. Disk block numbers can be stored in 4 bytes.

- a)2.0MB b)0.89GB c)1.08GB d)5GB

Answer:c

Number of blocks:

- Direct Blocks = 16 blocks**
- Single Indirect Blocks = $512 / 4 = 128$ blocks**
- Double Indirect Blocks = $128 * 128 = 16384$ blocks**
- Triple Indirect Blocks = $128 * 128 * 128 = 2097152$ blocks**

**Total number of blocks = direct + single + double + triple = 16 + 128 + 16384 + 2097152
= 2113680**

Total number of bytes = 2113680 * 512 = 1.08220416 E 9 = 1.08 GB

8. If the swap space is simply a large file, within the file system, _____ used to create it, name it and allocate its space.

- a) special routines must be
- b) normal file system routines can be
- c) normal file system routines cannot be
- d) swap space storage manager is

Answer:b

9. When a fixed amount of swap space is created during disk partitioning, more swap space can be added only by?

- I) repartitioning of the disk
 - II) adding another swap space elsewhere
- a) only I
 - b) only II
 - c) both I and II
 - d) neither I nor II

Answer:c

10. For swap space created in a separate disk partition where no file system or directory structure is placed, _____ used to allocate and deallocate the blocks.

- a) special routines must be
- b) normal file system routines can be
- c) normal file system routines cannot be

d) swap space storage manager is

Answer:d

UNIT 4

VIRTUAL MEMORY

PART B

1) Let us assume that the user process is of size 2048KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take

- a) **2000 milliseconds**
- b) 1000 milliseconds
- c) 100 milliseconds
- d) 200 milliseconds

Solution: $2048\text{KB} / 1024\text{KB}$ per second

$$= 2 \text{ seconds}$$

$$= 2000 \text{ milliseconds}$$

2) Consider a mechanism that performs demand paging on an operating system. The average device memory access time is M units if the corresponding memory page is accessible in memory, and D units if a page fault is triggered by memory access. It has been measured experimentally that X units are the average time taken for a memory access in the process.

- (A) $(D - M) / (X - M)$
- (B) $(X - M) / (D - M)$**
- (C) $(D - X) / (D - M)$
- (D) (D) $(X - M) / (D - X)$

Solution: Given, average time for a memory access = M units if page hits, and average time for a memory access = D units if page fault occurred.

And total/experimental average time taken for a memory access = X units.

Let page fault rate is p. Therefore,

$$\begin{aligned}\text{Average memory access time} &= (1 - \text{page fault rate}) * \text{memory access time when no page fault} \\ &+ \text{Page fault rate} * \text{Memory access time when page fault}\end{aligned}$$

3) Note that Belady's anomaly is that as the number of assigned frames increases, the page-fault rate can increase. Now, take a look at the following statements:

- S1: Belady's phenomenon suffers from a random page replacement algorithm (where a page chosen at random is replaced)
- S2: The replacement LUR page algorithm is suffering from Belady's anomaly. Which of the following is valid?
- (A) S1 is true, S2 is true
 - (B) S1 is true, S2 is false
 - (C) S1 is false, S2 is true
 - (D) S1 is false, S2 is false

Solution: Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm. For example, if we consider reference string 3 2 1 0 3 2 4 3 2 1 0 4 and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 page faults.

S1: Random page replacement algorithm (where a page chosen at random is replaced) suffers from Belady's anomaly.

-> **Random page replacement algorithm can be any including FIFO so its true**

S2: LRU page replacement algorithm suffers from Belady's anomaly .

-> **LRU doesn't suffer from Belady's anomaly .**

Therefore, **option B** is correct

4) Consider the following page reference string: 7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0, 1
Using the LRU, with 3 frames, show the page load order and number of page faults that would occur.

- (A) 18 Page fault**
- (B) 12 Page fault**

- (C) 24 Page fault
- (D) 32 Page fault

Solution: LRU (18 page faults): 7; 7 2; 7 2 3; 1 2 3; 1 2 5; 3 2 5; 3 4 5; 3 4 6; 7 4 6; 7 1 6; 7 1 0; 5 1 0; 5 4 0; 5 4 6; 2 4 6; 2 3 6; 2 3 0; 1 3 0

5) Let the reference for the page and the window for the working set be c c d b c e c e a d and 4, respectively. The original collection of jobs at the time t=0 Contains pages {a, d, e} where, at the time, a was referenced t=0, d At the time, was referenced t=-1, and e At the time, was referenced t=-2. Determine the average number of page frames.

- (A) 8
- (B) 6
- (C) 5**
- (D) 12

Solution: Window size of working set = 4

Initial pages in the working set window = {e, d, a} = {e, d, a}

Incoming pageccdbceeadTime12345678910Working set window{e,d,a,c}{d,a,c}{a,c,d}{c,d,b}{d,b,c}{d,b,c,e}{b,c,e}{c,e}{c,e,a}{c,e,a,d}Hit/ MissmissmisshitmisshitmisshitmissmissCurrent window size4333343234Incoming pageTimeWorking set windowHit/ MissCurrent window sizec1{e,d,a,c}miss4c2{d,a,c}hit3d3{a,c,d}hit3b4{c,d,b}miss3c5{d,b,c}hit3e6{d,b,c,e}miss4c7{b,c,e}hit3e8{c,e}hit2a9{c,e,a}miss3d10{c,e,a,d}miss4

Total number of page faults = 5

6) Find a device for computers with 40-bit virtual addressing method and 16 kilobytes of page size. If the computer system has a single-level page table per process and needs 48 bits for each page table entry, then the size of the page table per process is megabytes.

- (A) 384**
- (B) 426
- (C) 520
- (D) 628

Solution: No of bits in offset = $\log 16KB = 14$

No of pages = $2^{40 - 14} = 2^{26}$

Size of each PMT entry = $48 / 8 = 6 \text{ B}$

Size of PMT = $2^{26} * 6 = 384 \text{ MB}$

7) Consider a paging scheme that uses a primary memory 1-level page table and a TLB for translating addresses. 100 ns is required for each main memory access and 20 ns for TLB lookup. Each move of a page to/from the disk requires 5000 ns. Assume the TLB hit ratio is 95 percent, 10 percent of the page fault rate. Assume that a dirty page must be written back to the disk before the deadline for 20 percent of the total page faults.

(A) 154.7

(B) 150.4

(C) 154.4

(D) 162

Solution: Given,

1. Main Memory access time: 100
2. TLB lookup time: 20 ns
3. Time to transfer one page to/from disk: 5000 ns
4. TLB hit ratio: 0.95
5. Page fault rate: 0.10
6. 20 % of page faults needs to be written back to disk

Hence, effective memory access time =

$$0.95(20+100)+0.05\{0.90(20+100+100)+0.10[0.80(20+100+5000+100)+0.20(20+100+5000+500+100)]\}$$

= 155.0

8) A framework uses three page frames for the main memory storage of process files. It uses the substitution policy for the First in First out (FIFO) page. Assume all the frames of the page are initially empty. What is the total number of page faults that occur when the page reference string below is being processed? For eg, 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. Calculate the hit ratio.

(A) **0.4**

(B) 0.6

(C) 0.8

(D) 1

9) Find a series of references: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. The number of memory frames is 3. Find out the number of page errors corresponding to Optimal Page Replacement Algorithm

(A) Number of Page fault = 7

(B) Number of Page fault = 5

(C) Number of Page fault = 12

(D) Number of Page fault = 10

10) There are 3 page frames which are initially empty. If the page reference string is 1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6, the number of page faults using the optimal replacement policy is_____.

(A) 5

(B) 6

(C) 7

(D) 8

Solution: In optimal page replacement replacement policy, we replace the place which is not used for longest duration in future.

Given three page frames.

Reference string is **1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6**

Initially, there are three page faults and entries are

1 2 3

Page 4 causes a page fault and replaces 3 (3 is the longest distant in future), entries become

1 2 4

Total page faults = $3+1 = 4$

Pages 2 and 1 don't cause any fault.

5 causes a page fault and replaces 1, entries become

5 2 4

Total page faults = $4 + 1 = 5$

3 causes a page fault and replaces 5, entries become

3 2 4

Total page faults = $5 + 1 = 6$

3, 2 and 4 don't cause any page fault.

6 causes a page fault.

Total page faults = $6 + 1 = 7$

11) Given that he Memory access time = 200 nanoseconds. Average page-fault service time = 8 milliseconds EAT = $(1 - p) \times 200 + p$ (8 milliseconds) = $(1 - p) \times 200 + p \times 8,000,000 = 200 + p \times 7,999,800$ If one access out of 1,000 causes a page fault, find EAT.

(A) 8.2 microseconds

(B) 7.4 microseconds

(C) 6.4 microseconds

(D)7.2 microseconds

12) For instance, in a system with 62 frames, if there is a process of 10KB and another process of 127KB, how many frames are required by the first process and the other process

(A) 6 frames,54 frames

(B) 4 frames,57 frames

(C) 8 frames 62 frames

(D) 12 frames 74 frames

Solution: the first process will be allocated $(10/137)*62 = 4$ frames and the other process will get $(127/137)*62 = 57$ frames.

13) In a paged memory, the page hit ratio is 0.35. The time required to access a page in secondary memory is equal to 100 ns. The time required to access a page in primary memory is 10 ns. The average time required to access a page is

(A) 3.0 ns

(B) 68.5 ns

(C) 68.0 ns

(D) 78.5 ns

Solution: Hit ratio = 0.35

Time (secondary memory) = 100 ns

T(main memory) = 10 ns

Average access time = $h(T_m) + (1 - h)(T_s)$

$$= 0.35 \times 10 + (0.65) \times 100$$

$$= 3.5 + 65$$

$$= 68.5 \text{ ns}$$

14) Consider a computer with 8 Mbytes of main memory and a 128 K cache. The cache block size is 4 K. It uses a direct mapping scheme for cache management. How many different main memory blocks can map onto a given physical cache block ?

(A) 2048

(B) 64

(C) 256

(D) 1022

Solution: The computer with 8 Mbytes of main memory and a 128 K cache. The cache block size is 4K. It uses a direct mapping scheme for cache management. 64 different main memory blocks can map into a given physical cache block.

The cache provides in-memory storage and management for your data.

A cache is a hardware or software component that stores data so that future requests for that data can be served faster; the data stored in a cache might be the result of an earlier computation or a copy of data stored.

The simplest way of associating main memory blocks with cache block is the direct mapping technique.

15) Consider following page trace :4,3,2, 1,4,3,5,4,3,2, 1,5

Number of page faults that would occur if FIFO page replacement algorithm is used with

Number of frames for the Job M=3, will be

(A) 9

(B) 8

(C) 10

(D) 12

Solution:

When M-3

T	1	2	3	4	5	6	7	8	9	10	11	12
Page Sequence	4	3	2	1	4	3	5	4	3	2	1	5
Memory Frame 1st	4	4	4	1	1	1	5	5	5	5	5	5
2nd		3	3	3	4	4	4	4	4	4	1	1
3rd			2	2	2	3	3	3	3	2	2	2
Page fault	y	y	y	y	y	y	y	n	n	y	y	n

Total Page Fault = 9

PART C

- 1) An operating system supports a paged virtual memory. The central processor has a cycle time of 1 microsecond. It costs an additional 1 microsecond to access a page other than the current one. Pages have 1,000 words, and the paging device is a drum that rotates at 3,000 revolutions per minute and transfers 1 million words per second. The following statistical measurements were obtained from the system:
- One percent of all instructions executed accessed a page other than the current page.
 - Of the instructions that accessed another page, 80 percent accessed a page already in memory.
 - When a new page was required, the replaced page was modified 50 percent of the time.
- Calculate the effective instruction time on this system, assuming that the system is running one process only and that the processor is idle during drum transfers

(A) 42 microseconds

(B) 34 microseconds

(C) 68 microseconds

(D) 30 microseconds

Solution: Effective access time = $0.99*(1\text{sec} + 0.008*2) + 0.002*(10000 \text{ sec} + 1000 \text{ sec}) + 0.001 * (10000 \text{ sec} + 1000 \text{ sec})$

$$=(0.99 + 0.016 + 22.0 + 11.0) = 34 \text{ sec}$$

2) Consider the two-dimensional array A: int A[][] = new int[100][100]; where A[0][0] is at location 200 in a paged memory system with pages of size 200. A small process that manipulates the matrix resides in page 0 (locations 0 to 199). Thus, every instruction fetch will be from page 0. For three page frames, how many page faults are generated by the following array-initialization loops, using LRU replacement and assuming that page frame 1 contains the process and the other two are initially empty? **(a)** for (int j = 0; j < 100; j++) for (int i = 0; i < 100; i++) A[i][j] = 0; **(b)** for (int i = 0; i < 100; i++) for (int j = 0; j < 100; j++) A[i][j] = 0; Fill out the following table and provide a detailed elaboration. Without a detailed elaboration, you risk to receive a very low or even 0 points. Problem (a) (b)

(A) (a) A=5000,B=50;(b) A=10,000,B=200

(B) (a) A=500,B=100;(b) A=5000,B=500

(C) (a) A=1000,B=100;(b) A=200,B=500

(D)) (a) A=5000,B=100;(b) A=2000,B=500

Solution: Answer: The system has three page frames for this process. The first is for code and the second and third are for the array. Note that these three page frames do not have to be consecutive, even though the diagram below shows they are. The problem statement does not state the page size unit (i.e., byte or something else) and the size of an int. For simplicity, we just assume each unit of the size 200 page fits an int. In this case, each page can fit two rows or two

columns, depending on how the compiler stores a 2-dimensional array. If the compiler produces all entries of a row consecutively (i.e., row-major), then each page contains two rows. Otherwise, the compiler stores all entries of a column consecutively (i.e., column-major), then each page contains two columns. Because the code is in C style, the array is stored row-by-row (i.e., row-major). Because each entry in the array is only visited once, FIFO and LRU have no difference. In Part (a), the initialization goes column-by-column. Therefore, when a page is loaded into memory, it is only accessed twice, once per row. Refer to the diagram above for the details.

Because the two available page frames are initially empty, once the initialization procedure starts, the first page is loaded in and accessed twice for row 0 and row 1. When the initialization goes to row 2, a page fault occurs and the second page is loaded, which is accessed twice for row 2 and row 3. In this way, just for column 0 for every two rows there is a page fault. To complete initialization for column 0, there will be 50 page faults. Because we have 100 columns, the number of page faults is $5,000 = 50 \times 100$. Problem (a) (b) Answer 5,000 50 For Part (b), the initialization is done row-by-row. Consequently, we have one page fault for every two rows. Because we have 100 rows, the total number of page faults is $50 = 100/2$. CS4411 Operating Systems Final Solutions – Spring 2019 8 If you interpret the size of 200 as 200 bytes and each int requires 4 bytes, then each row has 400 bytes (i.e., 2 pages) and there are $200 = 2 \times 100$ pages for matrix A[][]. For Part (b), A[][] is initialized row-by-row and, as a result, initializing each row generates 2 page faults and the total number of page faults is $200 = 2 \times 100$. For Part (a), A[][] is initialized column-by-column, and, each access to a row will cause a page fault because two page frames can only hold one row. Consequently, for each column there will be 100 page faults. Because there are 100 columns, the total number of page faults is $10,000 = 100 \times 100$.

Problem (a) (b) Answer 10,000 200

3) Consider a demand-paged computer system where the degree of multiprogramming is currently fixed at four. The system was recently measured to determine utilization of the CPU and the paging disk. Three alternative results are shown below. For each case, what is happening? Can the degree of multiprogramming be increased to increase the CPU utilization? Is the paging helping? Match the following

- (i). CPU utilization 13 percent; disk utilization 97 percent a.CPU utilization is high and increase the degree of multiprogramming
- (ii). CPU utilization 87 percent; disk utilization 3 percent b.Thrashing is occurring

(iii). CPU utilization 13 percent; disk utilization 3 percent c. Increase the degree of multiprogramming

(A) **(i)-b,(ii)-a,(iii)-c**

(B) (i)-a,(ii)-b,(iii)-c

(C) (i)-c,(ii)-a,(iii)-b

(D) (i)-a,(ii)-c,(iii)-b

4) A certain computer provides its users with a virtual memory space of 2^{32} bytes. The computer has 2^{22} bytes of physical memory. The virtual memory is implemented by paging, and the page size is 4,096 bytes. A user process generates the virtual address 11123456. Find the corresponding physical location.

(A) **11123456**

(B) 12313511

(C) 10099988

(D) 12235578

Solution:

Given Data

Virtual memory: 2^{32} bytes

Physical memory: 2^{22} bytes

Page size: 4096 bytes = 2^{12} bytes

Given physical address: 11123456

5) Consider a paging system with the page table stored in memory.

(i) If a memory reference takes 200 nanoseconds, how long does a paged memory reference take?

(ii) If we add associative registers, and 75 percent of all page-table references are found in the associative registers, what is the effective memory reference time? (Assume that finding a page-table entry in the associative registers takes zero time, if the entry is there)

(A) 400 nanoseconds; 200 nanoseconds to access the page table and 200 nanoseconds to access the word in memory, Effective access time=250 nanoseconds

(B) 200 nanoseconds; 400 nanoseconds to access the page table and 200 nanoseconds to access the word in memory, Effective access time=200 nanoseconds

(C) 100 nanoseconds; 200 nanoseconds to access the page table and 200 nanoseconds to access the word in memory, Effective access time=400 nanoseconds

(D) 600 nanoseconds; 250 nanoseconds to access the page table and 100 nanoseconds to access the word in memory, Effective access time=450 nanoseconds

6) Suppose we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

(A) 0.000006

(B) 0.000012

(C) 0.000015

(D) 0.000024

7) A page-replacement algorithm should minimize the number of page faults. We can do this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages that are associated with that frame. Then, to replace a page, we search for the page frame with the smallest counter. How many page faults occur for your

algorithm for the following reference string, for four page frames? 1,2,3,4,5,3,4,1,6,7,8,7,8,9,7,8,9,5,4,5,4,2

- (A) 10 page fault
- (B) 14 page fault**
- (C) 7 page fault
- (D) 28 page fault

8) Consider a demand-paging system with a paging disk that has an average access and transfer time of 20 milliseconds. Addresses are translated through a page table in main memory, with an access time of 1 microsecond per memory access. Thus, each memory reference through the page table takes two accesses. To improve this time, we have added an associative memory that reduces access time to one memory reference, if the page-table entry is in the associative memory.

Assume that 80 percent of the accesses are in the associative memory, and that, of the remaining, 10 percent (or 2 percent of the total) cause page faults. What is the effective memory access time?

- (A) 0.4 ms**
- (B) 0.8 ms
- (C) 1.2 ms
- (D) 2.4 ms

Solution: Access Time = $(0.8) * (1 \text{ microsecond}) + (0.18) * (2 \text{ microsecond}) + (0.02) * (20002 \text{ microsecond})$

$$= 401.2 \text{ microsecond}$$

$$= 0.4 \text{ millisecond}$$

9) Consider the two-dimensional array A: **var A: array[1..100] of array[1..100] of integer;** where A[1][1] is at location 200, in a paged memory system with pages of size 200. A small process is in page 0 (location 0 to 199) for manipulating the matrix; thus, every instruction fetch will be from page 0.

For three page frames, how many page faults are generated by the following array-initialization

loops, using LRU replacement, and assuming page frame 1 has the process in it, and the other two are initially empty.

- a. for j:=1 to 100 do
 - for i:= 1 to 100 do
 - A[i][j]:=0;
- b. for i:=1 to 100 do
 - for j:=1 to 100 do
 - A[i][j]:=0;

- (A) 5000 page fault, 50 page fault**
(B) 2500 page fault, 100 page fault
(C) 6000 page fault, 200 page fault
(D) 2400 page fault, 120 page fault

Solution: The array is stored row-major; that is, the first data page contains A[1,1].A[1,2]....A[2,100] and the second page contains A[3,1],A[3,2]....A[4,100] and so on.

- a. The page reference string is 0,1,0,2,0,...,0,49,0,1,0,2,0,...,0,49,...
and thus there will be 5000 page faults.
- b. The page reference string is 0,1,0,1,0,...,0,49
and thus there will be 50 page faults.

10) Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100

3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

- a. 0,430 (i) 2310
- b. 1,10 (ii) 649
- c. 2,500 (iii) 1727
- d. 3,400 (iv) illegal reference, trap to operating system

(A) a- (ii),b-(i),c-(iv),d-(iii)

(B) a- (i),b-(ii),c-(iv),d-(iii)

(C) a- (ii),b-(iv),c-(i),d-(iii)

(D) a- (iIi),b-(i),c-(iv),d-(ii)

Solution:

- a. (ii) $219+430 = 649$
- b. (i) $2300 + 10 = 2310$
- c. (iv) illegal reference, trap to operating system
- d. (iii) $1327+ 400= 1727$

OPERATING SYSTEMS
TW0 MARK QUESTIONS AND ANSWERS

PART-B

1. Define operating system

An operating system is a set of program that controls, co-ordinates and supervises the activities of the computer hardware and software.

2. What is the role of an os?

An OS acts as an interface between the user and the computer. It acts as The manager of the resources of the computer.

3. Write the functions of an OS .

- (i) Memory Management.
- (ii) Processor management.
- (iii) Interrupt Handling.
- (iv) Accounting.
- (v) Automatic job sequencing.
- (vi) Management and control of I/O devices

4. What is the need for an OS?

A medium is needed to communicate between the user and the m/c. An OS acts as a medium of interface

5. What are the characteristics of an OS.

- (i) User friendly .
- (ii) Keep track of the status of each
- (iii) Allows sharing of resources(H/W and S/W).
- (iv) Provides adequate security.
- (v) Protection.

6. What is an interrupt ?

An event external to the currently executing process that causes a change in the normal flow of instruction execution; usually generated by hardware devices external to the CPU.

7.Define cache memory.

Cache memory is random access memory (RAM) that a computer microprocessor can access more quickly than it can access regular RAM. As the microprocessor processes data, it looks first in the cache memory and if it finds the data there (from a previous reading of data), it does not have to do the more time-consuming reading of data from larger memory.

8.What is the nucleus or kernel of an operating system?

Kernel is the part of the OS which directly makes interface with the Hardware system.

9.What are the main functions of the kernel?

To provide mechanism for

- (i) creation and deletion of processes
- (ii) inter process communication
- (iii) synchronization of processes.

10.What are the components of an OS?

OS which is a collection of programs are of 2 types

- (i) control program
- (ii) supervisory program

11.What is multi programming?

The ability of keeping several jobs in the memory at one time, where

The cpu is switched back and forth among them is called as

Multi programming

12.What is the use of Multi Programming ?

Multi programming helps to increase CPU utilization, and to decrease the total time needed to execute the jobs.

13.Illustrate the factors that usually determine the degree of Multi Programming

- (i) The number of Programs residing in Primary memory.
- (ii) Passing of the control of the CPU rapidly between these programs.
- (iii) Protection of user process from one another.

14.What are the Benefits of Multi Programming?

- (i) Improves the System Performance.
- (ii) Allows Time Sharing.
- (iii) Supports multiple simultaneous interactive users

15. Explain what is Multi Processing?

The Simultaneous Processing of a number of Processes by a number of Processors Simultaneously at the same time is Multi Processing.

16. What is the advantage of Multi Processing Systems?

A Multi Processing System is one in which there are more than one CPU, interleaved with each other. So it helps in improving the amount of work done.

17. What are the types of Multi Processing?

- (i) Symmetric Multi Processing
- (ii) Asymmetric Multi Processing.

18. What is Symmetric Multi Processing?

It is one in which each processor runs an identical copy of the OS and these copies communicate with one another as needed.

19. What is Asymmetric Multi Processing?

It is one in which each processor is assigned a specific task. A Master Processor controls the system and the other Processors are allocated work by the Master Processor.

20.what is Time Sharing?

Time Sharing (or Multi tasking) is a logical extension of Multi Programming. It is a form of Multi Programmed OS which operates in an interactive mode with Quick response time.

21.Explain the concept of Time Sharing?

Multiple Jobs are executed by the CPU switching between them, but the switches occur so frequently that the users may interact with each program while it is running.

22. What is the benefit of Time Sharing?

A Time Sharing system allows many users to simultaneously share the computer resources.

23.Define Real Time Systems .

It is another form of OS which are used in environments where a large number of events mostly external to the computer system must be accepted and processed in a short time or within certain deadlines.

24. Give examples of Real Time Application

Ex's are

- (i) Flight Control
- (ii) Real Time Simulation
- (iii) Military Application
- (iv) Petroleum Refinery
- (v) Process Control etc.

25. What is On-Line Processing?

Transferring the contents from the input directly on to the CPU and transferring the Processed contents onto the printer is On-Line Processing.

26. Explain Off-Line Processing ?

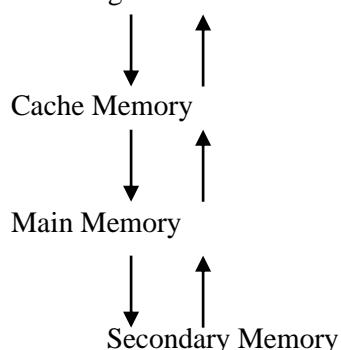
Rather than the CPU reading directly from the input, copying the content into CPU AND PROCESS.

27.What is Memory?

A Memory is the place for storage of data & information (or) it can be Defined as the work area of the computer where the microprocessor finds its data & instructions while the computer is working.

28.State the Memory Hierarchy :

CPU Registers



29.What are the types of memory?

- i) Internal Processor Memory
- ii) Primary or Main Memory
- iii) Secondary/Auxiliary/Backing Store are the types of memory.

30.What is primary memory?

This memory is directly accessible by the processor. It is mainly based on the Integrated circuits.

31.What is a process?

A process is basically a program in execution. It is the unit of work in a Modern operating system.

32.What is meant by a process state?

When a process executes, it changes, its status. This is known as process State.

33.What are the various process states ?

The various process states are

- a. New
- b. Ready
- c. Running
- d. Blocked
- e. exit.

34.How does a process differ from a job?

A process is an active entity with a program counter specifying the next instructions to execute and a set to associated resources, whereas a batchSystem executes jobs.(which is a collection of processes).

35.Differentiate program and a process?

A process is a program in execution(ie) A program is a passive entity,
Where as a process is an active entity.

36.What is process control Block?

Each process is represented in the operating system by a process control Block(PCB) also called a task control block.

37. What is the function of a process control block?.

A (PCB) contains many pieces of information associated with a specific Process. It serves as the repository for any information that may vary From process to process.

38.What are the information contained in a PCB?

A PCB contains pieces of information associated with a specific process,
Namely

- (i) Identifier

- (ii) process state
- (iii) program counter
- (iv) Context data
- (v) CPU scheduling information
- (vi) Memory management information
- (vii) Accounting information
- (viii) I/O status information

39.What are the operations on process?

- a.create a process
- b.destroy a process
- c.suspend a process
- d.resume a process
- e.change the priority of a process
- f.block a process
- g.wakeup a process
- h.dispatch a process
- i.enable a process to communicate with another

40. What are the operation involved in creating a process?

- a. name the process
- b. insert it in the system's known processes list(or) process table.
- c. Determine the process's initial priority
- d. Create the process control block
- e. Allocate the process's initial resource.

41.What is the function of the ready queue?

The ready queue stores threads that aren't currently running, that are capable of resuming execution.

There may be several ready queues for each priority level, depending on the scheduling algorithm.

The scheduler consults the ready queue to determine which process/thread to run next. As the name suggests, the ready queue is a *queue*, in order to schedule fairly.

42. What is the relationship between threads and processes?

A processes is a container for threads, which has it's own memory space. A process may contain one or more threads, which share that memory space, all of the file descriptors and other attributes. The threads are the units of execution within the process, they posess a register set, stack, program counter, and scheduling attributes - *per thread*.

43. . Give the difference between multiprogramming and multiprocessing

A multiprocessing system is a computer hardware configuration that includes more than one independent processing unit. The term multiprocessing is generally used to refer to large computer hardware complexes found in major scientific or commercial applications.

44. Differentiate between pre-emptive and non-pre-emptive scheduling.

In a pre-emptive scheduling approach, CPU can be taken away from a process if there is a need while in a non-pre-emptive approach if once a process has been given the CPU, the CPU cannot be taken away from that process, unless the process completes or leaves the CPU for performing an Input Output. Pre-emptive scheduling is more useful in high priority process which requires immediate response, for example in real time system. While in nonpreemptive systems, jobs are made to wait by longer jobs, but treatment of all processes is fairer.

45. . What is a semaphore? Explain busy waiting semaphores.

A semaphore is a protected variable or abstract data type which constitutes the classic method for restricting access to shared resources such as shared memory in a parallel programming environment.

Weak, Busy-wait Semaphores:

- The simplest way to implement semaphores.
- Useful when critical sections last for a short time, or we have lots of CPUs.
- S initialized to positive value (to allow someone in at the beginning).
- S is an integer variable that, apart from initialization, can only be accessed through atomic and mutually exclusive operations:

wait(s):

```
while (s.value != 0);
```

```
s.value--;
```

signal(s):

```
s.value++;
```

All happens atomically i.e. wrap pre and post protocols.

46. What are the four necessary conditions of deadlock prevention?

Four necessary conditions for deadlock prevention:

1. Removing the mutual exclusion condition means that no process may have exclusive access to a resource. This proves impossible for resources that cannot be spooled, and even with spooled resources deadlock could still occur. Algorithms that avoid mutual exclusion are called non-blocking synchronization algorithms.
2. The "hold and wait" conditions may be removed by requiring processes to request all the resources they will need before starting up. Another way is to require processes to release all their resources before requesting all the resources they will need.
3. A "no preemption" (lockout) 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. Algorithms that allow preemption include lock-free and wait-free algorithms and optimistic concurrency control.

4. The circular wait condition: Algorithms that avoid circular waits include "disable interrupts during critical sections", and "use a hierarchy to determine a partial ordering of resources" and Dijkstra's solution.

47 Define deadlock? Explain the necessary conditions for deadlock to occur.

Deadlock is a situation, in which processes never finish executing and system resources are tied up, preventing other jobs from starting. A process requests resources; if the resources are not available at that time, the process enters a wait state. Waiting processes may never again change state, because the resources they have requested are held by other waiting processes, thereby causing deadlock.

Necessary conditions for deadlock to occur are:

i Mutual exclusion:

At least one resource must be held in a nonshareable mode; that is, only one process at a time can use the resource.

ii. Hold and wait:

A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.

iii. No pre-emption:

Resources cannot be pre-empted; that is, a resource can be released only voluntarily by the process holding it, after the process holding it has its task.

iv. Circular wait:

A set $\{P_0, P_1, \dots, P_n\}$ of waiting processes must exist such that P_0 is waiting for a resource that is held by P_1 , P_1 is waiting for a resource that is held by P_2 , ..., P_{n-1} is waiting for a resource that is held by P_n and P_n is waiting for a resource that is held by P_0 . All four conditions must hold simultaneously for a deadlock to occur and conditions are not completely independent. For example, the circular-wait implies the hold-andwait condition.

48. What are the disadvantages of FCFS scheduling algorithm as compared to shortest job first (SJF) scheduling?

- (i) Waiting time can be large if short requests wait behind the long ones.
- (ii) It is not suitable for time sharing systems where it is important that each user should get the CPU for an equal amount of time interval.
- (iii) A proper mix of jobs is needed to achieve good results from FCFS scheduling.

49. Categorize the CPU scheduling algorithms? Explain non-pre-emptive algorithms?

The various CPU scheduling algorithms are classified as follows:

Non preemptive algorithms: In this method a job is given to CPU for execution as long as the job is non completed the CPU cannot be given to other processes.

There are three types of non preemptive algorithms.

(i) First-come-first-serve (FCFS):

This is simplest CPU scheduling algorithm . With this scheme, the process that requests the CPU at first is given to the CPU at first. The implementation of FCFS is easily managed by with a FIFO queue.

(ii) Shortest-job-first (SJF):

This is also called SPN (shortest process next). In this the burst times of all the jobs which are waiting in the queue are compared. The job which is having the least CPU execution time will be given to the processor at first. In this turnaround time and waiting times are least. This also suffers with starvation. Indefinite waiting time is called as starvation. It is complex than FCFS.

(iii) Priority:

In this algorithm every job is associated with CPU execution time, arrival time and the priority. Here the job which is having the higher priority will be given to the execution at first. This also suffers with starvation. And by using aging technique starvation effect may be reduced.

50. Describe the necessary conditions for Deadlock.

1. Mutual exclusion
2. Hold and wait
3. No preemption
4. Circular wait

51. What is a Process Scheduling? Explain the different sub-functions of Process Scheduling.

Scheduling is a key part of the workload management software which usually perform some or all of:

- Queuing
- Scheduling
- Monitoring
- Resource management
- Accounting

52. Explain the Features of Major scheduling algorithms

1. FCFS – First come first served scheduling
2. Shortest job – First scheduling

3. Priority scheduling
4. Round robin scheduling

FCFS-

1. Process that request the CPU first is allocated CPU first
2. Managed by FIFO queue.
3. Average waiting time is generally long
4. Non preemptive
5. Troublesome for time sharing systems

Shortest job first

1. The process which has the smallest CPU burst time gets first
2. It increases the waiting time of long processes.
3. Problem – difficult to predict the length of next CPU request
4. May be preemptive or non preemptive

Priority

1. Highest priority process gets CPU allocation first
2. The larger the CPU burst, lower the priority
3. Priority can be defined internally or externally
4. Can be preemptive or non-preemptive
5. Problem-starvation, blocking of process
6. Solution – aging – increases the priority

- Round robin Time quantum from 10 100 millisecond is defined
2. Processes are considered in circular queue
 3. CPU allocation is divided among processes accordingly
 4. Performance depends heavily on time quantum
 5. Preemptive

53. What is Starvation?

starvation is a multitasking-related problem, where a process is perpetually denied necessary resources. Without those resources, the program can never finish its task.

54. How binary semaphore is different from general semaphore?

General semaphore also called counting semaphore. It is a variable n whose value equal to no. of items in the buffer whereas binary is more restricted version having 0 and 1 value. Binary semaphore are easier to implement. If there are several processes are waiting for critical section. General semaphore value is extended to more negative but in case of binary semaphore, semaphore value only between 0 and 1.

55. State: Critical section.

Each processes has a segment of code, called critical section in which the process may be changing common variables, called a critical section. It is to design a protocol that the processes can use to cooperate.

56.Why is the capability to relocate processes desirable?

A process may occupy different partitions which means different absolute memory locations during execution (from swapping).Compaction will also cause a program to occupy a different partition which means different absolute memory locations

57.What is the difference between external and internal fragmentation?

External Fragmentation: External Fragmentation happens when a dynamic memory allocation algorithm allocates some memory and a small piece is left over that cannot be effectively used. If too much external fragmentation occurs, the amount of usable memory is drastically reduced. Total memory space exists to satisfy a request, but it is not contiguous.

Internal Fragmentation: Internal fragmentation is the space wasted inside of allocated memory blocks because of restriction on the allowed sizes of allocated blocks. Allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.

58.What are the distinctions among logical relative and physical addresses?

A logical address is a reference to a memory location independent of the current assignment of data to memory; a translation must be made to a physical address before the memory access can be achieved.

A relative address is a particular example of logical address, in which the address is expressed as a location relative to some known point, usually a value in a processor register.

A physical address, or absolute address, is an actual location in main memory.

59.What is the difference between Page and Frame?

In a paging system, programs and data stored on disk are divided into equal, fixed sized blocks called pages, and main memory is divided into blocks of the same size called frames. Exactly one page can fit in one frame. Physical memory is divided into parts called FRAME and logical memory is divided into parts called PAGE.

60.Explain the difference between page and segment?

In segmentation, the address space is typically divided into a preset number of segments like data segment (read/write), code segment(read-only), stack(read/write) etc. And the programs are divided into these segments accordingly. While with paging, the address space is divided into a sequence of fixed size units called "pages".

61.What is the advantage and disadvantage of fix partition?

A fixed partition cannot be moved or expanded; therefore, one advantage would be that you know how much space can be used, and might keep better track of what you are using. The

disadvantage is that it cannot be expanded; once you run out of space it takes a bit of work to get a larger partition, and that may not be possible if the physical space is exceeded.

62. what you mean by thrashing?

In a steady state partially all of main memory will be occupied with process pieces, so that the processor and OS have direct access to as many process as possible .Thus, when the OS brings one piece in, it must throw out another .if it throw out a piece just before it is used, then it just have to go get that piece again almost immediately too much of this lead to a condition called THRASHING,

63. What is the Translation Lookaside Buffer - TLB?

In a cached system, the base addresses of the last few referenced pages is maintained in registers called the TLB that aids in faster lookup. TLB contains those page-table entries that have been most recently used.

64. What is optimal algorithm?

It selects the page for which time to the next reference in the longest.

65. What is page fault?

The processor also update the TLB to include the new page table entry. If the present bit is not set, then the desired page is not in main memory and a memory access fault, called a page fault.

66. What is replacement policy?

When a non-resident page is needed by a process, the OS must decide which resident page is to be placed by the requested page, the part of virtual memory which makes this decision is called the replacement policy.

67. What is first in first out policy?

The pages to be replaced is the “oldest” page in the memory, the one which was loaded before all the other.

68. What is least recently used(LRU)?

The page to be replaced is the one which has not been referenced ,since all the other has been referenced.

69. What is demand paging?

Demand paging only brings pages into main memory when a reference is made to a location on the page. Many page faults when process first started.

70. What is prepaging?

Prepaging brings in more pages than needed more efficient to bring in pages that reside contiguously on the disk.

71. What is clock policy?

-ADDITIONAL bit called user bit.

-when a page is first loaded In memory, the user bit is set to 0.

-when the page is referred ,the user bit is set to 1.

-during the search for replacement,each user bit set to 1 is changed to 0.

72. Define Frame Locking.

– If frame is locked, it may not be replaced

- Kernel of the operating system
- Control structures
- I/O buffers
- Associate a lock bit with each frame

73. Define seek time and latency time?

The time taken by the head to move to the appropriate cylinder or track is called seek time. Once the head is at right track, it must wait until the desired block rotates under the read- write head. This delay is latency time.

74. What are the allocation methods of a disk space?

- Contiguous allocation
- Linked allocation
- Indexed allocation

75. What are the drawbacks of contiguous allocation of disk space?

- Suffers from external fragmentation
- Suffers from internal fragmentation
- Difficulty in finding space for a new file.
- Size of the file is to be declared in advance.

76. What are the disadvantages of linked allocation?

- Used only for sequential access of files.
- Space required for the pointers.
- Reliability.

77. What are the advantages of Indexed allocation?

- No external-fragmentation problem
- Solves the size-declaration problems.
- Supports direct access

78. What are the advantages of Contiguous allocation?

- Supports direct access and sequential access
- Number of disk seeks is minimal

79. What are the advantages of Linked allocation?

No external fragmentation
Size of the file does not need to be declared.

80. How Free-space management is implemented using bit vector?

Free-space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

81. Define rotational latency?

Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head.

82. What are the various Disk-Scheduling algorithms?

- First Come First Served Scheduling
- Shortest Seek Time First Scheduling
- SCAN Scheduling
- C-SCAN Scheduling
- LOOK Scheduling

83. What is a file?

A file is a named collection of related information that is recorded on the secondary storage. Commonly, files represent programs and data. In general, file is a sequence of bits, bytes, lines, or records.

84. List the various file attributes?

Name, Identifier, Type, Location, Size, Protection, Time, date and user identification.

85. What are the various file operations?

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

86. What are the Access methods for files?

Sequential Access

Direct Access

Indexed Access.

87. What is the use of relative block number in direct access?

The relative block number allows the operating system to decide where the file should be placed , and helps to prevent the user from accessing portions of the file system that may not be part of his file.

88. What is Directory?

The device directory or simply known as directory records information- such as name, location, size, and type for all files on that particular partition. The directory can be viewed as a symbol table that translates file names into their directory entries.

89. What are the most common schemes for defining the logical structure of a directory?

Single-Level Directory

Two-level Directory

Tree-Structured Directories

Acyclic-Graph Directories

General Graph Directory.

90.What is a file management system?

It is a way to store and retrieve information from disk drives; controls how files can be created, accessed, retrieved, and deleted.

91.What criteria is important in choosing a file organization?

file volatility

file activity

file size

file queries

data currency.

OPERATING SYSTEMS

PART-C

1.Explain the following

- i) The basic elements of a computer system
- ii) Processor register

2.Discuss Instruction execution with example.

3.Explain in detail

- i) how interrupts are processed.
- ii) how multiple interrupts are handled.

4.Explain in detail about memory hierarchy.

5.Explain the different I/O communication techniques.

6.Discuss the evolution of operating system.

7.Draw and explain Windows Architecture.

8.Explain the essential properties of the following operating systems.

- | | |
|-----------------|----------------|
| a) Batch | b) Interactive |
| c) Time sharing | d) Real Time |
| e) Network | f) parallel |
| g) Distributed | h) clustered |

9.Explain in detail the single thread and multithread process model with diagrams.

10.Compare user level and kernel level threads with necessary diagrams.

11.Explain how micro kernel architecture differs from layered kernel architecture.

12. i) Explain in detail the various reasons involved in process creation and termination.

- ii) Compare mode switching and process switching.

13.With neat diagram explain the five states involved in process model.

14. 1.Explain the following

- i) OS control structures
- ii) Process control structures

15. Describe about Mutual exclusion.
16. Write about Semaphores and give the algorithms wherever necessary.
17. Narrate about producer consumer problem with algorithm
18. Explain about Monitors
19. Narrate about Readers/Writers problem with algorithm
20. Explain in detail about Deadlock Avoidance with Banker's algorithm.
21. Write about the scheduling algorithms.
22. Consider the following set of processes. With the length of CPU burst time given in ms.

Process	Process time	Priority
P1	8	3
P2	3	1(high)
P3	4	4
P4	2	2

- i. Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, RR, Priority.(quantum=2)
- ii. Find the turnaround time, waiting time, average time of each process of scheduling.
23. Explain briefly about the conditions for deadlock to occur.
24. Discuss Deadlock Prevention

25. Explain in detail about the memory partitioning.
26. Discuss virtual memory paging and segmentation with neat sketch.

- 27.Explain in detail about various page replacement algorithms.
 28. Discuss combined paging and segmentation with neat sketch.
 - 29.Write in detail about Linux memory management.
-
30. Explain the various file organization techniques.
 31. Explain about various IO organization model.
 - 32.Write in detail about various file allocation techniques.
 - 33.Enumerate disk scheduling.
 - 34.Explain file directories.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
Department Of Computer Science and Engineering
18CSC205J_OPERATING SYSTEMS: CT3
(Total : 100 Marks) Time : 90 minutes

Part A (30*1=30) : Max Time : 30 minutes

1) During execution of a program if program references a page that is not available in main memory, it is called

- A. Demand Paging
- B. Frame Fault
- C. **page fault**
- D. processor fault

2) _____ has ability to mark an entry invalid through a valid–invalid bit or a special value of protection bits

- A. Secondary memory
- B. Swap Device
- C. **Page Table**
- D. None of the above

3) Below are the advantages of _____

- i. Large virtual memory.
- ii. More efficient use of memory.
- iii. There is no limit on degree of multiprogramming.

- A. **Demand Paging**
- B. Virtual Memory
- C. Page Fault
- D. None of the above

4) Consider the following sequence of addresses : 123,215,600,1234,76,96. If page size is 100, then the reference string is?

- A. 1,2,6,12
- B. 12,21,60,123,7,9
- C. **1,2,6,12,0,0**
- D. 0,2,6,12,0,0

5) Where does the Swap Space exists?

- A. Primary Memory
- B. **secondary memory**
- C. virtual memory
- D. CPU

6) If a process is spending more time paging than executing, then it is called

- A.Thrashing
- B.Page Fault
- C.Processor Fault
- D.Frame Fault

7) Effective access time is directly proportional to _____

- a) page-fault rate
- b) hit ratio
- c) memory access time
- d) none of the mentioned

8) In page replacement the working set model is based on assumption of _____

- a) modularity
- b) locality
- c) globalization
- d) random access

9) Consider a disk queue with requests for I/O to blocks on cylinders.

98 183 37 122 14 124 65 67. Considering FCFS (first cum first served) scheduling, the total number of head movements is, if the disk head is initially at 53 is?

- a) 600
- b) 620
- c) 630
- d) 640

10) Consider a disk queue with requests for I/O to blocks on cylinders.

98 183 37 122 14 124 65 67. Considering SSTF (shortest seek time first) scheduling, the total number of head movements is, if the disk head is initially at 53 is?

- a) 224
- b) 236
- c) 245
- d) 240

11) In the _____ algorithm, the disk arm goes as far as the final request in each direction, then reverses direction immediately without going to the end of the disk.

- a) LOOK
- b) SCAN
- c) C-SCAN
- d) C-LOOK

12) Total head movements in FCFS is _____ ; SSTF is _____

- A.632,232
- B.234,286
- C.331,234
- D.232,632

13) The two main aspects of Virtual memory is _____

- A. Frame allocation and Page Replacement
- B. Page Fault and Thrashing**
- C. Thrashing and Frame allocation
- D. None of the above

14) The algorithm replaces the page which will not be referred for so long in future. This is _____

- A. Optimal Page Replacement algorithm**
- B. LRU
- C. FIFO
- D. None of the above

15) The time taken to locate the disk arm to a specified track where the data is to be read or write.

- A. Seek Time**
- B. Transfer Time
- C. Disk Access Time
- D. None of the Above

16) Local network that uses fibre channel to connect several data storage devices

- A. Storage Area Network**
- B. Network attached Storage
- C. Both A and B
- D. None of the above

17) For systems that place a heavy load on the disk which algorithm can be selected?

- A. SSTF
- B. SCAN
- C. C-SCAN
- D. Both B and C**

18) _____ is series or collection of bits where each bit corresponds to a disk block.

- A. Bit Map**
- B. Bit Block
- C. Worksheet
- D. None of the above

19) _____ allows both parent and child processes to initially share the same pages in memory

- A. fork
- B. Copy-on-Write**
- C. Demand Paging
- D. None of the above

20) _____ works based on Locality of Reference

- A. Working – set Model
- B. Page Fault Frequency Model
- C. Thrashing
- D. None of the above

21) Dividing a disk into sectors that the disk controller can read and write is _____

- A. Low-level formatting
- B. Logical formatting
- C. Partition
- D. None of the above

22) In file attributes, only information stored in a human-readable form is _____

- A. Identifier
- B. Name
- C. Location
- D. Protection.

23) _____ allows a user to transfer files without having an account on the remote system.

- A. Anonymous access
- B. Read access
- C. Write access
- D. None of the above

24) _____ represent an important criterion for evaluating any file system that supports file sharing

- A. Consistency semantics
- B. File Sharing
- C. Remote file systems
- D. None of the above

25) Disk Access Time is calculated by _____

- A. Seek time + Rotational latency + Transfer time
- B. Seek time + Rotational latency
- C. Seek time + Transfer time
- D. None of the above

26. _____ is used to handle bad blocks

- A. Boot strap loader
- B. Sector sparing
- C. Raw disk
- D. None of the above

27) Bitmap or Bit vector can take two values 0 and 1: _____ indicates a free block and _____ indicates that the block is allocated

- A.0,1
- B.1,0**

28) _____ provide a common interface to multiple different file system types

A. Virtual File Systems

- B.Directory
- C.Hashtable
- D.None of the above

29) Consider a main memory with five page frames and the following sequence of page references: 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3. which one of the following is true with respect to page replacement policies First-In-First-out (FIFO) and Least Recently Used (LRU)?

- A. Both incur the same number of page faults**
- B. FIFO incurs 2 more page faults than LRU
- C. LRU incurs 2 more page faults than FIFO
- D. FIFO incurs 1 more page faults than LRU

30) _____ stores the address of the free blocks in the first free block

- A.Linked List
- B.Grouping**
- C.Counting
- D.None of the above

Part B (35*2=70) : Max Time : 60 minutes

1) Consider a system having 64 frames and there are 4 process with the following virtual memory sizes: V(1)=16, V(2)=128, V(3)=64, V(4)=48. The number of frames for each process using Equal allocation and Proportional allocation

- A.16,256**
- B.12,124
- C.256,24
- D.12,16

2) Consider a disk queue with request for input/output to block on cylinders 98, 183, 37, 122, 14, 124, 65, 67 in that order. Assume that disk head is initially positioned at cylinder 53 and moving towards cylinder number 0. The total number of head movements using Shortest Seek Time First (SSTF) and SCAN algorithms are respectively

- A.236 and 252 cylinders
- B.640 and 236 cylinders
- C.235 and 640 cylinders
- D.None of the above**

3) On a disk with 1000 cylinders (0 to 999) find the number of tracks, the disk arm must move to satisfy all the requests in the disk queue. Assume the last request service was at track 345 and the head is moving toward track 0. The queue in FIFO order contains requests for the following tracks :123, 874, 692, 475, 105, 376(Assume SCAN algorithm)

- A.2013
- B.1219**
- C.1967
- D.1507

4) Assuming that the disk head is located initially at 32, find the number of disk moves required with FCFS if the disk queue of I/O block requests are 98, 37, 14, 124, 65, 67 :

- A.310
- B.324
- C.320
- D.321**

5) Which of the following statements is not true about disk-arm scheduling algorithms ?

- A.SSTF (shortest seek time first) algorithm increases performance of FCFS.
- B.The number of requests for disk service are not influenced by file allocation method.**
- C.Caching the directories and index blocks in main memory can also help in reducing disk arm movements.
- D.SCAN and C-SCAN algorithms are less likely to have a starvation problem.

6) If the disk head is located initially at 32, find the number of disk moves required with FCFS if the disk queue of I/O blocks requests are 98, 37, 14, 124, 65, 67.

- A.239
- B.310
- C.321**
- D.325

7.Consider the following page reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1. calculate the total number of page faults when allocated page blocks are 3

- A.9**
- B.19
- C.10
- D.11

8. Consider a reference string: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. the number of frames in the memory is 3. the number of page faults respective to Optimal Page Replacement Algorithm & c. LRU Page Replacement Algorithm is

A.5,6

B.6,5

C.7,5

D.5,7

9. Consider the following disk request sequence for a disk with 100 tracks. 98, 137, 122, 183, 14, 133, 65, 78. Head pointer starting at 54 and moving in left direction. The number of head movements in cylinders using SCAN & C-SCAN scheduling.

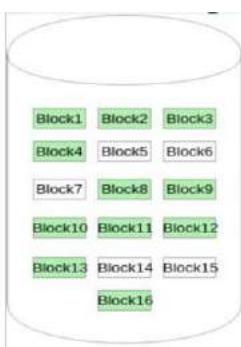
A.237,387

B.387,237

C.257,357

D.None of the above

10. The given instance of disk blocks as below. (where green blocks are allocated) can be represented by a bitmap as



A. 00001110000000110

B. 00001110000000010

C. 00011110000000110

D. 0000111001011110

11. A CPU generates 32-bit virtual address. The page size is 4 KB. The processor has a TLB which can hold a total of 128 page table entries and is a 4 way set associative. The minimum size of TLB tag is

A.11 bits

B.13 Bits

C.15 bits

D.20 bits

12. Match the following and select the appropriate answer

Disk Scheduling Algorithm	Process Description
i)FCFS	a)selects the request with the minimum seek time from the current head position.the request near the disk arm will get executed first.
ii)SCAN	b)The head moves from one end of the disk to the other, servicing requests as it goes– When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
iii)C-SCAN	c)The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues
iv)SSTF	d)The requests are addressed in the order they arrive in the disk queue.

- A.i)-d ; ii)-c;iii)-b;iv)a
B.i)-d ; ii)-c;iii)-a;iv)b
C.i)-b ; ii)-c;iii)-a;iv)d
D.i)-b ; ii)-a;iii)-c;iv)d

13) In Unix File Permission, -rwxr-xr-- represents that the owner has _____ -rwxr-xr-- represents that the group has _____

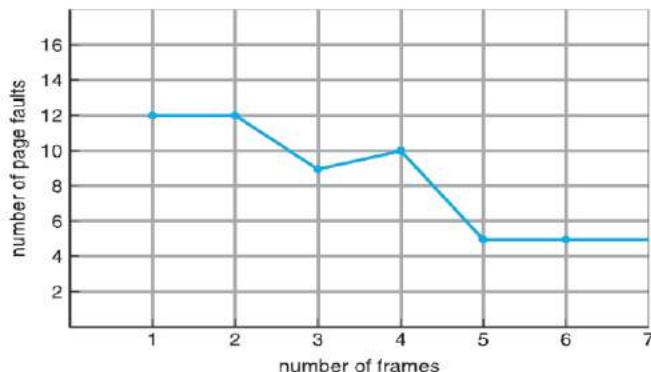
- A. read (r), write (w) and execute (x) permission; read (r) and execute (x) permission, but no write permission
B. read (r) and execute (x) permission, but no write permission; read (r), write (w) and execute (x) permission
C. read (r) only permission; read (r), write (w) and execute (x) permission
D. None of the above

14) VFS in Linux is based upon four key object types. Match appropriately.

Key Object types	Process Description
i)inode	a)representing a directory entry.
ii)file	b)representing a filesystem
iii)superblock	c)representing an individual file
iv)dentry	d)representing an open file

- A.i)-c ; ii)-d;iii)-b;iv)a
B.i)-d ; ii)-c;iii)-a;iv)b
C.i)-b ; ii)-c;iii)-a;iv)d
D.i)-b ; ii)-a;iii)-c;iv)d

15) What does the below graph indicate? What does it infer?



- A. FIFO Page replacement Algorithm; if number of page frames increases page fault also increases
- B. Belady's Anomaly; if number of page frames increases page fault also increases
- C. FIFO Page replacement Algorithm; if number of page frames decreases page fault increases
- D. Belady's Anomaly ; if number of page frames decreases page fault increases

16. Order of request is- (82,170,43,140,24,16,190) and current position of Read/Write head is : 50 total seek time using FCFS and SSTF is

- A.642; 208
- B.208;642
- C.332;208
- D.332;642

17) Which of the below statement about Acyclic graph directory is correct

- i) is a graph with no cycle and allows to share subdirectories and files
- ii) Searching is easy due to different-different paths
- iii) when two programmers are working on a joint project and they need to access files. In this situation acyclic graph can be used.
- iv) In acyclic graph directory cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.

- A.i,ii,ii are correct
- B.ii,iii,iv are correct
- C.i and iv are correct
- D.All are correct

18)The first entry in the stack frame contains _____ and second entry contains _____

- A.The previous value of the frame base; the return address of the function
- B. the return address of the function; The previous value of the frame base
- C. The previous value of the frame base; the next value of the frame base
- D. the next value of the frame base; The previous value of the frame base

19) Match the following commands

Key Object types	Process Description
i)Chmod	a)Change group permission(For files)
ii)Chown	b)Change Mode(For files)
iii)Chgroup	c)Change Owner(For Files)

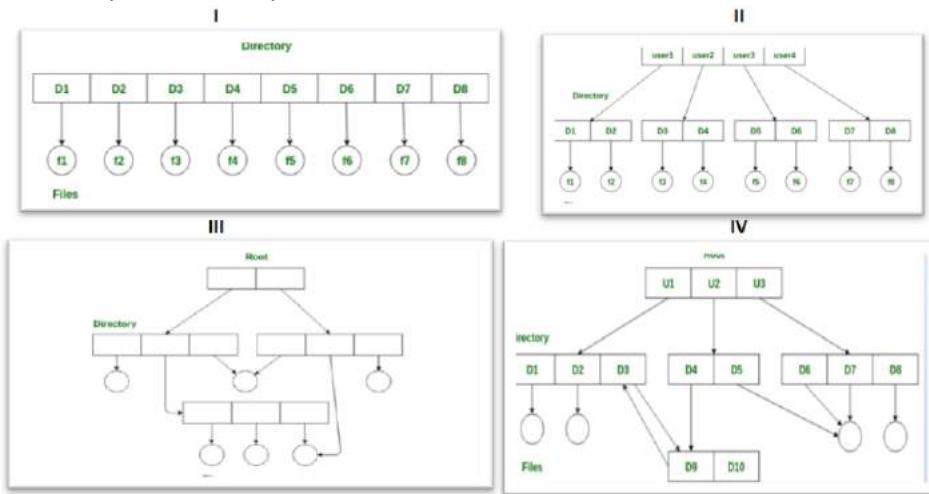
A.i)-b ; ii)-c;iii)-a

B.i)-c ; ii)-b;iii)-a

C.i)-a ; ii)-b;iii)-c

D.i)-a ; ii)-c;iii)-b

20) Identify the directory structure from the below



A. I.Single level Directory Structure;II.Two-level directory;III.Acyclic;IV.General graph directory

B. I.Acyclic;II.Two-level directory;III.Single level Directory Structure;IV.General graph directory

C. I.Acyclic;II.General graph directory;III.Single level Directory Structure;IV.Two-level directory

D. I.General graph directory;II.Acyclic;III.Single level Directory Structure;IV.Two-level directory

21) Consider a typical disk that rotates at 15000 rotations per minute (RPM) and has a transfer rate of 50×10^6 bytes/sec. If the average seek time of the disk is twice the average rotational delay and the controller's transfer time is 10 times the disk transfer time, the average time (in milliseconds) to read or write a 512 byte sector of the disk is _____

A.6.1

B.7.1

C.5.1

D.8.1

22) Suppose the following disk request sequence (track numbers) for a disk with 100 tracks is given: 45, 20, 90, 10, 50, 60, 80, 25, 70. Assume that the initial position of the R/W head is on track 50. The additional distance that will be traversed by the R/W head when the Shortest Seek Time First (SSTF) algorithm is used compared to the SCAN (Elevator) algorithm (assuming that SCAN algorithm moves towards 100 when it starts execution) is _____ tracks

- A.8
- B.9
- C.10
- D.11

23) Suppose a disk has 201 cylinders, numbered from 0 to 200. At some time the disk arm is at cylinder 100, and there is a queue of disk access requests for cylinders 30, 85, 90, 100, 105, 110, 135 and 145. If Shortest-Seek Time First (SSTF) is being used for scheduling the disk access, the request for cylinder 90 is serviced after servicing _____ number of requests

- A.1
- B.2
- C.3
- D.4

24) A virtual memory system uses First In First Out (FIFO) page replacement policy and allocates a fixed number of frames to a process. Consider the following statements:

- P: Increasing the number of page frames allocated to a process sometimes increases the page fault rate.
Q: Some programs do not exhibit locality of reference.

Which one of the following is TRUE?

- A.Both P and Q are true, and Q is the reason for P
- B.Both P and Q are true, but Q is not the reason for P
- C. P is false, but Q is true
- D. Both P and Q are false

25) A system uses FIFO policy for page replacement. It has 4 page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur?

- A.192
- B.195
- C.186
- D.196

26) A process has been allocated 3 page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string): 1, 2, 1, 3, 7, 4, 5, 6, 3, 1. If optimal page replacement policy is used, how many page faults occur for the above reference string?

- A. 10
- B. 9
- C. 7
- D. 8

27) Consider the virtual page reference string 1, 2, 3, 2, 4, 1, 3, 2, 4, 1. On a demand paged virtual memory system running on a computer system that main memory size of 3 pages frames which are initially empty. Let FIFO and OPTIMAL denote the number of page faults under the corresponding page replacements policy. Then

- A. OPTIMAL < FIFO < LRU
- B. OPTIMAL < LRU < FIFO
- C. OPTIMAL = LRU
- D. OPTIMAL = FIFO

28) Consider a disk queue with I/O requests on the following cylinders in their arriving order: 6, 10, 12, 54, 97, 73, 128, 15, 44, 110, 34, 45 . The disk head is assumed to be at cylinder 23 and moving in the direction of decreasing number of cylinders. Total number of cylinders in the disk is 150. The disk head movement using SCAN-scheduling algorithm is:

- A. 172
- B. 173
- C. 151
- D. 228

29) Consider a virtual page reference string 1, 2, 3, 2, 4, 2, 5, 2, 3, 4. Suppose LRU page replacement algorithm is implemented with 3 page frames in main memory. Then the number of page faults are _____.

- A. 7
- B. 5
- C. 9
- D. 10

30. Consider the following page reference string :1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6
Which of the following options, gives the correct number of page faults related to LRU, FIFO, and optimal page replacement algorithms respectively, assuming 05 page frames and all frames are initially empty ?

- A. 7, 10, 8
- B. 10, 14, 8
- C. 7, 10, 7
- D. 8, 10, 7

31.Identify the statements that are true for “Global replacement”

- i)Process selects a replacement frame from the set of all frames; one process can take a frame from another
- ii)process execution time can vary greatly
- iii)allows a high-priority process to increase its frame allocation at the expense of a low-priority process
- iv)the number of frames allocated to a process does not change

- A.i,ii,iii
- B.ii,iii,iv
- C.i,iii,iv
- D.All statements are true

32) NFS is standard _____ & CIFS is standard _____

- A. Windows protocol; UNIX client-server file sharing protocol
- B. **UNIX client-server file sharing protocol; Windows protocol**
- C. Windows protocol; Windows protocol
- D. UNIX client-server file sharing protocol; UNIX client-server file sharing protocol

33) The action of moving a process out from main memory to secondary memory is called _____ & the action of moving a process out from secondary memory to main memory is called _____

- A.**Swap out, Swap in**
- B.Swap in, Swap out
- C.Counting ,Gropuing
- D.Grouping, Counting

34) Are the below statement true to control Thrashing

- i) If actual rate is lower than lower bound, decrease the number of frames
- ii)If actual rate is larger than upper bound, increase the number of frames.

- A.TRUE
- B.FALSE

35) A computer system supports 32-bit virtual addresses as well as 32-bit physical addresses. Since the virtual address space is of the same size as the physical address space, the operating system designers decide to get rid of the virtual memory entirely. Which one of the following is true?

- A.Efficient implementation of multi-user support is no longer possible
- B. The processor cache organization can be made more efficient now
- C. CPU scheduling can be made more efficient now
- D. Hardware support for memory management is no longer needed**

Unit 4 -5 MCQ

Accessing a page marked as invalid causes a -----

- a) Page swap
- b) **Page fault**
- c) swap space
- d) Bit space

A page fault occurs

- a) **when the page is not in the memory**
- b) when the page is in the memory
- c) when the process enters the blocked state
- d) when the process is in the ready stat

When a process is swapped in, its pages are not swapped in all at once. Rather they are swapped in only when the process needs them .This particular method is called as

- a) **Lazy swapper**
- b) Busy Swapper
- c) Swapper
- d) Smart Swapper

PFF stands for -----

- a) Page Find Frequency
- b) **Page Fault Frequency**
- c) Peak Fault Frequency
- d) Peak Find Frequency

.Which is a technique to efficiently copy data resources in a computer system

- a) **Copy-on-write**
- b) Swapping
- c) Thrashing
- d) Paging

The two methods how LRU page replacement policy can be implemented in hardware are:

- a) Counters & register
- b) RAM & Registers
- c) **Stack & Counters**
- d) Registers and Additional memory

Operating system supports different page replacement policy. From the given below option which is not a valid page replacement policy?

- a) Least Recently Used
- b) First in first out
- c) **Currently used policy**
- d) Optimal page replacement policy

Sub routine, stack, symbol table and main program comes under

- a. **Logical view**
- b. Physical view

- c. Users view
- d. Segment Table

The page-replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

- a) least frequently used
- b) most frequently used**
- c) Dynamically used
- d) frequently used

Memory allocation based on Process size is called as

- a) Equal Allocation
- b) Dynamic allocation
- c) Proportional allocation**
- d) Static allocation

-----allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process

- a) Global replacement**
- b) Local replacement
- c) Uniform replacement
- e) Non-Uniform replacement

_____ has the lowest fault rate of all the page replacement algorithms.

- a) Optimal page replacement algorithm**
- b) LRU replacement algorithm
- c) FIFO
- d) Counting based

The situation where the processor spends most of its time swapping process pieces rather than executing instructions is called:

- a) Paging
- b) The Principle of Locality
- c) Thrashing**
- d) Swapping

Working set model for page replacement is based on the assumption of

- a) modularity
- b) locality**
- c) globalization

random access

If the quantity of frames designated to each process decreases, as a result the page-fault rate increases, thus _____ process execution.

- a. increasing
- b. decreasing
- c. slowing
- d. no changing

The set of tracks that are at one arm position make up a

- a) magnetic disks
- b) electrical disks
- c) assemblies
- d) cylinders**

Consider the scenario where a system has 210 frames that are accessible and 8 process are effectively running in the system, and in the event that you consider utilizing equal allocation algorithm, each procedure will get ____ frames evenly and remaining ____ frames will be pushed to the buffer pool

- a. 3,0
- b. 21,0
- c. 26,2**
- d. 14,8

A system has 250 frames and three process with size 10 kb,20 kb and 40kb are actively running in the system. Using proportional allocation algorithm we would split 250 frames between three processes as ___, _____ and _____ individually.

- a. 35,72,143
- b. 35,71,142**
- c. 83,83,83
- d. 83,83,84

In a system , there are 200 frames which can be effectively utilised . A process with size 100 kb enters in to the system, now the system has to allocate frames for that process. Which frame allocation algorithm would be the best to serve this situation ?

Priority allocation

Proportional allocation

System can randomly allocate

Equal allocation algorithm.

Which of the following is in correct order of size, smallest first?

- a) Cylinder, track, sector
- b) Cylinder, sector, track
- c) Sector, track, cylinder**
- d) Sector, cylinder, track

Magnetic disks provide the bulk of secondary storage for modern computer systems. Disk consists of various parts like

cylinders,platters,sectors,track,read-write head,arm, arm assembly.

The heads of the magnetic disk are attached to a _____ that moves all the heads as a unit.

- a) Spindle
- b) Disk Arm**
- c) Track
- d) Cylinder

"For a multiprogramming system with many processes, the disk queue may often have several pending requests. Thus, when one request is completed, the operating system chooses pending request to service next based on disk scheduling algorithms. In the _____ algorithm, the disk arm starts at one end of the disk and moves toward the other end, servicing requests till the other end of the disk. At the other end, the direction is reversed and servicing continues"

- a) LOOK
- b) SCAN**
- c) C-SCAN
- d) C-LOOK

Consider a disk queue with requests for I/O to blocks on cylinders.
98 183 37 122 14 124 65 67. Considering SSTF (shortest seek time first) scheduling, the total number of head movements is, if the disk head is initially at 53 is?"

- a) 224
- b) 236**
- c) 245
- d) 240

"Consider there are 100 row shops available in Street numbered from 1 to 100 in order, where each shop sells unique item. Mr. Mital is now purchasing an item at shop number 25 and has list of 8 remaining items which are available in shop number 53, 44, 95, 86, 22, 14, 36, 65 respectively. If Mr. Mital purchases items according to the order in the list find the number of shops he has to cross."

- a) 220**
- b) 55
- c) 650
- d) 8

"Consider a file consists of 58 bytes. The read/write head is at the end of file. To read 9th byte, direct access will require _____ number of access."

- a) 1
- b) 49
- c) 50
- d) 47

"In file system the information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editors and compilers usually access files in this fashion. The method described _____ access method of file."

- a) Sequential
- b) Direct**
- c) Random
- d) Structured

"Considering the linux file attributes from the output of ls command :
drwxrwxrwx 1 abcd abcd 104 Mar 9 11:15 AXBSUMS
What is the type of this file"

- a) Regular file
- b) directory file**
- c) FIFO file
- d) socket file

Consider the directory structures in the operating system, organizes the files and directory into a hierarchical manner. The single-level directory structure has?

- a) all directories must have unique names
- b) all files must have unique names**
- c) all files must have unique owners
- d) all files have unique permission

Suppose File Allocation Table consists of a file with length n blocks and starting from location b then the file occupies blocks b, b + 1, b + 2, ..., b + n-1. The directory for each file indicates the starting block address and length of the area allocated for the file. Contiguous allocation has two problems _____ and _____ that linked allocation solves.

- a) external – fragmentation & size – declaration**
- b) internal – fragmentation & external – fragmentation
- c) size – declaration & internal – fragmentation
- d) memory – allocation & size – declaration

"Consider a system maintaining a free space in which blocks 4, 5, 8, 9, 11, 12, 13, 22, and 23 are free and the rest of the blocks are allocated.

"

- a) 000011001101110000000011**
- b) 00011001101110000000011
- c) 11110011001000111111100
- d) 111001001

Considering the linux file attributes from the output of ls command, answer the following questions

drw-r-x--x 1 pqr pqr 73 Feb 4 21:53 MD5SUMS

i. What is the file permission given to other category of users who does not belongs to the same group of the owner

- a) Read, write and execute
- b) Read and execute
- c) Only execute permission**
- d) Write and execute

Consider a system maintaining free space available in disk. Consider a disk where blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 1718, 25, 26, and 27 are free and the rest of the blocks are allocated.

In a linked – grouping how many free space groupings will be available

- a) 1
- b) 4
- c) 11

OPERATING SYSTEMS

Chapter 1 Topics :

Operating systems Objectives and functions, Gaining the role of operating systems, The evolution of OS, Major achievements, OS Design considerations for multiprocessor and multicore.

Process Concept : Processes PCB, Threads – overview and its benefits, process scheduling : Scheduling queues, Schedulers, Context switch, Operations on processes – Process creation, process termination, understanding fork(), wait(), exit()

Interprocess Communication- Shared memory, message passing, pipe(), . Understanding the need of IPC.

Process synchronization :

Background, critical section problem, understanding the race condition and the need for the process synchronization.

1. Operating Systems :

Various Definitions of OS includes

- An interface between user and hardware
- Operating System - Operating systems are those programs that interface the machine with the applications programs. The main function of these systems is to dynamically allocate the shared system resources to the executing programs.
- A program that controls the execution of application programs

Main objectives of an OS

- Convenience - Make the computer system convenient to use
- Efficiency - OS allows the system resources to be used in efficient manner
- Ability to evolve – An OS should be constructed in such a way that it permits the effective testing and development.

OS Services/Functions

Briefly, the OS typically provides services in the following areas:

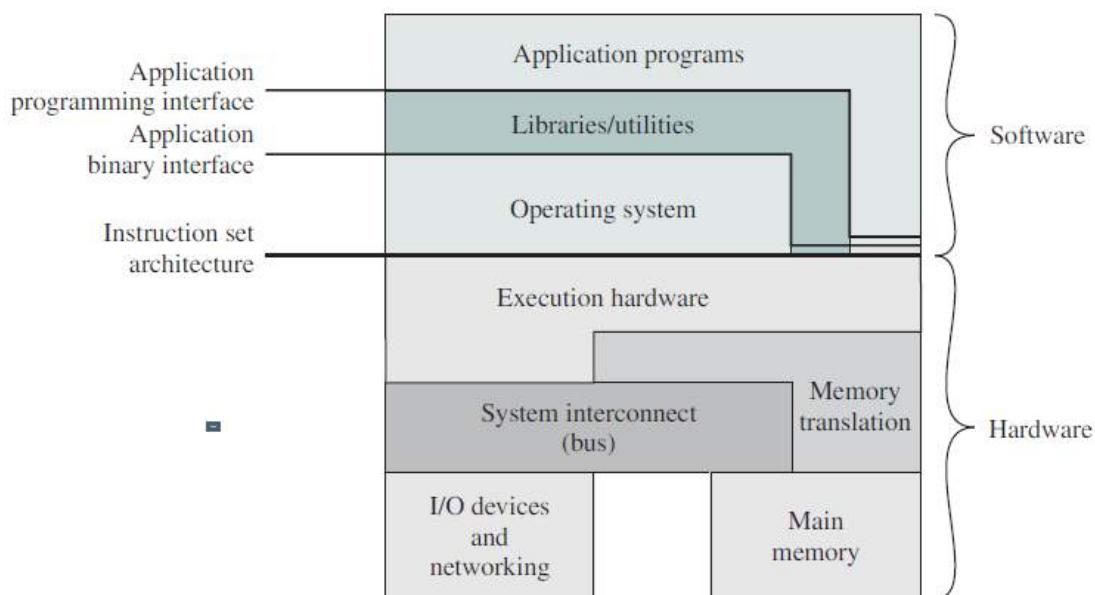
- **Program development:** The OS provides a variety of facilities and services, such as editors and debuggers, to assist the programmer in creating programs. and are referred to as application program development tools.
- **Program execution:** A number of steps need to be performed to execute a program. Instructions and data must be loaded into main memory, I/O devices and files must be initialized, and other resources must be prepared. The OS handles these scheduling duties for the user.
- **Access to I/O devices:** Each I/O device requires its own peculiar set of instructions or control signals for operation. The OS provides a uniform interface that hides these details so that programmers can access such devices using simple reads and writes.
- **Controlled access to files:** For file access, the OS must reflect a detailed understanding of not only the nature of the I/O device (disk drive, tape drive) but also the structure of the data contained in the files on the storage medium.
- **System access:** For shared or public systems, the OS controls access to the system as a whole and to specific system resources. The access function must provide protection of resources and data from unauthorized users and must resolve conflicts for resource contention.
- **Error detection and response:** A variety of errors can occur while a computer system is running. These include internal and external hardware errors, such as a memory error, or a device failure or malfunction; and various software errors, such as division by zero, attempt to access forbidden memory location, and inability of the OS to grant the request of an application.
- **Accounting:** A good OS will collect usage statistics for various resources and monitor performance parameters such as response time.

Role of Operating System :

A computer is a set of resources for the movement, storage, and processing of data and for the control of these functions. **The OS is responsible for managing these resources.**

a. The Operating System as a User/Computer Interface

The hardware and software used in providing applications to a user can be viewed in a layered or hierarchical fashion, as depicted in Figure . The user of those applications, the end user, generally is not concerned with the details of computer hardware. Thus, the end user views a computer system in terms of a set of applications. An application can be expressed in a programming language and is developed by an application programmer. If one were to develop an application program as a set of machine instructions that is completely responsible for controlling the computer hardware, one would be faced with an overwhelmingly complex undertaking.



The above figure indicates three key interfaces in a typical computer system:

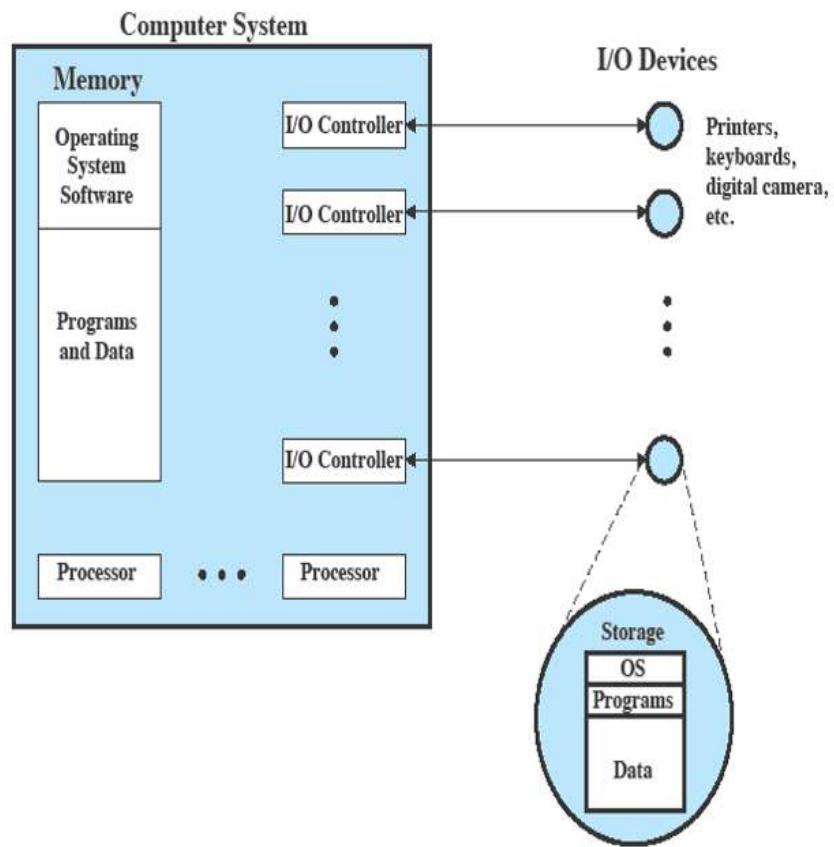
- **Instruction set architecture (ISA)** : The ISA defines the repertoire of machine language instructions that a computer can follow. This interface is the boundary between hardware and software. Note that both application programs and utilities may access the ISA directly. For these programs, a subset of the instruction repertoire is available (user ISA). The OS has access to additional machine language instructions that deal with managing system resources (system ISA).
- **Application binary interface (ABI)** : The ABI defines a standard for binary portability across programs. The ABI defines the system call interface to the operating system and the hardware resources and services available in a system through the user ISA.
- **Application programming interface (API)** : The API gives a program access to the hardware resources and services available in a system through the user ISA supplemented with high-level language (HLL) library calls.

B. Operating system as a resource manager

Figure suggests the main resources that are managed by the OS. A portion of the OS is in main memory. This includes the **kernel , or nucleus , which contains** the most frequently used

functions in the OS and, at a given time, other portions of the OS currently in use. The remainder of main memory contains user programs and data. The memory management hardware in the processor and the OS jointly control the allocation of main memory, as we shall see. The OS decides when an I/O device can be used by a program in execution and controls access to and use of files.

The processor itself is a resource, and the OS must determine how much processor time is to be devoted to the execution of a particular user program.



Evolution of Operating Systems

A major OS will evolve over time for a number of reasons:

- **Hardware upgrades plus new types of hardware:** The use of graphics terminals and page-mode terminals instead of line-at-a-time scroll mode terminals affects OS design. For example, a graphics terminal typically allows the user to view several applications at the same time through “windows” on the screen. This requires more sophisticated support in the OS.
- **New services:** In response to user demand or in response to the needs of system managers, the OS expands to offer new services.
- **Fixes: Any OS has faults.** These are discovered over the course of time and fixes are made. Of course, the fix may introduce new faults. The need to change an OS regularly places certain requirements on its design. The OS has a long series in evolution.

1. Serial Processing

With the earliest computers, from the late 1940s to the mid-1950s, the programmer interacted directly with the computer hardware; there was no OS. These computers were run from a console consisting of display lights, toggle switches, some form of input device, and a printer. Programs in machine code were loaded via the input device (e.g., a card reader). If an error halted the program, the error condition was indicated by the lights. If the program proceeded to a normal completion, the output appeared on the printer.

These early systems presented two main problems:

- **Scheduling :** Most installations used a hardcopy sign-up sheet to reserve computer time
- **Setup time:** A single program, called a **job**, could involve loading the compiler plus the high-level language program (source program) into memory, saving the compiled program (object program) and then loading and linking together the object program and common functions. Each

of these steps could involve mounting or dismounting tapes or setting up card decks. If an error occurred, the hapless user typically had to go back to the beginning of the setup sequence. Thus, a considerable amount of time was spent just in setting up the program to run. This mode of operation could be termed *serial processing*, reflecting the fact that users have access to the computer in series.

2. Simple Batch Processing

Monitor point of view: The monitor controls the sequence of events. For this to be so, much of the monitor must always be in main memory and available for execution. That portion is referred to as the **resident monitor**. The rest of the monitor consists of utilities and common functions that are loaded as subroutines to the user program at the beginning of any job that requires them. The monitor reads in jobs one at a time from the input device (typically a card reader or magnetic tape drive). As it is read in, the current job is placed in the user program area, and control is passed to this job. When the job is completed, it returns control to the monitor, which immediately reads in the next job. The results of each job are sent to an output device, such as a printer, for delivery to the user.

Early computers were very expensive, and therefore it was important to maximize processor utilization. The wasted time due to scheduling and setup time was unacceptable.

To improve utilization, the concept of a batch OS was developed. It appears that the first batch OS (and the first OS of any kind) was developed in the mid-1950s by General Motors for use on an IBM 701.

The central idea behind the simple batch-processing scheme is the use of a piece of software known as the **monitor**.

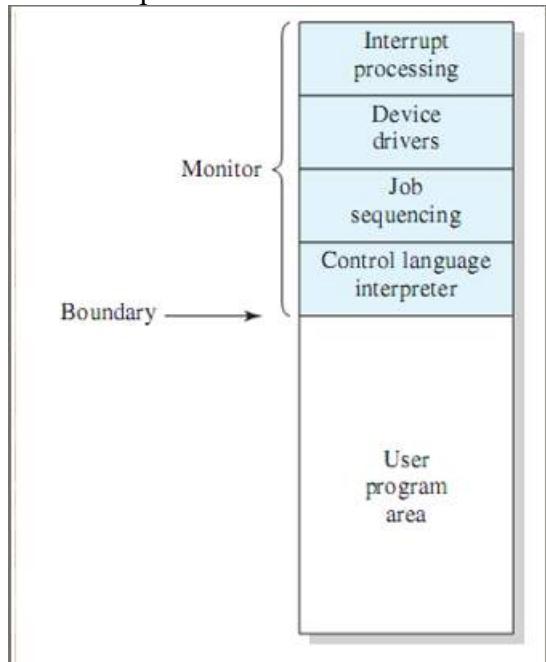
With this type of OS, the user no longer has direct access to the processor. Instead, the user submits the job on cards or tape to a computer operator, who batches the jobs together sequentially and places the entire batch on an input device, for use by the monitor. Each program is constructed to branch back to the monitor when it completes processing, at which point the monitor automatically begins loading the next program.

Job Control Language

The monitor performs a scheduling function: A batch of jobs is queued up, and jobs are executed as rapidly as possible, with no intervening idle time. With each job, instructions are included in a primitive form of **job control language (JCL)**. This is a special type of programming language used to provide instructions to the monitor. A simple example is that of a user submitting a program written in the programming language FORTRAN plus some data to be used by the program. All FORTRAN instructions and data are on a separate punched card or a separate record on tape. In addition to FORTRAN and data lines, the job includes job control instructions, which are denoted by the beginning \$. The monitor, or batch OS, is simply a computer program.

Memory protection: While the user program is executing, it must not alter the memory area containing the monitor.

Timer: A timer is used to prevent a single job from monopolizing the system. The timer is set at the beginning of each job. If the timer expires, the user program is stopped, and control returns to the monitor.



Privileged instructions: Certain machine level instructions are designated privileged and can be executed only by the monitor. If the processor encounters such an instruction while executing a user program, an error occurs causing control to be transferred to the monitor. Among the privileged instructions are I/O instructions, so that the monitor retains control of all I/O devices.

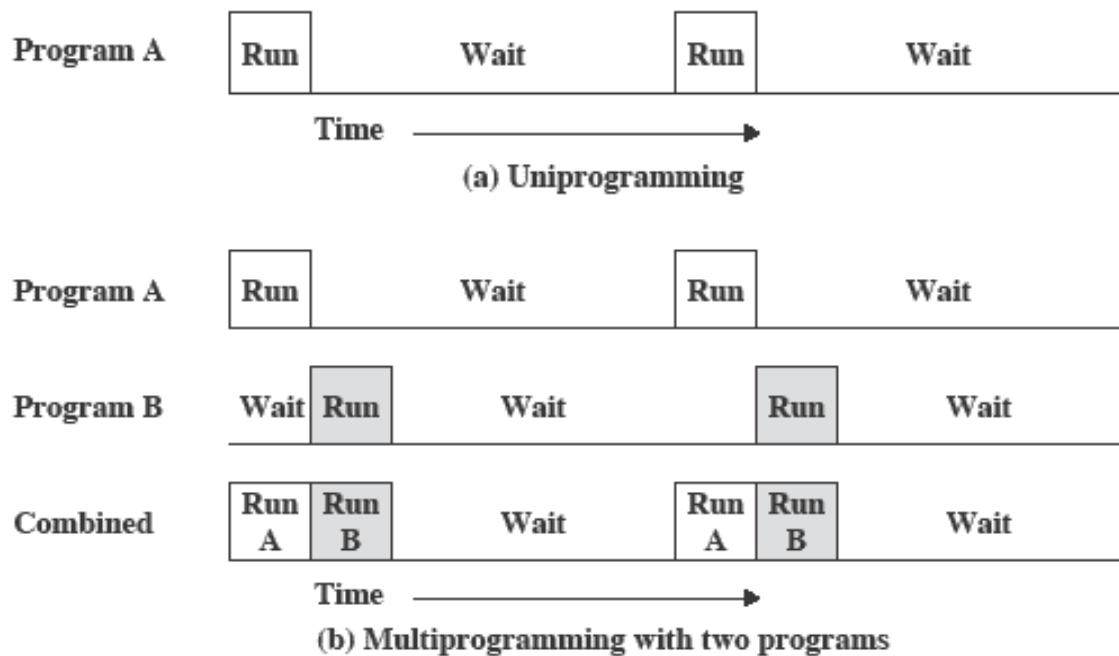
Interrupts: Early computer models did not have this capability.

Modes of Operation

Considerations of memory protection and privileged instructions lead to the concept of modes of operation. A user program executes in a **user mode**, in which certain areas of memory are protected from the user's use and in which certain instructions may not be executed. The monitor executes in a system mode, or what has come to be called **kernel mode**, in which **privileged instructions may be executed** and in which protected areas of memory may be accessed.

3. Multi programmed systems

Even with the automatic job sequencing provided by a simple batch OS, the processor is often idle. The problem is that I/O devices are slow compared to the processor. The processor spends a certain amount of time executing, until it reaches an I/O instruction; it must then wait until that I/O instruction concludes before proceeding



- There must be enough memory to hold the OS (resident monitor) and one user program
- When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O
- Multiprogramming
 - also known as multitasking
 - memory is expanded to hold three, four, or more programs and switch among all of them

4. Time-Sharing Systems

In multiprogrammed batch systems, the user cannot interact with the program during its execution. In time-sharing or multitasking systems, multiple jobs are executed by the CPU switching between them, but the switching occurs so frequently that the user may interact with each program while it is running. Time-sharing operating systems:

- uses CPU scheduling and multiprogramming,

- uses time-slice mechanism,
- allows interactive I/O,
- allows many users to share the computer simultaneously.

5. Personal Computer (PC) /Desktop Systems

A computer system dedicated to a single user is referred to as a PC. In the first PCs, the operating system was neither multiuser nor multitasking (eg. MS-DOS). The operating system concepts used in mainframes and minicomputers, today, are also used in PCs (eg. UNIX, Microsoft Windows NT, Macintosh OS).

6. Parallel Systems/Multiprocessor systems

Parallel systems have more than one processor. In multiprocessor systems (tightly coupled), processors are in close communication, such as they share computer bus, clock, memory or peripherals.

7. Clustered Systems

- Like parallel systems, clustered systems gather together multiple CPUs to accomplish computational work.
- Clustered systems differ from parallel systems, however, in that they are composed of two or more individual systems coupled together.
- The definition of the term clustered is **not concrete**; the general accepted definition is that clustered computers share storage and are closely linked via LAN networking.
- Clustering is usually performed to provide **high availability**.

8. Distributed Systems

Distributed systems also have more than one processor distributed in different geographical locations. Each processor has its local memory. Processors communicate through communication lines (eg. Telephone lines, high-speed bus, etc.). Processors are referred to as sites, nodes, computers depending on the context in which they are mentioned. Multicomputer systems are loosely coupled. Example applications are e-mail, web server, etc.

9. Real-time Systems

Real-time systems are special purpose operating systems. They are used when there are rigid time requirements on the operation of a processor or the flow of data, and thus it is often used as a control device in a dedicated application (eg. fuel injection systems, weapon systems, industrial control systems, ...). It has well defined, fixed time constraints. The processing must be done within the defined constraints, or the system fails.

Two types:

Hard real-time systems guarantee that critical tasks complete on time.

Soft real-time systems, a critical real-time task gets priority over other tasks, and the task retains that priority until it completes.

10. Handheld Systems

Handheld systems include **Personal Digital Assistants(PDAs)**, such as Palm-Pilots or Cellular Telephones with connectivity to a network such as the Internet. They are usually of limited size due to which most handheld devices have a small amount of memory, include slow processors, and feature small display screens.

Major Advances

Operating systems are among the most complex pieces of software ever developed. The following provides four major theoretical advances in the development of operating systems:

- Processes
- Memory management
- Information protection and security
- Scheduling and resource management

Process :

Three major lines of computer system development created problems in **timing and synchronization** that contributed to the development of the concept of the process: multiprogramming batch operation, time sharing, and real-time transaction systems. As we have seen, multiprogramming was designed to keep the processor and I/O devices, including storage devices, simultaneously busy to achieve maximum efficiency. The key mechanism is this: In response to signals indicating the completion of I/O transactions, the processor is switched among the various programs residing in main memory.

A second line of development was **general-purpose time sharing**. Here, the key design objective is to be responsive to the needs of the individual user and yet, for cost reasons, be able to support many users simultaneously. These goals are compatible because of the relatively slow reaction time of the user

A third important line of development has been **real-time transaction processing** systems. In this case, a number of users are entering queries or updates against a database. An example is an airline reservation system. The key difference between the transaction processing system and the time-sharing system is that the former is limited to one or a few applications, whereas users of a time-sharing system can engage in program development, job execution, and the use of various applications.

Memory management

The needs of users can be met best by a computing environment that supports modular programming and the flexible use of data. System managers need efficient and orderly control of storage allocation. The OS, to satisfy these requirements, has five principal storage management responsibilities:

- Process isolation: The OS must prevent independent processes from interfering with each other's memory, both data and instructions.
- Automatic allocation and management: Programs should be dynamically allocated across the memory hierarchy as required. Allocation should be transparent to the programmer. Thus, the programmer is relieved of concerns relating to memory limitations, and the OS can achieve efficiency by assigning memory to jobs only as needed.
- Support of modular programming: Programmers should be able to define program modules, and to create, destroy, and alter the size of modules dynamically.
- Protection and access control: Sharing of memory, at any level of the memory hierarchy, creates the potential for one program to address the memory space of another. This is desirable when sharing is needed by particular applications. At other times, it threatens the integrity of programs and even of the OS itself. The OS must allow portions of memory to be accessible in various ways by various users.
- Long-term storage: Many application programs require means for storing information for extended periods of time, after the computer has been powered down.

Information Protection and Security

The growth in the use of time-sharing systems and, more recently, computer networks has brought with it a growth in concern for the protection of information. The nature of the threat that concerns an organization will vary greatly depending on the circumstances. However, there are some general-purpose tools that can be built into computers and operating systems that

support a variety of protection and security mechanisms. In general, we are concerned with the problem of controlling access to computer systems and the information stored in them.

Much of the work in security and protection as it relates to operating systems can be roughly grouped into four categories:

- Availability: Concerned with protecting the system against interruption.
- Confidentiality: Assures that users cannot read data for which access is unauthorized.
- Data integrity: Protection of data from unauthorized modification.
- Authenticity: Concerned with the proper verification of the identity of users and the validity of messages or data.

Scheduling and resource management

- Key responsibility of an OS is managing resources
- Resource allocation policies must consider: efficiency, fairness and differential responsiveness

Multiprocessor and multicore design considerations:

In an SMP system, the kernel can execute on any processor, and typically each processor does self-scheduling from the pool of available processes or threads. The kernel can be constructed as multiple processes or multiple threads, allowing portions of the kernel to execute in parallel. The SMP approach complicates the OS. The OS designer must deal with the complexity due to sharing resources (like data structures) and coordinating actions (like accessing devices) from multiple parts of the OS executing at the same time. Techniques must be employed to resolve and synchronize claims to resources.

An SMP operating system manages processor and other computer resources so that the user may view the system in the same fashion as a multiprogramming uniprocessor system. A user may construct applications that use multiple processes or multiple threads within processes without regard to whether a single processor or multiple processors will be available. Thus, a multiprocessor OS must provide all the functionality of a multiprogramming system plus additional features to accommodate multiple processors. The key design issues include the following:

- **Simultaneous concurrent processes or threads:** Kernel routines need to be reentrant to allow several processors to execute the same kernel code simultaneously. With multiple processors executing the same or different parts of the kernel, kernel tables and management structures must be managed properly to avoid data corruption or invalid operations.
- **Scheduling:** Any processor may perform scheduling, which complicates the task of enforcing a scheduling policy and assuring that corruption of the scheduler data structures is avoided. If kernel-level multithreading is used, then the opportunity exists to schedule multiple threads from the same process simultaneously on multiple processors.
- **Synchronization:** With multiple active processes having potential access to shared address spaces or shared I/O resources, care must be taken to provide effective synchronization. Synchronization is a facility that enforces mutual exclusion and event ordering.
- **Memory management:** In addition to basic memory management need, the OS needs to exploit the available hardware parallelism to achieve the best performance.

Reliability and fault tolerance: The OS should provide graceful degradation in the face of processor failure. Because multiprocessor OS design issues generally involve extensions to solutions to multiprogramming uniprocessor design problems

Multicore OS Considerations

Current multicore vendors offer systems with up to eight cores on a single chip. With each succeeding processor technology generation, the number of cores and the amount of shared and dedicated cache memory increases, so that we are now entering the era of “many-core” systems.

The design challenge for a many-core multicore system is to efficiently harness the multicore processing power and intelligently manage the substantial on-chip resources efficiently. A central concern is how to match the inherent parallelism of a many-core system with the performance requirements of applications. The potential for **parallelism** in fact exists at three levels in contemporary multicore system.

First, there is **hardware parallelism** within each core processor, known as instruction level parallelism, which may or may not be exploited by application programmers and compilers. Second, there is the potential for **multiprogramming and multithreaded execution** within each processor. Finally, there is the potential for a **single application** to execute in **concurrent** processes or threads across multiple cores.

In essence, then, since the advent of multicore technology, OS designers have been struggling with the problem of how best to extract parallelism from computing workloads.

Process

Introduction

A process can be thought of as a program in execution. A process will need certain resources—such as CPU time, memory, files, and I/O devices—to accomplish its task. These resources are allocated to the process either when it is created or while it is executing.

- A process is the unit of work in most systems.
- Systems consist of a collection of processes:
 - Operating-system processes execute system code, and
 - user processes execute user code.
- All these processes may execute concurrently. Although traditionally a process contained only a single thread of control as it ran, most modern operating systems now support processes that have multiple threads.

Processes

Early computer systems allowed only one program to be executed at a time. This program had complete control of the system and had access to all the system's resources. In contrast, current-day computer systems allow multiple programs to be loaded into memory and executed concurrently. This evolution required firmer control and more compartmentalization of the various programs; and these needs resulted in the notion of a process, **which is a program in execution**.

A process is the unit of work in a modern time-sharing system. The more complex the operating system is, the more it is expected to do on behalf of its users. Although its main concern is the execution of user programs, it also needs to take care of various system tasks that are better left outside the kernel itself. A system therefore consists of a collection of processes: operating system processes executing system code and user processes executing user code. Potentially, all these processes can execute concurrently, with the CPU (or CPUs) multiplexed among them. By switching the CPU between processes, the operating system can make the computer more productive.

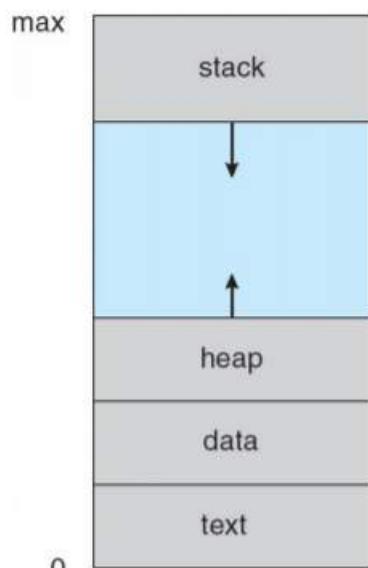
An operating system executes a variety of programs:

- Batch system – jobs
- Time-shared systems – user programs or tasks

We use the terms job and process almost interchangeably.

Process – a program in execution; process execution must progress in sequential fashion. A process is a program in execution. A process is more than the program code, which is sometimes known as the **text section**. A process also includes: **program counter, program stack, data section**. The process stack contains temporary data (such as function parameters, return addresses, and local variables), and a data section, which contains global variables. A process may also include a **heap**, which is memory that is dynamically allocated during process run time. The structure of a process in memory is shown in Figure.

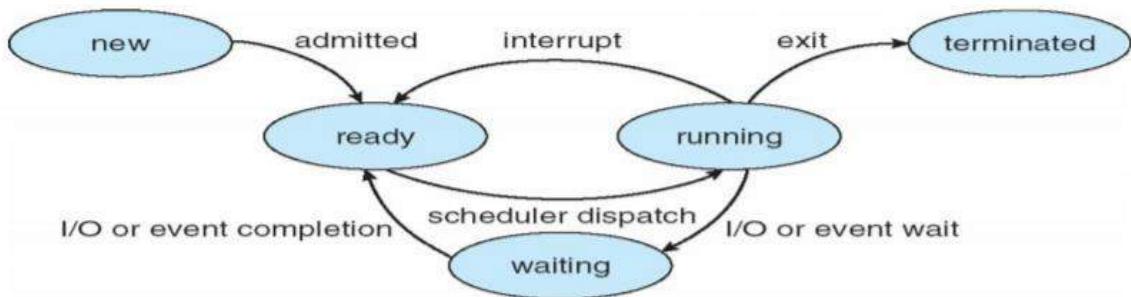
We emphasize that a program by itself is not a process; a program is a passive entity, such as a file containing a list of instructions stored on disk (often called an executable file). Whereas a **process is an active entity**, with a program counter specifying the next instruction to execute and a set of associated resources. A program becomes a process when an executable file is loaded into memory. It is also common to have a process that spawns many processes as it runs.



Process State

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. Each process may be in one of the following states:

- o New : The process is being created.
- o Running : Instructions are being executed.
- o Waiting : The process is waiting for some event to occur (such as an I/O completion or reception of a signal).



- o Ready : The process is waiting to be assigned to a processor.
- o Terminated : The process has finished execution.

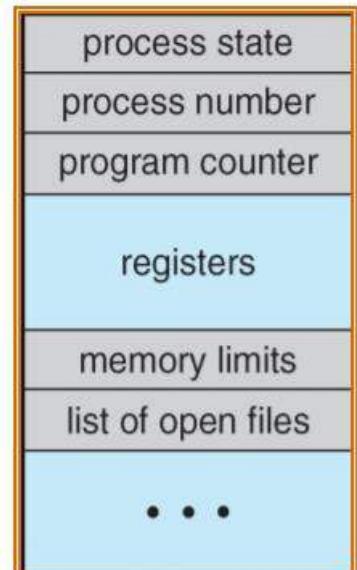
These names are arbitrary, and they vary across operating systems. The states that they represent are found on all systems, however.

Process Control Block (PCB)

Each process is represented in the operating system by a process control block (PCB)—also called a task control block. A PCB is shown in Figure.

It contains many pieces of information associated with a specific process: i. Process state ii. Program counter iii. CPU registers iv. CPU scheduling information v. Memory-management information vi. Accounting information vii. I/O status information

- Process state: - The state may be new, ready, running, waiting, halted, and so on.
- Program counter: The counter indicates the address of the next instruction to be executed for this process.
- CPU registers: The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information.
- CPU-scheduling information: This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters
- Memory-management information: This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.



Process Control Block (PCB)

- Accounting information: This information includes the amount of CPU and real time used, time limits, account members, job or process numbers, and so on.
- I/O status information: This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

In brief, the PCB simply serves as the repository for any information that may vary from process to process

Context switch

As we know interrupts cause the operating system to change a CPU from its current task and to run a kernel routine. Such operations happen frequently on general-purpose systems. When an interrupt occurs, **the system needs to save the current context of the process currently running on the CPU so that it can restore that context when its processing is done, essentially suspending the process and then resuming it.**

The context is represented in the PCB of the process; it includes the value of the CPU registers, the process state , and memory-management information. Generically, we perform a state save of the current state of the CPU, be it in kernel or user mode, and then a state restore to resume operations. **Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process is known as a context switch.** When a context switch occurs, the kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run. Context-switch time is pure overhead, because the system does no useful work while switching.

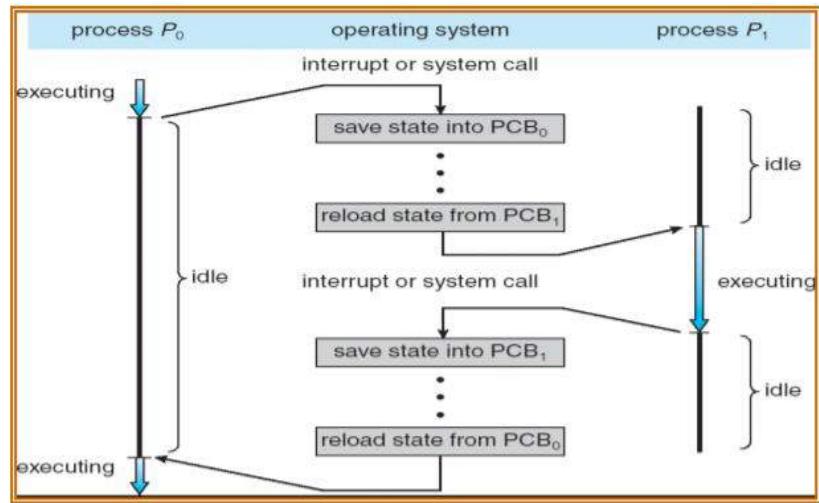
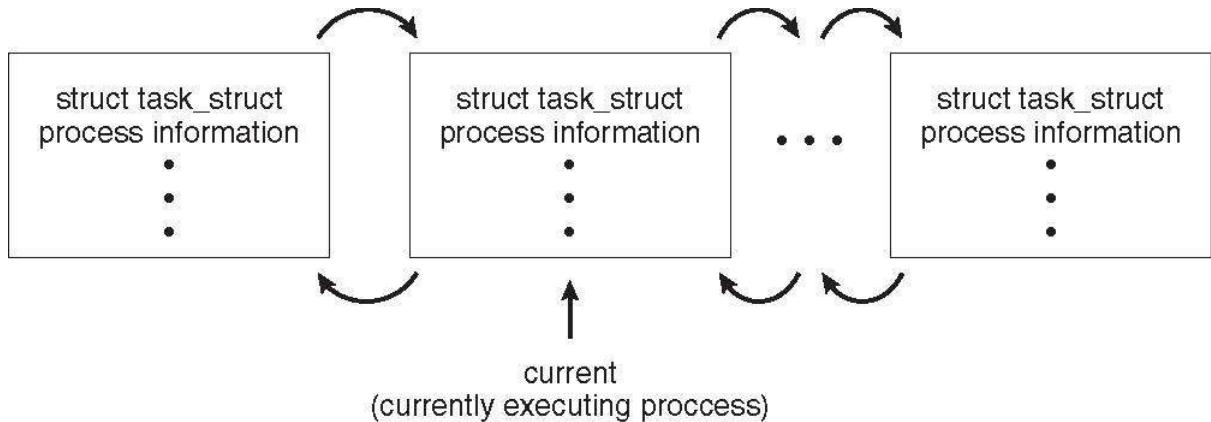


Diagram showing CPU Switch From Process to Process

Process representation in Linux

Represented by the C structure **task_struct**. This maintains PCB like data structure.

```
pid t_pid; /* process identifier */
long state; /* state of the process */
unsigned int time_slice /* scheduling information */
struct task_struct *parent; /* this process's parent */
struct list_head children; /* this process's children */
struct files_struct *files; /* list of open files */
struct mm_struct *mm; /* address space of this process */
```



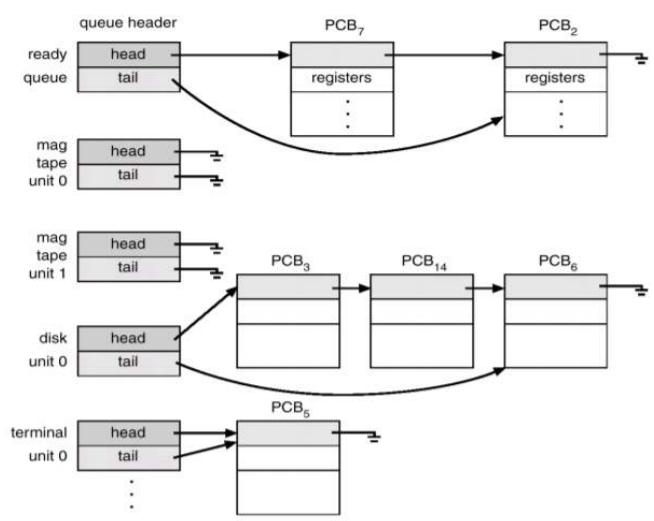
Process Scheduling

A uniprocessor system can have only one running process. If more process exists, the rest must wait until the CPU is free and can be rescheduled. The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization. The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running. To meet these objectives, the process scheduler selects an available process (possibly from a set of several available processes) for program execution on the CPU.

Process Scheduling Queues

Process migration between the various queues: i. Job queue ii. Ready queue iii. Device queues
As processes enter the system, they are put into a job queue, which consists of all processes in the system. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the ready queue. This queue is generally stored as a linked list. A ready-queue header contains pointers to the first and final PCBs in the list and each PCB includes a pointer field that points to the next PCB in the ready queue.

The Operating system also includes other queues. When a process is allocated the CPU, it executes for a while and eventually quits, is interrupted, or waits for the occurrence of a particular event, such as the completion of an I/O request. Suppose the process makes an I/O request to a dedicated tape drive, or to a shared device, such as a disk. Since there are many processes in the system, the disk may be busy with the I/O request of some other process. The process therefore may have to wait for the disk. The list of processes waiting for a particular I/O device is called a device queue. Each device has its own device queue. A common representation for a discussion of process scheduling is a queuing diagram.



Each rectangular box represents a queue. Two types of queues are present: the ready queue and a set of device queues. The circles represent the resources that serve the queues, and the arrows indicate the flow of processes in the system. A new process is initially put in the ready queue. It waits there until it is selected for execution, or is dispatched. Once the process is allocated the CPU and is executing, one of several events could occur:

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

A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated

Schedulers

A process migrates among the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler. Often, in a batch system, more processes are submitted than can be executed immediately. These processes are spooled to a mass-storage device (typically a disk), where they are kept for later execution.

- o The long-term scheduler, or job scheduler, selects processes from this pool and loads them into memory for execution.
- o The short-term scheduler, or CPU scheduler, selects from among the processes that are ready to execute and allocates the CPU to one of them.

The primary distinction between these two schedulers lies in frequency of execution. The short-term scheduler must select a new process for the CPU frequently. A process may execute for only a few milliseconds before waiting for an I/O request. Often, the short-term scheduler executes at least once every 100 milliseconds. Because of the short time between executions, the short-term scheduler must be fast.

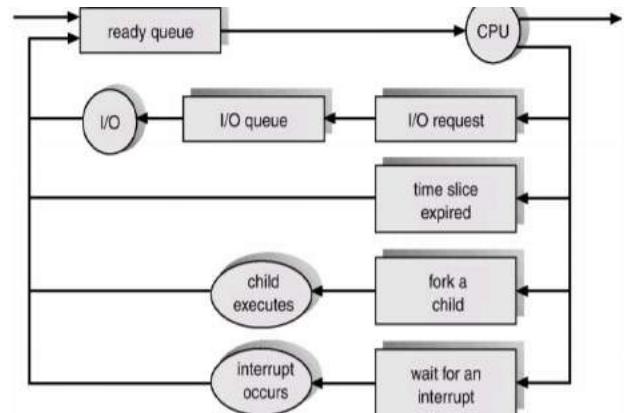
The long-term scheduler executes much less frequently; minutes may separate the creation of one new process and the next. The long-term scheduler controls the degree of multiprogramming (the number of processes in memory). Because of the longer interval between executions, the long-term scheduler can afford to take more time to decide which process should be selected for execution. It is important that the long-term scheduler make a careful selection.

In general, most processes can be described as either I/O bound or CPU bound.

- o An I/O-bound process is one that spends more of its time doing I/O than it spends doing computations.
- o A CPU-bound process, in contrast, generates I/O requests infrequently, using more of its time doing computations.

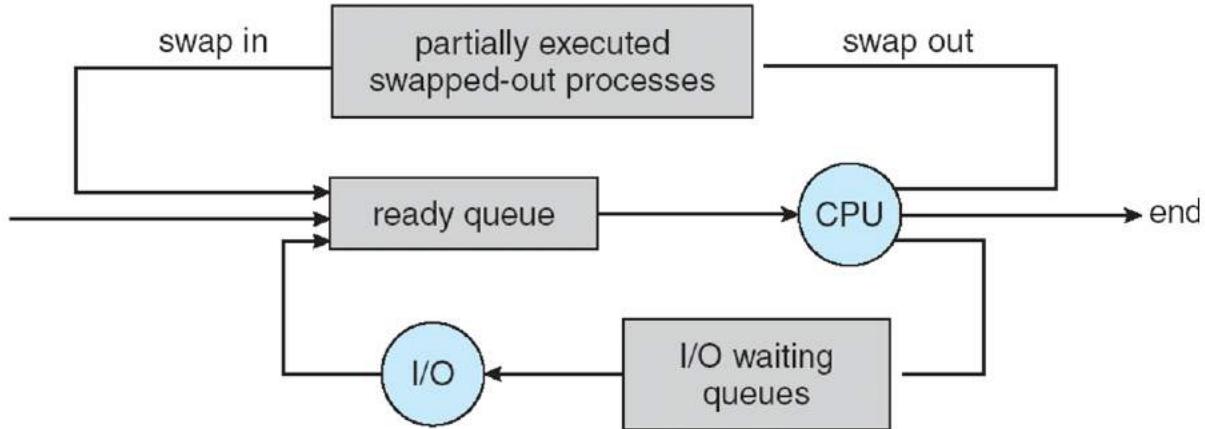
It is important that the long-term scheduler select a good process mix of I/O-bound and CPU-bound processes.

Some operating systems, such as time-sharing systems, may introduce an additional, intermediate level of scheduling. The Medium-term scheduler is diagrammed in Figure. The key idea behind a medium-term scheduler is that sometimes it can be advantageous to remove processes from memory (and from active contention for the CPU) and thus reduce the degree of



Queuing-diagram representation of Process Scheduling

multiprogramming. At sometime later, the process can be reintroduced into memory, and its execution can be continued where it left off. This scheme is called swapping. The process is swapped out, and is later swapped in, by the medium-term scheduler. Swapping may be necessary to improve the process mix or because a change in memory requirements has overcommitted available memory, requiring memory to be freed up.



Operations on Processes :

The processes in the system can execute concurrently, and they must be created and deleted dynamically. Thus, the operating system must provide a mechanism (or facility) for process creation and termination

- Process Creation

Parent process creates children processes, which, in turn create other processes, forming a tree of processes.

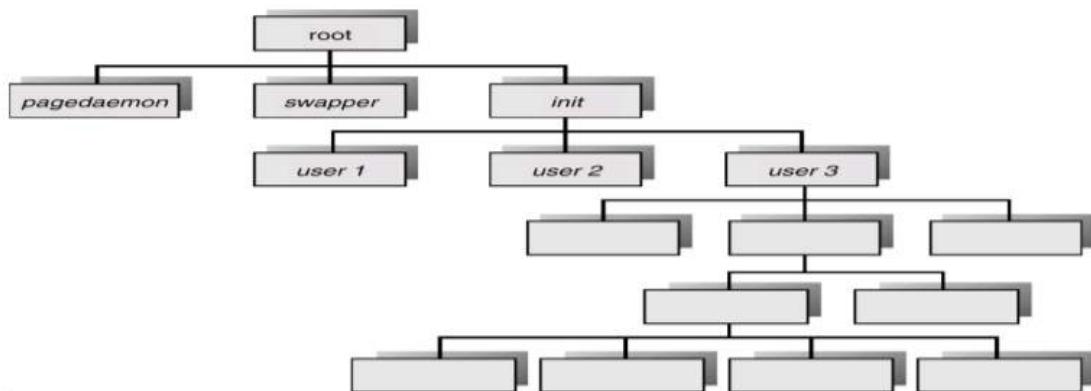
Resource sharing

- Parent and children share all resources.
- Children share subset of parent's resources.
- Parent and child share no resources.

- Execution

- Parent and children execute concurrently.
- Parent waits until children terminate.

A process may create several new processes, via a create-process system call, during the course of execution. The creating process is called a parent process, and the new processes are called the children of that process.



A Tree of Processes On A Typical UNIX System

Each of these new processes may in turn create other processes, forming a tree of processes. In general, a process will need certain resources (CPU time, memory, files, I/O devices) to accomplish its task. When a process creates a subprocess, that subprocess may be able to obtain its resources directly from the operating system, or it may be constrained to a subset of the resources of the parent process. The parent may have to partition its resources among its children, or it may be able to share some resources (such as memory or files) among several of its children.

Restricting a child process to a subset of the parent's resources prevents any process from overloading the system by creating too many subprocesses. In addition to the various physical and logical resources that a process obtains when it is created, initialization data (input) may be passed along by the parent process to the child process. For example, consider a process whose function is to display the contents of a file—say, img.jpg—on the screen of a terminal. When it is created, it will get, as an input from its parent process, the name of the file img.jpg, and it will use that file name, open the file, and write the contents out. It may also get the name of the output device. Some operating systems pass resources to child processes. On such a system, the new process may get two open files, img.jpg and the terminal device, and may simply transfer the datum between the two.

When a process creates a new process, two possibilities exist in terms of execution:

- o The parent continues to execute concurrently with its children.
- o The parent waits until some or all of its children have terminated.

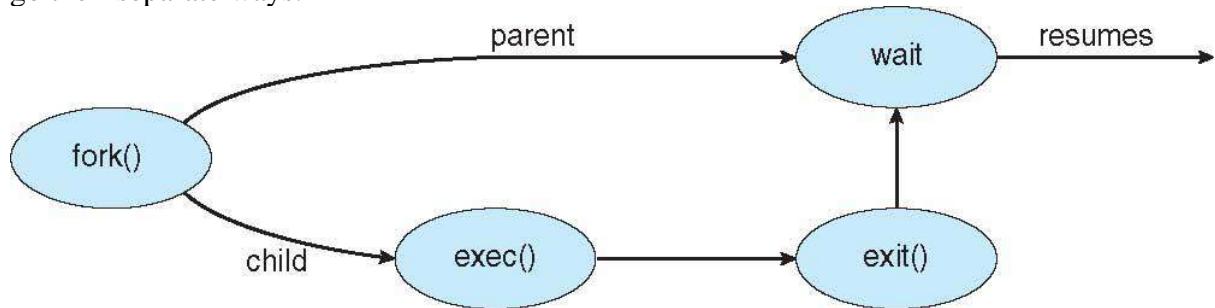
There are also two possibilities in terms of the address space of the new process:

- o The child process is a duplicate of the parent process (it has the same program and data as the parent).
- o The child process has a new program loaded into it.

In UNIX, as we've seen, each process is identified by its process identifier, which is a unique integer.

A new process is created by the **fork()** system call. The new process consists of a copy of the address space of the original process. This mechanism allows the parent process to communicate easily with its child process. Both processes (the parent and the child) continue execution at the instruction after the **fork()**, with one difference. The return code for the **fork()** is zero for the new (child) process, whereas the (nonzero) process identifier of the child is returned to the parent.

Typically, the **exec()** system call is used after a **fork()** system call by one of the two processes to replace the process's memory space with a new program. The **exec()** system call loads a binary file into memory (destroying the memory image of the program containing the **exec()** system call) and starts its execution. In this manner, the two processes are able to communicate and then go their separate ways.



The parent can then create more children; or, if it has nothing else to do while the child runs, it can issue a `wait()` system call to move itself off the ready queue until the termination of the child.

To illustrate these differences, let's first consider the UNIX operating system.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
pid_t pid;

/* fork a child process */
pid = fork();

if (pid < 0) { /* error occurred */
    fprintf(stderr, "Fork Failed");
    return 1;
}
else if (pid == 0) { /* child process */
    execl("/bin/ls", "ls", NULL);
}
else { /* parent process */
    /* parent will wait for the child to complete */
    wait(NULL);
    printf("Child Complete");
}

return 0;
}
```

The C program illustrates the UNIX system calls previously described.

We now have two different processes running a copy of the same program.

The value of `pid` for the child process is zero; that for the parent is an integer value greater than zero. The child process overlays its address space with the UNIX command `/bin/ls` (used to get a directory listing) using the `execl()` system call (`execlp()` is a version of the `exec()` system call). The parent waits for the child process to complete with the `wait()` system call. When the child process completes (by either implicitly or explicitly invoking `exit()`) the parent process resumes from the call to `wait()`, where it completes using the `exit()` system call.

Process Termination

A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the `exit()` system call. At that point, the process may return a status value (typically an integer) to its parent process (via the `wait()` system call). All the resources of the process—including physical and virtual memory, open files, and I/O buffers—are deallocated by the operating system.

A parent may terminate the execution of one of its children for a variety of reasons, such as these:

- o The child has exceeded its usage of some of the resources that it has been allocated. (To determine whether this has occurred, the parent must have a mechanism to inspect the state of its children.)
- o The task assigned to the child is no longer required.
- o The parent is exiting, and the operating system does not allow a child to continue if its parent terminates.

If a process terminates (either normally or abnormally), then all its children must also be terminated. This phenomenon, referred to as **cascading termination**, is normally initiated by the operating system

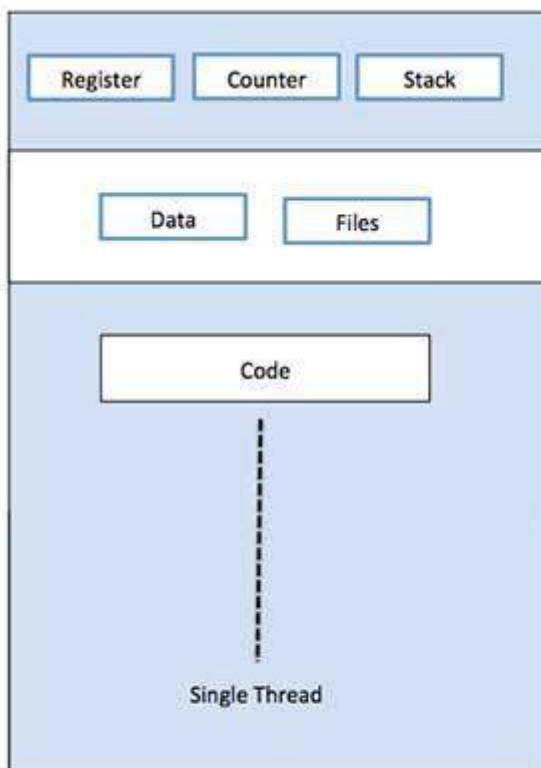
What is Thread?

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

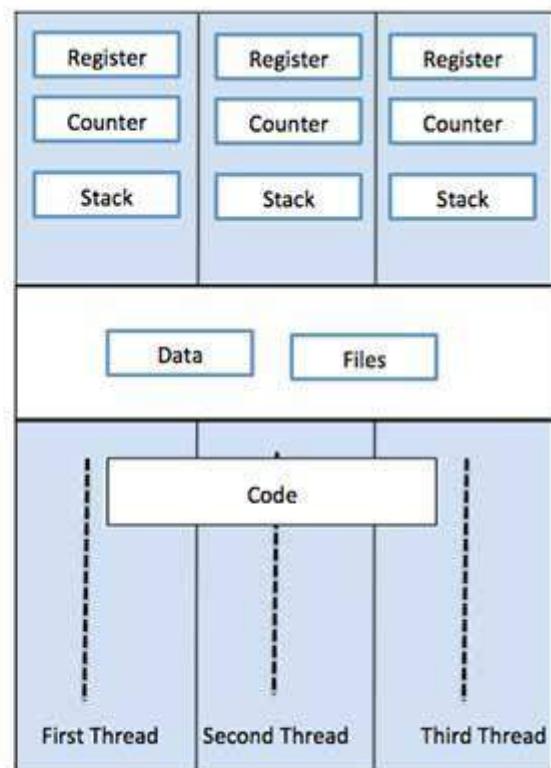
A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. The following figure shows the working of a single-threaded and a multithreaded process.



Single Process P with single thread



Single Process P with three threads

Difference between Process and Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.

3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

Types of Thread

Threads are implemented in following two ways –

- **User Level Threads** – User managed threads.
- **Kernel Level Threads** – Operating System managed threads acting on kernel, an operating system core.

User Level Threads

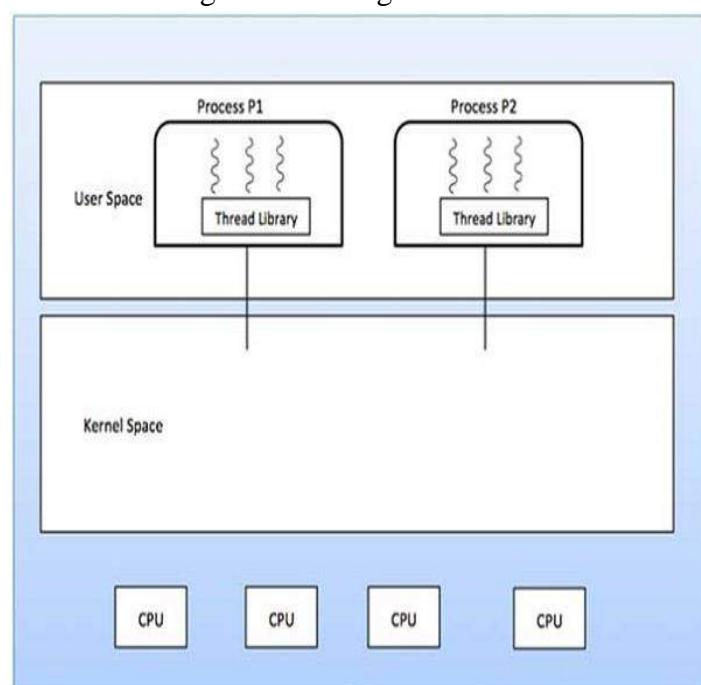
In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.

Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.



Kernel Level Threads

In this case, thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.
-

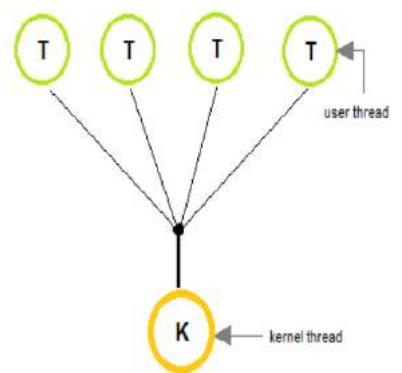
Multithreading Models

Some operating systems provide a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

- Many to many relationship.
- Many to one relationship.
- One to one relationship.

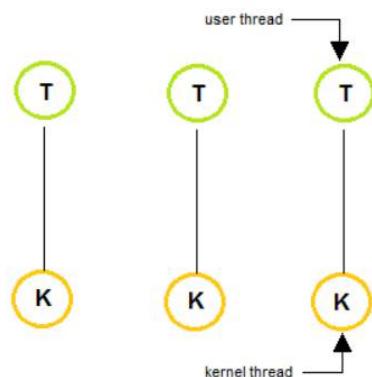
Many to one Model

- The many-to-many model multiplexes any number of In the **many to one** model, many user-level threads are all mapped onto a single kernel thread.
- Thread management is handled by the thread library in user space, which is efficient in nature.



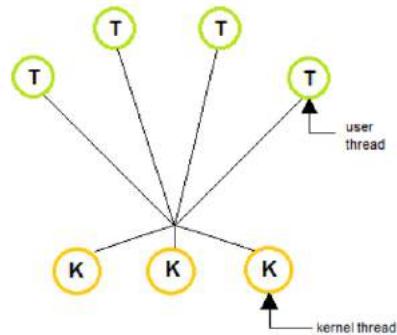
One to One Model

- The one to one model creates a separate kernel thread to handle each and every user thread.
- Most implementations of this model place a limit on how many threads can be created.
- Linux and Windows from 95 to XP implement the one-to-one model for threads.



Many to Many Model

- The many to many model multiplexes any number of user threads onto an equal or smaller number of kernel threads, combining the best features of the one-to-one and many-to-one models.
- Users can create any number of the threads.
- Blocking the kernel system calls does not block the entire process.
- Processes can be split across multiple processors



Difference between User-Level & Kernel-Level Thread

S.N.	User-Level Threads	Kernel-Level Thread
1	User-level threads are faster to create and manage.	Kernel-level threads are slower to create and manage.
2	Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads.
3	User-level thread is generic and can run on any operating system.	Kernel-level thread is specific to the operating system.
4	Multi-threaded applications cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.

Benefits of Multithreading

1. Responsiveness
2. Resource sharing, hence allowing better utilization of resources.
3. Economy. Creating and managing threads becomes easier.
4. Scalability. One thread runs on one CPU. In Multithreaded processes, threads can be distributed over a series of processors to scale.
5. Context Switching is smooth. Context switching refers to the procedure followed by CPU to change from one task to another.

Multithreading Issues

Below we have mentioned a few issues related to multithreading. Well, it's an old saying, *All good things, come at a price.*

fork() System Call

fork() is a system call executed in the kernel through which a process creates a copy of itself. Now the problem in Multithreaded process is, if one thread forks, will the entire process be copied or not? And same applies to exec()

Thread Cancellation

Thread cancellation means terminating a thread before it has finished working. There can be two approaches for this, one is **Asynchronous cancellation**, which terminates the target thread immediately. The other is **Deferred cancellation** allows the target thread to periodically check if it should be cancelled.

Signal Handling

Signals are used in UNIX systems to notify a process that a particular event has occurred. Now in when a Multithreaded process receives a signal, to which thread it must be delivered? It can be delivered to all, or a single thread.

Security Issues

Yes, there can be security issues because of extensive sharing of resources between multiple threads.

Inter Process Communication (IPC)

A process can be of two type:

- Independent process.
- Co-operating process.

An independent process is not affected by the execution of other processes while a co-operating process can be affected by other executing processes. Though one can think that those processes, which are running independently, will execute very efficiently but in practical, there are many situations when co-operative nature can be utilised for increasing computational speed, convenience and modularity. **Inter process communication (IPC)** is a mechanism which allows processes to communicate each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them. Processes can communicate with each other using these two ways:

1. Shared Memory
2. Message passing

i. Shared memory model.

Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other. All POSIX systems, as well as Windows operating systems use shared memory.

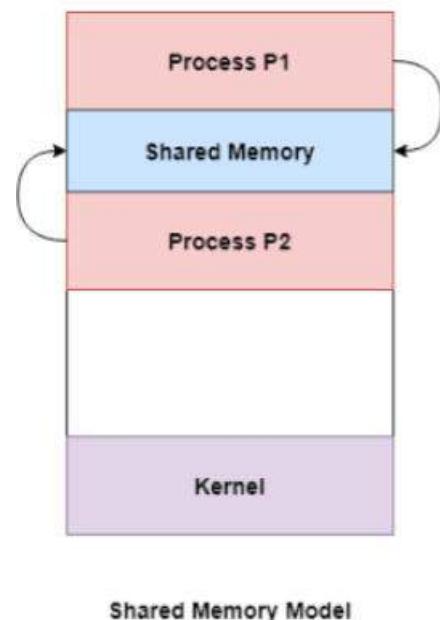
Advantage of Shared Memory Model

Memory communication is faster on the shared memory model as compared to the message passing model on the same machine.

Disadvantages of Shared Memory Model

Some of the disadvantages of shared memory model are as follows:

- All the processes that use the shared memory model need to make sure that they are not writing to the same memory location.
- Shared memory model may create problems such as synchronization and memory protection that need to be addressed.



Ex: Producer-Consumer problem :

There are two processes: Producer and Consumer. Producer produces some item and Consumer consumes that item. The two processes shares a common space or memory location known as buffer where the item produced by Producer is stored and from where the Consumer consumes the item if needed. There are two version of this problem: first one is known as unbounded buffer problem in which Producer can keep on producing items and there is no limit on size of buffer, the second one is known as bounded buffer problem in which producer can produce up to a certain amount of item and after that it starts waiting for consumer to consume it. We will discuss the bounded buffer problem. First, the Producer and the Consumer will share some common memory, then producer will start producing items. If the total produced item is equal to the size of buffer, producer will wait to get it consumed by the Consumer. Similarly, the consumer first check for the availability of the item and if no item

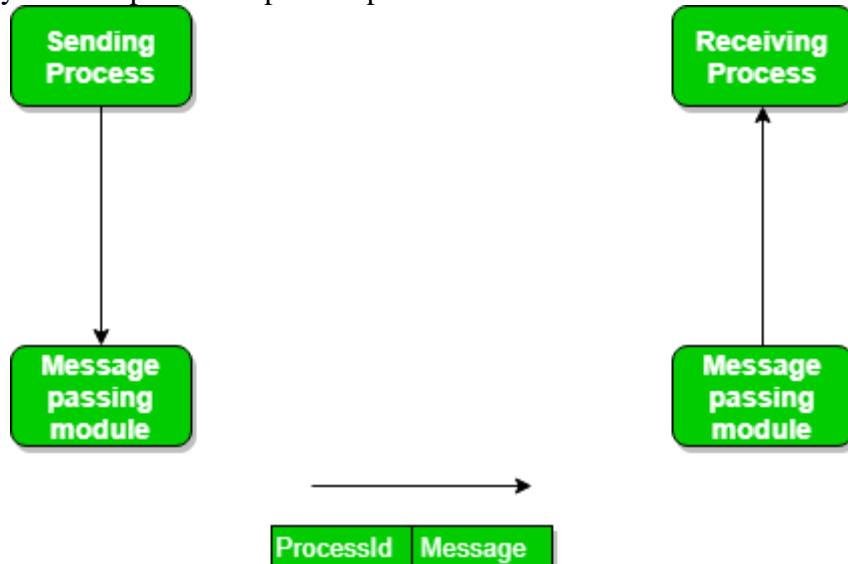
is available, Consumer will wait for producer to produce it. If there are items available, consumer will consume it. The pseudo code are given below:

Producer process	Consumer Process
<pre>item nextProduced; while(1){ // check if there is no space for production. // if so keep waiting. while((free_index+1) mod buff_max == full_index); shared_buff[free_index] = nextProduced; free_index = (free_index + 1) mod buff_max; }</pre>	<pre>item nextConsumed; while(1){ // check if there is an available // item for consumption. // if not keep on waiting for // get them produced. while((free_index == full_index); nextConsumed = shared_buff[full_index]; full_index = (full_index + 1) mod buff_max; }</pre>

In the above code, The producer will start producing again when the $(\text{free_index}+1) \bmod \text{buff_max}$ will be free because if it is not free, this implies that there are still items that can be consumed by the Consumer so there is no need to produce more. Similarly, if free index and full index points to the same index, this implies that there are no item to consume.

ii) Messaging Passing Method

Now, We will start our discussion for the communication between processes via message passing. In this method, processes communicate with each other without using any kind of shared memory. If two processes p1 and p2 want to communicate with each other, they proceed as



follow:



- Establish a communication link (if a link already exists, no need to establish it again.)
- Start exchanging messages using basic primitives.
We need at least two primitives:
 - **send(message, destination)** or **send(message)**
 - **receive(message, host)** or **receive(message)**

a. Message Passing through Communication Link.

Direct and Indirect Communication link

Now, We will start our discussion about the methods of implementing communication link. While implementing the link, there are some questions which need to be kept in mind like :

1. How are links established?
2. Can a link be associated with more than two processes?
3. How many links can there be between every pair of communicating processes?
4. What is the capacity of a link? Is the size of a message that the link can accommodate fixed or variable?
5. Is a link unidirectional or bi-directional?

A link has some capacity that determines the number of messages that can reside in it temporarily for which Every link has a queue associated with it which can be either of zero capacity or of bounded capacity or of unbounded capacity. In zero capacity, sender wait until receiver inform sender that it has received the message. In non-zero capacity cases, a process does not know whether a message has been received or not after the send operation. For this, the sender must communicate to receiver explicitly. Implementation of the link depends on the situation, it can be either a Direct communication link or an In-directed communication link. **Direct Communication links** are implemented when the processes use specific process identifier for the communication but it is hard to identify the sender ahead of time. **For example: the print server.**

In Direct message passing, The process which want to communicate must explicitly name the recipient or sender of communication.

e.g. `send(p1, message)` means send the message to p1. similarly, `receive(p2, message)` means receive the message from p2. In this method of communication, the communication link get established automatically, which can be either unidirectional or bidirectional, but one link can be used between one pair of the sender and receiver and one pair of sender and receiver should not possess more than one pair of link. Symmetry and asymmetry between the sending and receiving can also be implemented i.e. either both process will name each other for sending and receiving the messages or only sender will name receiver for sending the message and there is no need for receiver for naming the sender for receiving the message. The problem with this method of communication is that if the name of one process changes, this method will not work.

In-directed Communication is done via a shred mailbox (port), which consists of queue of messages. Sender keeps the message in mailbox and receiver picks them up. **In Indirect message passing**, processes uses mailboxes (also referred to as ports) for sending and receiving messages. Each mailbox has a unique id and processes can communicate only if they share a mailbox. Link established only if processes share a common mailbox and a single link can be associated with many processes. Each pair of processes can share several communication links and these link may be unidirectional or bi-directional. Suppose two process want to communicate though Indirect message passing, the required operations are: create a mail box, use this mail box for sending and receiving messages, destroy the mail box. The standard primitives used are : `send(A,`

message) which means send the message to mailbox A. The primitive for the receiving the message also works in the same way e.g. **received (A, message)**.

b. Message Passing through Exchanging the Messages.

Synchronous and Asynchronous Message Passing:

A process that is blocked is one that is waiting for some event, such as a resource becoming available or the completion of an I/O operation. IPC is possible between the processes on same computer as well as on the processes running on different computer i.e. in networked/distributed system. In both cases, the process may or may not be blocked while sending a message or attempting to receive a message so Message passing may be blocking or non-blocking. Blocking is considered **synchronous** and **blocking send** means the sender will be blocked until the message is received by receiver. Similarly, **blocking receive** has the receiver block until a message is available. Non-blocking is considered **asynchronous** and Non-blocking send has the sender sends the message and continue. Similarly, Non-blocking receive has the receiver receive a valid message or null. After a careful analysis, we can come to a conclusion that, for a sender it is more natural to be non-blocking after message passing as there may be a need to send the message to different processes But the sender expect acknowledgement from receiver in case the send fails. Similarly, it is more natural for a receiver to be blocking after issuing the receive as the information from the received message may be used for further execution but at the same time, if the message send keep on failing, receiver will have to wait for indefinitely. That is why we also consider the other possibility of message passing. There are basically three most preferred combinations:

- Blocking send and blocking receive
- Non-blocking send and Non-blocking receive
- Non-blocking send and Blocking receive (Mostly used)

The other message passing variants include :

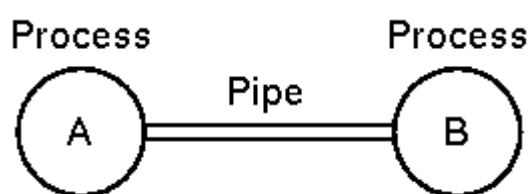
- Send by copy / Send by reference

Buffering - queue of messages attached to the link; implemented in one of three ways.

- Zero capacity - 0 messages Sender must wait for receiver (rendezvous).
- Bounded capacity - finite length of n messages Sender must wait if link full.
- Unbounded capacity - infinite length Sender never waits.

Pipes :

A pipe is a simple method for communicating between two processes.



- As far as the processes are concerned the pipe appears to be just like a file.
- When A performs a write, it is buffered in the pipe.

- When B reads then it reads from the pipe, blocking if there is no input.
- in UNIX (and DOS) one process can be piped into another pipe using the '|' character.
e.g.

```
cat classlist | sort | more
```

the **cat** command prints the contents of the file 'classlist', this is piped into the **sort** command which sorts the list. Finally, the sorted list is sent to the **more** command that prints it one screenfull at a time.

- Pipes may be implemented using shared memory (UNIX) or even with temporary files (DOS).

pipe() System call

Conceptually, a pipe is a connection between two processes, such that the standard output from one process becomes the standard input of the other process. In UNIX Operating System, Pipes are useful for communication between related processes(inter-process communication).

- Pipe is one-way communication only i.e we can use a pipe such that One process write to the pipe, and the other process reads from the pipe.
- The pipe can be used by the creating process, as well as all its child processes, for reading and writing. One process can write to this "virtual file" or pipe and another related process can read from it.
- If a process tries to read before something is written to the pipe, the process is suspended until something is written.
- The pipe system call finds the first two available positions in the process's open file table and allocates them for the read and write ends of the pipe.

Syntax

```
int pipe(int fds[2]);
```

Parameters :

fd[0] will be the fd(file descriptor) for the read end of pipe.
fd[1] will be the fd for the write end of pipe.

Returns : 0 on Success.

-1 on error.

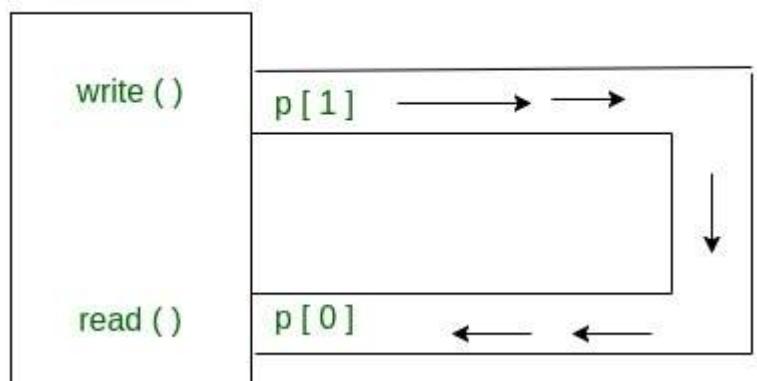
```
// C program to illustrate pipe system call  
in C
```

```
#include <stdio.h>
```

```
#include <unistd.h>
```

```
#define MSGSIZE 16
```

Process



```
char* msg1 = "hello, world #1";
char* msg2 = "hello, world #2";
char* msg3 = "hello, world #3";

int main()
{
    char inbuf[MSGSIZE];
    int p[2], i;

    if (pipe(p) < 0)
        exit(1);

    /* continued */
    /* write pipe */

    write(p[1], msg1, MSGSIZE);
    write(p[1], msg2, MSGSIZE);
    write(p[1], msg3, MSGSIZE);

    for (i = 0; i < 3; i++) {
        /* read pipe */
        read(p[0], inbuf, MSGSIZE);
        printf("% s\n", inbuf);
    }
    return 0;
}
```

Chapter 2 Topics:

Process synchronization:

- Background, critical section problem, understating the race condition and the need for the process synchronization. [From Chapter 1]
- Petersons solution, synchronization Hardware
- Semaphores implementation
- Classical synchronization problem: Bounder buffer, Reader Writer, Dining philosophers

CPU scheduling:

- FCFS, SJF, Priority, Round robin, Multilevel queue and multilevel feedback queue scheduling, Real Time scheduling – Rate Monotonic Scheduling and Deadline scheduling

Deadlocks:

- Necessary conditions, Resource allocation graph, Deadlock prevention, Avoidance, detection and recovery.

PROCESS SYNCHRONIZATION:

Introduction

CPU scheduler switches rapidly between processes to provide concurrent execution. This means that one process may only partially complete execution before another process is scheduled. In fact, a process may be interrupted at any point in its instruction stream, and the processing core may be assigned to execute instructions of another process.

Now consider the following producer Consumer Code

Producer Process	Consumer Process
<pre>while (true) { /* produce an item in next produced */ while (counter == BUFFER SIZE) ; /* do nothing */ buffer[in] = next produced; in = (in + 1) % BUFFER SIZE; counter++; }</pre>	<pre>while (true) { while (counter == 0) ; /* do nothing */ next consumed = buffer[out]; out = (out + 1) % BU FFER SIZE; counter--; /* consume the item in next consumed */ }</pre>

The statements “counter++” and “counter—” may be implemented in machine language (on a typical machine) as follows:

Counter ++	Counter--
<pre>register1 = counter register1 = register1 + 1 counter = register1</pre>	<pre>register2 = counter register2 = register2 - 1 counter = register2</pre>

The concurrent execution of “counter++” and “counter—” is equivalent to a sequential execution in which the lower-level statements presented are interleaved in some arbitrary order.

T0: *producer* execute *register1* = *counter* {*register1* = 5}

T1: *producer* execute *register1* = *register1* + 1 {*register1* = 6}

T2: *consumer* execute *register2* = *counter* {*register2* = 5}

T3: consumer execute $register2 = register2 - 1$ { $register2 = 4$ }

T4: producer execute $counter = register1$ { $counter = 6$ }

T5: consumer execute $counter = register2$ { $counter = 4$ }

Several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called a **race condition**. **Maintaining data consistency requires** while writing into shared data must be mutually exclusive and need mechanisms to ensure the orderly execution

The Critical-Section Problem

- Consider a system consisting of n processes $\{P_0, P_1, \dots, P_{n-1}\}$. Each process has a segment of code, called a **critical section**, in which the process may be changing common variables, updating a table, writing a file, and so on.
- The important feature of the system is that, when one process is executing in its critical section, no other process is allowed to execute in its critical section. That is, no two processes are executing in their critical sections at the same time. The **critical-section problem** is to design a protocol that the processes can use to cooperate.
- Each process must request permission to enter its critical section. The section of code implementing this request is the **entry section**. The critical section may be followed by an **exit section**. The remaining code is the **remainder section**.
- The general structure of a typical process P_i is shown below :

```
do {  
    entry section  
    critical section  
    exit section  
    remainder section  
} while (true);  
•
```

A solution to the **critical-section problem** must satisfy the following **three requirements**:

1. **Mutual exclusion**. If process P_i 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 some processes wish to enter their critical sections, for which a selection is made which cannot be postponed indefinitely.
3. **Bounded waiting**. There exists a bound, or limit, on the number of times that **other processes** are allowed to enter their critical sections and makes a bounded waiting for a process

Types of Solutions

- Software solutions : Petersons Algorithm - Algorithms whose correctness does not rely on any other assumptions.
- Hardware solutions - Rely on some special machine instructions.
- Operating System solutions – Semaphore - Provide some functions and data structures to the programmer through system/library calls.
- Programming Language solutions -Monitors - Constructs provided as part of a high level language

Peterson's Solution

The classic software-based two process solution to the critical-section problem known as Peterson's solution

The two processes share two variables:

```
int turn;  
Boolean flag[2]
```

The variable turn indicates whose turn it is to enter the critical section. The flag array is used to indicate if a process is ready to enter the critical section.

flag[i] = true implies that process Pi is ready!

Algorithm for Process Pi

```
do  
{  
flag[i] = TRUE;  
turn = j;  
while (flag[j] && turn == j);  
.....  
critical section  
flag[i] = FALSE;  
....  
remainder section  
}  
while (TRUE);
```

Algorithm with explanation

Process Pi	Process Pj
<pre>do { flag[i]=true;//Pi indicates it wants to enter critical Section// turn = j;//sets turn as j allowing Pj to enter Critical section If Pj wants to// while (flag [j] and turn = j); {//wait//} // if Pj wants to enter & it is Pj's turn to enter its critical section then Pi waits until Pi completes its critical section// critical section//critical section of Pi// flag [i]=false;</pre>	<pre>do { flag[j]=true;//Pj indicates it wants to enter critical Section// turn = i;s//sets turn as i allowing Pi to enter Critical section If Pi wants to// while (flag [j] and turn = j); {//wait//} // if Pi wants to enter & it is Pi's turn to enter its critical section then Pj waits until Pi completes its critical section// critical section//critical section of Pj// flag [j]=false;</pre>

//sets flag as false indicating Pi no longer wants to enter its critical section//	//sets flag as false indicating Pj no longer wants to enter its critical section//
remainder section	remainder section
//rest of Pi's non critical code//	//rest of Pj's non critical code//

This solution guarantees the three requirements: mutual exclusion, progress and bounded waiting.

Multi-process solution

Bakery algorithm supports multiple cooperating processes and solves critical section problem for multiple processes.

General functioning of Bakers algorithms

Its based on a scheduling algorithm commonly used in bakeries

On entering the store, each customer receives a number. The customer with the lowest number is served next. The Customer with the lowest name is served first

Functioning of the Bakery algorithm for process

Each process wanting to enter its critical section chooses a number (choose). The numbering scheme always generates numbers in increasing order. If no processes are in their critical section the process with lowest number is allowed to enter its critical section. If two or more processes have been waiting the process with smallest process token number is allowed

Bakery Algorithm for Process i

```

boolean choosing[n];//willingness to enter//
//initialized to false//
int number[n];//number assigned to each process willing to enter its critical section//
//initialized to 0//
do {
    choosing[i]=true; //process i wants to enter its critical section//
    number[i]=max(number[0], number[1],.....number [n- 1])+1;//a number is given to the
process//
    choosing[i]=false; //sets willing to enter as false to check status of other processes
before entering into its critical section//
    for (j=0; j<n; j++)
    {
        while (choosing[j]);
        // checks if there are any other processes (j = process 0 to n)willing to enter their critical section
        //still choosing a number) wait.
        //all willing processes have chosen a number//
        While ((number[j]!= 0)&&(number [j,j] < number [i,i]));
        //process with least token number
        critical section
        number[i] = 0;
        //process i has finished its critical section //
        remainder section
        //the non critical section of process i//
    } while (1);//continuous loop//
}

```

Note:- the processes enter in their critical-section on a first-come, first-serve basis.

Bakery Algorithm satisfies mutual exclusion, progress and bounded waiting

Drawback of Software Solutions

Processes that are requesting to enter their critical section are busy waiting (consuming processor time needlessly).

Synchronization hardware:

Hardware instructions are available on many systems and showing how they can be used effectively in solving the critical-section problem. Hardware features can make any programming task easier and improve system efficiency particularly for multiprocessor systems. These hardware instructions are executed **atomically**- as one uninterruptable unit. Two such instructions are test_and_set() and swap()

Test and Set

Test and sets lock if no other process in in Critical section

Test and Set Definition	Test and Set
<pre>boolean test and set(boolean *target) { boolean rv = *target; *target = true; return rv; } //keeps testing the variable target if false sets it to true & return's false//</pre>	<pre>do { while (test and set(&lock)); /* do nothing */ //test variable lock if false set to true// //if lock = false => no other process is in its critical section// //if lock is already true => some other process is already in its critical section and wait until released// /* critical section */ lock = false; //sets the lock false /* remainder section */ } while (true);</pre>

Swap

Process i will wait to enter critical section until lock is true. Once lock becomes false I sets the lock and enters to critical section.

<pre>void Swap (boolean *a, boolean *b) { boolean temp = *a; *a = *b; *b = temp; }</pre>	<pre>do { key = TRUE; while (key == TRUE) Swap (&lock, &key); //if lock=false=> no other process is in its critical section// //(Key = true ,lock = false) swap =>(lock = true key = false) =>process has set the lock// // Key = false process can exit while loop & enter critical section// //If lock = true => some other process is in its critical section// // critical section lock = FALSE; //remainder section }while(True)</pre>
------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Semaphore

A **semaphore** S is an **integer variable** that, apart from initialization, is accessed only through two standard **atomic** operations: **wait()** and **signal()**. The wait() operation was termed P and signal() as V.

The definition of wait() is as follows:	The definition of signal() is as follows:
<pre>wait(S) { while (S <= 0) ; // busy wait S--; }</pre>	<pre>signal(S) { S++; }</pre>

When one process modifies the semaphore value, no other process can simultaneously modify that same semaphore value. In addition, in the case of wait(S), the testing of the integer value of S ($S \leq 0$), as well as its possible modification ($S--$), must be executed without interruption.

The two types of semaphores are counting and binary semaphores. The value of a **counting semaphore** can range over an unrestricted domain. The value of a **binary semaphore** can range only between 0 and 1.

Mutual exclusion using semaphore

Semaphore mutex;

Mutex is a semaphore variable set to 1 .

*//If mutex ≤ 0 => there is some process in its critical section All other processes has to wait//
// if mutex $=> 0$ => there is no process in critical section next process that sets mutex as ≤ 0 can enter its critical section//*

Process Pi:

```
do {  
    wait(mutex); //process tries to set mutex as  $\leq 0$ //  
    Critical section; //process i has set mutex as  $\leq 0$  & entered the CS//  
    signal(mutex);  
    Remainder section;  
} while (1);
```

Advantages of semaphores - Easily implemented for n processes

Disadvantages of Semaphores - Require's busy waiting(spinlock)

Semaphore avoiding busy waiting

When a process executes the wait() operation and finds that the semaphore value is not positive, it must wait. Rather than engaging in busy waiting, the process can block itself. The block operation places a process into a waiting queue associated with the semaphore and the state of the process is switched to the waiting state.

A process that is blocked waiting on a semaphore S, should be restarted when some other process executes a signal() operation. The process is restarted by a wakeup() operation which changes the process from the waiting state to the ready state. Process is then placed in the ready queue.

Structure definition <pre>typedef struct{ int value;//semaphore value// struct process *L; } semaphore.</pre>	The definition of the semaphore operations wait and signal should be modified Wait:- <ul style="list-style-type: none"> o If semaphore is not +ve then the process can block itself o The block operation:- places the process in a waiting queue. Signal:- <ul style="list-style-type: none"> o If Queue is not empty wakes the next process from queue
Definition of Wait (S) <pre>wait(S) { S.value--;//process wants to enter CS// if (S.value < 0)//another process is already in its CS// { add this process to S.L;//insert the process to Q// block;//process blocks itself// } }</pre>	Definition of Signal(S) <pre>signal(S) { S.value++;//process finished CS & exiting// if (S.value <= 0)//if Q is not empty// { remove a process P from S.L; //select next process from Q// wakeup(P);//wake the next process from Q// } }</pre>

Deadlocks and Starvation

The implementation of a semaphore with a waiting queue may result in a situation where two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes. The event in question is the execution of a signal() operation. When such a state is reached, these processes are said to be deadlocked. To illustrate this, we consider a system consisting of two processes, P0 and P1, each accessing two semaphores, S and Q, set to the value 1:

P0	P1
wait(S);	wait(Q);
wait(Q);	wait(S);
.....
signal(S);	signal(Q);
signal(Q);	signal(S);

Suppose that P0 executes wait (S) and then P1 executes wait (Q). When P0 executes wait(Q), it must wait until P1 executes signal(Q). Similarly, when P1 executes wait(S), it must wait until P0 executes signal(S). Since these signal () operations cannot be executed, P0 and P1 are deadlocked.

Starvation a situation in which processes wait indefinitely within the semaphore. Indefinite blocking may occur if we add and remove processes from the list associated with a semaphore in LIFO (last-in, first-out) order.

Classical synchronization Problem

- Bounded-Buffer Problem
- Readers and Writers Problem
- Dining-Philosophers Problem

The Bounded-Buffer Problem

/ Producer Consumer Problem

- Producer produces (data) into buffer and Consumer consumes (data) from buffer
- Producer is trying to full the buffer & Consumer it trying to empty the buffer
- Synchronization should ensure
 - Producer should not produce on full buffer
 - Consumer should not consume from empty buffer
 - Producer & consumer should not function at the same time. Production & consumption should be mutually exclusive.

Bounded-Buffer Problem

Shared data

```
semaphore full; //semaphore to ensure producer dose produce to a full buffer//  
semaphore empty; //semaphore to ensure consumer dose consume from a empty buffer//  
semaphore mutex; //semaphore to ensure producer & consumer remain mutually exclusive //
```

```
// initially full = 0, empty = n, mutex = 1//
```

<u>Process Producer</u>	<u>Process Consumer</u>
<pre>do { ... produce an item in next_p //producer has produced the data to be written into buffer// ... wait(empty); //if empty≤0 => no empty buffer space =>producer wait// //ensures producer dose not write into a full buffer// //empty --;producer has filled one empty// wait(mutex);//ensures mutual exclusion// ... add next_p to buffer ... signal(mutex); //producer has exited CS (writing into buffer)// signal(full); //full ++; producer has filled buffer by 1 entry// } while (1);</pre>	<pre>do { wait(full); //if full ≤ 0 => no elements in buffer =>buffer empty=>consumer wait// //ensures consumer dose not read from empty buffer// wait(mutex); //ensures mutual exclusion// ... remove an item from buffer to next_c ... signal(mutex); //signals consumer is exiting CS(reading from buffer)// signal(empty); //empty ++;consumer created 1 empty space by consuming// ... consume the item in next_c //consumer is consuming the data from buffer// ... } while (1)</pre>

Readers and Writers Problem

Readers and writers that share a data resource. Some of these processes may want only to read (readers) whereas others may want to update (read and write) (writers)

- If two readers access the shared data simultaneously - No adverse effects
- Two writers access the resource simultaneously - Data might become inconsistent
- A writer and a reader access the resource simultaneously - Data might become inconsistent

Synchronization tool has to ensure

- o Any no of readers can access the shared data - No reader should wait for other readers
- o When writer is accessing neither reader nor writer should access shared resource
- o Once a writer is ready, that writer performs its write as soon as Possible
 - If a writer is waiting to access the object, no new Readers can start
- o Provide mutual exclusion for updating read count/read/write semaphore

Shared data

```

semaphore write; //semaphore to ensure consumer dose consume from a empty buffer//
semaphore mutex; //semaphore to ensure producer & consumer remain mutually exclusive //
int read_count = 0; //maintains count of number of readers//

// initially write = 1, mutex = 1, read_count = 0//

```

Process Reader:

```

wait(mutex);
//provides mutual exclusion for updating read_count//
//process increments readers count before starting Read//
read_count++;
if (read_count == 1)
  wait(write); r//writer is blocked on first reader
Signal(mutex);
// mutual exclusion for updating read_count is completed//
Reading is performed
wait(mutex);
//provides mutual exclusion for updating read_count//
//process decrements readers count after finishing Read//
read_count--;
if (read_count == 0)
  signal(write);
//writer signalled on last reader//
signal(mutex);
// mutual exclusion for updating read_count is completed//

```

Process Writer:

```

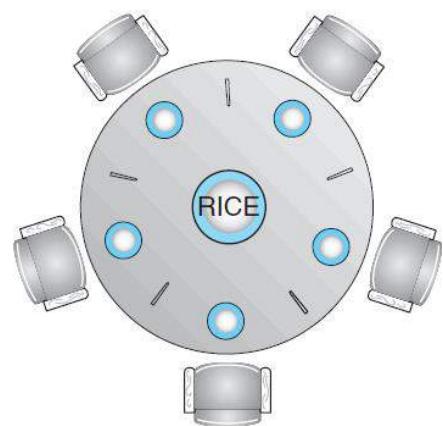
Wait(write);
//requesting Mutual exclusion for
writing//
//=> readers =0 && no other writer
accessing Data//
Writing is performed
Signal(write);
//writing completed exiting mutual
exclusion//

```

The Dining-Philosophers Problem

5 philosophers share a common circular table surrounded by five chairs

- In the center of the table is a bowl of rice, A chopstick is placed between each pair of adjacent philosophers.
- Each philosopher can either think or eat.
- If a philosopher wants to eat, he will need both chopsticks
- Each philosopher can pick up an adjacent chopstick, when available and put down the chop stick after eating.
- Chopsticks must be picked up and put down one by one.



<p>Solution :</p> <p>Semaphore chopstick[5]</p> <p>Algorithm</p> <pre> do { wait(chopstick[i]); wait(chopstick[(i+1) % 5]); ... /* eat for awhile */ ... signal(chopstick[i]); signal(chopstick[(i+1) % 5]); ... /* think for awhile */ ... } while (true); </pre>	<p>This solution is incorrect: -</p> <ul style="list-style-type: none"> • it allows the system to reach deadlock <p>Consider situation</p> <ul style="list-style-type: none"> • all philosopher has picked up their left chopsticks • each philosopher is waiting for his right chopstick <p>Proposed solutions</p> <ul style="list-style-type: none"> • Allow at most four philosophers to be sitting simultaneously at the table (allow only even No of philosophers) • Allow a philosopher to pick up her chopsticks only if both chopsticks are available <p>asymmetric solution; that is,</p> <ul style="list-style-type: none"> • an odd philosopher first picks up left chopstick and then right chopstick, • an even philosopher first picks up right chopstick and then left chopstick
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

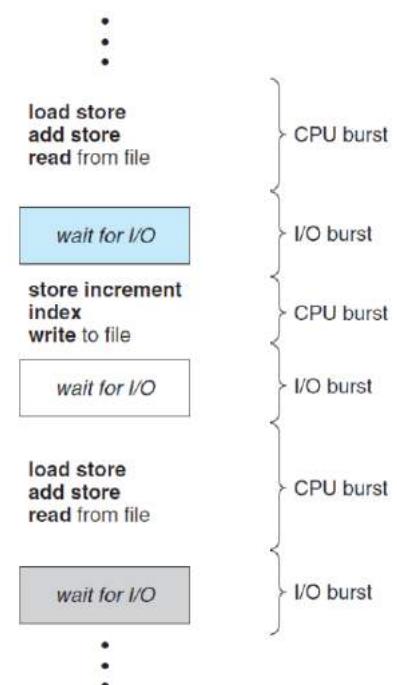
CPU SCHEDULING

Introduction

CPU-I/O Burst Cycle

The success of CPU scheduling depends on an observed property of processes:

- Process execution consists of a cycle of CPU execution and I/O wait.
- Processes alternate between these two states. Process execution begins with a CPU burst. That is followed by an I/O burst, which is followed by another CPU burst, then another I/O burst, and so on.
- Eventually, the final CPU burst ends with a system request to terminate execution (Fig a).
- The durations of CPU bursts have been measured extensively.



CPU Scheduler

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler).

The scheduler selects a process from the processes in memory that are ready to execute and allocates the CPU to that process. Note that the ready queue is not necessarily a first-in, first-out (FIFO) queue. Conceptually, however, all the processes in the ready queue are lined up waiting for a chance to run on the CPU. The records in the queues are generally process control blocks (PCBs) of the processes.

Dispatcher

Another component involved in the CPU-scheduling function is the dispatcher. The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves:

- o Switching context
- o Switching to user mode
- o Jumping to the proper location in the user program to restart that program

The time it takes for the dispatcher to stop one process and start another running is known as the dispatch latency.

Types of CPU schedulers

CPU Scheduler can be classified in to two types

Based on when the scheduler will choose the next job for execution



Non Preemptive Scheduler

Once a process is in the running state, it will continue until it terminates or blocks itself

- Cannot force stop a running process Thus a Non-preemptive scheduler will select next process Only when previous process has released the CPU & CPU is idle

Preemptive Scheduler

Currently running process may be interrupted and moved to the Ready state by OS

- Can force stop a running process if a higher importance
- (priority) process is available in ready Queue
- Allows for better service
- Prevents any one process from monopolizing
- Provides provision for priority of important processes

Thus, a preemptive scheduler will select next process when

- Previous process has released the CPU & CPU is idle
- Whenever a new process enters the ready queue

Scheduling Criteria

Different CPU scheduling algorithms have different properties, and the choice of a particular algorithm may favour one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms.

The criteria include the following:

- **CPU Utilization:** Keep CPU utilization as high as possible.
- **Throughput:** Number of processes completed per unit time.
- **Turnaround Time:** Mean time from submission to completion of process.
- **Waiting Time:** Amount of time spent ready to run but not running.
Amount of time spent waiting in ready Q waiting for CPU
- **Response Time:** Time between submission of requests and first response to the request.

Scheduling Algorithms

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. There are many different CPU scheduling algorithms:

- i. First-come, First served scheduling
- ii. Shortest-Job-First scheduling
- iii. Shortest-remaining-time-first scheduling
- iv. Priority scheduling
- v. Round-Robin scheduling
- vi. Multilevel Queue scheduling
- vii. Multilevel Feedback Queue scheduling
- viii. Multiprocessor scheduling
- ix. Real time scheduling

First-Come First-Served (FCFS) Scheduling

- Runs the processes in the order they arrive in the Ready Queue
- Non preemptive algorithm
- Removes a process from the processor only if it goes into the waiting state or terminates

Advantages

- Very simple
- Very little scheduler overhead (low dispatch latency)

Disadvantages

- Long average and worst-case waiting times
- Poor dynamic behavior (convoy effect - short process behind long process)

Analysis

- Good for long processes /batch processing
- Poor performance for short /interactive processes

Example:

Process	Burst Time
P1	24
P2	3
P3	3

Suppose that the processes arrive in the order : P1 , P2 , P3 and are served in FCFS order,

- The Gantt Chart for the schedule is:



Waiting time:

- The waiting time is 0 milliseconds for process P1,

- 24 milliseconds for process P2, and
- 27 milliseconds for process P3.
- Thus, the average waiting time is $(0 + 24 + 27)/3 = 17$ milliseconds.

Turnaround time:

- Turnaround time for P1 = 24
- Turnaround time for P2 = 24 + 3
- Turnaround time for P3 = 24 + 3 + 3

Shortest-Job-First scheduling

- A different approach to CPU scheduling is the shortest-job-first (SJF) scheduling algorithm. This algorithm associates with each process the length of the process's next CPU burst.
- When the CPU is available, it is assigned to the process that has the **smallest next CPU burst**. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.
- The SJF algorithm can be either preemptive or nonpreemptive. The basic SJF is non preemptive and shortest remaining time next is SJF preemptive version.

Advantages

- Very simple
- Minimal waiting times

Disadvantages

- little scheduler overhead(some dispatch latency)
- has to rely on user estimates of burst times
- might cause starvation of jobs with long CPU bursts

Analysis

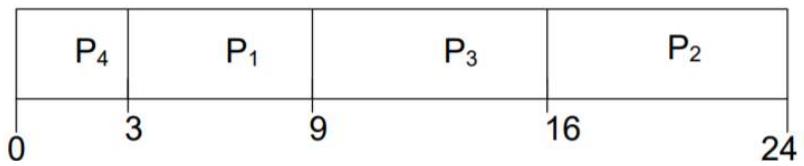
- Good for lots of short processes
- No effect for batch processing

Example:

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

Suppose that the processes arrive in the order : P1 , P2 , P3 and P4 are served in SJF order,

The Gantt Chart for the schedule is:



Waiting time:

- The waiting time is 3 milliseconds for process P1,
- 16 milliseconds for process P2,
- 16 milliseconds for process P3, and
- 0 milliseconds for process P4.
- Thus, the average waiting time $=(3 + 16 + 9 + 0)/4 = 7$ milliseconds.

Turnaround time:

- Turnaround time for P1 = 9
- Turnaround time for P2 = 24
- Turnaround time for P3 = 16
- Turnaround time for P4 = 3
- Average Turnaround time = $(9+24+16+3)/4 = 13$ milliseconds

Pre-emptive SJF / Shortest remaining time process next (SRTN)

A pre-emptive SJF algorithm will pre-empt the currently executing process, whereas nonpreemptive SJF algorithm will allow the currently running process to finish its CPU burst.

Advantages

- Very simple
- Minimal waiting times

Disadvantages

- little scheduler overhead(some dispatch latency)
- has to rely on user estimates of burst times
- might cause starvation of jobs with long CPU bursts

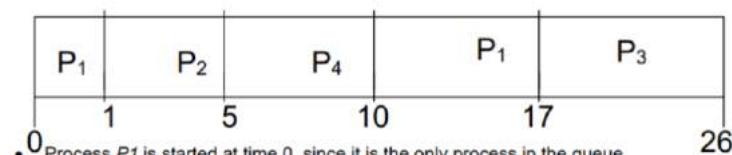
Analysis

- Good for lots of short processes
- No effect for batch processing

Process	Arrival time	Burst Time
P1	0	6
P2	1	8
P3	2	7
P4	3	3

Suppose that the processes arrive in the order : P1 , P2 , P3 and P4 are served in SJF order,

The Gantt Chart for the schedule is:



26

- Process P2 arrives at time 1.
- Among P1 and P2 as P2 is having shortest time, P2 will get executed first.
- The remaining time for process P1 (7 milliseconds) is larger than the time required by process P2 (4 milliseconds), so process P1 is preempted, and process P2 is scheduled.
- After completion of P2, P1 is scheduled.
- And then in the last P4 which is having larger time is scheduled.

Waiting time:

- Waiting time for process p1 = $(10 - 1)$
- Waiting time for process p2 = $(1-1)$
- Waiting time for process p3 = $(17-2)$
- Waiting time for process p4 = $(5-3)$
- The average waiting time = $((10 - 1) + (1 - 1) + (17-2) + (5-3))/4 = 26/4 = 6.5$ milliseconds.

Nonpreemptive SJF scheduling would result in an average waiting time of 7.75 milliseconds.

Turnaround time:

- Turnaround time for P1 = $(0+8)=8$

- Turnaround time for P2 = $(7 + 4)$
- Turnaround time for P3 = $(15+9)$
- Turnaround time for P4 = $(9 + 5)$
- Average Turnaround time = $(8+11+24+14) / 4 = 14.25$ milliseconds.

Priority Scheduling

- Can be pre-emptive or non-pre-emptive.
- A priority (an integer) is associated with each process. Priorities can be assigned either as Externally (by user) or Internally (by Operating System).
- The CPU is allocated to the process with the highest priority (Smallest integer means highest priority).
- In preemptive priority scheduling, If running process is not the process with highest priority it is pre-empted (forced to release CPU) and CPU will be scheduled for the newly arrived high priority process.
- Algorithm is initiated each time a process enters ready Q

Advantages

- Important tasks can be processed quicker

Disadvantages

- Starvation or indefinite blocking – low priority processes may never execute
- Solution Aging – increasing the priority of the process. With time

Analysis

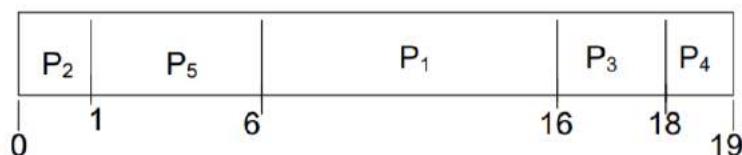
- Good for implementing system rules & priorities
- Might result in starvation of low priority jobs

Non preemptive priority Example

As an example, consider the following set of processes, assumed to have arrived at time 0, in the order P1, P2, ..., P5, with the length of the CPU burst given in milliseconds:

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

Gantt Chart



- Process P2 is started at time 0, since it is having the highest priority among the processes in the queue.
- After execution of P2 process, process P5 is started at time 1.
- Then P1,P3 and P4 as P2 processes are executed. As P2 has the shortest time, P2 will get executed first.

Waiting time:

- Waiting time for process p1 = 6
- Waiting time for process p2 = 0

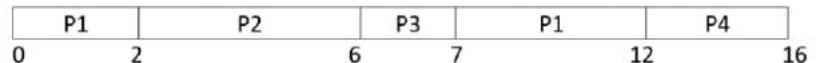
- Waiting time for process p3 = 16
- Waiting time for process p4 = 18
- Waiting time for process p5 = 1
- The average waiting time = $(6+0+16+18+1)/5 = 41/5 = 8.2$ milliseconds.

Turnaround time:

- Turnaround time for P1 = 6 + 10
- Turnaround time for P2 = 0 + 1
- Turnaround time for P3 = 16 + 2
- Turnaround time for P4 = 18 + 1
- Turnaround time for P5 = 1 + 5
- Average Turnaround time = $(16+1+18+19+6) / 5 = 12$ milliseconds

Preemptive priority example

Process	Arrival time	Burst Time	Priority
P1	0	7	3
P2	2	4	1
P3	4	1	2
P4	5	4	4



Waiting time:

- Waiting time for process p1 = $7-2-0=5$
- Waiting time for process p2 = $2-2=0$
- Waiting time for process p3 = $6-4=2$
- Waiting time for process p4 = $12-5=7$
- The average waiting time = $(5+0+2+7)/4 = 3.5$ milliseconds.

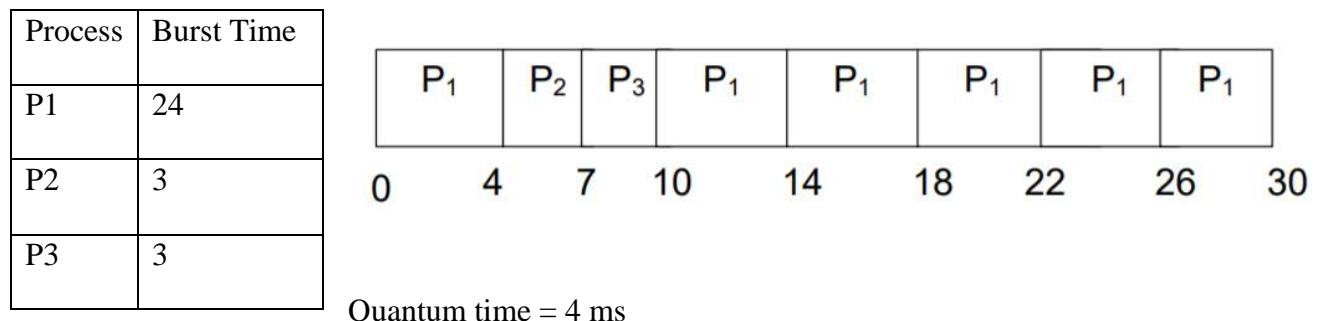
Turnaround time:

- Turnaround time for P1 = 11
- Turnaround time for P2 = 4
- Turnaround time for P3 = 3
- Turnaround time for P4 = 11
- Average Turnaround time = $(11+4+3+11) / 4 = 7.25$ milliseconds

Round Robin Scheduling

- The round-robin (RR) scheduling algorithm is designed especially for timesharing systems. It is similar to FCFS scheduling, but preemption is added to switch between processes.
- A small unit of time, called a time quantum or time slice, is defined. A time quantum is generally from 10 to 100 milliseconds.

- The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to t time quantum.
 - The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after t time quantum, and dispatches the process. One of two things will then happen:
 - The process may have a CPU burst of less than t time quantum.
 - In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue.
 - Otherwise, if the CPU burst of the currently running process is longer than t time quantum, the timer will go off and will cause an interrupt to the operating system.
 - A context switch will be executed, and the process will be put at the tail of the ready queue.
 - The CPU scheduler will then select the next process in the ready queue.
- The average waiting time under the RR policy is often long.
- Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:



If we use a time quantum of 4 milliseconds, then process P1 gets the first 4 milliseconds.

- Since it requires another 20 milliseconds, it is preempted after the first time quantum, and the CPU is given to the next process in the queue, process P2
- Since process P2 does not need 4 milliseconds, it quits before its time quantum expires.
- The CPU is then given to the next process, process P3.
- Once each process has received t time quantum, the CPU is returned to process P1 for an additional time quantum. The resulting RR schedule is:

Waiting time:

- Waiting time for process p1 = $(10 - 4)$
- Waiting time for process p2 = 4
- Waiting time for process p3 = 7
- The average waiting time = $((10-4)+4+7) / 3 = 17/3 = 5.66$ milliseconds.

Turnaround time:

- Turnaround time for P1 = $(10-4) + 24$
- Turnaround time for P2 = $4 + 3$
- Turnaround time for P3 = $7 + 3$
- Average Turnaround time = $(30+7+10) / 3 = 15.6$ milliseconds

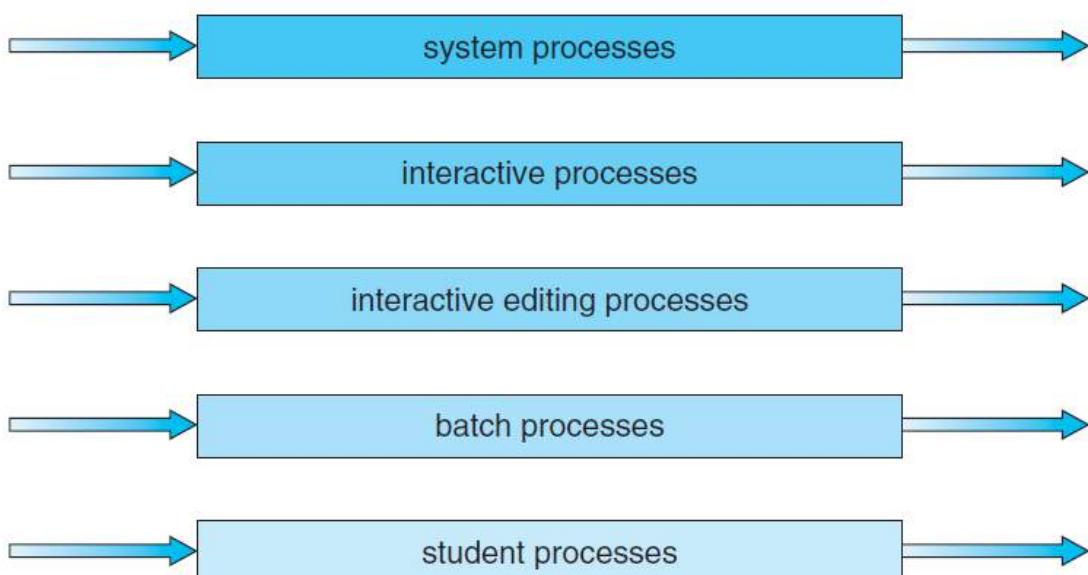
Multilevel Queue scheduling

- Another class of scheduling algorithm has been created for situations in which processes are easily classified into different groups. For example, a common division is made between foreground (interactive) processes and background (batch) processes. These two types of

processes have different response-time requirements and so may have different scheduling needs. In addition, foreground processes may have priority (externally defined) over background processes.

- A multilevel queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type
- Each queue has its own scheduling algorithm. For example, separate queues might be used for foreground and background processes.
 - The foreground queue might be scheduled by an RR algorithm,
 - while the background queue is scheduled by an FCFS algorithm.

highest priority



lowest priority

Advantages

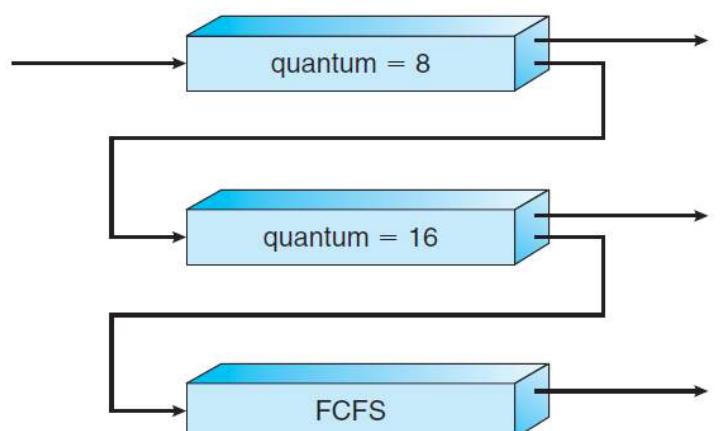
Tasks can be categorized

Disadvantages

- Could result in starvation of low priority jobs
- Once a process is assigned to a Q it cannot be changed
- Solution Multi Level Feedback Queues

Multilevel Feedback Queue scheduling

- The multilevel feedback-queue scheduling algorithm, allows a process to move between queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves I/O-bound and



interactive processes in the higher-priority queues.

- Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.

For example, consider a multilevel feedback-queue scheduler with three queues, numbered from 0 to 2 .

- The scheduler first executes all processes in queue 0.
- Only when queue 0 is empty will it execute processes in queue 1.
- Similarly, processes in queue 2 will only be executed if queues 0 and 1 are empty.
- A process that arrives for queue 1 will preempt a process in queue 2.
- A process that arrives for queue 0 will in turn, preempt a process queue 1.

In general, a multilevel feedback-queue scheduler is defined by the following parameters:

- The number of queues
- The scheduling algorithm for each queue
- The method used to determine when to upgrade a process to a higher priority queue
- The method used to determine when to demote a process to a lower priority queue
- The method used to determine which queue a process will enter when that process needs service

Real time scheduling

Correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced. Tasks or processes attempt to control or react to events that take place in the outside world during execution. These events occur in “real time” and process must be able to keep the events with them.

Examples: process control in industrial plants, robotics, air traffic control, telecommunications, military command and control systems.

There are two types of real time systems. They are:

- Hard real-time task - one that must meet its deadline, otherwise it will cause unacceptable damage or a fatal error to the system
- Soft real-time task - has an associated deadline that is desirable but not mandatory and may find small relaxation to schedule and complete the task even if it has passed its deadline.

Real-time operating systems have requirements in five general areas:

- Determinism - whether the system has sufficient capacity to handle all requests within the required time
- Responsiveness - Concerned with how long, after acknowledgment, it takes an operating system to service the interrupt
- User control - User should be able to distinguish between hard and soft tasks and to specify relative priorities within each class
- Reliability - Real-time systems respond to and control events in real time so loss or degradation of performance may have catastrophic consequences.
- Fail-safe operation - A characteristic that refers to the ability of a system to fail in such a way as to preserve as much capability and data as possible.
- Real-time operating systems are designed with the objective of starting real-time tasks as rapidly as possible and emphasize rapid interrupt handling and task dispatching. Priorities

provide a crude tool and do not capture the requirement of completion (or initiation) at the most valuable time

A. Rate Monotonic scheduling

The rate-monotonic scheduling algorithm schedules periodic tasks using a static priority policy with preemption. If a lower-priority process is running and a higher-priority process becomes available to run, it will preempt the lower-priority process. Upon entering the system, each periodic task is assigned a priority inversely based on its period. The shorter the period, the higher the priority; the longer the period, the lower the priority. The rationale behind this policy is to assign a higher priority to tasks that require the CPU more often.

Furthermore, rate-monotonic scheduling assumes that the processing time of a periodic process is the same for each CPU burst.

B. Dead Line Scheduling

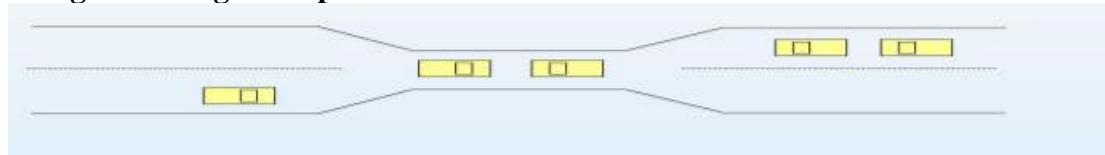
The earlier the deadline, the higher the priority; the later the deadline, the lower the priority.

Deadlocks

Introduction to deadlock

A process requests resources; if the resources are not available at that time, the process enters a waiting state. Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting processes. This situation is called a **deadlock**.

Bridge Crossing Example



Traffic only in one direction. Each section of a bridge can be viewed as a resource. If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback). Several cars may have to be backed up if a deadlock occurs. Starvation is possible.

Another Eg.,

System has 2 tape drives. P1 and P2 each hold one tape drive and each needs another one.

System Model

1. Resource types R_1, R_2, \dots, R_m
2. CPU cycles, memory space, I/O devices includes resources
3. Each resource type R_i has W_i instances.

Each process utilizes a resource as follows:

- request
- use
- release

Deadlock Characterization

Deadlock can arise if **four conditions hold simultaneously**.

- **Mutual exclusion:** only one process at a time can use a resource.
- **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes.
- **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task.
- **Circular wait:** there exists a set $\{P_0, P_1, \dots, P_n\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1 , P_1 is waiting for a resource that is held by P_2 , ..., P_{n-1} is waiting for a resource that is held by P_n , and P_n is waiting for a resource that is held by P_0 .

Resource-Allocation Graph

A set of vertices V and a set of edges E.

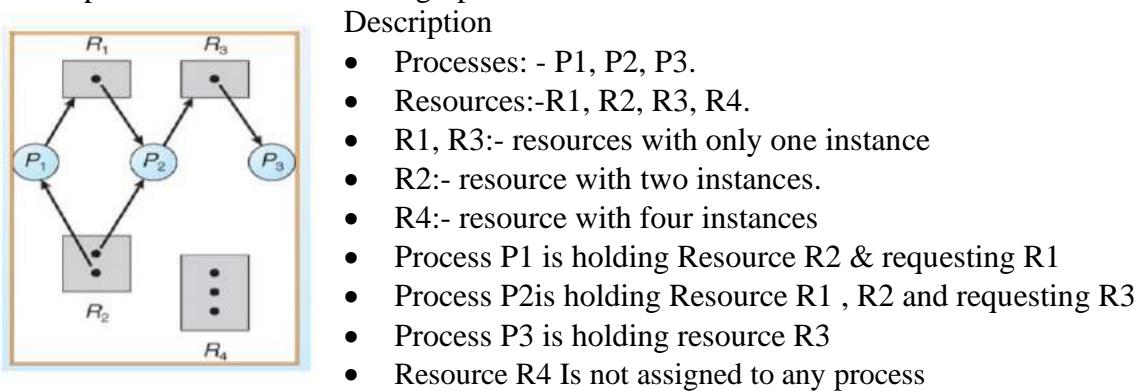
V is partitioned into two types: $P = \{P_1, P_2, \dots, P_n\}$, the set consisting of all the processes in the system. $R = \{R_1, R_2, \dots, R_m\}$, the set consisting of all resource types in the system.

E (edges) is partitioned into two types as well:

Request edge – directed edge $P_i \rightarrow R_j$ [defining the process is requesting the resource]

Assignment edge – directed edge $R_j \rightarrow P_i$ [defining the resource has been assigned to process]

Example of resource allocation graph



Basic Facts - Describing a deadlock

- if the graph contains no cycles, then no process in the system is deadlocked.
- If the graph does contain a cycle, then a deadlock may exist
- If each resource type has exactly one instance, then a cycle \Rightarrow that a deadlock has occurred.
- If each resource type has several instances, then a cycle does not necessarily imply that a deadlock has occurred.

Methods for Handling Deadlocks

- Deadlock Prevention - Ensure that the system will never enter a deadlock state
- Deadlock Avoidance - Allow the system to enter a deadlock state and then recover.
- Deadlock Detection and recovery - Ignore the problem and pretend that deadlocks never occur in the system; used by most operating systems, including UNIX. On detecting the deadlock the system will be recovered.

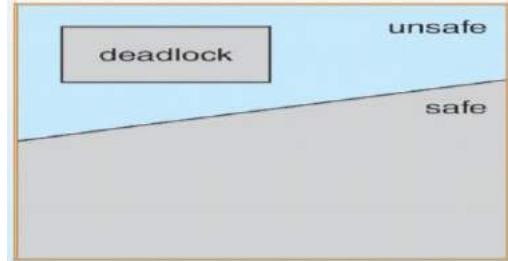
Safe State

- When a process requests an available resource, system must decide if immediate allocation leaves the system in a safe state. System is in safe state if there exists a safe sequence of all processes.
- Sequence $\langle P_1, P_2, \dots, P_n \rangle$ is safe if for each P_i , the resources that P_i can still request can be satisfied by currently available resources + resources held by all the P_j , with $j < i$.

- If P_i resource needs are not immediately available, then P_i can wait until all P_j have finished. When P_j is finished, P_i can obtain needed resources, execute, return allocated resources, and terminate.
- When P_i terminates, P_{i+1} can obtain its needed resources, and so on.

Safe, Unsafe , Deadlock State

- If a system is in safe state \Rightarrow no deadlocks.
- If a system is in unsafe state \Rightarrow possibility of deadlock.
- Avoidance \Rightarrow ensure that a system will never enter an unsafe state.



Deadlock Prevention

For a deadlock to occur each of the four necessary conditions must hold. By ensuring that at least one of these conditions cannot hold we can prevent the occurrence of a deadlock.

Denying/prevent one of these 4 necessary conditions

- o Mutual Exclusion condition
- o Hold and Wait condition
- o No Preemption condition
- o Circular Wait condition
- To Prevent/deny Mutual Exclusion condition
Use only shareable resource and make all resources as shareable. But it's impossible for practical system, where some of the resources like printer cannot be shared.
- Prevent/Deny Hold and Wait condition
 - o Method 1 Pre-allocation protocol
 - Require processes to request all the resources before starting. So, process will be allocated all the resources before starting and a process never has to wait for what it needs. The main drawback here is it leads to low utilization of resources and sometimes process may not know required resources at start of run.
 - o Method 2
 - Process can request resources only when the process has none. Process must release all holding resources. Request all immediately needed resources. Process never goes into waiting while holding resources.
- Prevent/deny No Preemption condition
 - o If a process that is holding some resources requests another resource that cannot be immediately allocated then the process goes into waiting, all resources currently held by the waiting process are released.
 - o Preempted resources are added to the list of resources required by the process
 - o Process will be restarted only when it can regain its old resources & the new requested resources
 - o The draw back here is Some resources (e.g. printer, tap drives) cannot be preempted without serious problems and it may require the job to restart
- Prevent/Deny Circular Wait condition
(Order resources) each resource type is assigned a unique integer. Process can request for resources only in increasing order. If a process needs several instances of the same resource. It should issue a single request for all of them.

Deadlock Avoidance

- Processes must tell OS in advance how many resources they will request
- OS Should never allocates resources in a way that could lead to a deadlock

Deadlock Avoidance with Resource-Allocation Graph

- This algorithm can be used only for one instance of each resource type.

Definitions

In addition to the request edge and assignment edge a claim edge is also introduced.

Claim edge $P_i \rightarrow R_j$ indicated that process P_j may request resource R_j in future. Claim edge is Represented by a dashed line.

The algorithm is executed when a process requests a resource. Suppose that process P_i requests resource R_j . **The request can be granted only If converting the request edge $P_i \rightarrow R_j$ to an assignment edge does not result in a cycle in the resource-allocation graph.** i.e., a cycle detection algorithm is used.

- o If no cycle exists, the resource R_j is assigned to process P_i
- o Else the process P_i will have to wait.

Banker's Algorithm

Applicable to system with multiple instances of resource types.

Each process must declare max requirement of each resource type. Max requirement should not exceed total resources. When a process requests a resource it may have to wait. When a process gets all its resources it must return them in a finite amount of time.

Working

Banker's algorithm runs each time when a process requests resource. Bankers algorithm will make requested allocation only when it leads the system in safe state.

Data Structures for the Banker's Algorithm

Let n = number of processes, and m = number of resources types.

- Available: Vector of length m . If available $[j] = k$, there are k instances of resource type R_j available.
- Max: $n \times m$ matrix. If Max $[i,j] = k$, then process P_i may request at most k instances of resource type R_j .
- Allocation: $n \times m$ matrix. If Allocation $[i,j] = k$ then P_i is currently allocated k instances of R_j .
- Need: $n \times m$ matrix. If Need $[i,j] = k$, then P_i may need k more instances of R_j to complete its task. Need $[i,j] = \text{Max}[i,j] - \text{Allocation}[i,j]$

A. Safety Algorithm

1. Let Work and Finish be vectors of length m and n , respectively.

Initialize:

Work = Available

Finish $[i] = \text{false}$ for $i = 1, 2, 3, \dots, n$.

2. Find i such that both:

Finish $[i] = \text{false}$

$\text{Need}_i \leq \text{Work}$

If no such i exists, go to step 4.

3. $\text{Work} = \text{Work} + \text{Allocation}_i$.

Finish $[i] = \text{true}$ go to step 2.

4. If Finish $[i] == \text{true}$ for all i , then the system is in a safe state.

B. Resource-Request algorithm for Process P_i

Request = request vector for process P_i .

- If $\text{Request}_i[j] = k$ then process P_i wants k instances of resource type R_j .
1. If $\text{Request}_i \leq \text{Need}_i$ go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim.
 2. If $\text{Request}_i \leq \text{Available}$, go to step 3. Otherwise P_i must wait, since resources are not available.
 3. Pretend to allocate requested resources to P_i by modifying the state as follows:
 $\text{Available} = \text{Available} - \text{Request}_i;$
 $\text{Allocation}_i = \text{Allocation}_i + \text{Request}_i;$
 $\text{Need}_i = \text{Need}_i - \text{Request}_i$

If safe \Rightarrow the resources are allocated to P_i .

If UNsafe $\Rightarrow P_i$ must wait, and the old resource-allocation state is restored.

Example of Banker's Algorithm

Total No. processes:- 5 [P_0, P_1, P_2, P_3, P_4 .]

Resource types:- 3 [A, B, C.]

Total Instances of each resource type

- o A:- 10 ; B:- 5 ; C:- 7

Snapshot at time T0:

	Allocation	Max	Available
	ABC	ABC	ABC
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Each process has declared the maximum Nos for each resource type required to complete execution. Declaration is listed in under **Max** Allocations listed under **Allocation**

Available shows Nos of free resources available with the system

Finished indicates if process is executing /finishes.

Initialize: calculate Need For each process

- Need should list Nos for each resource type further required to complete execution
- Need = Max – Allocation.
- Mark all executing processes finished as false

	Finished	Allocation	Max	Need	Available
		ABC	ABC	A B C	ABC
P_0	False	0 1 0	7 5 3	7 4 3	3 3 2
P_1	False	2 0 0	3 2 2	1 2 2	
P_2	False	3 0 2	9 0 2	6 0 0	
P_3	False	2 1 1	2 2 2	0 1 1	
P_4	False	0 0 2	4 3 3	4 3 1	

Repeat until all processes finished execution

- Find next executing process such that $\text{Need} \leq \text{Available}$
- Checks Need for P_0 , as Need is higher it checks for P_1 .
- For P_1 Need (1,2,2) \leq Available (3,3,2)
- //assume Resources are allocated to P_1
- P_1 has completed execution P_1 has released resources//
- Mark P_1 as finished (i.e., P_1 finished =true)

- Like this all other process will execute?
- Executing safety algorithm shows that there is safe sequence $\langle P1, P3, P4, P0, P2 \rangle$ and satisfies safety requirement. So there is no deadlock in the system.
- Can request for (3,3,0) by P4 be granted?
- Can request for (0,2,0) by P0 be granted?

Note : Look notes for problems

Deadlock Detection and Recovery

Allow system to enter deadlock state, detects the deadlock and recovers from deadlock

=>Requires

- a deadlock detection algorithm
- a deadlock recovery algorithm

Deadlock detection with Single Instance of Each Resource Type

- Maintain wait-for graph
- Nodes are processes.
- $P_i \rightarrow P_j$ if P_i is waiting for P_j [inherited from RAG]
- Periodically invoke an algorithm that searches for a cycle in the graph.
- An algorithm to detect a cycle in a graph, where n is the number of vertices in the graph.

Several Instances of a Resource Type

- **Available:** A vector of length m indicates the number of available resources of each type.
- **Allocation:** An $n \times m$ matrix defines the number of resources of each type currently allocated to each process.
- **Request:** An $n \times m$ matrix indicates the current request of each process. If Request $[i,j] = k$, then process P_i is requesting k more instances of resource type R_j .

Detection Algorithm

1. Let Work and Finish be vectors of length m and n , respectively
 Initialize:
 a. Work = Available
 b. For $i = 1, 2, \dots, n$,
 if Allocation $_i \neq 0$ then
 Finish[i] = false;
 Otherwise
 Finish[i] = true.
2. Find an index i such that both:
 a. Finish[i] == false
 b. Request $_i \leq$ Work

 If no such i exists, go to step 4
3. Work = Work + Allocation $_i$
 Finish[i] = true
 go to step 2.
4. If Finish[i] == false, for some i , $1 \leq i \leq n$, then the system is in deadlock state. Moreover, if Finish[i] == false, then P_i is deadlocked.

Example of Detection Algorithm

- Five processes P0 through P4; three resource types A (7 instances), B (2 instances), and C (6 instances).
- Snap Shot at Time T0

	<u>Allocation</u>	<u>Request</u>	<u>Available</u>
	A B C	A B C	A B C
P ₀	0 1 0	0 0 0	0 0 0
P ₁	2 0 0	2 0 2	
P ₂	3 0 3	0 0 0	
P ₃	2 1 1	1 0 0	
P ₄	0 0 2	0 0 2	

Sequence <P0,P2,P3,P4,P1> will result in Finish[i] = true for all i and so concludes that there is no deadlock.

But if the Process P2 request an additional resource type as [0,0,1] it will lead to deadlock.

Detection-Algorithm Usage

When, and how often, to invoke depends on:

1. How often a deadlock is likely to occur?
2. How many processes will need to be rolled back?

one for each disjoint cycle If detection algorithm is invoked arbitrarily, there may be many cycles in the resource graph and so we would not be able to tell which of the many deadlocked processes “caused” the deadlock.

Recovery from Deadlocks

- Process Termination
- Resource Preemption

Recovery from Deadlock with Process Termination

- Abort all deadlocked processes.
- Abort one process at a time until the deadlock cycle is eliminated.

To abort one process at a time in which order should we choose to abort?

- Priority of the process.
- How long process has computed, and how much longer to completion?
- Resources the process has used. Resources process needs to complete.
- How many processes will need to be terminated?
- Is process interactive or batch?

Recovery from Deadlock: Resource Pre-emption

- Selecting a victim – minimize cost.
- Rollback – return to some safe state, restart process for that state.
- Starvation – same process may always be picked as victim, include number of roll-back in cost factor.

Chapter 3 Topics:

Memory Management

- Logical Vs Physical address space, Swapping
- Contiguous memory allocation – Fixed and dynamic partition
- Strategies for selecting free holes in dynamic partition
- Paged Memory management
- Structure of Page map table
- Example: Intel 32 bit and 64-bit architectures
- ARM architecture
- Segmented memory management
- Paged segmentation technique

Memory management Introduction

Main memory is a resource that must be allocated and deallocated. Memory Management Techniques determine:

- How the memory is to be (logically) subdivided?
- Where and how a process resides in memory?
- How addressing is performed?
- How process can be relocated?
- How memory is to be protection?
- How memory can be shared by processes?
- How to logical and physically organize memory

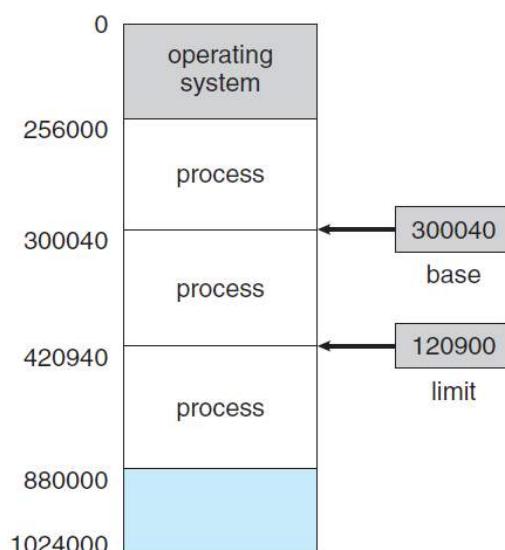
The memory manager is a component in Operating system used to manage memory. Managing and sharing of primary memory and minimizing the access time is the primary goal of memory manager.

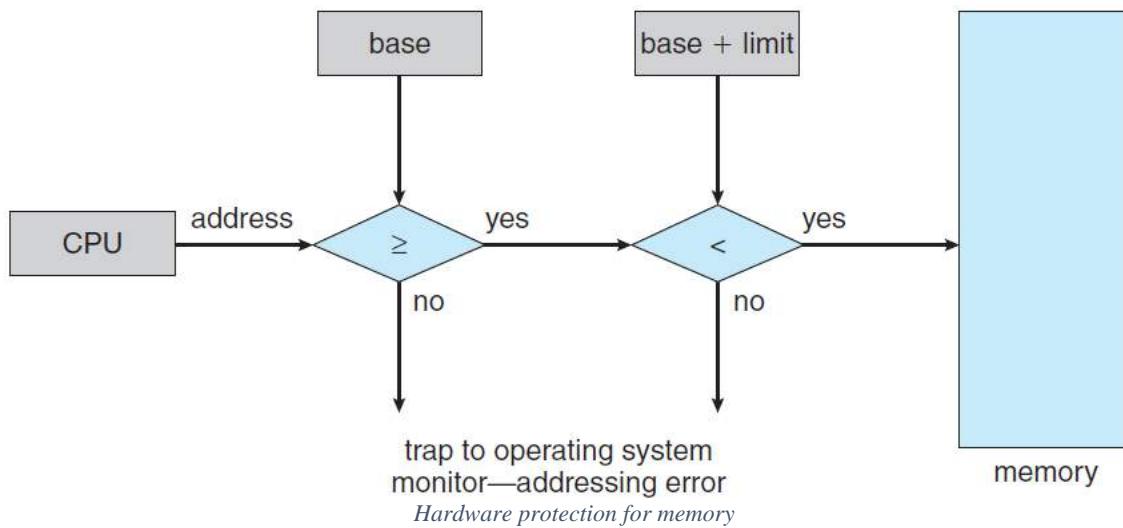
Primary memory management functions include:

1. Allocate primary memory space to processes.
2. Minimize access time
3. Determining allocation policy for memory
4. Deallocation technique and policy.

Basic Hardware

- Each process has a separate memory space.
- The **base register** holds the smallest legal physical memory address; the **limit register** specifies the size of the range.
- Protection of memory space is accomplished by having the CPU hardware compare every address generated in user mode with the registers. Any attempt by a program executing in user mode to access operating-system memory or other users' memory results in a trap to the operating system, which treats the attempt as a **fatal** error. This scheme prevents a user program from (accidentally or deliberately) modifying the code or data structures of either the operating system or other users.





Addressing Requirements of a Process

User programs go through several steps before being run

- source program - Addresses are generally symbolic (variable name).
- compiler time - Addresses are in relocatable format (in terms of offsets).
- linkage editor or loader - Addresses are in absolute addresses format (physical addresses).

Binding of Instructions and Data to Memory

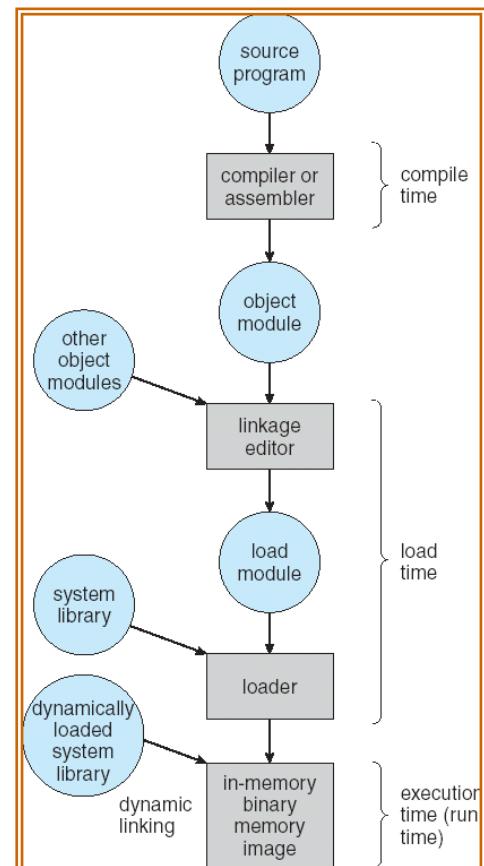
To be executed, the program must be brought into memory and placed within a process.

Address binding of instructions and data to memory addresses can happen at three different stages.

- **Compile time:** If memory location known a priori, **absolute code** can be generated; must recompile code if starting location changes
- **Load time:** Must generate **relocatable code** if memory location is not known at compile time
- **Execution time:** Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., base and limit registers)

Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate **physical address space** is central to proper memory management



- **Logical address** – generated by the CPU; also referred to as **virtual address**
- **Physical address** – address seen by the memory unit
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme

Memory-Management Unit (MMU)

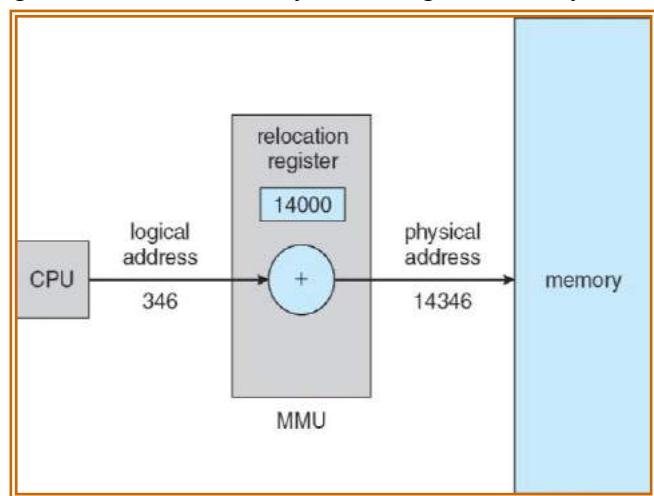
The memory management unit translates the logical address to physical address. It is a hardware device that maps virtual to physical address

In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.

The user program deals with *logical* addresses; it never sees the *real* physical addresses.

Simple MMU scheme

- Relocation register contains the starting address of the program
- The value in the relocation register is added to every logical address generated by the CPU
- Dynamic binding can be implemented using a relocation register



Dynamic loading :

With dynamic loading, a routine is not loaded until it is called. All routines are kept on disk in a relocatable load format. The main program is loaded into memory and is executed.

Methods to execute programs larger than available memory

Overlays

Overlay is one of the techniques to run a program that is bigger than the size of the physical memory. The idea of overlays is to keep in memory only those instructions and data that are needed working.

- divide the program into modules in such a way that
- all modules are not needed in the memory at the same time.
- Programmer specifies which modules can overlay each other
- The linker inserts commands to invoke the loader
- When invoked linker replaces older modules with newer modules

Advantages

- Reduced memory requirements

Disadvantages

- Overlap map must be specified by programmer
- Programmer must know memory requirements

Swapping

A process needs to be in the memory to be executed. However, a process can be swapped temporarily out of memory to a backing store & then brought back into memory for continued execution.

Requirements for swap is a backing store – secondary storage disk. That it should be a fast disk ,large enough to accommodate copies of all memory images and provide direct access.

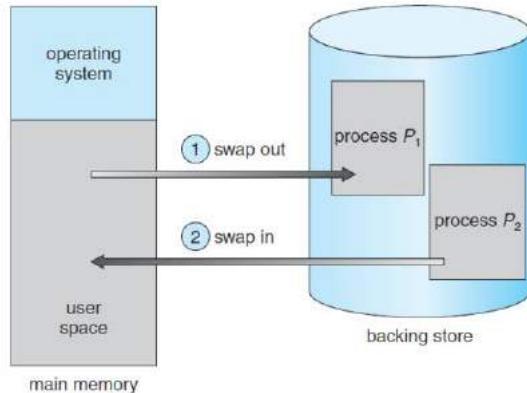
Steps involved

- Roll out :-Copying memory image of the process into backing store
- Roll in :-Copying memory image of the process from backing store to main memory

Uses

Used for implementing priority-based scheduling algorithms

- o lower-priority process is swapped out
- o higher-priority process can be loaded and executed.



Disadvantages

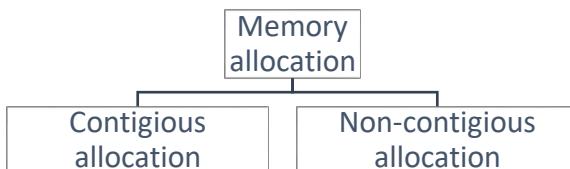
- Major part of swap time is transfer time
 - o copying from memory to backing store
 - o copying from backing store to memory
- Total transfer time is directly proportional to the amount of memory swapped.
- Context switching time is quite high in this scheme.
- Before swapping out a process, we have to ensure that it is completely idle.

Allocating Main Memory for programs

The operating system has to allocate memory for both user programs & System programs

Memory Allocation Techniques

The operating system follows different methods for allocating memory space for executing programs



Contiguous memory allocation

- Main memory usually divided into two partitions:
- Partition for Resident operating system
- Partition for User processes
- Every process is loaded into a single contiguous partition of the memory

Non-Contiguous memory allocation

- Main memory usually divided into multiple partitions:
- Process also can be split into multiple parts
- Each part of the program can be loaded into different parts of the memory
- Entire program need not be contiguous

Contiguous memory allocation

Types of Contiguous memory allocation

- Single-partition Allocation
- Multiple-Partition Allocation – Fixed Partitions
- Multiple-partition Allocation – Dynamic Partitions

Single-partition Allocation

- Memory split into two partitions one for Operating system and the other one for user programs. The user programs exist as one single partition

- A single process is loaded into the memory at a time. Next program can be loaded only after finishing current process

Advantages:

- Simplicity and no special hardware required

Disadvantages:

- Cannot support multiprogramming & multi user
- Cannot support multi-user.

Multiple-Partition Allocation – Fixed Partitions

Desirable in multiprogramming environment.

- Main memory is divided into number of static partitions at system generation time. A process may be loaded in to partition of equal or greater size. Partition size may be either equal or unequal as shown in the figure.
- The degree of multiprogramming is bounded by the number of partitions.
- Originally used in IBM OS/360 and no longer in use now.



Advantages

Simple to implement and little overhead

Disadvantage

Inefficient use of memory due to internal fragmentation; maximum number of active process is fixed.

Multiple-Partition Allocation – Dynamic Partitions

Partitions are of variable length and variable numbers. Process is allocated exactly the memory as required.

Working of dynamic partition scheme

Operating system maintains information about: Allocated partitions and Free partitions (hole).

Initially, all the memory is available and is considered as a single block (one big hole).

When a process arrives, we search for a hole large enough for the process

- If a hole of required size is found, it is allocated to the process
- If we find a hole, which is too large, it is split into two one part is allocated to the arriving process and the other is returned to the set of holes.
- When a process completes, the memory is freed and is placed in to the set of holes
- If the new hole is adjacent to other holes they are packed & merged into a single hole

Dynamic Allocation Placement Algorithms

How to satisfy a request of size n bytes from a list of free holes? Solutions to the problem of allocating memory

First-fit:

- Allocate the first hole that is big enough
- The search can start at the beginning
- or start from where the previous first-fit search was ended.

Best-fit:

- Allocate the smallest hole that is big enough
- Must search entire list
- Produces the smallest leftover hole.

Worst-fit:

- Allocate the largest available hole
- Must also search entire list

Produces Given five memory partitions of 100Kb, 500Kb, 200Kb, 300Kb, 600Kb (in order), how would the first-fit, best-fit, and worst-fit algorithms place

processes of 212 Kb, 417 Kb, 112 Kb, and 426 Kb (in order)? Which algorithm makes the most efficient use of memory?

First-fit:

212K is put in 500K partition

417K is put in 600K partition

112K is put in 288K partition (new partition 288K = 500K - 212K)

426K must wait

Best-fit:

212K is put in 300K partition

417K is put in 500K partition

112K is put in 200K partition

426K is put in 600K partition

Worst-fit:

212K is put in 600K partition

417K is put in 500K partition

112K is put in 388K partition

426K must wait

Internal Fragmentation

- Allocated memory may be slightly larger than requested memory.
- This size difference is internal to a memory partition
- Hence cannot be used by any other process.

External Fragmentation

- Total memory space exists to satisfy a request, but it is not contiguous.
- Hence cannot be used by any process.

External fragmentation can be reduced by compaction.

Compaction/De-fragmentation

Place all free memory together in one large block and all the allocated blocks together one side.

UNIT - 5

DISK SCHEDULING ALGORITHMS

FCFS Disk Scheduling Algorithms

Given an array of disk track numbers and initial head position, our task is to find the total number of seek operations done to access all the requested tracks if **First Come First Serve (FCFS)** disk scheduling algorithm is used.

First Come First Serve (FCFS)

FCFS is the simplest [disk scheduling algorithm](#). As the name suggests, this algorithm entertains requests in the order they arrive in the disk queue. The algorithm looks very fair and there is no starvation (all requests are serviced sequentially) but generally, it does not provide the fastest service.

Algorithm:

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. ‘head’ is the position of disk head.
2. Let us one by one take the tracks in default order and calculate the absolute distance of the track from the head.
3. Increment the total seek count with this distance.
4. Currently serviced track position now becomes the new head position.
5. Go to step 2 until all tracks in request array have not been serviced.

Example:

Input:

Request sequence = { 176, 79, 34, 60, 92, 11, 41, 114 }

Initial head position = 50

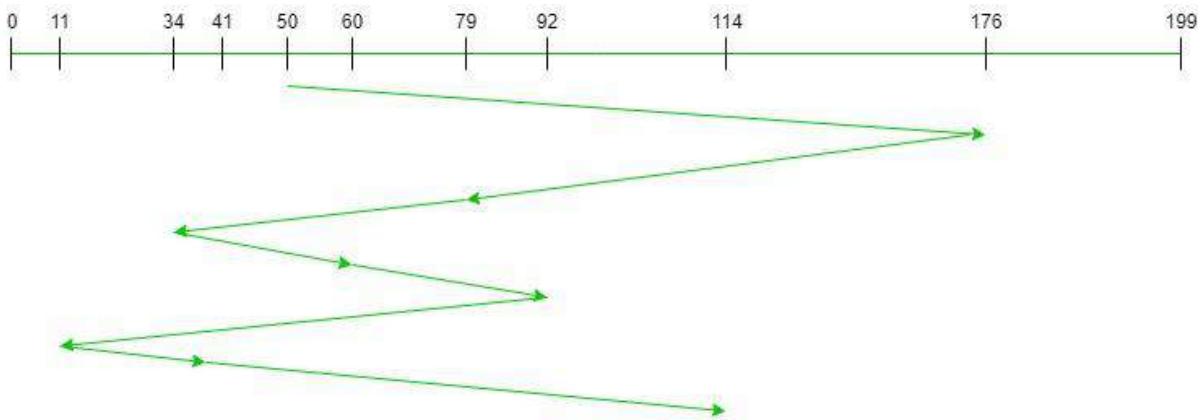
Output:

Total number of seek operations = 510

Seek Sequence is

176
79
34
60
92
11
41
114

The following chart shows the sequence in which requested tracks are serviced using FCFS



SSTF disk scheduling algorithm

Given an array of disk track numbers and initial head position, our task is to find the total number of seek operations done to access all the requested tracks if **Shortest Seek Time First (SSTF)** is a disk scheduling algorithm is used.

Shortest Seek Time First (SSTF) –

Basic idea is the tracks which are closer to current disk head position should be serviced first in order to *minimise the seek operations*.

Algorithm –

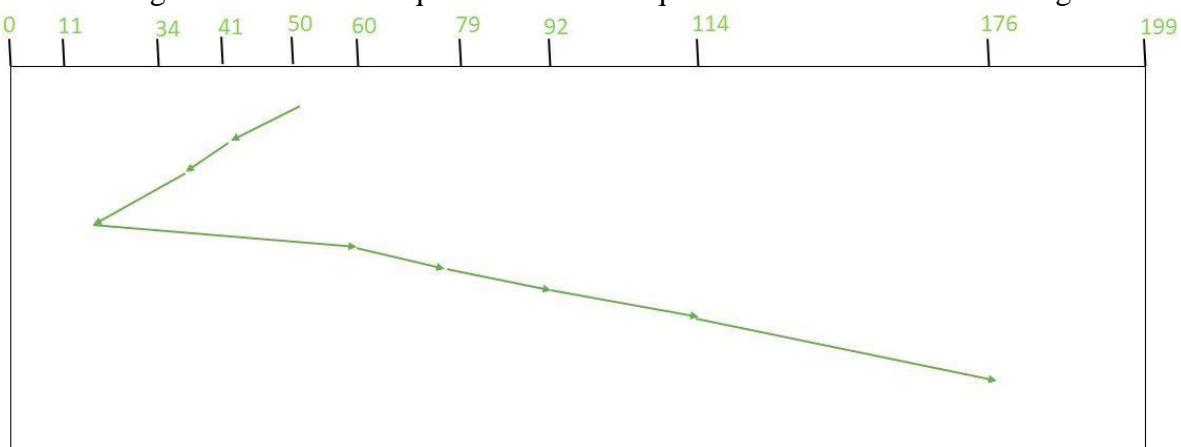
1. Let Request array represents an array storing indexes of tracks that have been requested. ‘head’ is the position of disk head.
2. Find the positive distance of all tracks in the request array from head.
3. Find a track from requested array which has not been accessed/serviced yet and has minimum distance from head.
4. Increment the total seek count with this distance.
5. Currently serviced track position now becomes the new head position.
6. Go to step 2 until all tracks in request array have not been serviced.

Example –

Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}

Initial head position = 50

The following chart shows the sequence in which requested tracks are serviced using SSTF.



Therefore, total seek count is calculates as:

$$\begin{aligned} &= (50-41)+(41-34)+(34-11)+(60-11)+(79-60)+(92-79)+(114-92)+(176-114) \\ &= 204 \end{aligned}$$

SCAN (Elevator) Disk Scheduling Algorithms

Given an array of disk track numbers and initial head position, our task is to find the total number of seek operations done to access all the requested tracks if SCAN disk scheduling algorithm is used.

SCAN (Elevator) algorithm

In SCAN disk scheduling algorithm, head starts from one end of the disk and moves towards the other end, servicing requests in between one by one and reach the other end. Then the direction of the head is reversed and the process continues as head continuously scan back and forth to access the disk. So, this algorithm works as an elevator and hence also known as the **elevator algorithm**. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Algorithm:-

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. ‘head’ is the position of disk head.
2. Let direction represents whether the head is moving towards left or right.
3. In the direction in which head is moving service all tracks one by one.
4. Calculate the absolute distance of the track from the head.
5. Increment the total seek count with this distance.
6. Currently serviced track position now becomes the new head position.
7. Go to step 3 until we reach at one of the ends of the disk.
8. If we reach at the end of the disk reverse the direction and go to step 2 until all tracks in request array have not been serviced.

9. Example:

10. Input:

11. Request sequence = { 176, 79, 34, 60, 92, 11, 41, 114 }
12. Initial head position = 50
13. Direction = left (We are moving from right to left)
- 14.

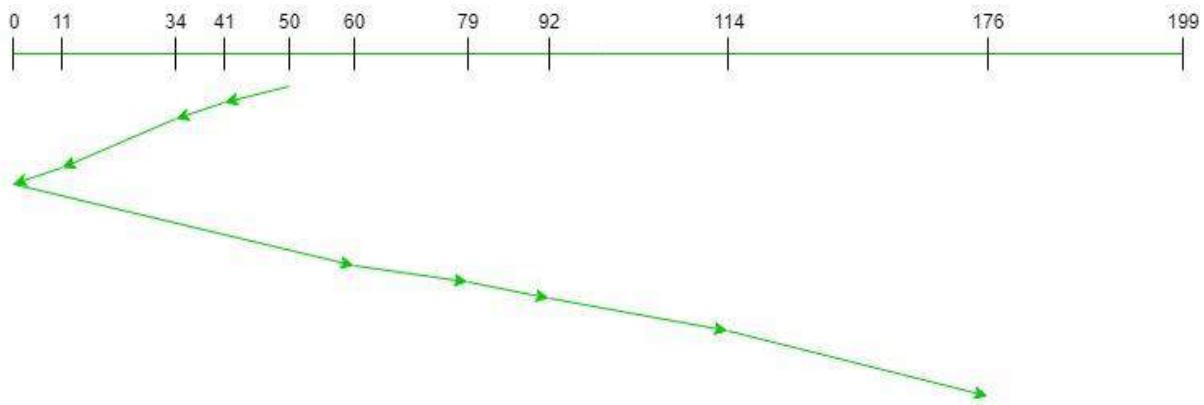
15. Output:

16. Total number of seek operations = 226
17. Seek Sequence is

18. 41
19. 34
20. 11
21. 0
22. 60
23. 79
24. 92
25. 114

26. 176

The following chart shows the sequence in which requested tracks are serviced using SCAN.



Therefore, the total seek count is calculated as:

$$\begin{aligned} &= (50-41)+(41-34)+(34-11) \\ &+ (11-0)+(60-0)+(79-60) \\ &+ (92-79)+(114-92)+(176-114) \\ &= 226 \end{aligned}$$

LOOK Disk Scheduling Algorithm

Given an array of disk track numbers and initial head position, our task is to find the total number of seek operations done to access all the requested tracks if *LOOK* disk scheduling algorithm is used. Also, write a program to find the seek sequence using *LOOK* disk scheduling algorithm.

LOOK Disk Scheduling Algorithm:

LOOK is the advanced version of **SCAN (elevator) disk scheduling algorithm** which gives slightly better seek time than any other algorithm in the hierarchy (*FCFS->SRTF->SCAN->C-SCAN->LOOK*). The LOOK algorithm services request similarly as SCAN algorithm meanwhile it also “looks” ahead as if there are more tracks that are needed to be serviced in the same direction. If there are no pending requests in the moving direction the head reverses the direction and start servicing requests in the opposite direction.

The main reason behind the better performance of LOOK algorithm in comparison to SCAN is because in this algorithm the head is not allowed to move till the end of the disk.

Algorithm:

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. ‘head’ is the position of disk head.
2. The intial direction in which head is moving is given and it services in the same direction.
3. The head services all the requests one by one in the direction head is moving.

4. The head continues to move in the same direction until all the requests in this direction are not finished.
5. While moving in this direction calculate the absolute distance of the track from the head.
6. Increment the total seek count with this distance.
7. Currently serviced track position now becomes the new head position.
8. Go to step 5 until we reach at last request in this direction.
9. If we reach where no requests are needed to be serviced in this direction reverse the direction and go to step 3 until all tracks in request array have not been serviced.

Examples:

Input:

Request sequence = { 176, 79, 34, 60, 92, 11, 41, 114 }

Initial head position = 50

Direction = right (We are moving from left to right)

Output:

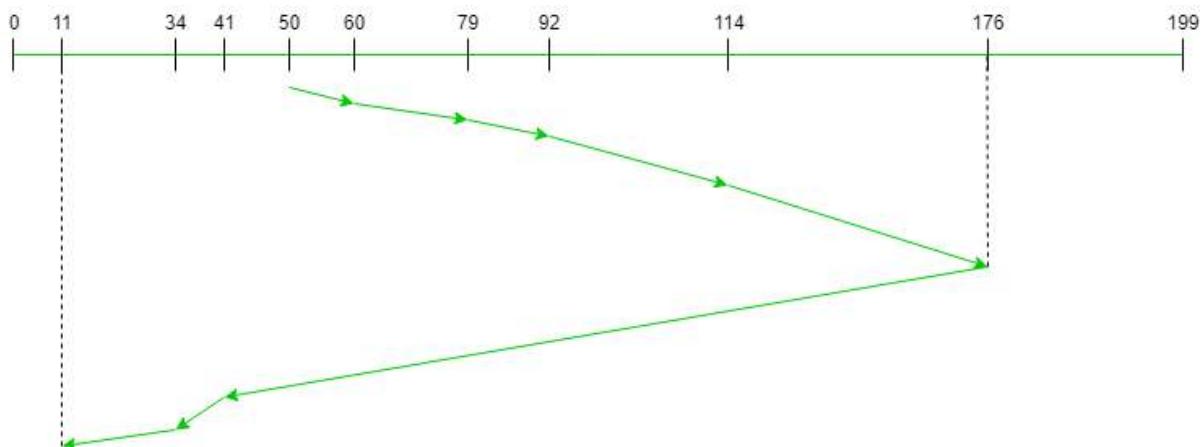
Initial position of head: 50

Total number of seek operations = 291

Seek Sequence is

60
79
92
114
176
41
34
11

The following chart shows the sequence in which requested tracks are serviced using LOOK.



Therefore, the total seek count is calculated as:

$$\begin{aligned}
 &= (60-50)+(79-60)+(92-79) \\
 &\quad +(114-92)+(176-114) \\
 &\quad +(176-41)+(41-34)+(34-11)
 \end{aligned}$$

C-LOOK Disk Scheduling Algorithm

Given an array of disk track numbers and initial head position, our task is to find the total number of seek operations done to access all the requested tracks if **C-LOOK** disk scheduling algorithm is used. Also, write a program to find the seek sequence using **C-LOOK** disk scheduling algorithm.

C-LOOK (Circular LOOK) Disk Scheduling Algorithm:

C-LOOK is an enhanced version of both **SCAN** as well as **LOOK** disk scheduling algorithms. This algorithm also uses the idea of wrapping the tracks as a circular cylinder as **C-SCAN** algorithm but the seek time is better than **C-SCAN** algorithm. We know that **C-SCAN** is used to avoid starvation and services all the requests more uniformly, the same goes for **C-LOOK**.

In this algorithm, the head services requests only in one direction(either left or right) until all the requests in this direction are not serviced and then jumps back to the farthest request on the other direction and service the remaining requests which gives a better uniform servicing as well as avoids wasting seek time for going till the end of the disk.

Algorithm:-

1. Let Request array represents an array storing indexes of the tracks that have been requested in ascending order of their time of arrival and **head** is the position of the disk head.
2. The initial direction in which the head is moving is given and it services in the same direction.
3. The head services all the requests one by one in the direction it is moving.
4. The head continues to move in the same direction until all the requests in this direction have been serviced.
5. While moving in this direction, calculate the absolute distance of the tracks from the head.
6. Increment the total seek count with this distance.
7. Currently serviced track position now becomes the new head position.
8. Go to step 5 until we reach the last request in this direction.
9. If we reach the last request in the current direction then reverse the direction and move the head in this direction until we reach the last request that is needed to be serviced in this direction without servicing the intermediate requests.
10. Reverse the direction and go to step 3 until all the requests have not been serviced.

Examples:

Input:

Request sequence = {176, 79, 34, 60, 92, 11, 41, 114}

Initial head position = 50

Direction = right (Moving from left to right)

Output:

Initial position of head: 50

Total number of seek operations = 156

Seek Sequence is

60

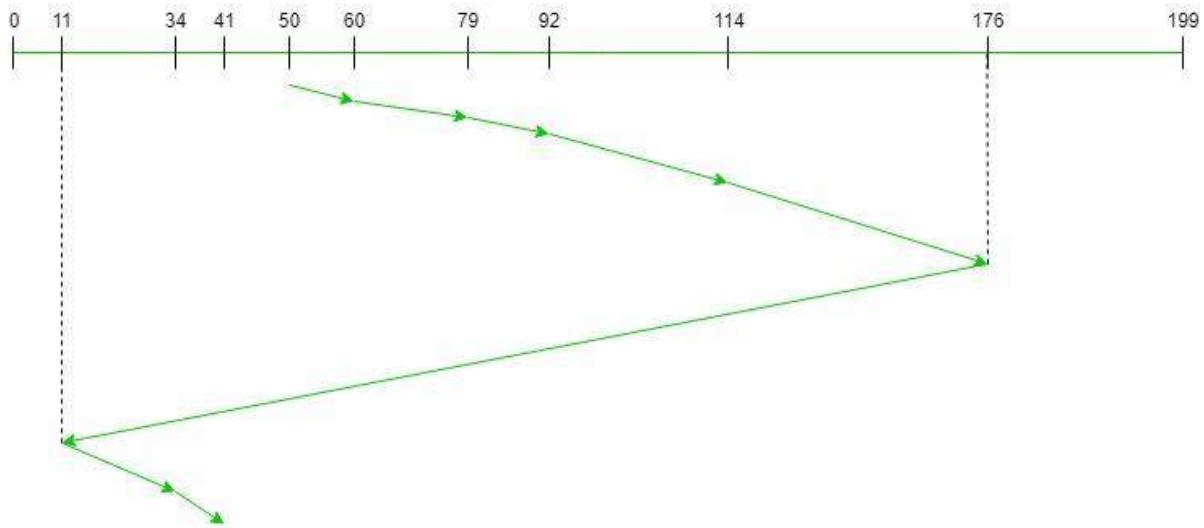
79

92

114

176
11
34
41

The following chart shows the sequence in which requested tracks are serviced using C-LOOK.



C-SCAN Disk Scheduling Algorithm

Given an array of disk track numbers and initial head position, our task is to find the total number of seek operations done to access all the requested tracks if **C-SCAN** disk scheduling algorithm is used.

What is C-SCAN (Circular Elevator) Disk Scheduling Algorithm?

Circular SCAN (C-SCAN) scheduling algorithm is a modified version of SCAN disk scheduling algorithm that deals with the inefficiency of SCAN algorithm by servicing the requests more uniformly. Like SCAN (Elevator Algorithm) C-SCAN moves the head from one end servicing all the requests to the other end. However, as soon as the head reaches the other end, it immediately returns to the beginning of the disk without servicing any requests on the return trip (see chart below) and starts servicing again once reaches the beginning. This is also known as the “Circular Elevator Algorithm” as it essentially treats the cylinders as a circular list that wraps around from the final cylinder to the first one.

Algorithm:

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. ‘head’ is the position of disk head.
2. The head services only in the right direction from 0 to size of the disk.
3. While moving in the left direction do not service any of the tracks.
4. When we reach at the beginning(left end) reverse the direction.
5. While moving in right direction it services all tracks one by one.
6. While moving in right direction calculate the absolute distance of the track from the head.
7. Increment the total seek count with this distance.
8. Currently serviced track position now becomes the new head position.
9. Go to step 6 until we reach at right end of the disk.

10. If we reach at the right end of the disk reverse the direction and go to step 3 until all tracks in request array have not been serviced.

Examples:

Input:

Request sequence = { 176, 79, 34, 60, 92, 11, 41, 114 }

Initial head position = 50

Output:

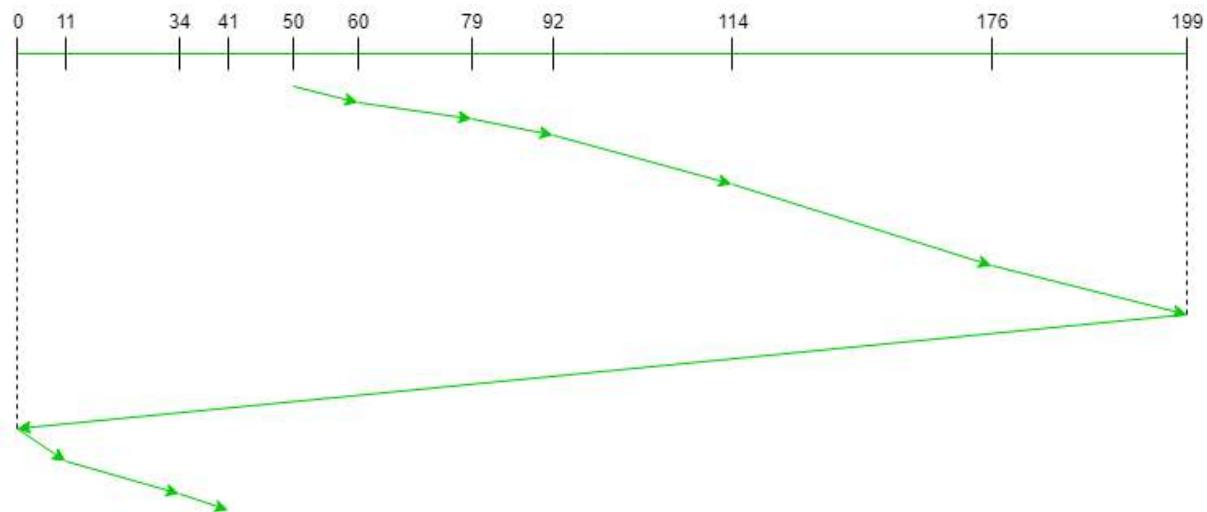
Initial position of head: 50

Total number of seek operations = 190

Seek Sequence is

60
79
92
114
176
199
0
11
34
41

The following chart shows the sequence in which requested tracks are serviced using SCAN.



Therefore, the total seek count is calculated as:

$$\begin{aligned} &= (60-50)+(79-60)+(92-79) \\ &\quad +(114-92)+(176-114)+(199-176)+(199-0) \\ &\quad +(11-0)+(34-11)+(41-34) \end{aligned}$$

18CSC205J Operating Systems

UNIT I

Course Learning Rationale

- Introduce the key role of an Operating System
- Insist the Process Management functions of an Operating System

Course Learning Outcomes

- Identify the need of an Operating system
- Know the Process management functions of an Operating system

Learning Resources

- 0.Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Operating systems, 9th ed., John Wiley & Sons, 2013
- 1.William Stallings, Operating Systems-Internals and Design Principles, 7th ed., Prentice Hall, 2012
- 2.Andrew S.Tanenbaum, Herbert Bos, Modern Operating systems, 4th ed., Pearson, 2015
- 3.Bryant O'Hallaxn, Computer systems- A Programmer's Perspective, Pearson, 2015



Edit with WPS Office

Contents

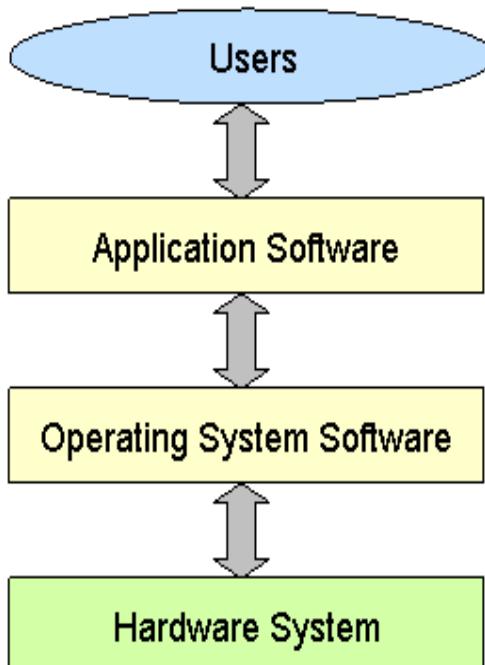
- Operating System Objectives and functions
- Gaining the role of Operating systems
- The evolution of operating system, Major achievement
- Understanding the evolution of Operating systems from early batch processing systems to modern complex systems
- OS Design considerations for Multiprocessor and Multicore
- Understanding the key design issues of Multiprocessor Operating systems and Multicore Operating systems
- PROCESS CONCEPT– Processes, PCB
- Understanding the Process concept and Maintenance of PCB by OS
- Threads – Overview and its Benefits
- Understanding the importance of threads
- Process Scheduling : Scheduling Queues, Schedulers, Context switch
- Understanding basics of Process scheduling
- Operations on Process – Process creation, Process termination
- Understanding the system calls – fork(),wait(),exit()
- Inter Process communication : Shared Memory, Message Passing ,Pipe()
- Understanding the need for IPC
- PROCESS SYNCHRONIZATION: Background, Critical section Problem
- Understanding the race conditions and the need for the Process synchronization

What is operating system?

Definition:

- The operating system acts as an Interface between the user and computer hardware
- Without an operating system, a computer would be useless

“OS is designed in such a way, that it is convenient to use in an efficient manner”



Hardware

- CPU Central processing unit
- Memory
- I/O devices



• Computer Hardware is the physical parts or components of a computer

Software

- Application Programs
- Word Processors
- Microsoft Office



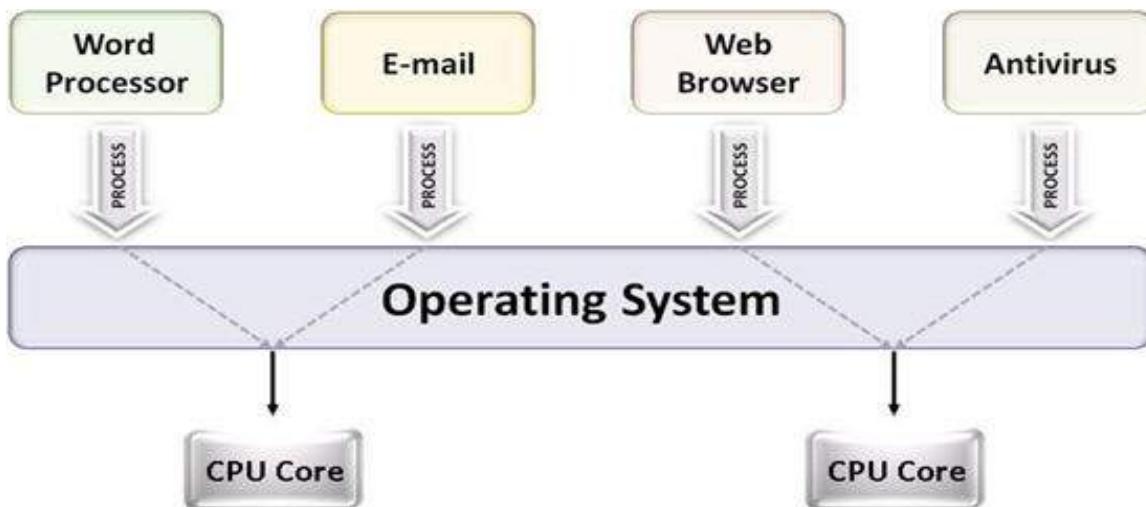
- Computer software or simply **software** is any set of instructions that directs a **computer** to perform specific operations.

Some Examples of Operating System



Tasks performed by Operating System

1. It manages computer hardware
2. It controls and coordinates the computer H/W
3. It specifies, how various resources like hardware and software can be used in an efficient manner.



Two view points of Operating System

User's view

1. Users want → **ease of use** and **good performance & highly convenience**
 - Don't care about **resource utilization**
2. Users can share the terminals, connected to mainframe or minicomputer.
 - Resources are shared, hence maximum resource utilization is possible.
3. Individual Users with dedicate systems such as **workstations** use shared resources from **servers or Printers**.
4. In case of hand held devices, the operating system is designed for individual usability. Example: OS on iPhone, or any Android phone.

Two view points of Operating System

System view

1. The computer system, need to consider CPU time, memory, I/O Devices.

Hence,

OS is a **resource allocator**

- ⊗ Manages all resources
- ⊗ Conflicting requests between resources are managed for efficient use
- OS is a **control program**
 - ⊗ Controls execution of programs to prevent errors and improper use of the computer

Objectives of Operating system

- Convenience
- Efficiency
- Ability to evolve
- It specifies how resources are convenient to use in an efficient manner.

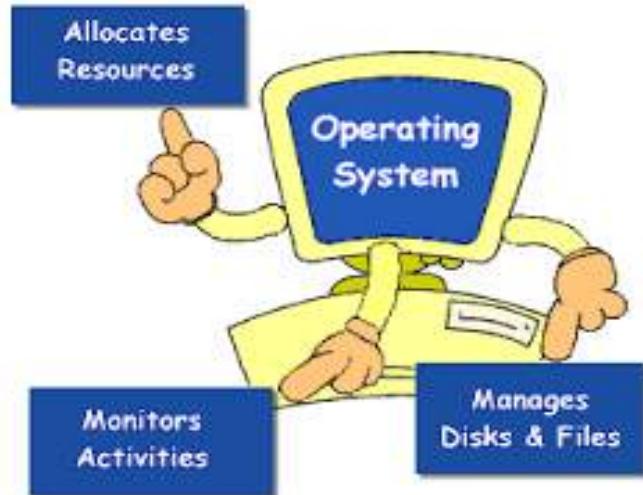
Note:

- The Operating system is like a government. It does not perform any useful function by itself.

Functions of Operating system

OS as a service routine

1. Program development.
Ex: Editors & Debuggers
1. Program Execution
2. Access I/O devices and controlling access to I/O devices
3. Error detection & response
4. Performance maintenance



Role of Operating System

The OS as a User/ Computer Interface

- Computer Hardware-Software Structure
 - Layered organization
 - OS services to users

Operating System as Software

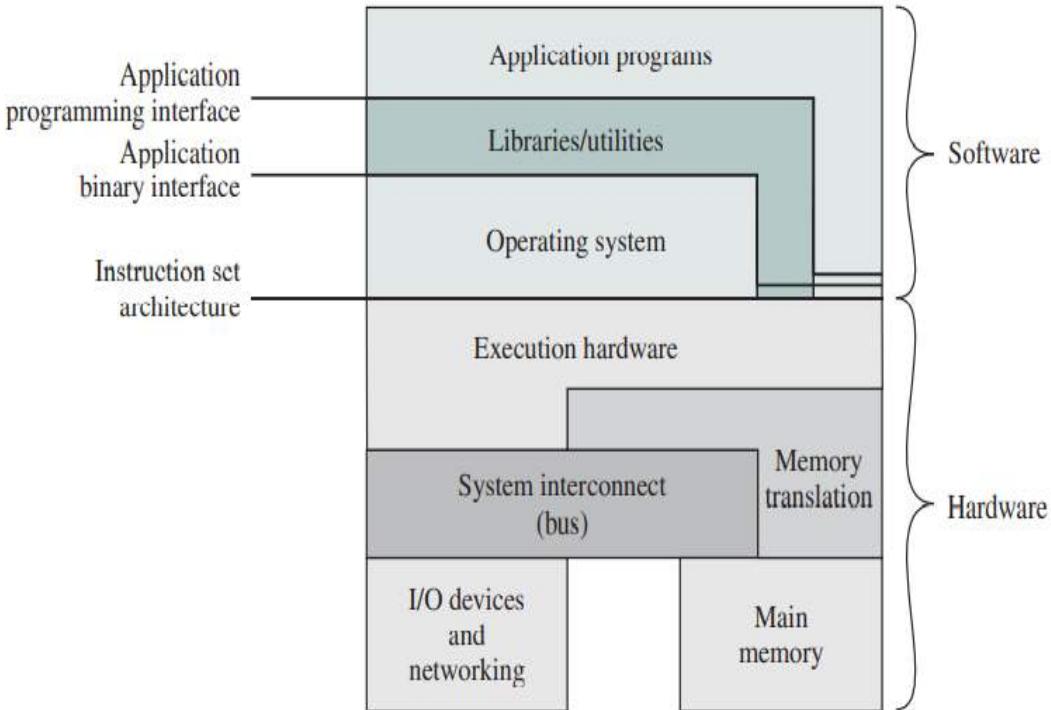
- Functions in the same way as ordinary computer software
- Program, or suite of programs, executed by the processor
- Frequently relinquishes control and must depend on the processor to allow it to regain control

The Operating System as a Resource Manager

- A computer is a set of resources for moving, storing, & processing data
- The OS is responsible for managing these resources
- The OS exercises its control through software

The OS as a User/Computer Interface

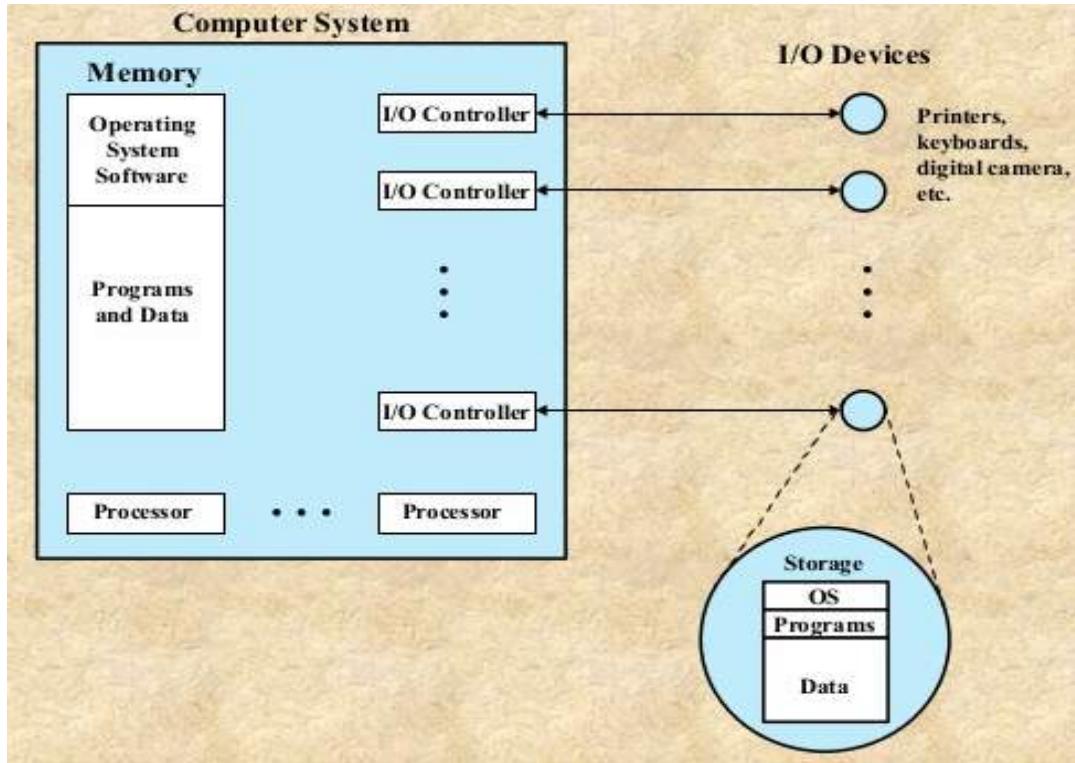
- OS provides services in following areas
 - Program development
 - Program execution
 - Access to I/O devices
 - Controlled Access to files
 - System access
 - Error detection and response
 - Accounting
 - Instruction set Architecture(ISA)
 - Application Binary Interface (ABI)
 - Application Programming Interface(API)



Computer Hardware & Software Structure

The Operating System as a Resource Manager

- OS is responsible for managing resources for movement, storage, processing and control.
- By managing OS is controller of these resources
- OS has control mechanism which is unusual in two aspects
- OS functions in same way as ordinary computer software, i.e it's a program or s/w suit of programs executed by processor
- The OS frequently relinquish the control and depends on the processor to regain the control



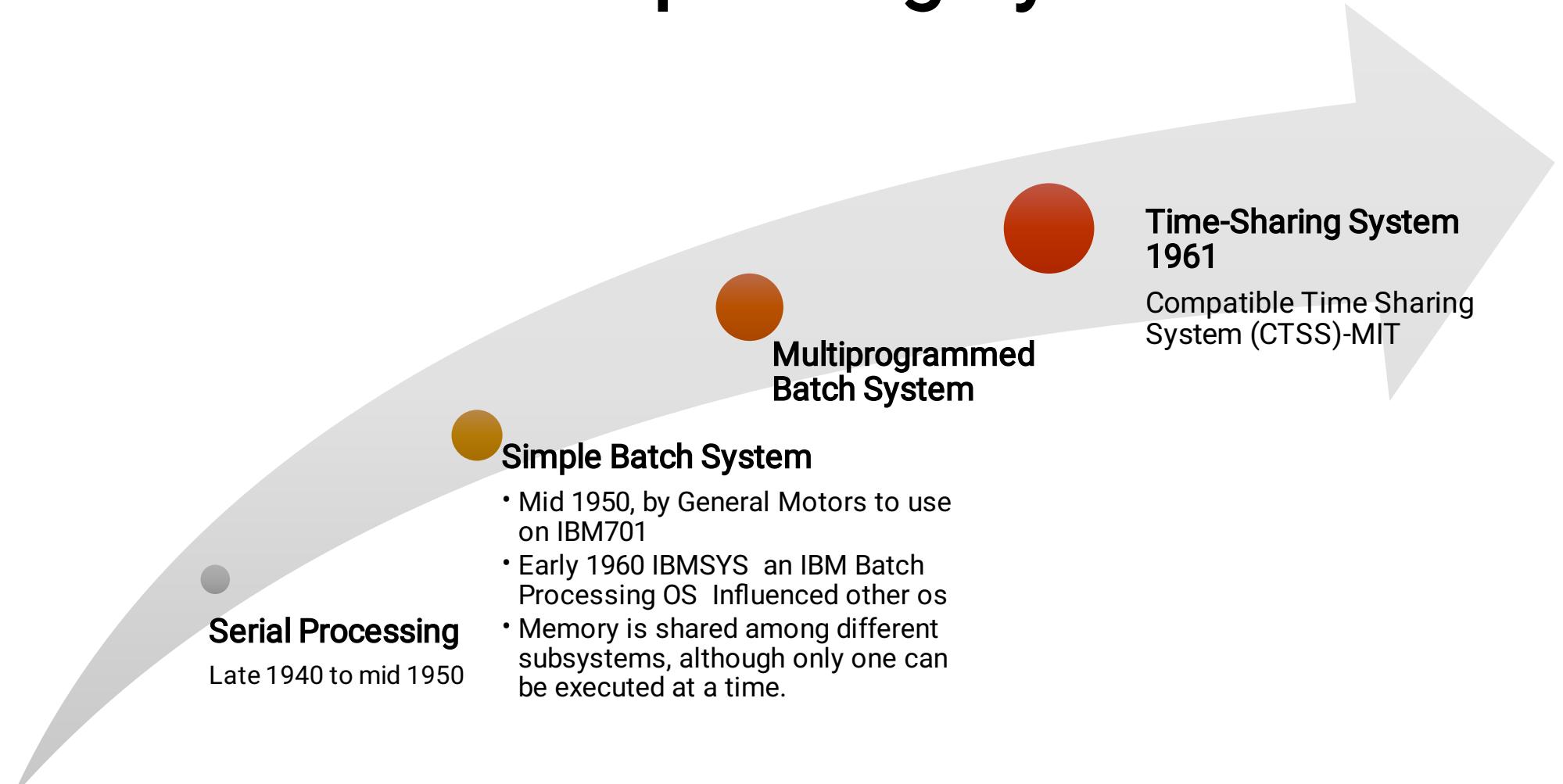
OS as Resource Manager

Figure shows main resources managed by OS

- So, how OS differs from other computer programs ??

- It directs the processor in use of other system resources'
- It makes processor to coordinate and work with other resources
- If processor wants to do any other program it ceases the execution of OS program and do
- Then OS relinquishes the control and prepares processor to do next useful task
- The (Ref Figure) part of OS remains in main memory known as **Kernal/nucleus**, which contains most frequently used programs.
- The user program and data are also available in main memory
- The memory management Hardware (in the processor) and OS jointly controls the memory allocations
- The OS decides when and I/O devices and files can be used by the user programs and controls the access accordingly.

The Evolution of Operating System



Serial Processing (Late 1940 and Mid 1950)

- No Operating System
- Programmers interacted Straight away with hardware
- Lights, toggles, input devices and printers
- Inputs are given through input devices (card readers), errors are intimated through lights and normal completion is done through printers
- These systems had two problems
 - **Scheduling**
 - Used hard copy sign-off sheets for reserving computing time (multiples of 30 minutes)
 - User block 2 slots and use only 45 min
 - User block 2 slots and unable complete within it
 - **Setup time**
 - Single program (Job) need to load compiler and source program in to the memory
 - Later save the compiled program and then loading and linking
 - All these steps cards are to be mounted and unmounted in sequence
 - Failing in any of these steps needs repetition of entire steps
- Later there were many System software developed to make this process efficient however these process are done one after another

Order over
and over again



Simple Batch Systems

- Executes jobs in batches controlled by Monitors
- User submits jobs to the operator, who make them it to batches loads the tape in to input.
- jobs branch back to monitor, which loads another batch of jobs
- Batch system can be seen in two views

Monitor Point of View	Processor Point of View
<ul style="list-style-type: none"> • Controls sequence of events • Partially Resides on main memory for execution – Called Resident Monitor • Rest is loaded as subroutine in user program • It reads the jobs at once (from card/tape)and loaded in to user program area , control passed to user job • Once completed user program transfer the control to monitor, • Monitor starts to read next batch • Results are sent to Output devices 	<ul style="list-style-type: none"> • At a point in time Current batch programs executed in main memory • On completion (sent control to monitor) brings next batch get loaded in another portion in user area • Once loaded, job gains control and execute the instructions • This continues util end of program or error • Process gains control means- processor executes the current program instructions.



Edit with WPS Office

Simple Batch System..

- Monitor performs scheduling of batches & improves job setup time

- The programs instruction are given to the monitor using special language called Job Control Language(JCL)
- Eg: Program is written in FORTRAN, the program & data is fed in a separate card, in addition to this job is described using JCL

JCL (Executed by Monitor)

- \$JOB
- \$FTN
- FORTRON Instruction
- \$LOAD
- \$RUN
- PROGRAM DATA
- \$END

JCL is Special type of programming language used to provide instructions to the monitor

what compiler to use

what data to use

Monitor Reads \$FTN and loads appropriate language compiler

Compiler translates the source/ User program into object code

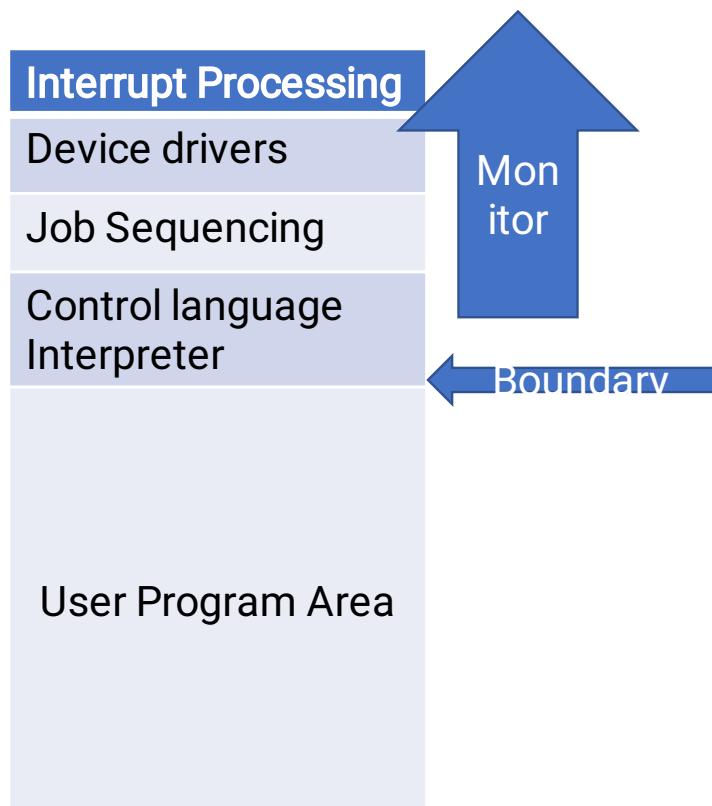
Object code is stores in Memory / Mass Storage

If stored in Mass storage Monitor executes "Compile, Load, Go" instructions, if on Memory \$LOAD instruction is required

- The Input instruction read a line of data and invoke input routine which is part of OS
- The user programs makes sure that it does not call JCL inst
- At compilation time monitor scan input line until next JCL inst to make ensure there is no too large / too low data lines



Simple Batch System..



Modes in Batch Processing

User Mode : Executes user program

Kernel mode/Monitor Mode :
Executes Privileged Instruction

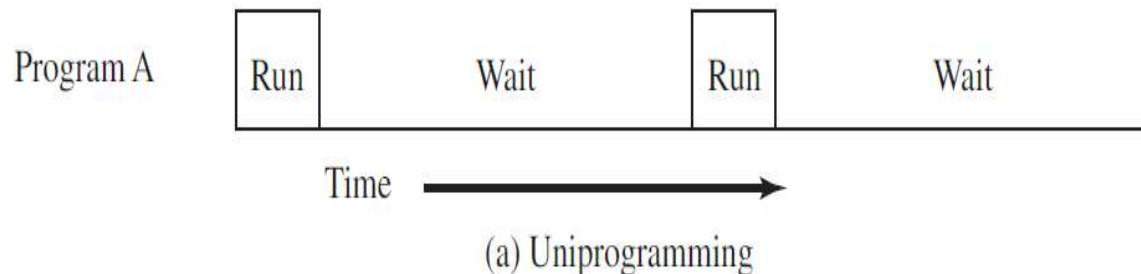


Simple Batch System..

- The hardware features by monitors (Apart from fetching inst, seize and relinquish control)
- **Memory Protection :**
 - When user program in under execution the it must not attempt to access the other users memory area, if so the processor Hardware detects is as error and transfers the control to the monitor. Monitor aborts the job with an error message and takes up a next job
- **Timer :**
 - To avoid job monopolising the processor, it sets timer at the beginning of job, if timer expires user job is stopped and control transferred to the monitor
- **Privileged Instructions:**
 - Few special/ Privileged instructions are executed under monitor mode, if such instructions are happened to attempted by user, then control is transferred to monitor. All I/O instructions are Privileged instructions, which prevents the user programs from reading the Job control instructions
- **Interrupts**
 - This facilitates the OS in relinquish control and regaining control from user programs
- **Limitations**
 - Thought it is advantageous than serial processing, it often requires same process done by the programmer in batches.

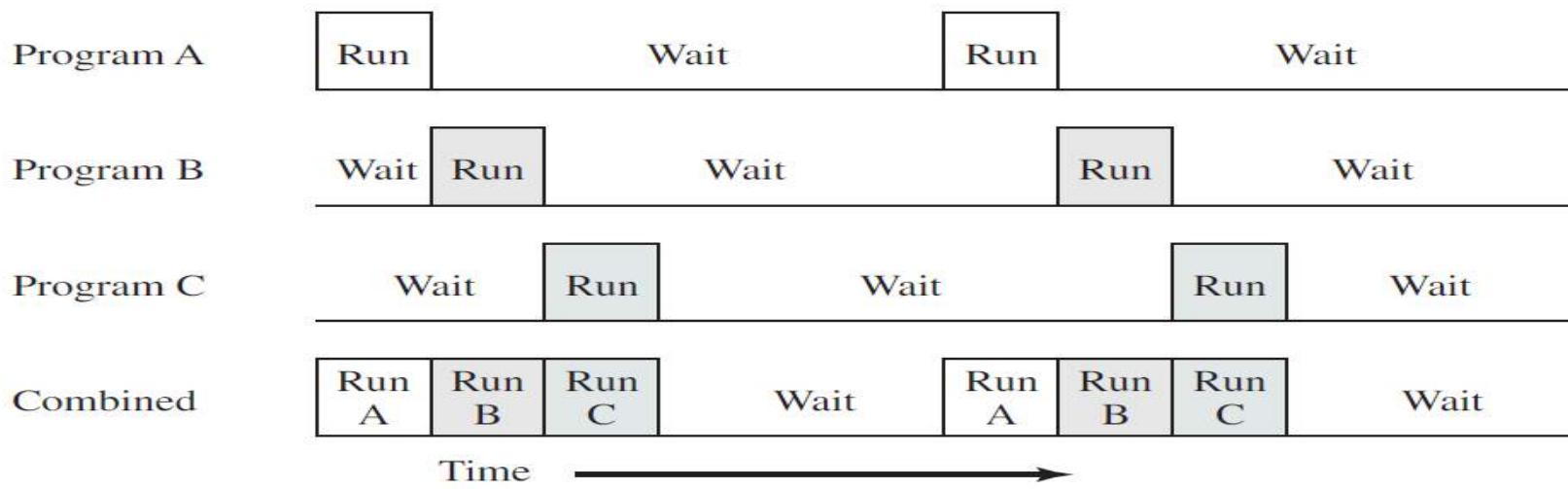
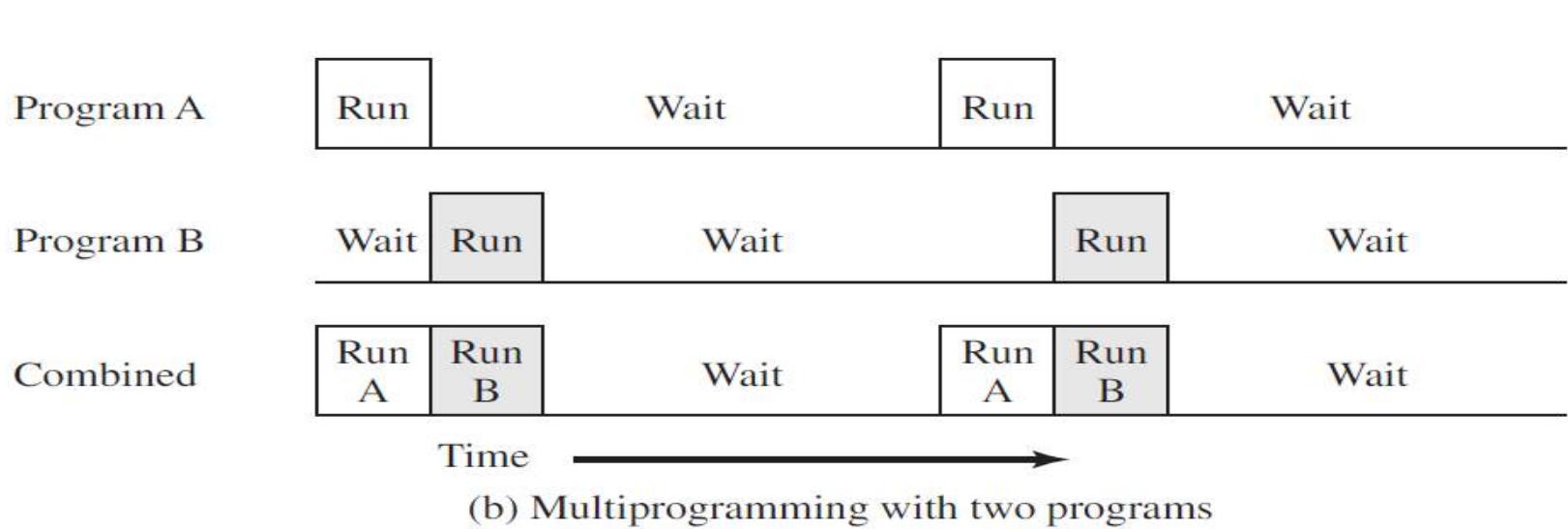
Multiprogrammed Batch Systems

- In batch processing system, memory holds resident monitor (OS) and one user program
- Batch processing hold Processor on idle state when I/O operations are being done.
- If the memory is sufficiently large to hold more than one user program then, while one program is getting I/O , processor can be switched to another program, this is called as **Multiprogramming/ Multitasking**.



Read one record from file	15 μ s
Execute 100 instructions	1 μ s
Write one record to file	15 μ s
TOTAL	31 μ s
Percent CPU Utilization	$= \frac{1}{31} = 0.032 = 3.2\%$

Figure 2.4 System Utilization Example



Time-Sharing Systems

- In 1960 programs need dedicated computers, which wasn't possible as it was too costly in those days
- Application with more user interactions, needs dedicated computers
- Multiprogramming designed to handle multiple interactive jobs referred as time slicing
- Used to share processor among multiple user, and makes each user to feel that they have dedicated system but interleaved among all of them.
- Hence if n user using the system, then each user see $1/n$ of effective computer capacity.

Table 2.3 Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

CTSS First Time sharing OS

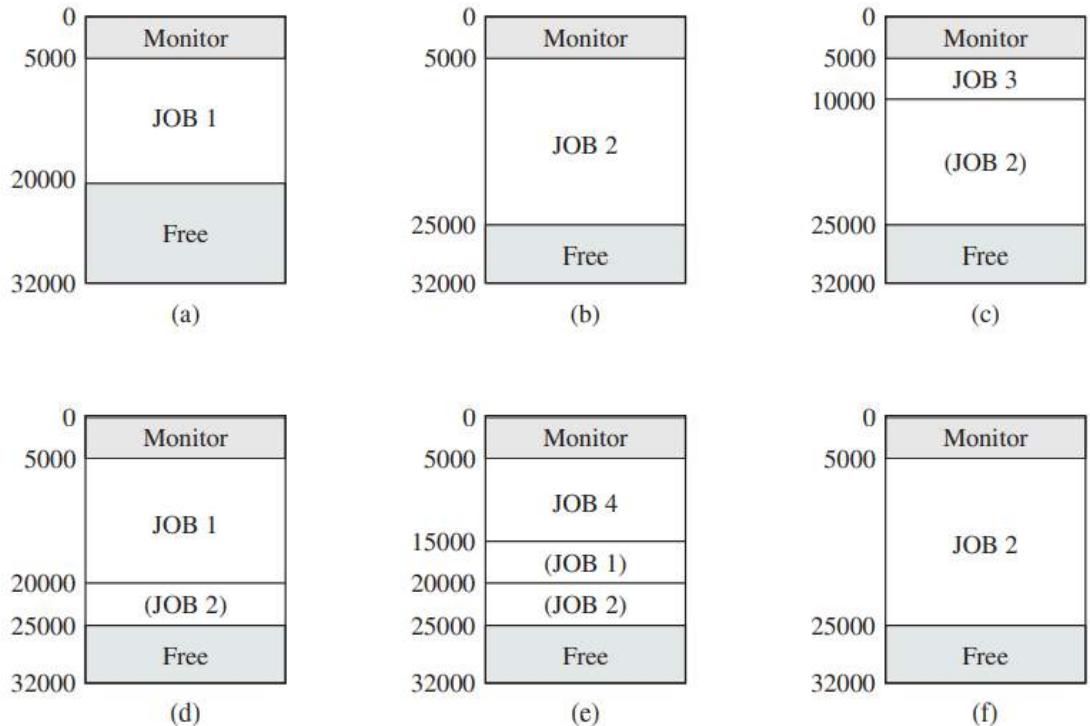


Figure 2.7 CTSS Operation

Resident Monitor: 0-5000 words
 → User program : 5000- 32000
 words timeslice : 0.2 sec

Example :

- J1->Job 1: 15000
- J2->Job 2: 20000
- J3->Job3 : 5000
- J4->Job 4: 10000

- (a) Job 1 is loaded in the memory,
- (b) Job2 is loaded since job2 and 1 can not accommodate together job 1 is flushed out
- (c) Job 3 is loaded in memory since job 2 & 3 can accommodate in memory job 2 remains there
- (d) Now processor takes job1, only portion of job2 is accommodated in memory
- (e) Now the timeslice of j1 is over processor takes J4 so portion of J1 and J2 remains in memory, Now to activate J1 or J2 only partial data is to be loaded.
- (f) JOB2 is loaded for completion.

Compatible Time-Sharing Systems

CTSS



- One of the first time-sharing operating systems
- Developed at MIT by a group known as Project MAC
- Ran on a computer with 32,000 36-bit words of main memory, with the resident monitor consuming 5000 of that To simplify both the monitor and
- memory management a program was always loaded to start at the location of the 5000th word

Time Slicing



- System clock generates interrupts at a rate of approximately one every 0.2 seconds
- At each interrupt OS regained control and could assign processor to another user
- At regular time intervals the current user would be preempted and another user loaded in
- Old user programs and data were written out to disk
- Old user program code and data were restored in main memory when that program was next given a turn



MAJOR ACHIEVEMENTS

Operating Systems are among the most complex pieces of software ever developed.

Four major theoretical advances in the development of operating systems:

- 1) Process
- 2) Memory management
- 3) Information protection and security
- 4) Scheduling and resource management

MAJOR ACHIEVEMENTS-The Process

- Central to the design of operating systems is the **concept of process**.
- Three major lines of computer system development created problems in timing and synchronization that contributed to the development of concept of the process:
 - Multiprogramming batch operation -
 - Time sharing
 - Real time transaction systems
- Four main course of error
 - 1)Improper Synchronization
 - 2) Failed mutual exclusion
 - 3) Nondeterministic program operation
 - 4) Deadlocks

Process consisting of three components:

- 1) An executable program
- 2) The associated data needed by the program
- 3) The execution context of the program

The Process...

- The execution context, or process state is the internal data by which the OS is able to supervise and control the process.
- This internal information is separated from the process, because the OS has information not permitted to the process
- Fig.1 indicates a way in which processes may be managed. Two processes, a and B, exist in portions of main memory.
- The process list contains one entry for each process, which includes a pointer to the location of the block of memory that contains the process.
- The entry may also include part or all of the execution context of the process.

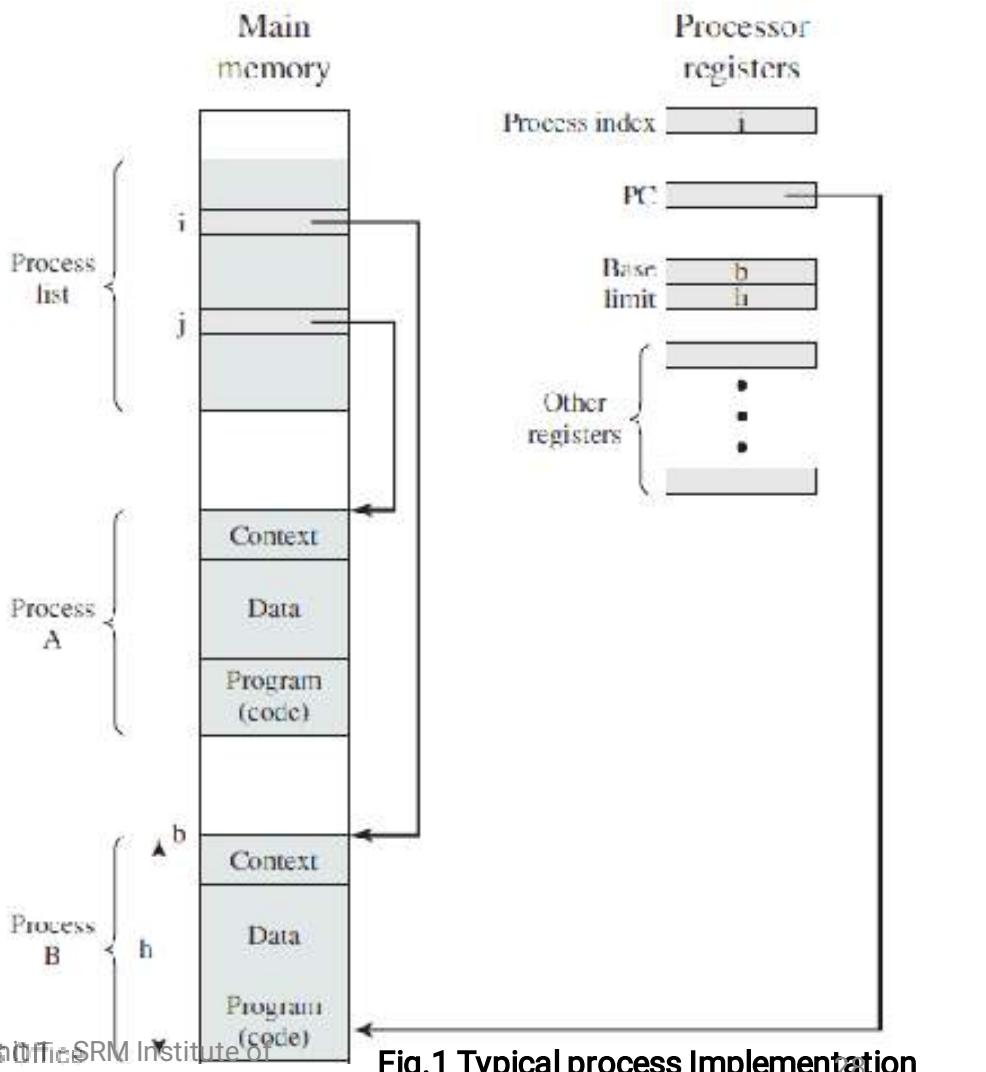
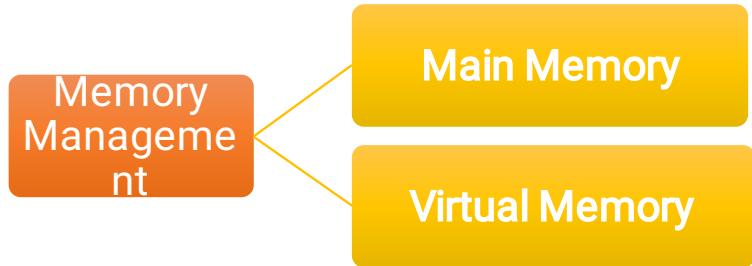


Fig.1 Typical process Implementation

The Process...

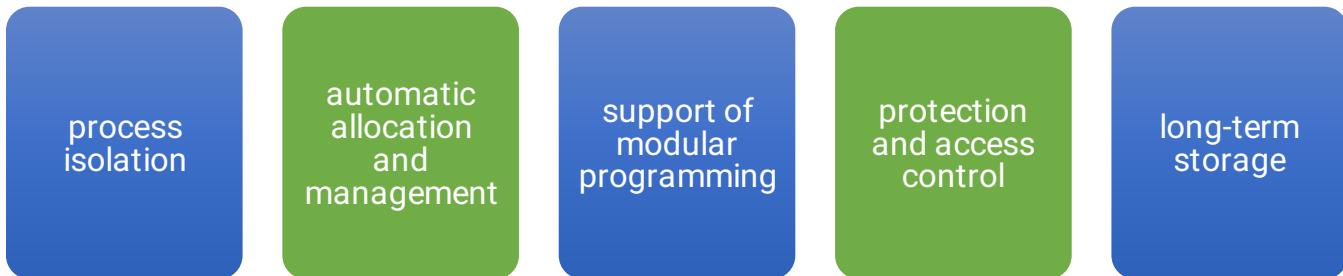
- The remainder of the execution context is stored elsewhere, perhaps with the process itself or frequently in a separate region of memory.
- The process index register contains the index into the process list of the process currently controlling the processor.
- The program counter points to the next instruction in that process to be executed.
- The base and limit registers define the region in memory occupied by the process.

MAJOR ACHIEVEMENTS - Memory Management



Main Memory

- The needs of users can be met by a computing environment that supports modular programming and the flexible use of data.
- System managers need efficient and orderly control of storage allocation.
- The OS, to satisfy these requirements, has five principal storage management responsibilities:



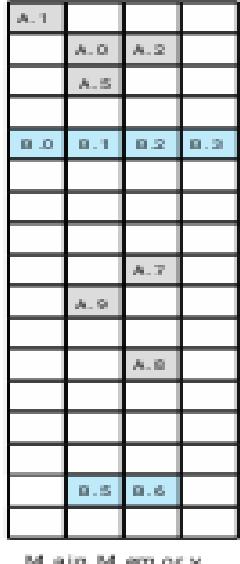
- Typically, operating system meet these requirements with virtual memory and file system facilities.

Virtual Memory

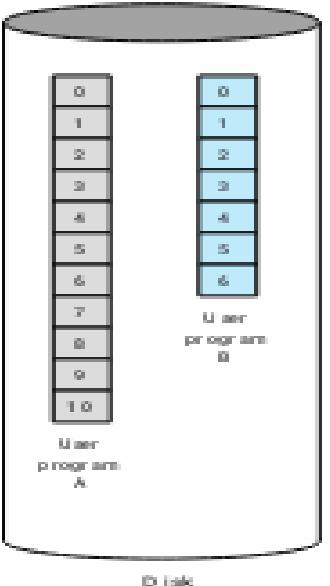
- Virtual memory is a facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available.
- Paging systems were introduced, which allow Processes to be comprised of a number of fixed size blocks, called pages.
- A program references a word by means of a virtual address consisting of a page number and an offset within the page.
- The paging system provides for a dynamic mapping between the virtual address used in the program and a real address, or physical address, in main memory. Fig 2. represents virtual memory addressing.

Memory Management...

Paging Concepts



Main memory consists of a number of fixed-length frames, each equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.



Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages for all programs plus the operating system are on disk, as are files.

Virtual memory addressing

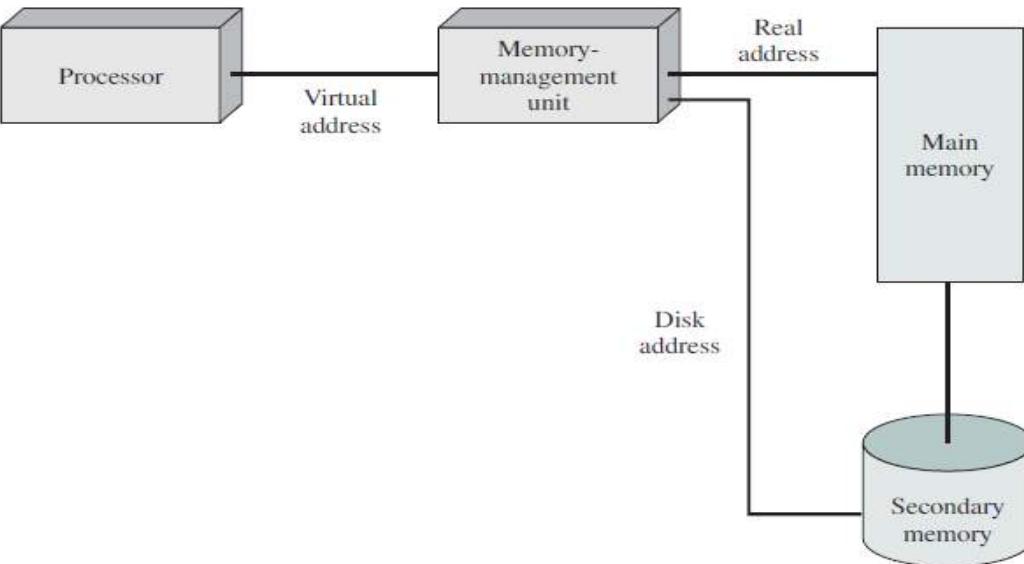


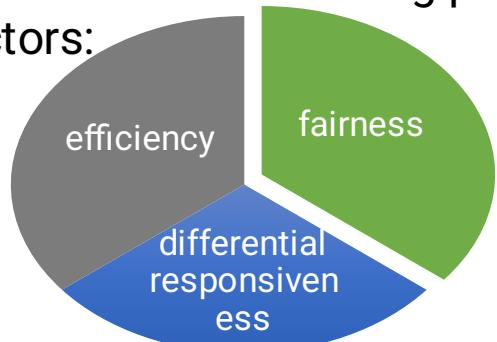
Figure 2.9: Virtual Memory Concepts

MAJOR ACHIEVEMENTS - Information Protection and Security

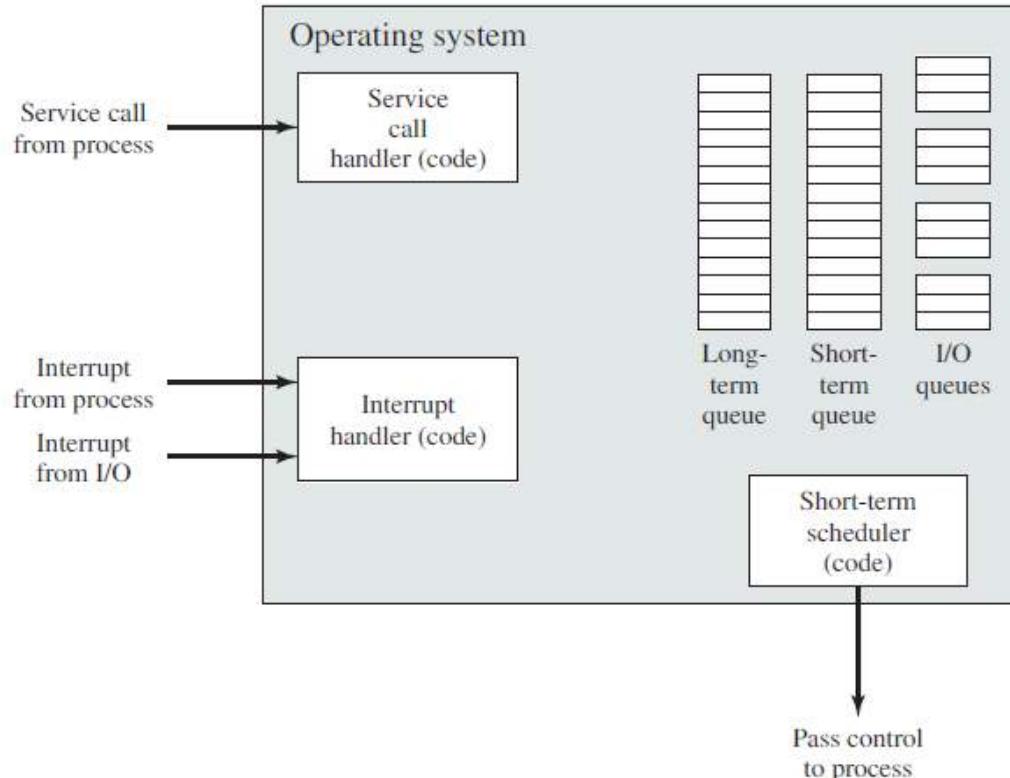
- The growth in the use of time-sharing systems and, more recently, computer networks has brought with it a growth in concern for the protection of information.
- Much of the work in security and protection as it relates to operating systems can be roughly grouped into four categories:
 - Availability
 - Confidentiality
 - Data Integrity
 - Authenticity

MAJOR ACHIEVEMENTS- *Scheduling and Resource Management*

- A key responsibility of the OS is to manage the various resources available to it (main memory space, I/O devices, processors) and to schedule their use by the various active processes.
- Any resource allocation and scheduling policy must consider three factors:



- Figure suggests the major elements of the OS involved in the scheduling of processes and the allocation of resources in a multiprogramming environment.



Key elements of an operating system for multiprogramming

Understanding the evolution of Operating systems from early batch processing systems to modern complex systems

Different approaches and design elements have been tried:

microkernel architecture

multithreading

symmetric multiprocessing

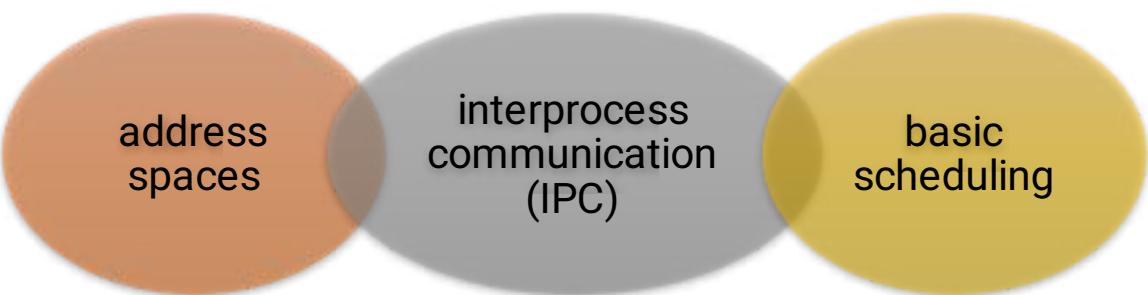
distributed operating systems

object-oriented design

Understanding the evolution of Operating systems from early batch processing systems to modern complex systems

microkernel architecture

- Assigns only a few essential functions to the kernel:



- The approach:



Understanding the evolution of Operating systems from early batch processing systems to modern complex systems

multithreading

- Technique in which a process, executing an application, is divided into threads that can run concurrently

Thread

- dispatchable unit of work
- includes a processor context and its own data area to enable subroutine branching
- executes sequentially and is interruptible

Process

- a collection of one or more threads and associated system resources
- programmer has greater control over the modularity of the application and the timing of application related events

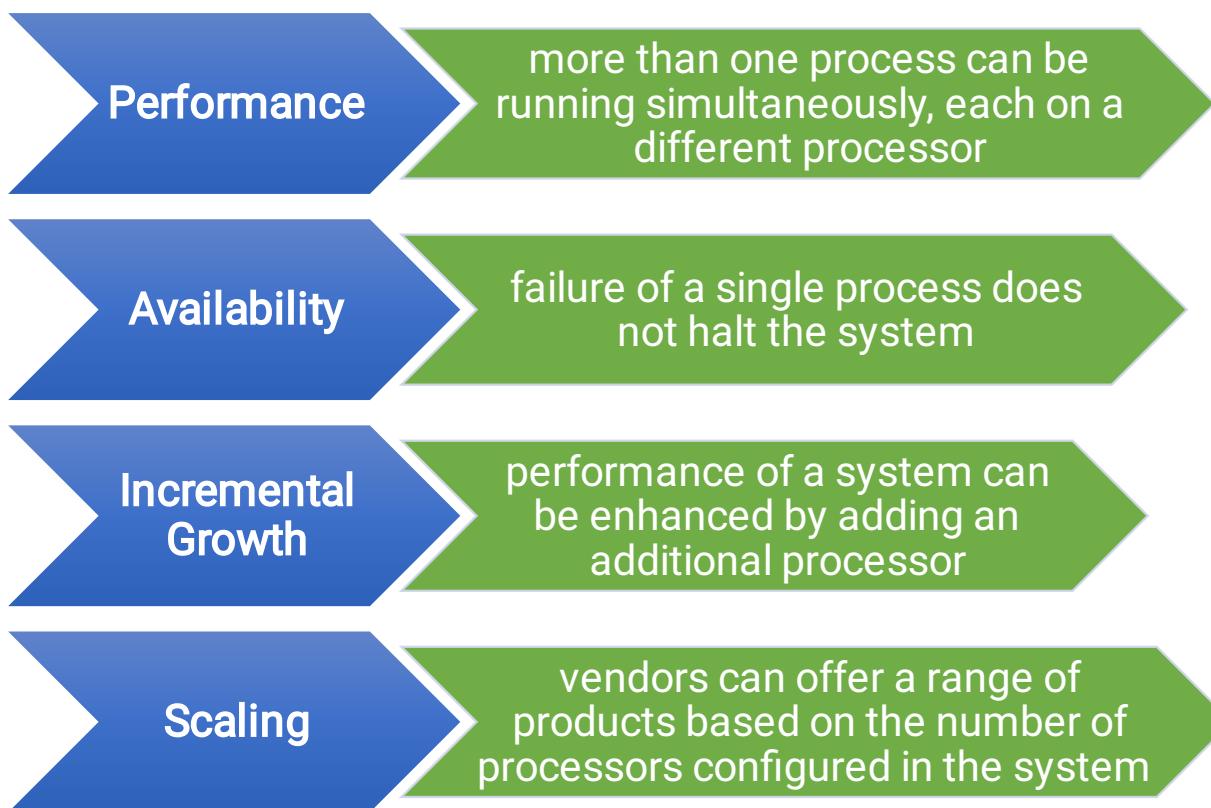
Understanding the evolution of Operating systems from early batch processing systems to modern complex systems

symmetric multiprocessing

- Term that refers to a computer hardware architecture and also to the OS behavior that exploits that architecture
- Several processes can run in parallel
- Multiple processors are transparent to the user
 - these processors share same main memory and I/O facilities
 - all processors can perform the same functions
- The OS takes care of scheduling of threads or processes on individual processors and of synchronization among processors

Understanding the evolution of Operating systems from early batch processing systems to modern complex systems

symmetric multiprocessing - Advantages



Understanding the evolution of Operating systems from early batch processing systems to modern complex systems

Multiprogramming Vs Multiprocessing

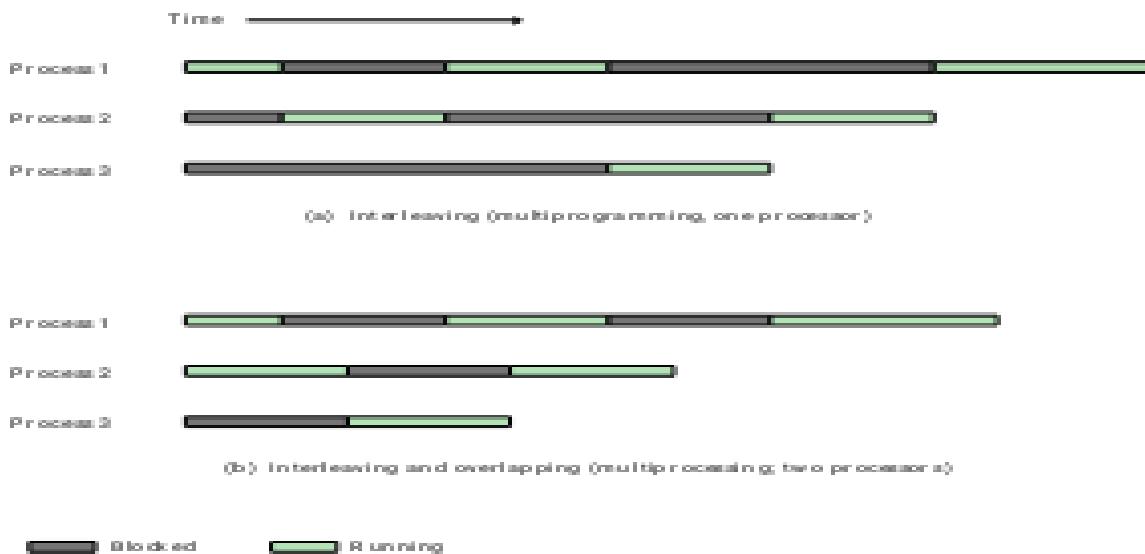


Figure 2.12 Multiprogramming and Multiprocessing

Understanding the evolution of Operating systems from early batch processing systems to modern complex systems

Distributed operating systems Vs Object-Oriented Design

Distributed operating systems

- Provides the illusion of
 - a single main memory space
 - single secondary memory space
 - unified access facilities
- State of the art for distributed operating systems lags that of uniprocessor and SMP operating systems

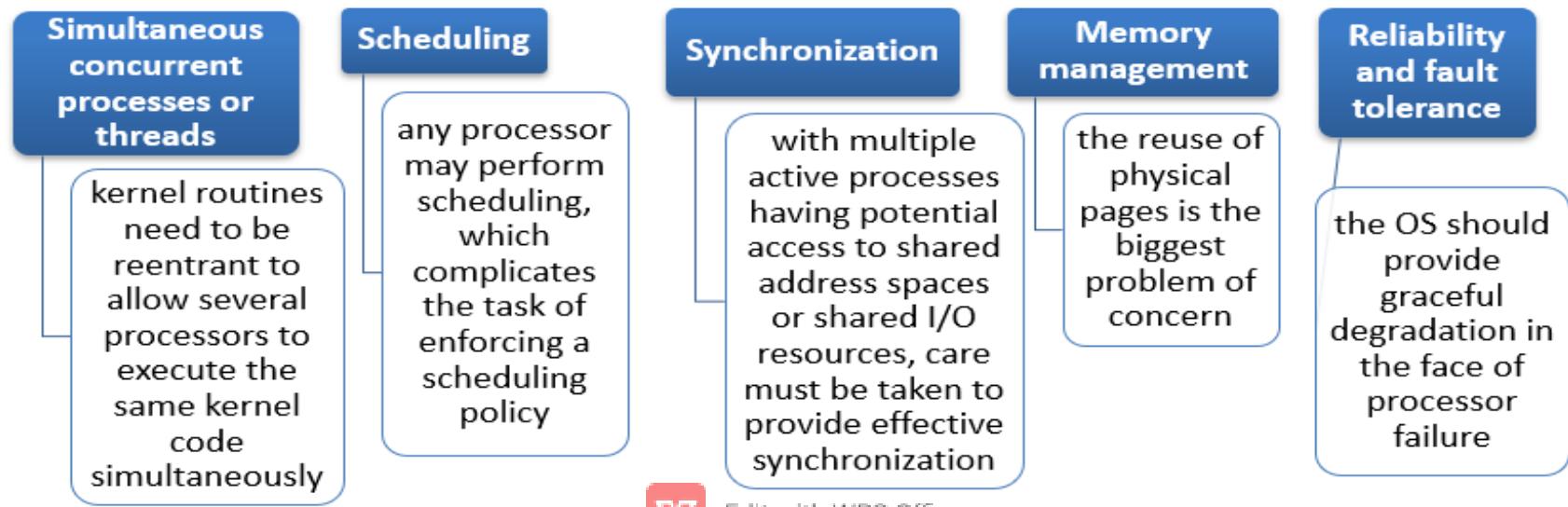
Object-Oriented Design

- Used for adding modular extensions to a small kernel
- Enables programmers to customize an operating system without disrupting system integrity
- Eases the development of distributed tools and full-blown distributed operating systems

OS Design considerations for Multiprocessor and Multicore Symmetric Multiprocessor OS Considerations

- A multiprocessor OS must provide all the functionality of a multiprogramming system plus additional features to accommodate multiple processors

Key design issues:



Edit with WPS Office

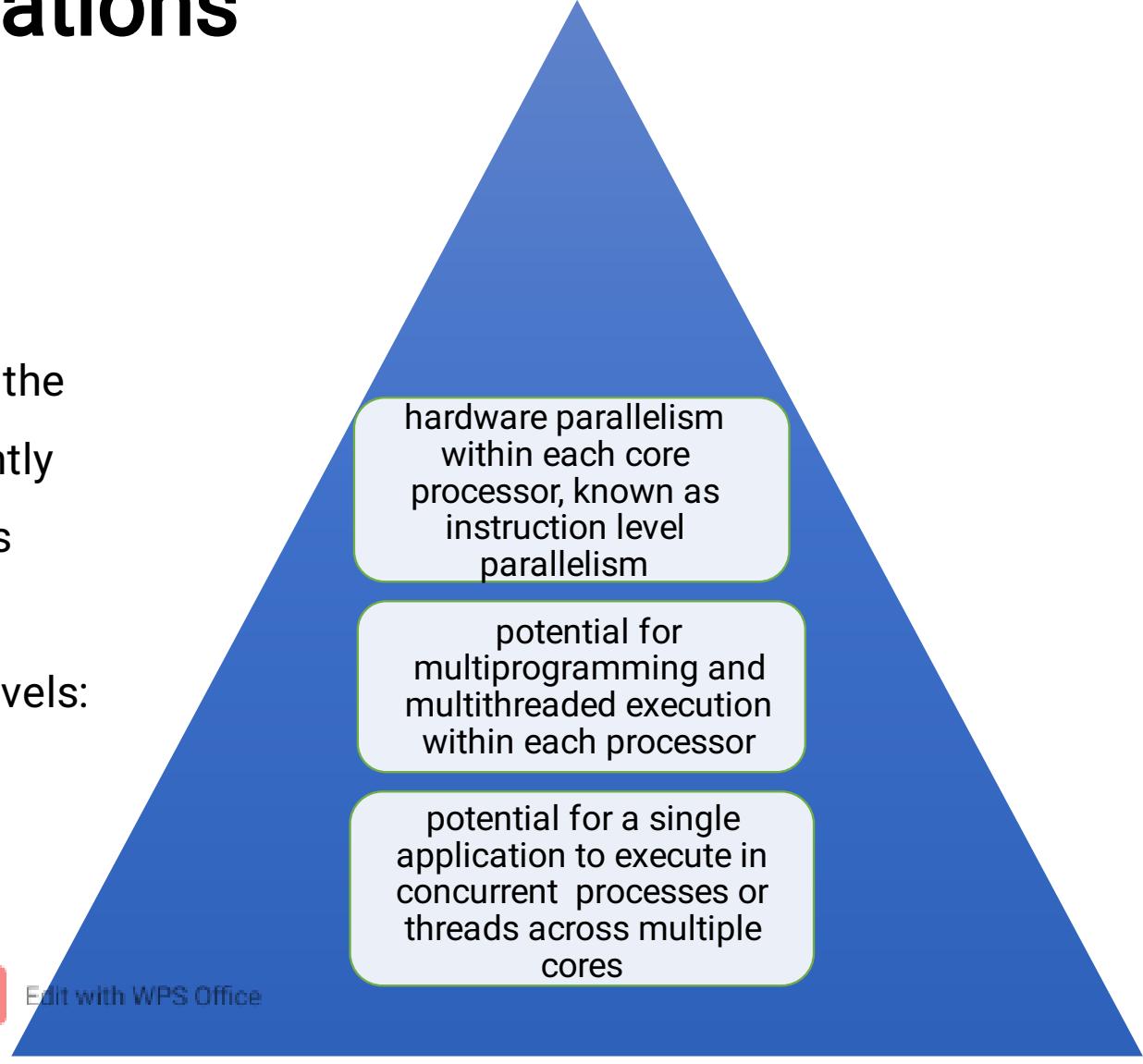
Multicore OS Considerations

Key design Issues:

- The design challenge for a many-core multicore system is to efficiently harness the multicore processing power and intelligently manage the substantial on-chip resources efficiently
- Potential for parallelism exists at three levels:



Edit with WPS Office

- 
- hardware parallelism within each core processor, known as instruction level parallelism
 - potential for multiprogramming and multithreaded execution within each processor
 - potential for a single application to execute in concurrent processes or threads across multiple cores

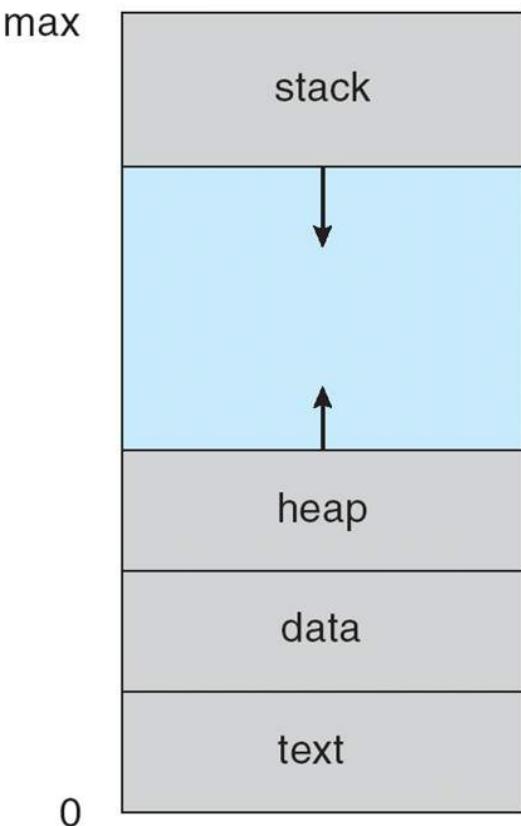
Process

- Process Concept and States
- PCB
- Process Scheduling
- Operations on Processes
- Interprocess Communication
- Examples of IPC Systems

Process Concept

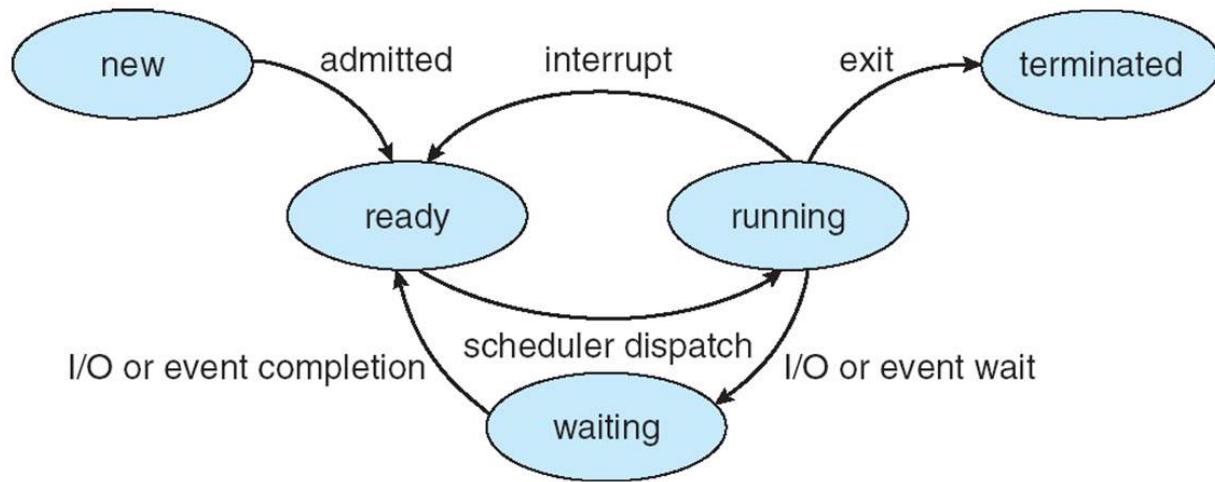
- An operating system executes a variety of programs:
 - Batch system – **jobs**
 - Time-shared systems – **user programs** or **tasks**
- Textbook uses the terms **job** and **process** almost interchangeably
- **Process** – a program in execution; process execution must progress in sequential fashion
- Multiple parts
 - The program code, also called **text section**
 - Current activity including **program counter**, processor registers
 - **Stack** containing temporary data
 - Function parameters, return addresses, local variables
 - **Data section** containing global variables
 - **Heap** containing memory dynamically allocated during run time
- Program is **passive** entity stored on disk (**executable file**), process is **active**
 - Program becomes process when executable file loaded into memory
- Execution of program started via GUI mouse clicks, command line entry of its name, etc
- One program can be several processes
 - Consider multiple users executing the same program

Process in Memory



Process State

- As a process executes, it changes **state**
 - new**: The process is being created
 - running**: Instructions are being executed
 - waiting**: The process is waiting for some event to occur
 - ready**: The process is waiting to be assigned to a processor
 - terminated**: The process has finished execution



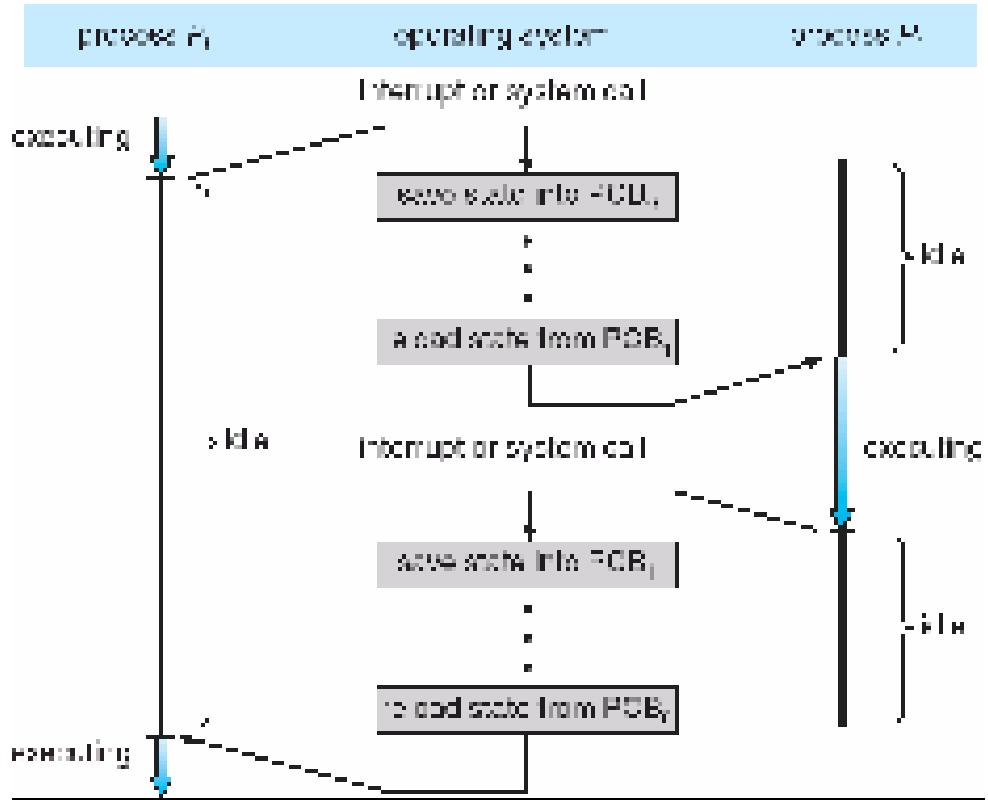
Process Control Block (PCB)

Information associated with each process
(also called **task control block**)

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files

process state
process number
program counter
registers
memory limits
list of open files
• • •

CPU Switch From Process to Process



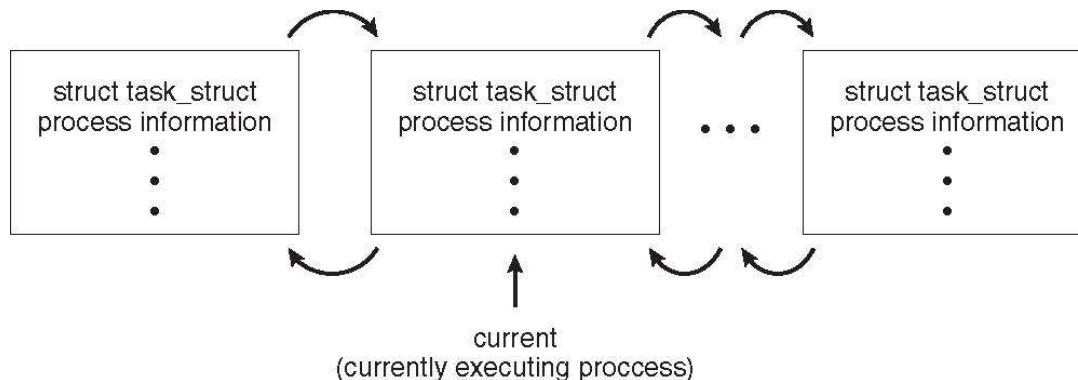
Threads

- So far, process has a single thread of execution
- Consider having multiple program counters per process
 - Multiple locations can execute at once
 - Multiple threads of control -> **threads**
- Must then have storage for thread details, multiple program counters in PCB

Process Representation in Linux

Represented by the C structure task_struct

```
pid t_pid; /* process identifier */  
long state; /* state of the process */  
unsigned int time_slice /* scheduling information */  
struct task_struct *parent; /* this process's parent */  
struct list_head children; /* this process's children */  
struct files_struct *files; /* list of open files */  
struct mm_struct *mm; /* address space of this process */
```



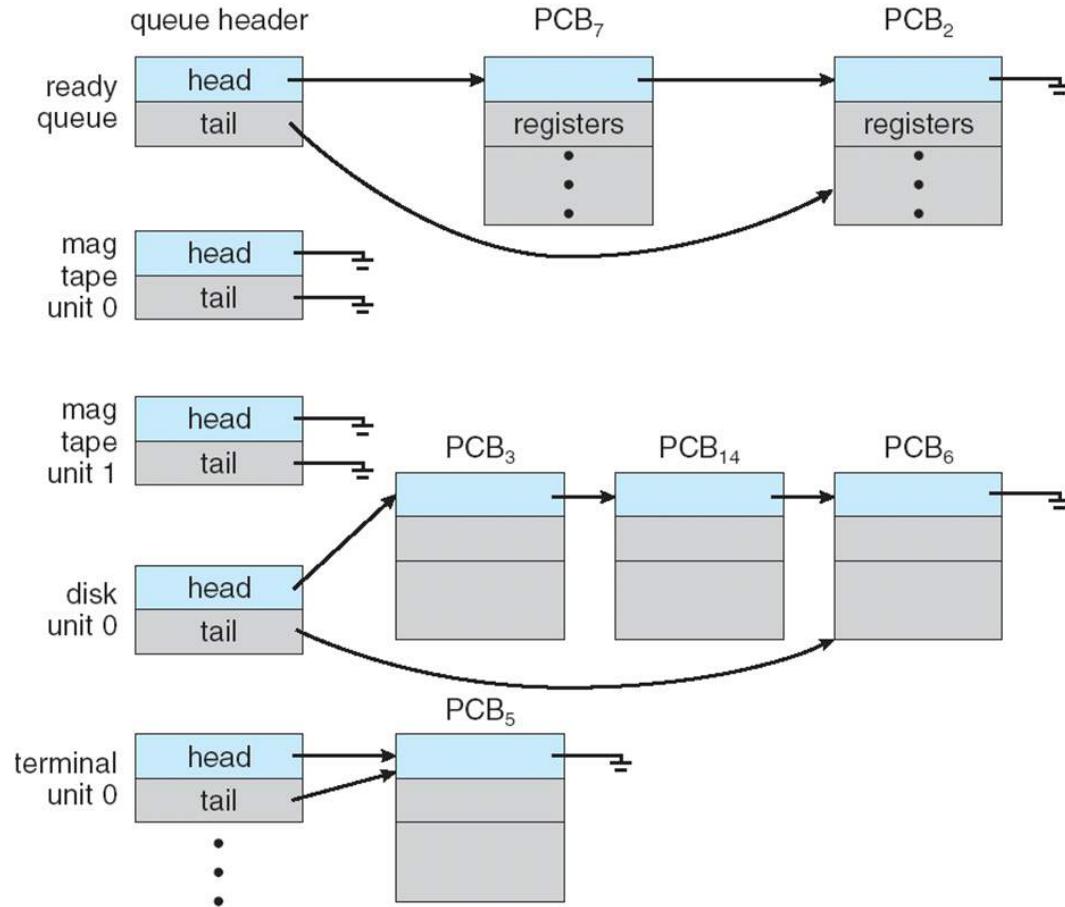
Process Scheduling

- The aim is **to assign processes to be executed by the processor or processors over time, in a way that meets system objectives,**
 - Such as **response time, throughput, and processor efficiency.**
- In many systems, this scheduling activity is **broken down into three separate functions:**
 - **long-, medium-, and short term scheduling**
- The names suggest the **relative time scales with which these functions are performed.**

Process Scheduling

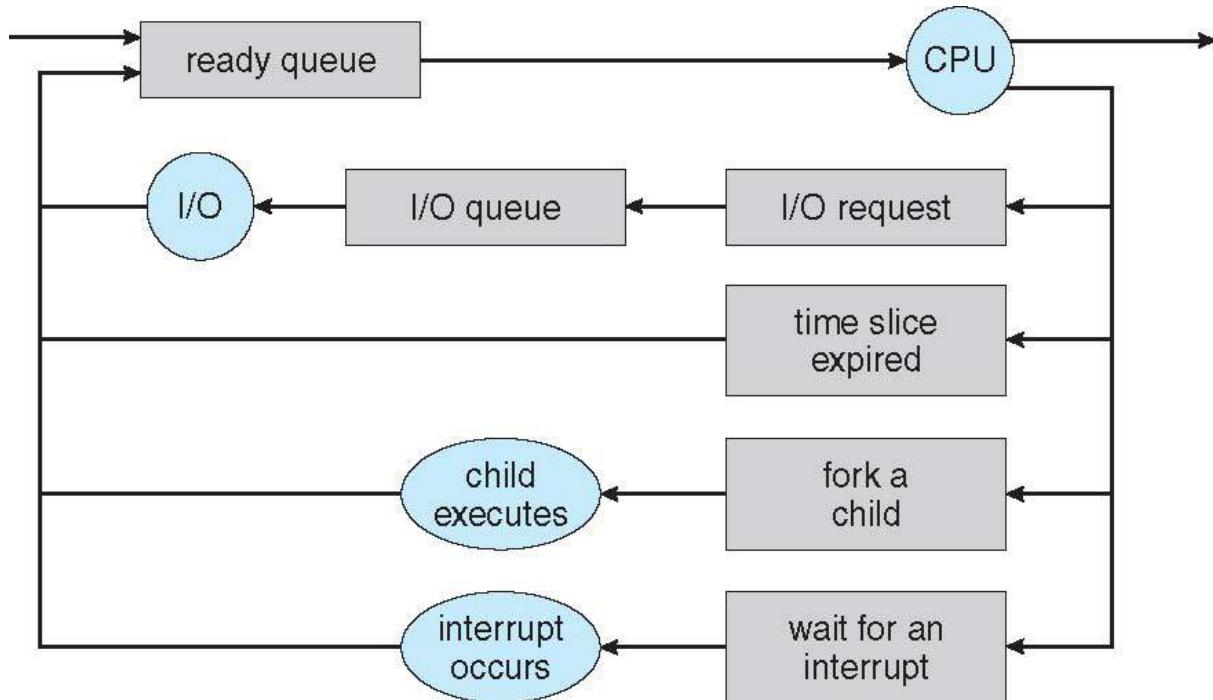
- Maximize CPU use, quickly switch processes onto CPU for time sharing
- **Process scheduler** selects among available processes for next execution on CPU
- Maintains **scheduling queues** of processes
 - **Job queue** – set of all processes in the system
 - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
 - **Device queues** – set of processes waiting for an I/O device
 - Processes migrate among the various queues

Ready Queue And Various I/O Device Queues



Representation of Process Scheduling

- **Queueing diagram** represents queues, resources, flows



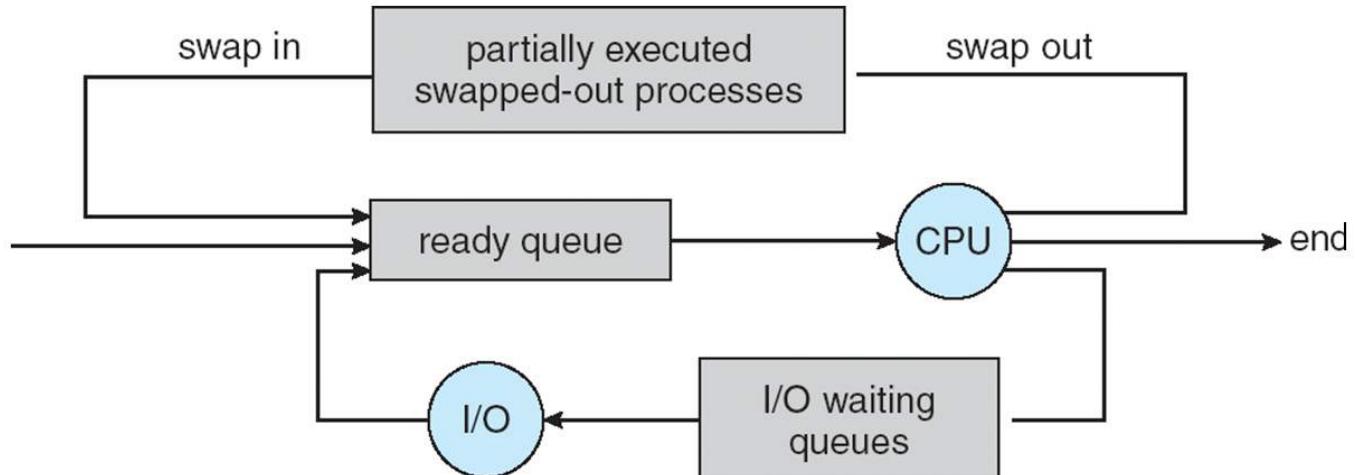
Scheduler

- A **process migrates among the various scheduling queues** throughout its lifetime.
- The operating system must select, for scheduling purposes, processes from these queues in some fashion.
- The **selection process is carried out by the appropriate scheduler.**

Schedulers

- **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
 - Sometimes the only scheduler in a system
 - Short-term scheduler is invoked frequently (milliseconds) \Rightarrow (must be fast)
- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
 - Long-term scheduler is invoked infrequently (seconds, minutes) \Rightarrow (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
- Long-term scheduler strives for good ***process mix***

- **Medium-term scheduler** can be added if degree of multiple programming needs to decrease
 - Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**
 - **A part of the swapping function .**



Context Switch

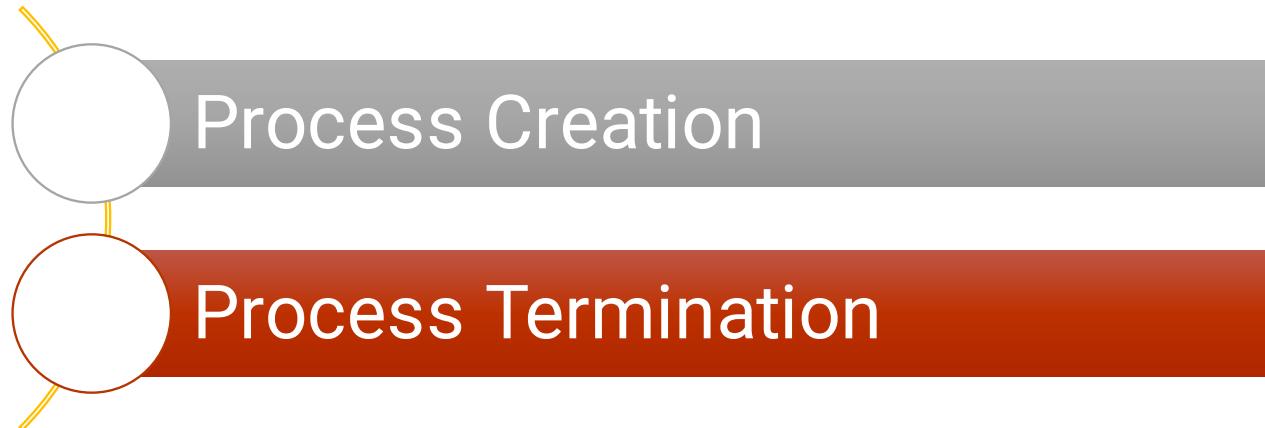
- When CPU switches to another process, the system must **save the state** of the old process and load the **saved state** for the new process via a **context switch**
- Context** of a process represented in the PCB
- Context-switch time is overhead; the system does no useful work while switching
 - The more complex the OS and the PCB → the longer the context switch
- Time dependent on hardware support
 - Some hardware provides multiple sets of registers per CPU → multiple contexts loaded at once

CONTEXT SWITCHING



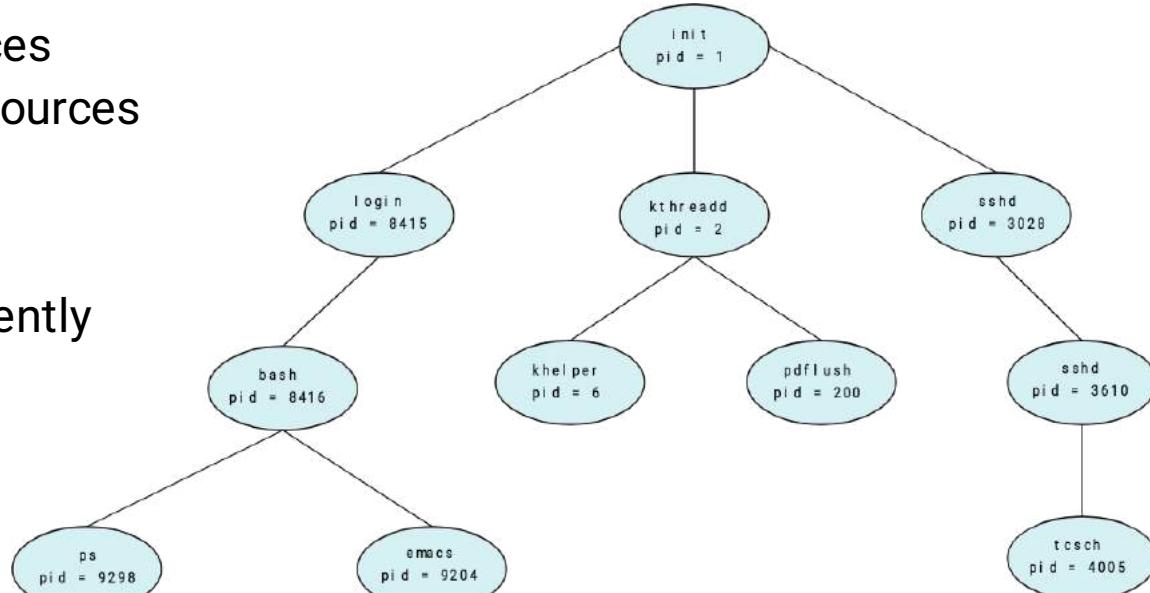
Operations on Processes

- System must provide mechanisms for:



Process Creation

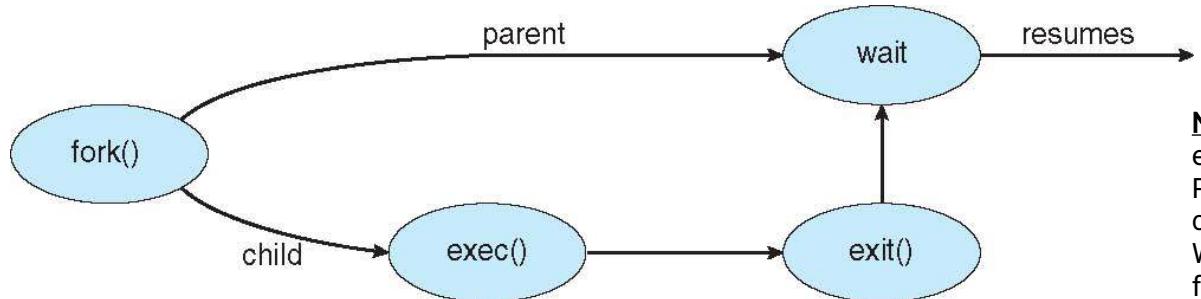
- Parent process create children processes, which, in turn create other processes, forming a tree of processes
- Generally, process identified and managed via a process identifier (pid)
- Resource sharing options
 - Parent and children share all resources
 - Children share subset of parent's resources
 - Parent and child share no resources
- Execution options
 - Parent and children execute concurrently
 - Parent waits until children terminate



Process tree in Linux

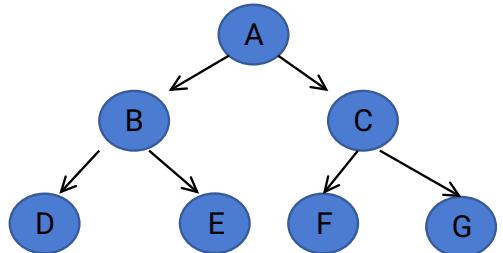
Process Creation

- Address space
 - Child duplicate of parent
 - Child has a program loaded into it
- UNIX examples
 - **fork()** system call creates new process
 - **exec()** system call used after a **fork()** to replace the process' memory space with a new program



Note:

execvp() – version of exec()
Parent waits for the child process to complete
When child completes exit(), resumes from wait().



A forks B and C
B forks D and E
C forks F and G

Process Creation

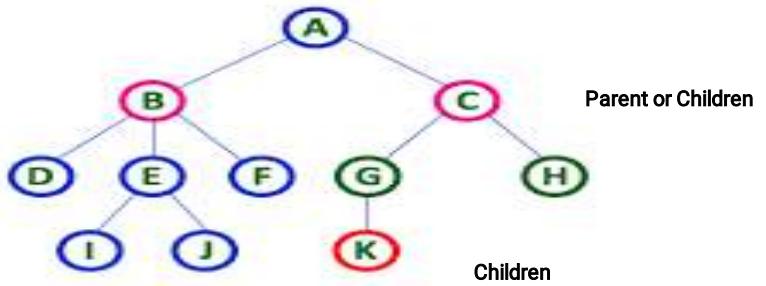
- **Steps involved in process creation :**
- (i). When a new process is created, operating system assigns a unique Process Identifier (PID) to it and inserts a new entry in primary process table.
- (ii). Then the required memory space for all the elements of process such as program, data and stack is allocated including space for its Process Control Block (PCB).
- (iii). Next, the various values in PCB are initialized such as,
 - Process identification part is filled with PID assigned to it in step (1) and also its parent's PID.
 - The processor register values are mostly filled with zeroes, except for the stack pointer and program counter. Stack pointer is filled with the address of stack allocated to it in step (ii) and program counter is filled with the address of its program entry point.
 - The process state information would be set to 'New'.
 - Priority would be lowest by default, but user can specify any priority during creation.
- In the beginning, process is not allocated to any I/O devices or files. The user has to request them or if this is a child process it may inherit some resources from its parent.
- (iv). Then the operating system will link this process to scheduling queue and the process state would be changed from 'New' to 'Ready'. Now process is competing for the CPU.
- (v). Additionally, operating system will create some other data structures such as log files or accounting files to keep track of processes activity.

[createprocess()

- Process that create new process (via)
createprocess system call

Creating process → Parent Process

The created process → child



Note:
Each process is identified by Process Identifier (**PID**)

Process Creation

- Resource Sharing(CPU time, Memory files, I/O devices)
 - Parent and children share all resources
 - Children share subset of parent resources
 - Parent & child share no resources
- Execution
 - Parent & children execute concurrently
 - **(Asynchronous Process creation)**
 - Parent waits until children terminate
 - **(Synchronous Process creation)**
- Address Space
 - Child process, copy the address space of the parent
 - Child process occupy the separate address space
- Note:
 - When one process creates a new process, the identity of the newly created process is passed to the parent.



Example

- Both Parent & Child continue creation of the new process with `fork()`.
- Return code of `fork()` → 0 (**Child Process**)
- Return code of `fork()` → Non Zero (**Parent Process**)

Creating a Separate Process using fork()

Example 1

```
#include <sys/types.h>
#include <studio.h>
#include <unistd.h>

int main()
{
    pid_t processID;
    /* create a process */
    pid = fork();
    if (pid < 0)           /* error occurred */
        fprintf(stderr, "ERROR");
    else if (pid == 0)      /* child process */
        execvp("/bin/ls", "ls", NULL);
    else                   /* parent process */
        /* parent waits for the child to complete */
        wait(NULL);
        printf("Child Complete");
}
```

Example 2

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
    pid_t pid;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        execlp("/bin/ls", "ls", NULL);
    }
    else { /* parent process */
        /* parent will wait for the child to complete */
        wait(NULL);
        printf("Child Complete");
    }
    return 0;
}
```

Process Termination

- Processes are terminated by themselves when they finish¹ executing their last statement, then operating system USES exit() system call to delete its context.
- Then all the resources held by that process like physical and virtual memory, 10 buffers, open files etc., are taken back by the operating system. A process P can be terminated either by operation system or by the parent process of P.
- A parent may terminate a process due to one of the following reasons,
- (i). When task given to the child is not required now.
- (ii). When child has taken more resources than its limit.
- (iii). The parent of the process is exiting, as a result all its children are deleted. This is called as cascaded termination.

Exit()

- Process terminates by itself (ie) when the process finish executing the final statement.
 - **System call: exit()**
 - Returns status data from child to parent (via **wait()**)
 - Process' resources are deallocated by operating system

Abort()

- The parent process terminating the child process
 - **System call: Terminate Process()**
- Parent may terminate the execution of children processes using the **abort()** system call. Some reasons for doing so:
 - Child has exceeded allocated resources
 - Task assigned to child is no longer required
 - The parent is exiting and the operating systems does not allow a child to continue if its parent terminates
- Reasons for termination
 - Child has exceeded allocated resources
 - Task assigned to child is no longer required
 - If parent is exiting, some OS doesn't allow child to process
 - **Cascading termination (2 mark Question)**
 - If the process terminates (either normally or abnormally), then all its children must be terminated.

Kill()

- By system admin for administration purpose



Process Termination

- Some operating systems do not allow child to exists if its parent has terminated. If a process terminates, then all its children must also be terminated.
 - **cascading termination.** All children, grandchildren, etc. are terminated.
 - The termination is initiated by the operating system.
- The parent process may wait for termination of a child process by using the **wait()** system call. The call returns status information and the pid of the terminated process

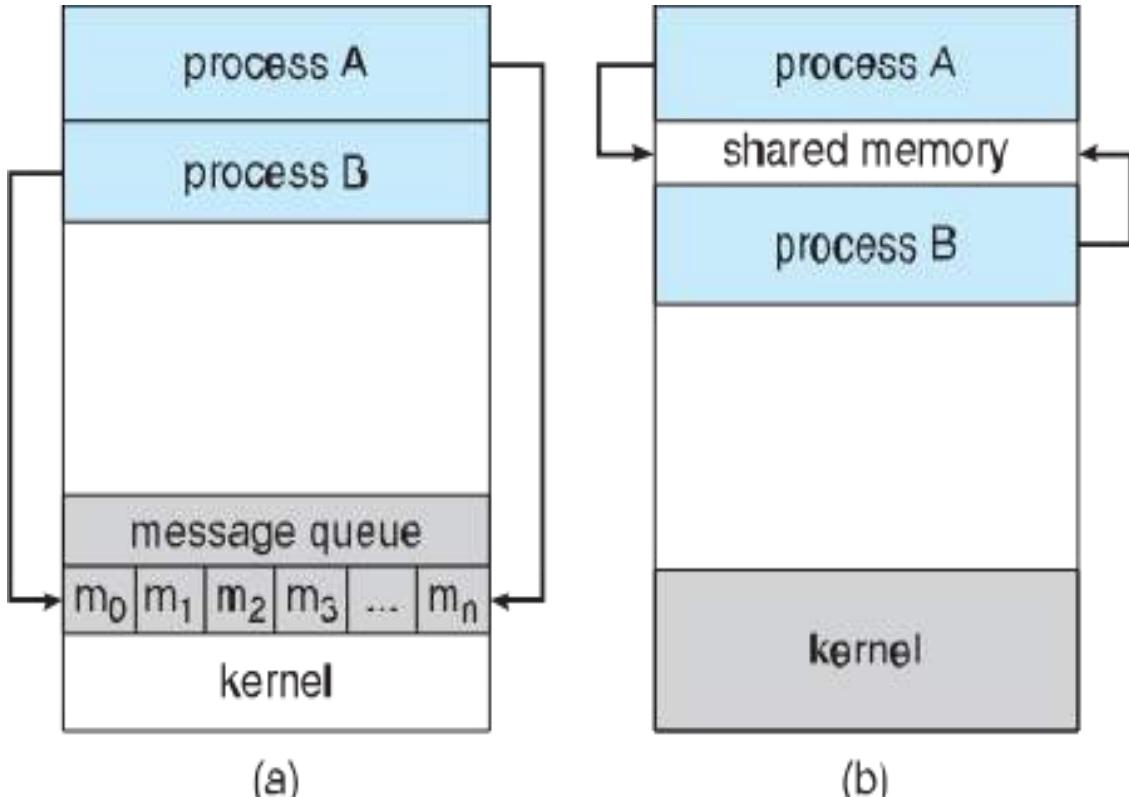
pid = wait(&status);
- If no parent waiting (did not invoke **wait()**) process is a **zombie**
- If parent terminated without invoking **wait**, process is an **orphan**

Interprocess Communication

- Processes within a system may be *independent* or *cooperating*
- Cooperating process can affect or be affected by other processes, including sharing data
- Reasons for cooperating processes:
 - Information sharing
 - Computation speedup
 - Modularity
 - Convenience
- Cooperating processes need **interprocess communication (IPC)**
- Two models of IPC
 - **Shared memory**
 - **Message passing**

Communications Models (Modes of IPC)

(a) Message passing. (b) shared memory.



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
 - Convenience

IPC Example - Producer-Consumer Problem

- Paradigm for cooperating processes, *producer* process produces information that is consumed by a *consumer* process
 - **unbounded-buffer** places no practical limit on the size of the buffer
 - **bounded-buffer** assumes that there is a fixed buffer size

Bounded-Buffer – Shared-Memory Solution

- Shared data

```
#define BUFFER_SIZE 10
```

```
typedef struct {
```

```
    ...
```

```
} item;
```

```
item buffer[BUFFER_SIZE];
```

```
int in = 0;
```

```
int out = 0;
```

- Solution is correct, but can only use BUFFER_SIZE-1 elements

Bounded-Buffer – Producer

```
item next_produced;
while (true) {
    /* produce an item in next produced */
    while (((in + 1) % BUFFER_SIZE) == out)
        ; /* do nothing */
    buffer[in] = next_produced;
    in = (in + 1) % BUFFER_SIZE;
}
```

Bounded-Buffer – Consumer

```
item next_consumed;
while (true) {
    while (in == out)
        ; /* do nothing */
    next_consumed = buffer[out];
    out = (out + 1) % BUFFER_SIZE;

    /* consume the item in next consumed */
}
```

Interprocess Communication – Shared Memory

- An area of memory shared among the processes that wish to communicate
- The communication is under the control of the users processes not the operating system.
- Major issues is to provide mechanism that will allow the user processes to synchronize their actions when they access shared memory.
- Synchronization is discussed in great details in Chapter 5.

Interprocess Communication – Message Passing

- Mechanism for processes to communicate and to synchronize their actions
- Message system – processes communicate with each other without resorting to shared variables
- IPC facility provides two operations:
 - send(*message*)**
 - receive(*message*)**
- The *message* size is either fixed or variable

Message Passing (Cont.)

- If processes P and Q wish to communicate, they need to:
 - Establish a ***communication link*** between them
 - Exchange messages via send/receive
- Implementation issues:
 - How are links established?
 - Can a link be associated with more than two processes?
 - How many links can there be between every pair of communicating processes?
 - What is the capacity of a link?
 - Is the size of a message that the link can accommodate fixed or variable?
 - Is a link unidirectional or bi-directional?

Message Passing (Cont.)

- Implementation of communication link
 - Physical:
 - Shared memory
 - Hardware bus
 - Network
 - Logical:
 - Direct or indirect
 - Synchronous or asynchronous
 - Automatic or explicit buffering

Naming: Direct Communication

- 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

Naming: 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 (port)
 - 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

Naming: Indirect Communication

- Mailbox sharing
 - $P_1, P_2,$ and P_3 share mailbox A
 - P_1 , sends; P_2 and P_3 receive
 - Who gets the message?
- Solutions
 - Allow a link to be associated with at most two processes
 - Allow only one process at a time to execute a receive operation
 - Allow the system to select arbitrarily the receiver. Sender is notified who the receiver was.

Synchronization

- Message passing may be either blocking or non-blocking
- **Blocking** is considered **synchronous**
 - **Blocking send** -- the sender is blocked until the message is received
 - **Blocking receive** -- the receiver is blocked until a message is available
- **Non-blocking** is considered **asynchronous**
 - **Non-blocking send** -- the sender sends the message and continue
 - **Non-blocking receive** -- the receiver receives:
 - A valid message, or
 - Null message
- Different combinations possible
 - If both send and receive are blocking, we have a **rendezvous**

Synchronization (Contd)

Producer-consumer becomes trivial

```
message next_produced;
while (true) {
    /* produce an item in next produced */
    send(next_produced);
}
```

```
message next_consumed;
while (true) {
    receive(next_consumed);

    /* consume the item in next consumed */
}
```

Buffering

- Queue of messages attached to the link.

Implemented in one of three ways

1. Zero capacity – no messages are queued on a link.

Sender must wait for receiver (rendezvous)

2. Bounded capacity – finite length of n messages

Sender must wait if link full

3. Unbounded capacity – infinite length

Sender never waits



Pipes

- Act as a channel allowing **two processes to communicate**
- One of the first IPC mechanisms in early UNIX systems
- Typically provide **one of the simpler ways for processes to communicate with one another**,
 - Communication medium between **two or more related or interrelated processes**.
 - Either **within one process** or a communication between **the child and the parent processes**.
 - Communication can also be **multi-level** such as communication between
 - **The parent,**
 - **The child and**
 - **The grand-child, etc.**
 - Communication is achieved by **one process writing into the pipe and other reading from the pipe**.

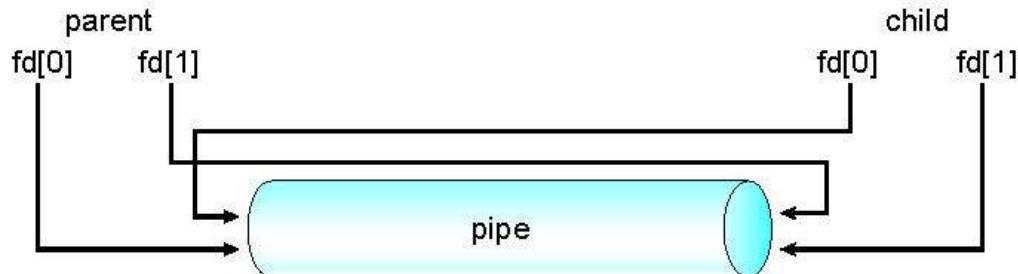


Pipes

- In implementing a pipe, four issues must be considered:
 - Is communication unidirectional or bidirectional?
 - In the case of two-way communication, is it half or full-duplex?
 - Must there exist a relationship (i.e., ***parent-child***) between the communicating processes?
 - Can the pipes be used over a network?
- Ordinary pipes – cannot be accessed from outside the process that created it. Typically, a parent process creates a pipe and uses it to communicate with a child process that it created.
- Named pipes – can be accessed without a parent-child relationship.

Ordinary Pipes

- Ordinary Pipes allow communication in standard producer-consumer style
- Producer writes to one end (the **write-end** of the pipe)
- Consumer reads from the other end (the **read-end** of the pipe)
- Ordinary pipes are therefore unidirectional
- Require parent-child relationship between communicating processes



- Windows calls these **anonymous pipes**
- See Unix and Windows code samples in textbook

Ordinary Pipes

```
#include<unistd.h>

int pipe(int pipedes[2]);
```

```
int pipe(int fds[2]);
```

Parameters :

fd[0] will be the fd(file descriptor) for the read end of pipe.

fd[1] will be the fd for the write end of pipe.

Returns : 0 on Success.

-1 on error.

- This system call would create a pipe for one-way communication
- It creates two descriptors,
 - **First one is connected to read from the pipe and**
 - **Other one is connected to write into the pipe**
- Descriptor **pipedes[0]** is for reading and **pipedes[1]** is for writing.
- Whatever is written into pipedes[1] can be read from pipedes[0].

This call would **return zero on success** and **-1 in case of failure**.

Cannot be accessed from outside the process that creates it.

A parent process **creates a pipe and uses it to communicate with a child process** it creates via fork().

The **child process inherits open files from its parent**.

A **pipe is a special type of file**, the **child inherits the pipe from its parent process**.

Named Pipes

- Named Pipes are more powerful than ordinary pipes
- Communication is bidirectional
- No parent-child relationship is necessary between the communicating processes
- Several processes can use the named pipe for communication
- Provided on both UNIX and Windows systems

Need for IPC

- Processes can execute concurrently
 - May be interrupted at any time, partially completing execution
- Concurrent access to shared data may result in data inconsistency
- Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes
- Illustration of the problem:
Suppose that we wanted to provide a solution to the consumer-producer problem that fills ~~all~~ the buffers. We can do so by having an integer **counter** that keeps track of the number of full buffers. Initially, **counter** is set to 0. It is incremented by the producer after it produces a new buffer and is decremented by the consumer after it consumes a buffer.

Process Synchronization

- Process Synchronization means sharing system resources by processes in a such a way that, Concurrent access to shared data is handled thereby minimizing the chance of inconsistent data.
- Maintaining data consistency demands mechanisms to ensure synchronized execution of cooperating processes.
- Process Synchronization was introduced to handle problems that arose while multiple process executions
 - **Background**
 - Processes can execute concurrently
 - May be interrupted at any time, partially completing execution
 - Concurrent access to shared data may result in data inconsistency
 - Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes
 - Illustration of the problem:
Suppose that we wanted to provide a solution to the consumer-producer problem that fills **all** the buffers. We can do so by having an integer **counter** that keeps track of the number of full buffers. Initially, **counter** is set to 0. It is incremented by the producer after it produces a new buffer and is decremented by the consumer after it consumes a buffer.

Need for Process Synchronization & Race Condition

- **Race condition:** The situation where several processes access – and manipulate shared data concurrently.
- The final value of the shared data depends upon which process finishes last.
- To prevent race conditions, concurrent processes must be **synchronized**.

Process Synchronization - Example

Producer

```
while (true) {
    /* produce an item in next produced */

    while (counter == BUFFER_SIZE) ;
        /* do nothing */
    buffer[in] = next_produced;
    in = (in + 1) % BUFFER_SIZE;
    counter++;
}
```

Consumer

```
while (true) {
    while (counter == 0)
        ; /* do nothing */
    next_consumed = buffer[out];
    out = (out + 1) % BUFFER_SIZE;
    counter--;
    /* consume the item in next consumed */
}
```

Race Condition

- **counter++** could be implemented as

```
register1 = counter
register1 = register1 + 1
counter = register1
```

- **counter--** could be implemented as

```
register2 = counter
register2 = register2 - 1
counter = register2
```

- Consider this execution interleaving with “count = 5” initially:

S0: producer execute register1 = counter	{register1 = 5}
S1: producer execute register1 = register1 + 1	{register1 = 6}
S2: consumer execute register2 = counter	{register2 = 5}
S3: consumer execute register2 = register2 - 1	{register2 = 4}
S4: producer execute counter = register1	{counter = 6 }
S5: consumer execute counter = register2	{counter = 4}

Critical Section Problem

- Consider system of n processes $\{p_0, p_1, \dots p_{n-1}\}$
- Each process has **critical section** segment of code
 - Process may be changing common variables, updating table, writing file, etc
 - When one process in critical section, no other may be in its critical section
- ***Critical section problem*** is to design protocol to solve this
- Each process must ask permission to enter critical section in **entry section**, may follow critical section with **exit section**, then **remainder section**

Critical Section

- General structure

```
do {  
    entry section  
    critical section  
    exit section  
    remainder section  
} while (true);
```

Algorithm for Process P_i

```
do {
    while (turn == j);
    critical section
    turn = j;
    remainder section
} while (true);
```

Solution to Critical-Section Problem

1. **Mutual Exclusion** - If process P_i 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
 - ☒ Assume that each process executes at a nonzero speed
 - ☒ No assumption concerning **relative speed** of the n processes

18CSC205J

OPERATING SYSTEMS UNIT – II

Course Learning Rationale (CLR):

CLR-2 : Insist the Process Management functions of an Operating system

Course Learning Outcomes (CLO):

CLO-2 : Know the Process management functions of an Operating system

TOPICS COVERED

- **PROCESS SYNCHRONIZATION :**

Peterson's solution, Synchronization Hardware, Semaphores, usage, implementation, Classical Problems of synchronization – Readers writers problem, Bounded Buffer Problem, Dining Philosophers problem (Monitor)

- **CPU SCHEDULING :**

FCFS, SJF, Priority scheduling, Round robin, Multilevel queue Scheduling, Multilevel feedback Scheduling.

- **REAL TIME SCHEDULING:**

Rate Monotonic Scheduling and Deadline Scheduling

- **DEADLOCKS:**

Necessary conditions, Resource allocation graph, Deadlock prevention methods, Deadlock Avoidance, Detection and Recovery



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

Process synchronization

What is Process Synchronization ?

- **Process Synchronization** is the process of coordinating the execution of processes in such a way that no two processes can have access to the same shared data and resources.

What is Critical section ?

- The portion in any program which accesses a shared resource (such as a shared variable in the memory) is called as critical section or Critical region.

Types of solutions to CS problem

- Software solution
 - Peterson's solution
- Hardware solutions
 - Synchronization Hardware – TSL Instruction
 - Compare and swap Instruction
- MUTEX Locks (Spin lock- Software Tool)
- Programming language construct
 - Semaphores

Software Solution to CS problem

Peterson's Solution (overview)

Helps to solve the critical section problem – applicable for 2 processes.
(Scenarios - Classical problems of synchronization – Bounded Buffer, Producer-consumer, Dining philosophers)

Peterson's solution

- The algorithm deals with 2 variables
- Turn and flag

Using these 2 variables , the critical section problem is addressed by Peterson.



Contd..

Two variables are used:

- **int turn**

Turn \square whose turn to enter critical section

- **boolean flag[2]**

Flag \square to indicate if the process is ready to enter critical section

flag[i] == true implies that process P_i is ready to enter its critical section



Peterson's Algorithm

```
do {  
    flag[i] = true;  
    turn = j;  
    while (flag[j] && turn == j);  
        critical section  
        flag[i] = false;  
    remainder section  
} while (true);
```

Explanation :

Peterson's Algorithm is used to synchronize two processes.

- In this algorithm , the variable i can be the (Process i) Producer and j can be Consumer (Process j).
- Initially the flags are false.
- When a particular process wants to enter its critical section, **it sets it's flag to true and turn as the index of the other process**. This means that the process wants to execute but it will allow the other process to run first.
- **The process performs busy waiting** until the other process has finished it's own critical section.



Peterson's sol. Contd..

The solution addresses all the 3 conditions required for solving a Critical section problem.

- (i) Mutual exclusion
- (ii) Progress
- (iii) Bounded waiting

Limitations of Peterson's sol

- Applicable only between processes
- Busy waiting.

Hardware solution - Overview

- Many systems provide hardware support for implementing the critical section code.
- All solutions below based on idea of **locking**
 - Protecting critical regions via locks
- 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-interruptible
 - Either test memory word and set value
 - Or swap contents of two memory words

Hardware sol..

- Many modern computer systems therefore provide special hardware instructions that allow us either to test and modify the content of a word or to swap the contents of two words atomically— **that is, as one uninterrupted unit (TSL/Compare and Swap Instruction).**



Hardware solutions to CS problem

- (i) TSL
- (ii) Compare and swap

General Structure of a process using locks :

```
do {  
    acquire lock  
    critical section  
    release lock  
    remainder section  
} while (TRUE);
```

(i) Test and set lock (TSL)

- This instruction reads the contents of a memory location, stores it in a register and then stores a non-zero value at the address.
- This operation is guaranteed to be indivisible (TSL- atomic).
- That is, no other process can access that memory location until the TSL instruction has finished.

(i) TSL Instruction (

Test and Set Lock)

- Defining the Test and Set Lock:

```
boolean TestAndSet(boolean *target) {  
    boolean rv = *target;  
    *target = TRUE;  
    return rv;  
}
```

Test and Set Lock Instruction

- If two TestAndSet() instructions are executed simultaneously (each on a different CPU), they will be executed sequentially in some arbitrary order.
- If the machine supports the TestAndSet() instruction, then we can implement mutual exclusion by declaring a Boolean variable lock, initialized to false. The structure of a Process using TSL is presented as follows:

General structure of a process using Test and set lock (TSL)

```
do {
    while (TestAndSet(&lock))
        ; // do nothing

    // critical section

    lock = FALSE;

    // remainder section
} while (TRUE);
```

(ii) Compare and swap

- The Swap() instruction, in contrast to the TestAndSet() instruction, operates on the contents of two words.
- Like the TestAndSet() instruction, it is executed atomically. If the machine supports the Swap() instruction, then mutual exclusion can be provided as follows:
 - A global Boolean variable lock is declared and is initialized to false. In addition, each process has a local Boolean variable key.
 - The structure of process Pi is given in the next slide.

(ii) Compare and swap Instruction

```
do {  
    while (compare_and_swap(&lock, 0 , 1)!=0)  
        ; /* do nothing */  
    /* critical section */  
    lock = 0;  
    /* remainder section */  
} while (true);
```

Note:

- Although mutual exclusion exclusion is guaranteed. Bounded waiting is not met in both Test and set Lock & Compare and swap methods. **Hence, to fulfill all 3 conditions for the Critical section problem following improved version of TSL is introduced.**

Data structures used.. (Improved version) TSL

- boolean waiting[n];
- boolean lock;
- Initially all the variables are initialized to False.

Bounded waiting Mutual exclusion with Test and set Lock

```
do {
    waiting[i] = TRUE;
    key = TRUE;
    while (waiting[i] && key)
        key = TestAndSet(&lock);
    waiting[i] = FALSE;

    // critical section

    j = (i + 1) % n;
    while ((j != i) && !waiting[j])
        j = (j + 1) % n;

    if (j == i)
        lock = FALSE;
    else
        waiting[j] = FALSE;

    // remainder section
} while (TRUE);
```

Explanation

- These data structures are initialized to false.
- To prove that the mutual exclusion requirement is met, we note that process P_i can enter its critical section only if either $\text{waiting}[i] == \text{false}$ or $\text{key} == \text{false}$.
- The value of key can become false only if the $\text{TestAndSet}()$ is executed.
- The first process to execute the $\text{TestAndSet}()$ will find $\text{key} == \text{false}$; all others must wait.
- The variable $\text{waiting}[i]$ can become false only if another process leaves its critical section; only one $\text{waiting}[i]$ is set to false, maintaining the mutual-exclusion requirement

Contd..

- To prove that the progress requirement is met, we note that the arguments presented for mutual exclusion also apply here, since a process exiting the critical section either sets lock to false or sets waiting[j] to false.
- Both allow a process that is waiting to enter its critical section to proceed.

Contd..

- To prove that the bounded-waiting requirement is met, we note that, when a process leaves its critical section, it scans the array waiting in the cyclic ordering $(i + 1, i + 2, \dots, n - 1, 0, \dots, i - 1)$.
- It designates the first process in this ordering that is in the entry section ($\text{waiting}[j] == \text{true}$) as the next one to enter the critical section. Any process waiting to enter its critical section will thus do so within $n - 1$ turns.

Mutex Locks

- Previous solutions are complicated and generally inaccessible to application programmers
- OS designers build software tools to solve critical section problem
- Simplest is mutex lock
- Protect a critical section by first **acquire()** a lock then **release()** the lock
 - Boolean variable indicating if lock is available or not
- Calls to **acquire()** and **release()** must be atomic
 - Usually implemented via hardware atomic instructions
- But this solution requires **busy waiting**
 - This lock therefore called a **spinlock**

Mutex locks Contd..

- `acquire() {
 while (!available)
 ; /* busy wait */
 available = false;;
}
■ release() {
 available = true;
}
■ do {
 acquire lock
 critical section
 release lock
 remainder section`

Semaphores

- Discovered by Dijkstra
 - Synchronization tool that does not require busy waiting
- Semaphore or mutex variable **S** □ integer variable
- Two operations:
 - Wait()
 - Signal()
- Less complicated

Semaphore – Contd...

General Structure of critical section using semaphores

```
do {
    Wait(S)
    critical section
    Signal(S)
    remainder section
} while (TRUE);
```

Definition-signal

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

Definition-Wait

```
wait(S)
{
    while(S<=0)
        ; //no-op
    S--;
}
```

Operations on semaphores (wait and signal / Down and Up)

- Two operations:
 - block()** – place the process in the waiting queue
 - wakeup()** – remove one of the processes in the waiting queue and place it in the ready queue

```
wait(semaphore *S) {
    S->value--;
    if (S->value < 0) {
        add this process to S->list;
        block();
    }
}
```

```
signal(semaphore *S) {
    S->value++;
    if (S->value <= 0) {
        remove a process P from S->list;
        wakeup(P);
    }
}
```

Semaphore usage

- Counting semaphore
 - Values are unrestricted
- Binary semaphore
 - Values can range only between 0 and 1
- We are using binary semaphores

Mutual Exclusion implementation with semaphores

```
do {  
    Wait(mutex)  
    critical section  
    Signal(mutex)  
    remainder section  
} while (TRUE);
```

Example 1:

- Semaphore is initialized with the number of resources available.
 - Semaphore is having 10 printers
 - Each process P_i , that needs the resource perform the **wait()** **operation** on semaphore [thereby decrementing the count]
 - When the process P_i , releases the resources, it performs **signal()** **operation** [incrementing the count value]
 - Count = 0 \square all resources are used
 - Count = n \square all processes released the resources

Example 2:

- Assume 2 concurrent running processes
 - P1 with statement S1;
 - P2 with statement S2;
- s2 is executed after s1 has completed
 - P1 and p2 share a common semaphore
 - **Synch = 0**

S1;
Signal(synch);

Fig. Statements in P1

wait(synch);
S2;

Fig. Statements in P2

Disadvantages of semaphores

● Busy waiting

- (ie) when a process is in critical section, the other process loop indefinitely.
- Wasting its time

Semaphore Implementation with no Busy Waiting

- Two operations:
 - block()** – place the process in the waiting queue
 - wakeup()** – remove one of the processes in the waiting queue and place it in the ready queue

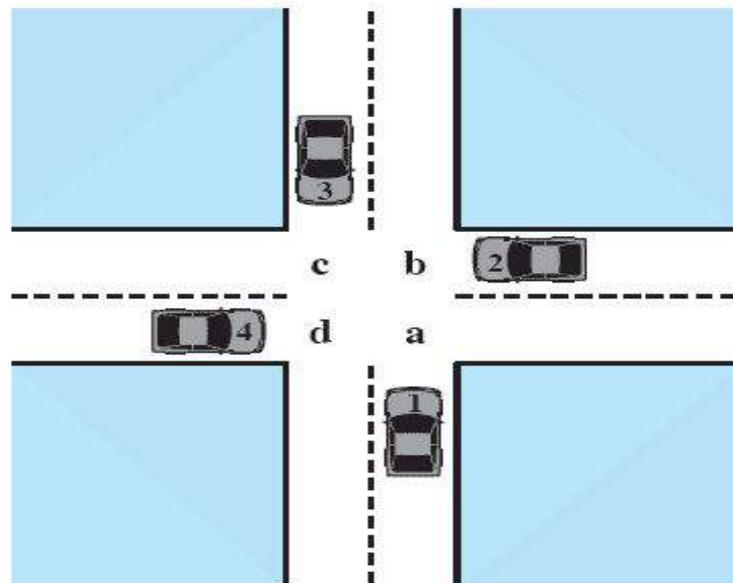
```
wait(semaphore *S) {  
    S->value--;  
    if (S->value < 0) {  
        add this process to S->list;  
        block();  
    }  
}
```

```
signal(semaphore *S) {  
    S->value++;  
    if (S->value <= 0) {  
        remove a process P from S->list;  
        wakeup(P);  
    }  
}
```

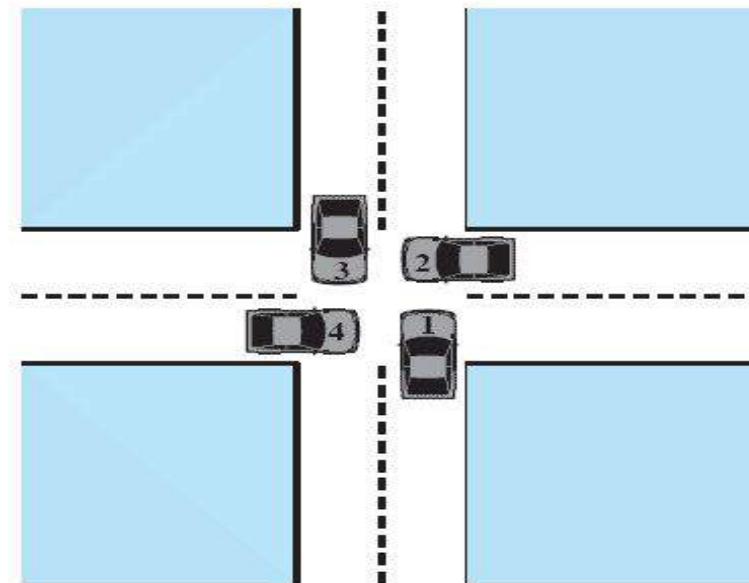
Deadlocks

- Definition:

- Two or more processes waiting indefinitely for an event to be completed is called as **Deadlock**.



(a) Deadlock possible



(b) Deadlock

Figure 6.1 Illustration of Deadlock

Example

- System has 2 tape drives.
 - P_1 and P_2 each hold one tape drive and each needs another one.
-
- Consider 2 processes P_0 and P_1 .
 - semaphores A and B , initialized to 1

P_0	P_1
wait (A);	wait(B)
wait (B);	wait(A)
signal(B);	signal(A)
signal(A);	signal(B)

P0 and P1 are deadlocked

Classical problems of Synchronization

- Bounded-Buffer Producer/Consumer Problem
- Readers and Writers Problem
- Dining Philosophers Problem (Monitor)

Bounded-Buffer Producer/Consumer Problem

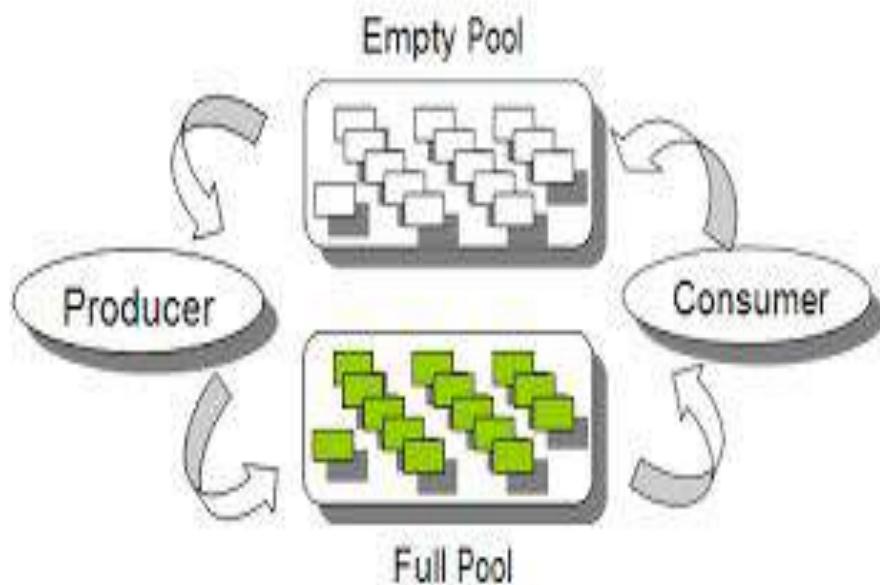
- Shared data:

semaphore full, empty, mutex

- Initially:

full = 0, empty = n, mutex = 1

where n is the buffer size



Empty Empty Buffer

Full Full Buffer

Producer make buffer full

Consumer empty the buffer

Explanation

- Consider a pool contains “n” buffers and each buffer can hold an item. The shared variable “mutex” provides the required mutual exclusion for the buffer pool, “empty” and “full” are semaphore variables used to count the number of empty and full buffers.
- Need to initialize the mutex variable as ‘1’, empty=‘n’ and full=‘0’.
- According to the production and consumption by the producer and the consumer, the variables empty and full will get modified.
- With the help of “wait” and “signal” method of semaphores, the bounded buffer problem can be handled properly.

The structure of the Producer process

```

do {

    // produce an item in nextp

    wait (empty);
    wait (mutex);

    // add the item to the buffer

    signal (mutex);
    signal (full);

} while (TRUE);

```

The structure of the consumer process

```

do {

    wait (full);
    wait (mutex);

    // remove an item from buffer to nextc

    signal (mutex);
    signal (empty);

    //consume the item in nextc

} while (TRUE);

```

Readers-Writers Problem

- A data set is shared among many processes
 - Readers – only read the data set; they do **not** perform any updates
 - Writers – can both read and write

- **Problem**

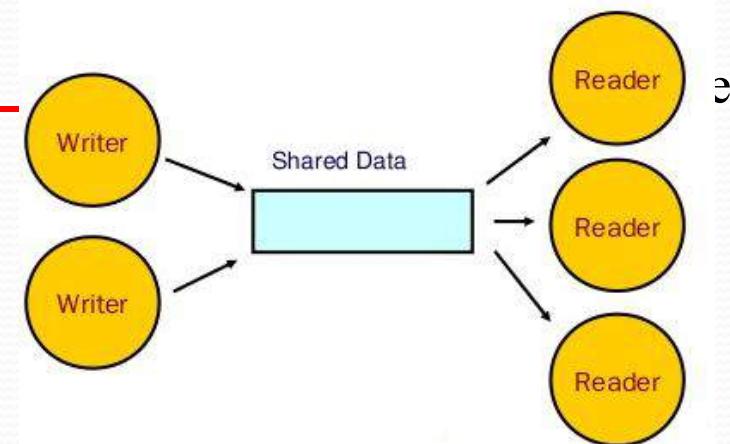
- Allow multiple readers to read at the same time
- But Only one writer can write

- **Shared data:**

mutex, rw_mutex

- **Initially:**

mutex = 1, rw_mutex = 1,
int readcount = 0



Explanation

Consider a situation of having concurrent read and write operation over a common resource like database. In which, many users wants to read and write on the same database. If many users are concurrently perform read operation, it will not create any problem. Whereas if a write operation and any other operation(may be read or write) are concurrently performed on the common field may leads to inconsistencies in the database content.

This synchronization problem is called as “reader-writers” problem. The order in which the read and write operation performed may leads to starvation if they are not synchronized properly.



The structure of the Reader process

```
do {  
    wait (mutex) ;  
    readcount ++ ;  
    if (readcount == 1)  
        wait (rw_mutex) ;  
  
    signal (mutex)  
  
    //reading is performed  
  
    wait (mutex) ;  
    readcount -- ;  
    if (readcount == 0)  
        signal (rw_mutex) ;  
    signal (mutex) ;  
} while (TRUE);
```

The structure of the Writer process

```
do {  
    wait (rw_mutex) ;  
  
    // writing is performed  
  
    signal (rw_mutex) ;  
} while (TRUE);
```

The structure of reader and writer process is given in the above figure.

“rw_mutex” semaphore variable is shared among readers and writer processes.

“rw_mutex” act as a mutual exclusion semaphore for writer processes.

“read_count” variable get updated whenever a new reader process will come or when it gets completed.

“read_count” will specify the number of current read processes waiting for the resource.

With the help of “wait” and “signal” methods of semaphore, the synchronization between readers and writers processes can be achieved.

Dining-Philosophers Problem

- Five philosophers spend their lives **thinking** and **eating**
- Circular table with 5 chairs, 5 philosophers, 5 plates and 5 chopsticks
- Each philosopher need 2 chopsticks to eat. After eating, they drop chopsticks
- One person cannot pickup chopstick (ie) already in hand of a neighbour
- Shared data
 - Bowl of rice (data set)
 - Semaphore **chopstick [5]** initialized to 1





The structure of the Philosopher

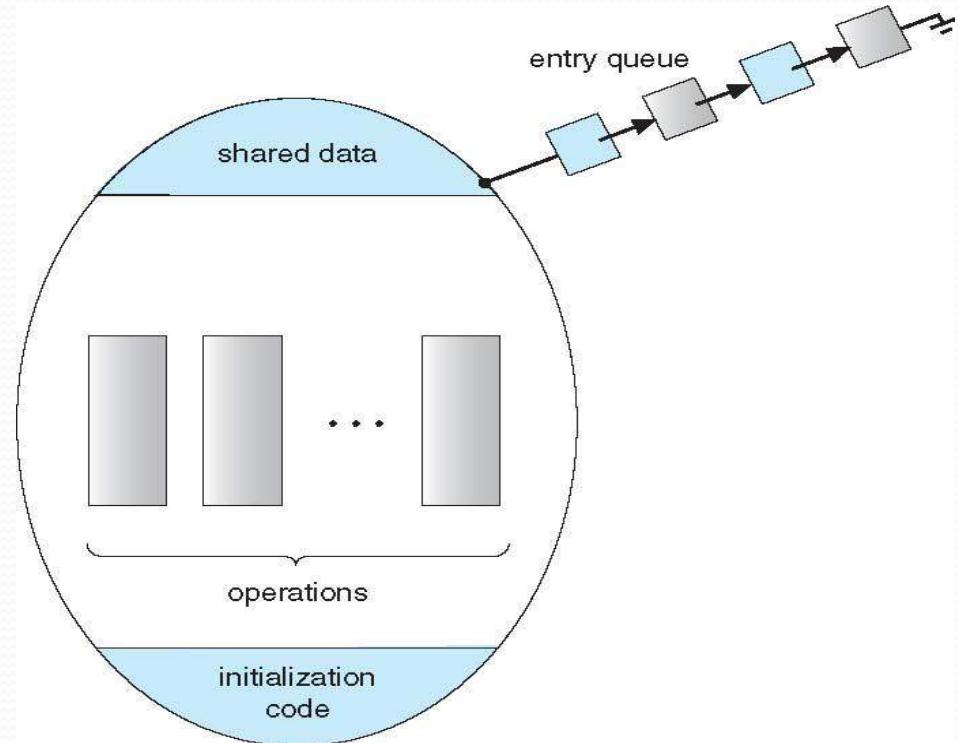
```
do {  
    wait ( chopstick[i] );  
    wait ( chopStick[ (i + 1) % 5] );  
  
    // eat  
  
    signal ( chopstick[i] );  
    signal (chopstick[ (i + 1) % 5] );  
  
    // think  
  
} while (TRUE);
```

To avoid deadlock in Dining Philosopher problem

- At most 4 philosophers to sit simultaneously
- Allow philosophers to pick chopsticks, only when both chopsticks are available
- Use an asymmetric solution
 - Odd philosopher picks up **left** chopstick first
 - Even philosopher picks up **right** chopstick

Monitors

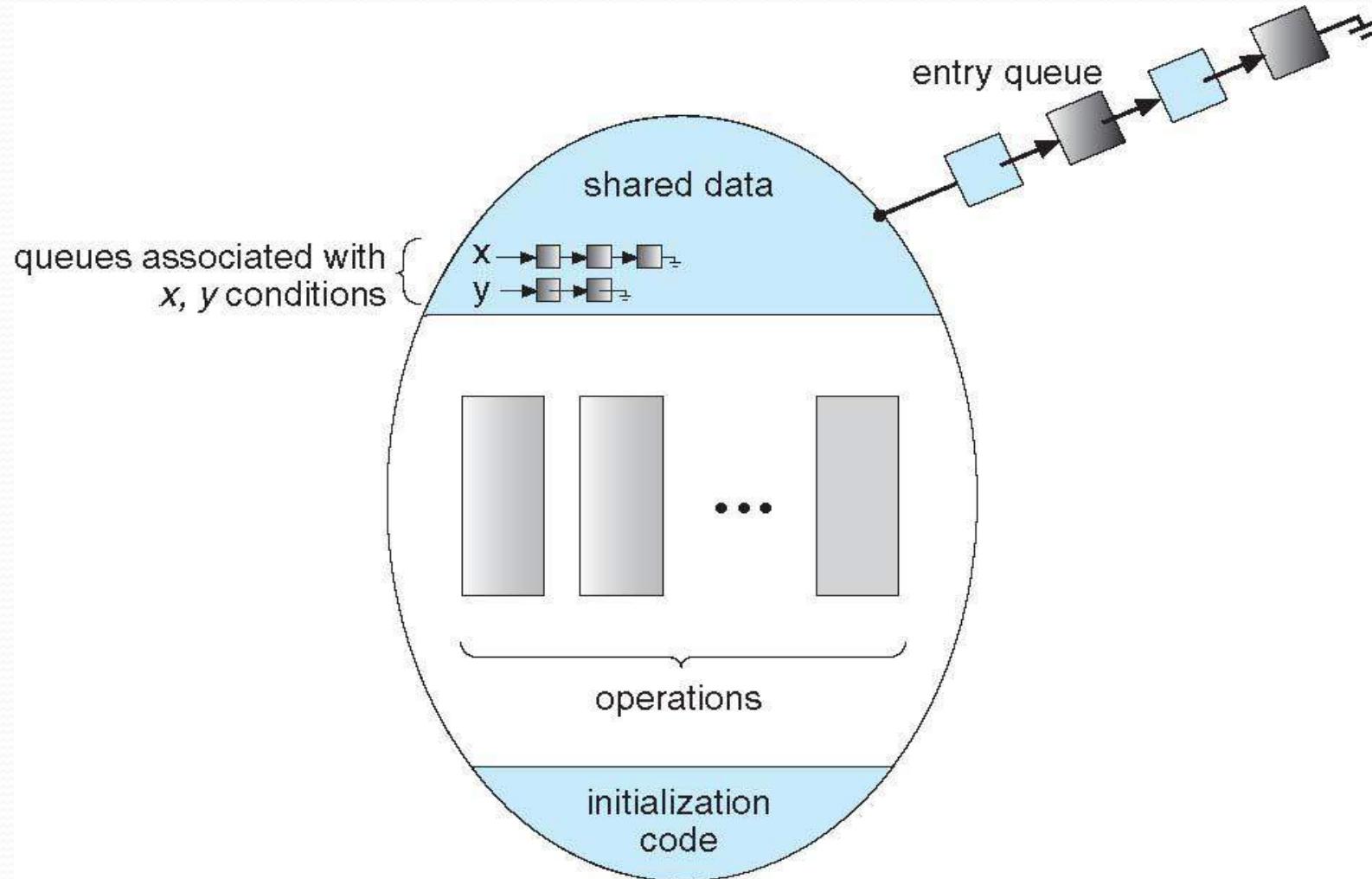
- Monitor is **Abstract data type**, where the internal variables only accessible by code within the procedure
 - Programming languages like PASCAL, C# implement the concept of monitor
 - Only one process is active within the monitor at a time
 - Not powerful



Variables & Operations

- Two variables of data type condition is used
condition x, y;
- Two operations
 - **x.wait ()**
 - A process that invokes the operation until x.signal ()
 - **x.signal ()**
 - Resumes one of processes that invoked x.wait ()

Monitor with Condition Variables





Monitor (syntax)

- High-level synchronization construct that allows the safe sharing of an abstract data type among concurrent processes.

```
type monitor-name = monitor
    variable declarations
    procedure entry P1 :(...);
        begin ... end;
    procedure entry P2(...);
        begin ... end;
        :
    procedure entry Pn (...);
        begin...end;
begin
    initialization code
end
```

Monitor solution for Dining Philosopher problem

```

monitor DiningPhilosophers
{
    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((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;
    }
}

```

~~General structure of a Philosopher i using Pickup() and Putdown()~~

Syntax:

DiningPhilosophers.pickup(i);

...

eat

...

DiningPhilosophers.putdown(i);

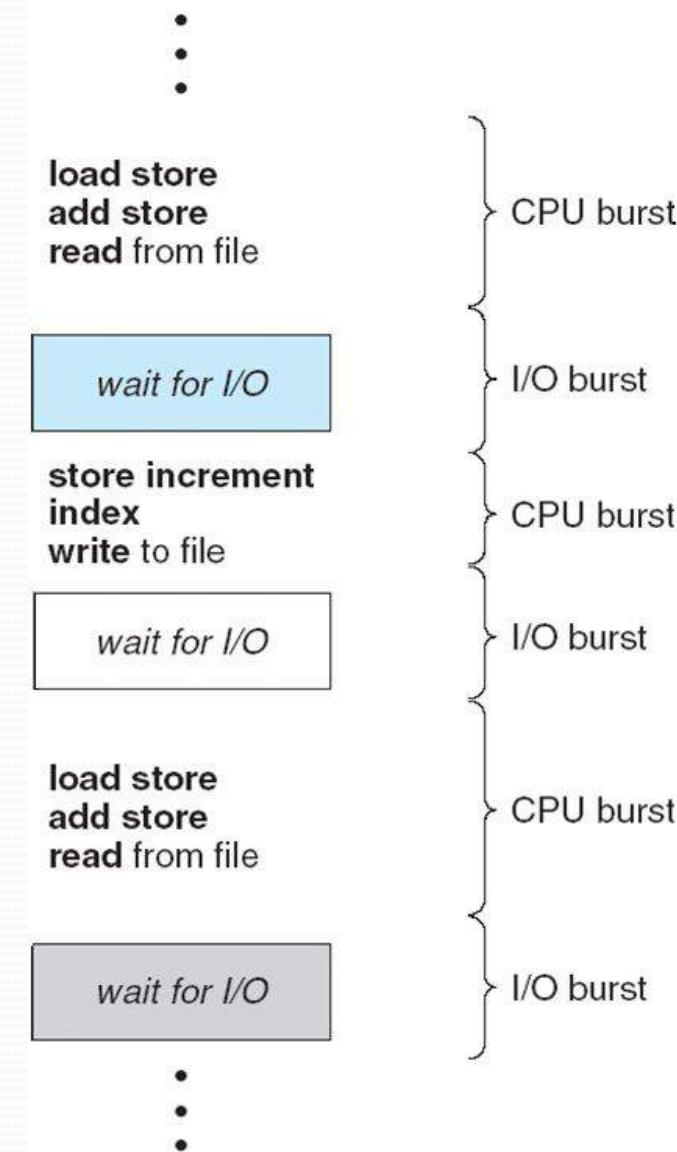
Note : This solution ensures that **no two neighbors** are eating simultaneously and that no deadlocks will occur.

PROCESS SCHEDULING



Basic Concepts

- Maximum CPU utilization is obtained with multiprogramming
- Process execution consists of a *cycle* of a **CPU time burst** and an **I/O time burst**
 - Processes alternate between these two states (i.e., CPU burst and I/O burst)
 - Eventually, the final CPU burst ends with terminate execution



Preemptive:

The CPU is allocated to the process, if any higher priority process comes it releases the CPU and gets the service once the higher priority process completes.

Non Preemptive:

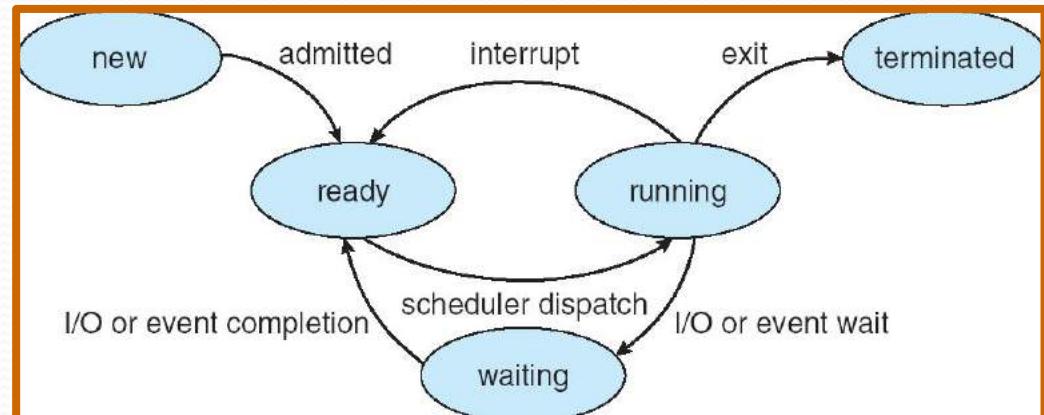
- Once the CPU is allocated to the process, the process keeps the CPU until it releases the CPU either by terminating or switching to waiting state.

CPU Scheduler

- The CPU scheduler selects from among the processes in memory that are ready to execute and allocates the CPU to one of them

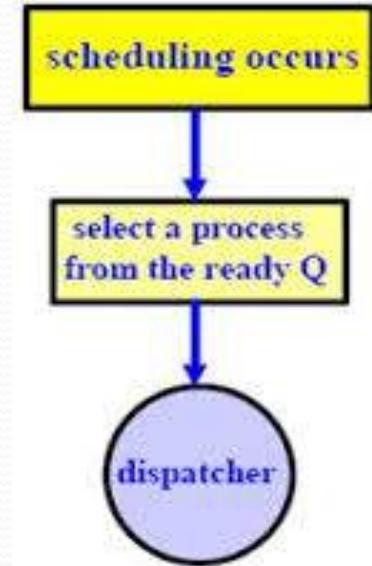
Ready Queue \square CPU

- When CPU scheduling takes place?
 - (N) A process switches from **running** to **waiting** state
 - (P) A process switches from **running** to **ready** state
 - (P) A process switches from **waiting** to **ready** state
 - (N) A processes switches from **running** to **terminated** state
- Circumstances 1 and 4 are **non-preemptive**
- Circumstances 2 and 3 are **pre-emptive**



Dispatcher

- The dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
 - switching context
 - switching to user mode
 - jumping to the proper location in the user program to restart that program
- The time it takes for the dispatcher to stop one process and start another process is called **dispatch latency**





Scheduling Criteria

- Different CPU scheduling algorithms have different properties
 - **CPU utilization** – keep CPU as busy as possible
 - CPU utilization ranges from 0% to 100%
 - Lightly loaded system 40%
 - Heavily loaded system 90%
 - **Throughput** = Number of processes completed /Unit time
 - **Response time** – amount of time it takes from when a request was submitted until the first response occurs
 - **Waiting time** – the amount of time the processes has been waiting in the ready queue
 - **Turnaround time** – amount of time to execute a particular process from the time of submission through the time of completion

Scheduling Algorithms

1. First-Come, First-Served (FCFS) Scheduling
2. Shortest-Job-First (SJF) Scheduling
 - Simultaneous arrival times
 - Varied arrival times
 - Preemptive SJF with varied arrival times = Shortest-remaining time First (SRT) Scheduling
3. Priority Scheduling
 - Preemptive & non preemptive
4. Round robin scheduling
5. Multi-level Queue Scheduling
6. Multilevel Feedback Queue Scheduling



Scheduling

- The first entered job is the first one to be serviced.
- Example: Three processes arrive in order P1, P2, P3.
 - P1 burst time: 24
 - P2 burst time: 3
 - P3 burst time: 3
- Draw the Gantt Chart and compute Average Waiting Time and Average Completion Time.

First-Come, First-Served (FCFS)

- Example: Three processes arrive in order P1, P2, P3.

- P1 burst time: 24
- P2 burst time: 3
- P3 burst time: 3

- Waiting Time**

- P1: 0
- P2: 24
- P3: 27

- Completion Time**

- P1: 24
- P2: 27
- P3: 30

- Average Waiting Time: $(0+24+27)/3 = 17$

- Average Turnaround time: $(24+27+30)/3 = 27$



Convoy effect (2 mark)

All the other processes wait for one long process to finish its execution

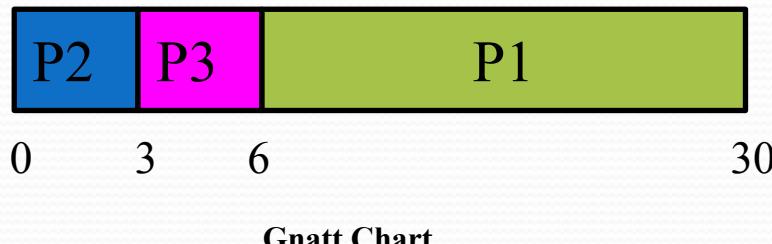
First-Come, First-Served (FCFS)

- What if their order had been P2, P3, P1?
 - P1 burst time: 24
 - P2 burst time: 3
 - P3 burst time: 3

First-Come, First-Served (FCFS)

- What if their order had been P2, P3, P1?

- P1 burst time: 24
- P2 burst time: 3
- P3 burst time: 3



- Waiting Time**

- P2: 0
- P3: 3
- P1: 6

- Turn-around Time**

- P2: 3
- P3: 6
- P1: 30

Average Waiting Time: $(0+3+6)/3 = 3$ (compared to 17)

Average turn-around Time: $(3+6+30)/3 = 13$ (compared to 27)

FIFO (First In and First Out) or FCFS

Advantages:

- Simple

Disadvantages:

- Short jobs get stuck behind long ones
- There is no option for pre-emption of a process. If a process is started, then CPU executes the process until it ends.
- Because there is no pre-emption, if a process executes for a long time, the processes in the back of the queue will have to wait for a long time before they get a chance to be executed.
-

Shortest-Job-First (SJF) Scheduling

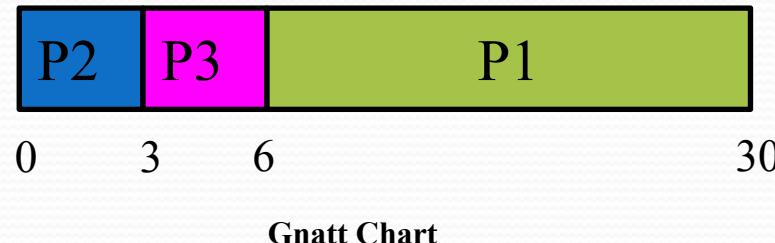
(simultaneous arrival ie. all jobs arrive at the same time)

Example 1

- P1 burst time: 24
- P2 burst time: 3
- P3 burst time: 3

Waiting Time

- P2: 0
- P3: 3
- P1: 6



Turn-around Time

- P2: 3
- P3: 6
- P1: 30

● **Average Waiting Time:** $(0+3+6)/3 = 3$

● **Average turn-around Time:** $(3+6+30)/3 = 13$

Shortest-Job-First (SJF) Scheduling

Here come the concept of arrival time.

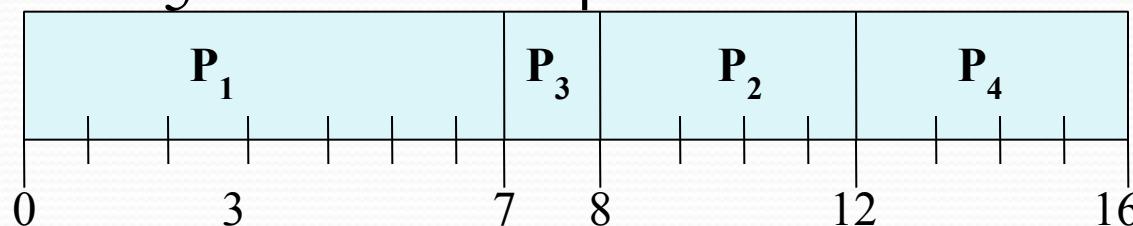
SJF (non-preemptive, varied arrival times)

Example 2

<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
----------------	---------------------	-------------------

P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

Gnatt Chart



- Average waiting time

$$\begin{aligned}
 &= ((0 - 0) + (8 - 2) + (7 - 4) + (12 - 5)) / 4 \\
 &= (0 + 6 + 3 + 7) / 4 = 4
 \end{aligned}$$

- Average turn-around time:

$$\begin{aligned}
 &= ((7 - 0) + (12 - 2) + (8 - 4) + (16 - 5)) / 4 \\
 &= (7 + 10 + 4 + 11) / 4 = 8
 \end{aligned}$$

Waiting time : sum of time that a process has spent waiting in the ready queue

Shortest-remaining time First (SRT) Scheduling

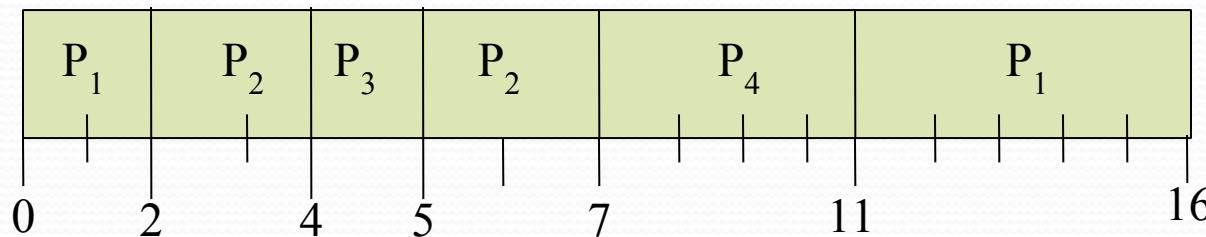
Preemptive SJF with varied arrival times



<u>Process</u>	<u>Arrival Time</u>	<u>Burst Time</u>
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

Example 3

Gnatt Chart



- Average waiting time
- $= (([0 - 0] + (11 - 2)] + [(2 - 2) + (5 - 4)] + (4 - 4) + (7 - 5))/4$
- $= 9 + 1 + 0 + 2)/4$
- $= 3$
- Average turn-around time $= (16-0) + (7-2) + (5-4) + 11-4)/4 = 7$



Shortest-Job-First (SJF) Scheduling

Pros and Cons

Advantages:

- Works based on the next process CPU burst
- It gives optimal waiting time

Disadvantages:

- Long jobs get stuck behind short ones

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer \equiv highest priority)
 - Preemptive
 - Nonpreemptive
- SJF is priority scheduling where priority is the inverse of predicted next CPU burst time
- Problem \equiv **Starvation** – low priority processes may never execute
- Solution \equiv **Aging** – as time progresses increase the priority of the process

Priority Scheduling



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

(non –Preemptive)

- A priority number (integer) is associated with each process (smallest integer = highest priority)

<u>Process</u>	<u>Burst Time</u>	<u>Priority</u>
A	8	2
B	1	1
C	1	3



0 1 9 10

Gnatt Chart

- Avg Wait Time $(0 + 1 + 9) / 3 = 3.3$

Priority Scheduling

(Preemptive)

- Consider the example with seven process.

PID	Priority	Arrival Time	Burst Time	Completion Time(CT)	Turn Around Time(TAT)	Waiting Time (WT)
P1	2(low)	0	4	25	25	21
P2	4	1	2	22	21	19
P3	6	2	3	21	19	16
P4	10	3	5	12	9	4
P5	8	4	1	19	15	14
P6	12(high)	5	4	9	4	0
P7	9	6	6	18	12	6

Gantt chart

P1	P2	P3	P4	P6	P4	P7	P5	P3	P2	P1	
0	1	2	3	5	9	12	18	19	21	22	25

Average waiting time

- **Starvation**
- It is a situation in which the continuous arrival of higher priority process keeps the lowest priority process always in waiting state. The waiting process will starve (in other words, the deadline of the waiting process will never meet). We can resolve the starvation problem in the priority scheduling with the help of Aging technique.
- **Aging Technique**
- In Aging technique, the priority of every lower priority processes has to be increased after a fixed interval of time.



Priority Scheduling

Pros and Cons

Advantages:

- Higher priority job executes first

Disadvantages:

- Starvation ie. low priority processes never execute.
- To overcome the above problem “AGING” □ the priority of a process is increased



Round Robin (RR) Scheduling

- In the round robin algorithm, each process gets a small unit of CPU time (**a *time quantum***), usually **10-100 ms**.

- After this time has elapsed, the process is preempted and added to the end of the ready queue.

- Performance of the round robin algorithm
 - q large \Rightarrow FCFS
 - q small \Rightarrow q must be greater than the context switch time; otherwise, the overhead is too high

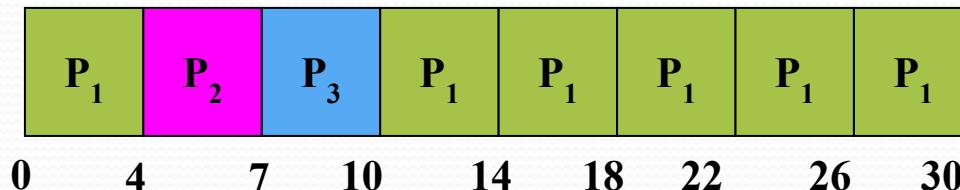


Example of RR with Time Quantum = 4

Example 1

<u>Process</u>	<u>Burst Time</u>
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



Average turn around time is larger than SJF
But more context switching

Average waiting time = $(6+4+7)/3 = 5.6$ ms



Example of RR with Time Quantum = 20

Process Burst Time

P_1 53

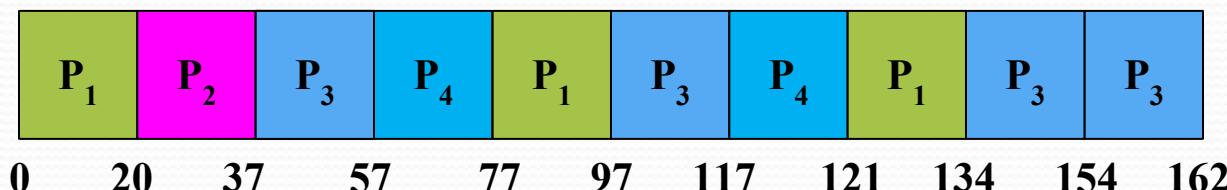
P_2 17

P_3 68

P_4 24

Example 2

Gantt chart is:



- Average waiting time

$$= (([0 - 0] + (77 - 20) + (121 - 97)] + (20 - 0) + [(37 - 0) + (97 - 57) + (134 - 117)] + [(57 - 0) + (117 - 77)]) / 4$$

$$= (0 + 57 + 24) + 20 + (37 + 40 + 17) + (57 + 40) / 4$$

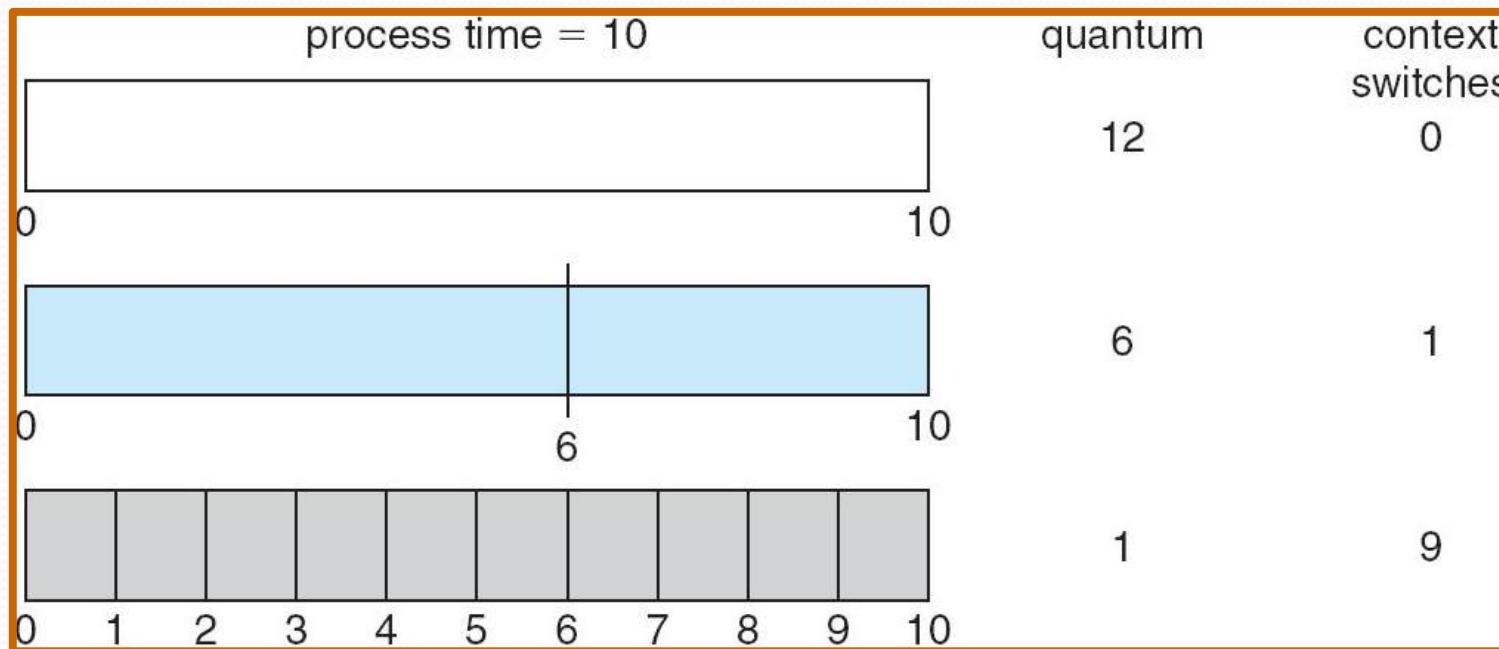
$$= (81 + 20 + 94 + 97) / 4$$

$$= 292 / 4 = 73$$

- Average turn-around time $= (134 + 37 + 162 + 121) / 4 = 113.5$



Time Quantum and Context Switches





Round Robin (RR) Scheduling

Pros and Cons

Advantages:

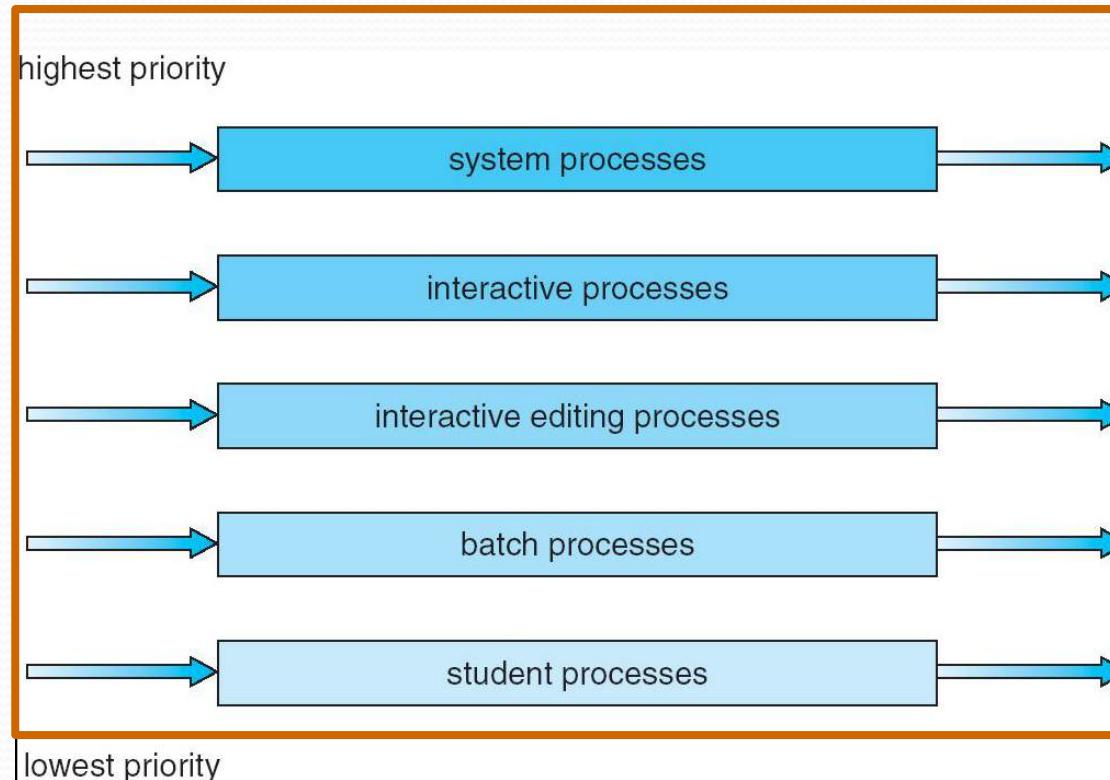
- Fair for smaller tasks

Disadvantages:

- More context switching

Multi-level Queue Scheduling

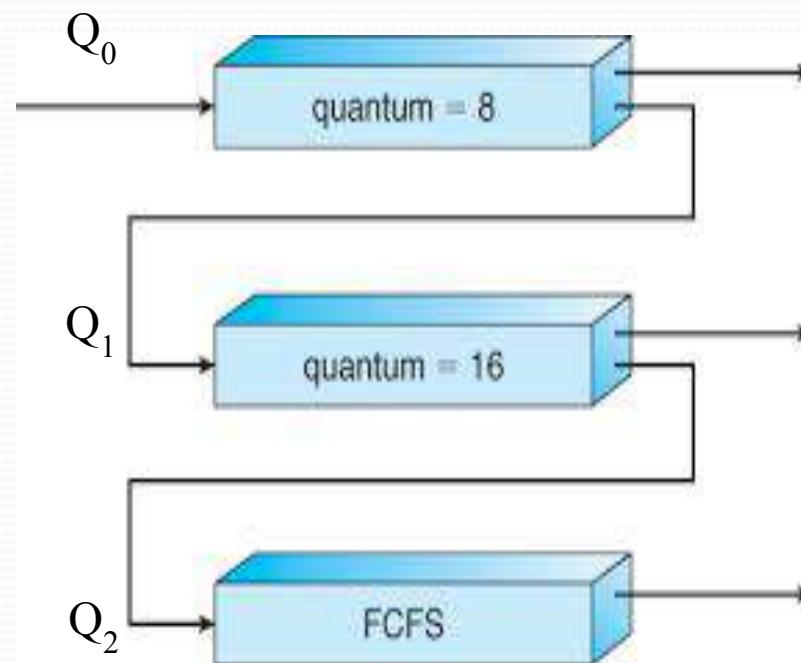
- Multi-level queue scheduling is used when processes can be classified into groups
 - For example, foreground (interactive) processes and background (batch) processes
 - 80% of the CPU time to foreground queue using RR.
 - 20% of the CPU time to background queue using FCFS



Multilevel Feedback Queue Scheduling

- In multi-level feedback queue scheduling, a process can move between the various queues;

- A new job enters queue Q_0 (RR) and is placed at the end. When it gains the CPU, the job receives 8 milliseconds. If it does not finish in 8 milliseconds, the job is moved to the end of queue Q_1 .
- A Q_1 (RR) job receives 16 milliseconds. If it still does not complete, it is preempted and moved to queue Q_2 (FCFS).



Real-Time CPU Scheduling

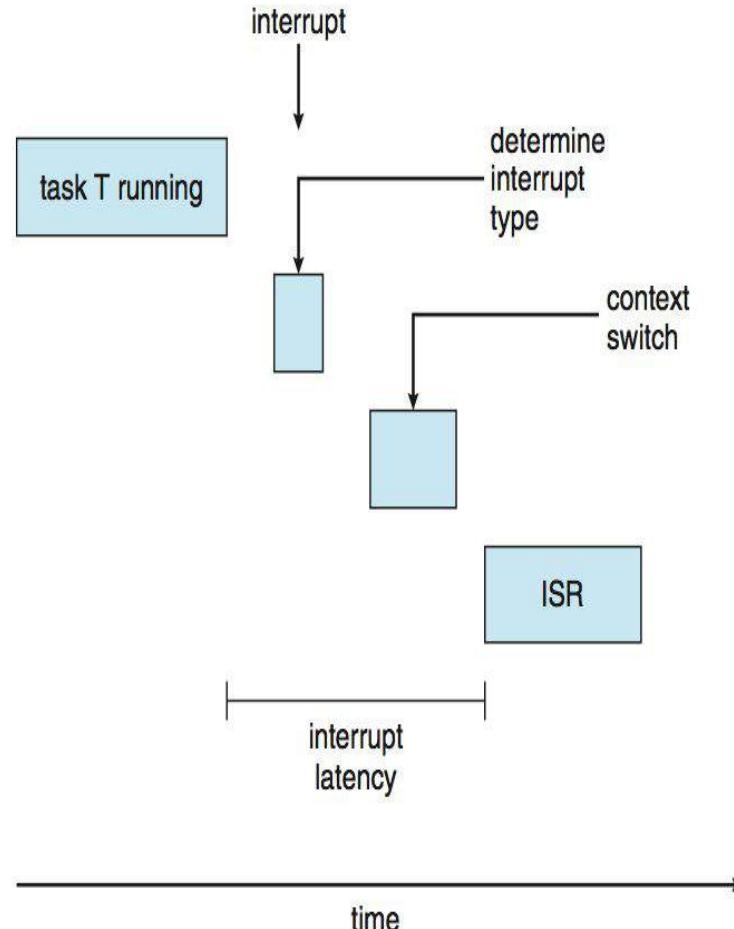
- **CPU scheduling for real-time operating systems involves special issues.**
- In general, we can distinguish between soft real-time systems and hard real-time systems.
- **Soft real-time systems** provide no guarantee as to when a critical real-time process will be scheduled. They guarantee only that the process will be given preference over noncritical processes.
- **Hard real-time systems** have stricter requirements. A task must be serviced by its deadline; service after the deadline has expired is the same as no service at all.
- In this section, we explore several issues related to process scheduling in both soft and hard real-time operating systems

Real-Time CPU Scheduling

- Two types of latencies affect performance

1. **Interrupt latency** – time from arrival of interrupt to start of routine that services interrupt

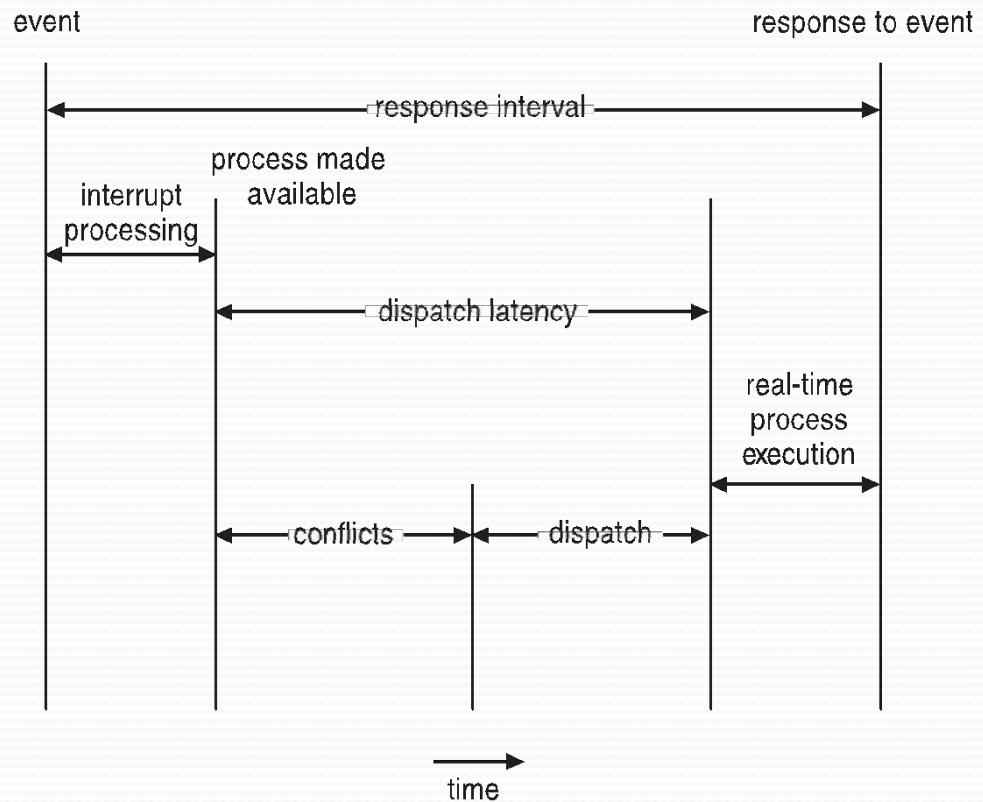
2. **Dispatch latency** – time for scheduler to take current process off CPU and switch to another



Real-Time CPU Scheduling (Cont.)

- Conflict phase of dispatch latency:

1. Preemption of any process running in kernel mode
2. Release by low-priority process of resources needed by high-priority processes



Priority-based Scheduling

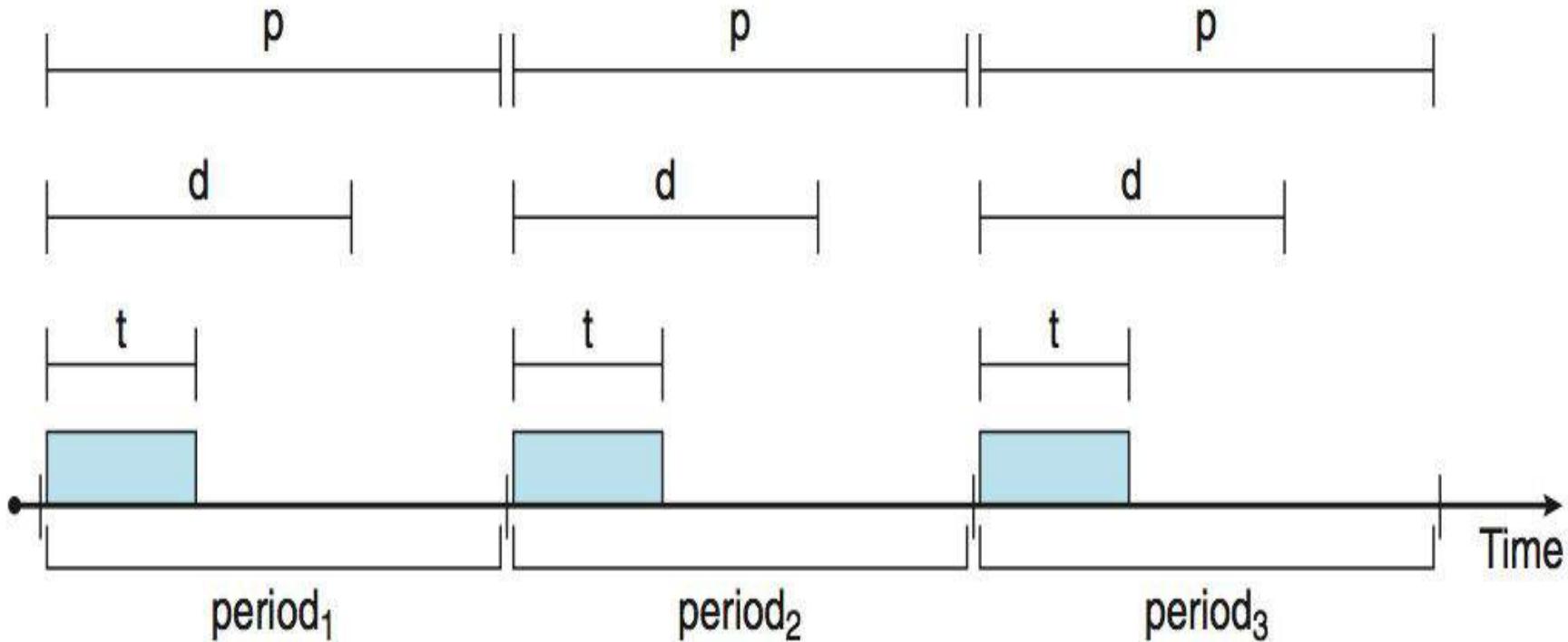


- The most important feature of a real-time operating system is to **respond immediately to a real-time process as soon as that process requires the CPU.**
- As a result the scheduler for a real-time operating system must support a **priority-based algorithm with preemption.**
- Recall that priority-based scheduling algorithm **assign each process a priority based on its importance;** more are assigned higher priorities than those deemed less important.
- If the scheduler also supports preemption, a process currently running on the CPU will preempted if a higher-priority process becomes available to run.

Priority-based Scheduling

- Before we proceed with the details of the individual schedulers, how we must **define certain characteristics of the processes that are to be scheduled.**
- Periodic processes require the CPU at specified intervals (periods).
- **p is the duration of the period.**
- **d is the deadline** by when the process must be serviced.
- **t is the processing time.**
- The **relationship of the processing time, the deadline, and the period can be expressed as $0 \leq t \leq d \leq p$.**

Priority-based Scheduling



Rate Monotonic Scheduling

- The rate-monotonic scheduling algorithm schedules periodic tasks using static priority policy with preemption.
- If a lower-priority process is running and a higher-priority process becomes available to run, it will preempt the lower priority process.
- Upon entering the system, **each periodic tasks and Priority inversely based on its period.**
- **Shorter periods = higher priority;**
- **Longer periods = lower priority**
- **The rationale behind this policy is to assign a higher priority to tasks that require the CPU more often.**
- Furthermore, rate-monotonic scheduling assumes that the processing time of A periodic process is the same for each CPU burst. That is, every time a process acquires the CPU, the duration of its CPU burst is the same

Rate Monotonic Scheduling



- Let's consider an example. We have two processes, P1 and P2.
- The periods for P1 and P2 are $p_1=50$, $p_2=100$. The processing times are $t_1=20$, $t_2=35$.
- The deadline for each it complete its CPU burst by the start of its next period.
- We must ask ourselves whether it is possible to schedule these tasks so that each meets its deadlines.
- If we measure the CPU utilization of a process P_i as the ratio of its burst to its period t_i/p_i .
- The CPU utilization of P1 is $20/50=0.40$ and that P2 is $35/100 = 0.35$, for a total CPU utilization of 75percent.
- Therefore, it seems we can schedule these tasks in such a way that both meet their deadlines and still leave the CPU with available cycles.

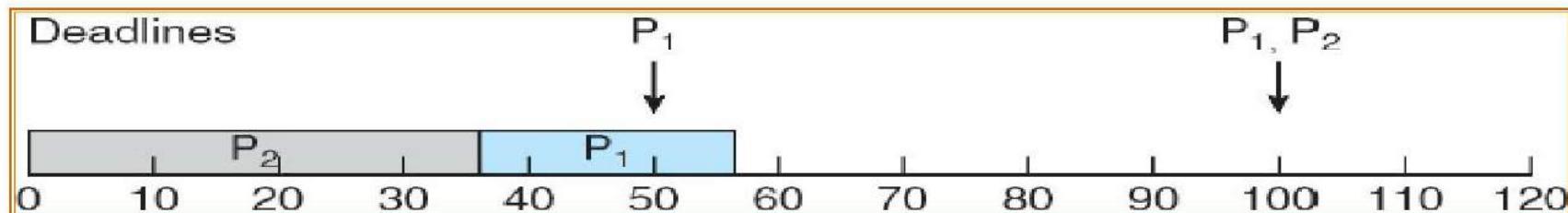
Rate Monotonic Scheduling



Example(Why not Priority scheduling)

- Suppose we assign P2 a higher priority than P1.
- The execution of P1 and P2 in this situation is shown in the below Figure.
- As we can see, P2 starts execution first and completes at time 35.
- At this point, P1 starts; it completes its CPU burst at time 55.
- However, the first deadline for P1 was at time 50, so the scheduler has caused **P1 to miss its deadline**.

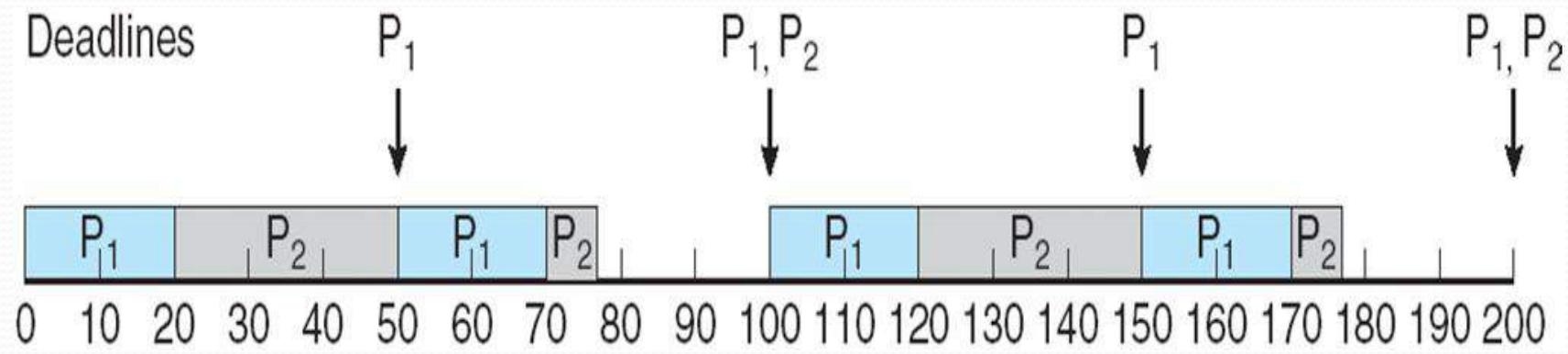
Scheduling of tasks when P2 has a higher priority than P1



Rate Monotonic Scheduling

Now suppose we use rate-monotonic scheduling, in which we assign P1 a higher priority than P2 because the period of P1 is shorter than that of P2.

- The execution of these processes in this situation is shown in the below Figure.
- P1 starts first and completes its CPU burst at time 20, thereby meeting its first deadline.
- P2 starts running at this point and runs until time 50.
- At this time, it is preempted by P1, although **it still has 5 milliseconds remaining in its CPU burst**.
- P1 completes its CPU burst at time 70, at which point the scheduler resumes P2.



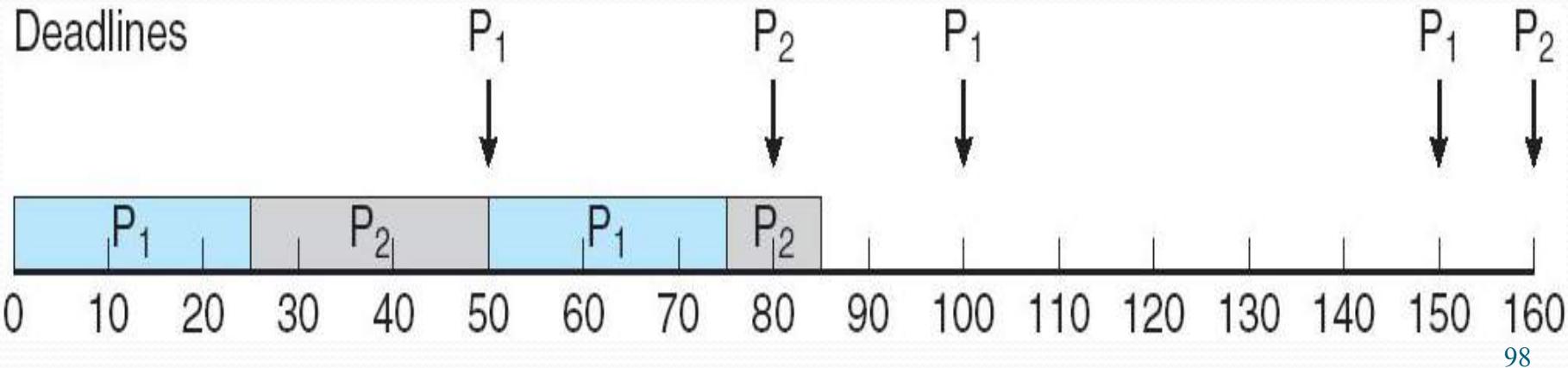
Missed Deadlines with Rate Monotonic Scheduling

- Let's next examine a set of processes that cannot be scheduled using the rate Monotonic algorithm.
- Assume that process P1 has a period of **p1 = 50** and a CPU burst of **t1=25**.
- For P2, the corresponding values are **p2= 80** and **t2= 35**.
- Rate-monotonic scheduling would assign process P1 a higher priority as it has the shorter period.
- The total CPU utilization of the two processes is $(25/50)+(35/80)=0.94$, and it therefore seems logical that the two processes could be scheduled and leave the CPU with 6 percent available time.

Missed Deadlines with Rate Monotonic Scheduling

- Below figure shows the scheduling processes P1 and P2. Initially P1, runs until it completes its CPU burst at time 25.
- Process P2 then begins running and runs until time 50, when it is preempted by P1.
- At this point, P2, still has 10 milliseconds remaining in its CPU burst. Process P1 runs until time 75; consequently, P2 finishes its burst at time 85, after the deadline for completion of its CPU burst at time 80.

$$p_1=50, p_2=80 \quad t_1=25, \\ t_2=35$$



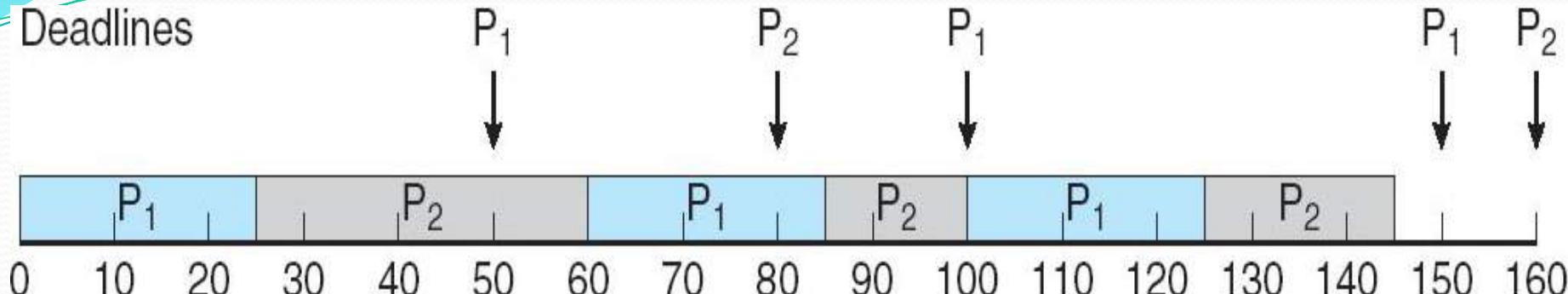
~~Earliest Deadline First Scheduling (EDF)~~

- Earliest-deadline-first (EDF) scheduling dynamically **assigns priorities according to deadline.**
- **The earlier the deadline, the higher the priority; the later the deadline, the lower the priority.**
- Under the EDF policy, when a process becomes runnable, it must announce its deadline requirements to the system. Priorities may have to be adjusted to reflect the deadline of the newly runnable process.
- Note how this differs from **rate-monotonic scheduling, where priorities are fixed.**
- **To illustrate EDF scheduling, we again schedule the processes which failed to meet deadline requirement under the rate-monotonic scheduling.**

Earliest Deadline First Scheduling (EDF)

- Recall that P1 has values of **p1=50** and **t1=25** and that values of **p2 = 80** and **t2= 35**.
- The EDF scheduling of these processes in below figure. **Process P1 has the earliest deadline, so its initial priority is higher than that of process P2.**
- Process P2 begins running at the end of the CPU burst for P1. However, whereas **rate-monotonic scheduling allows P1 to preempt P2 at the beginning of its next period at time 50, EDF scheduling allows process P2 to continue running.**
- P2 now has a higher priority than P1, because its next deadline (at time 80) is earlier than that of P1 (at time 100).
- Thus, **both P1 and P2 meet their first deadlines.** Process P1 again begins running, at time 60 and completes its second CPU burst at time 85, also meeting its second deadline at time 100.
- P2 begins running at this point, only to be preempted by P1 at the start of its next period at time 100. P2 is preempted because P1 has an earlier deadline (time 150) than P2(time 160).
- At time 125, P1 completes its CPU burst and P2 resumes execution, finishing at time 145 and meeting its deadline a well.
- The system is idle until time 150, when P1 is scheduled to run once again.

Earliest Deadline First Scheduling (EDF)

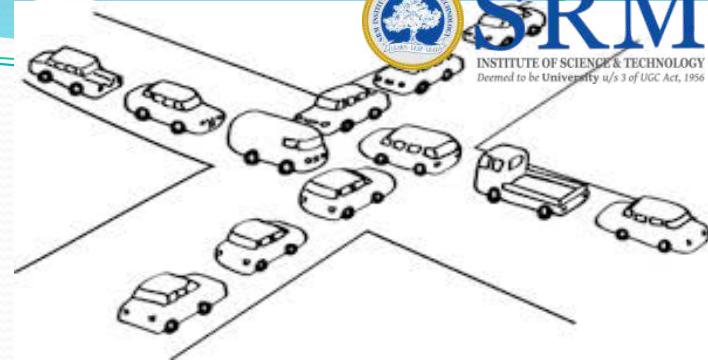


- Unlike the rate-monotonic algorithm, **EDF scheduling does not require that processes be periodic, nor must a process require a constant amount of CPU time per burst.**
- The **only requirement is that a process announce its deadline to the scheduler when it becomes runnable.**
- The appeal of EDF scheduling is that it is theoretically optimal -- theoretically, it can schedule processes so that each process **can meet its deadline requirements and CPU utilization will be 100 percent.**
- In practice, however, it is **impossible to achieve this level of CPU utilization** due to the cost of context switching between processes and interrupt handling.

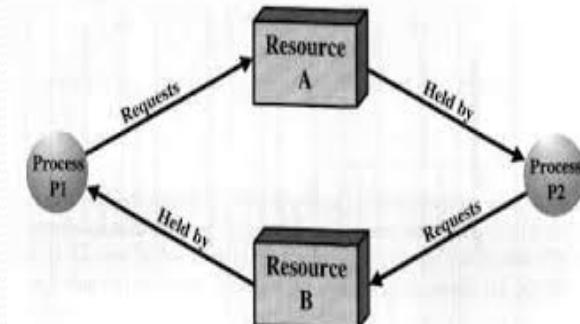
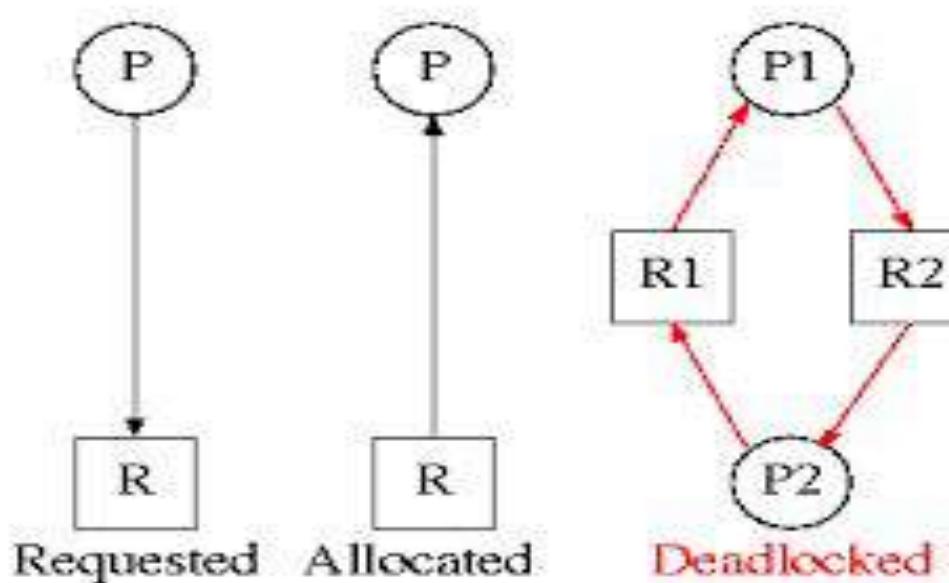


Deadlocks

Deadlocks



- Assume 2 process, P1 and p2.
- When p1 process is holding resource R1 and requesting for resource R2, where it is hold by process P2. This state is **DEADLOCK**.





System Model

- Assume resource types R_1, R_2, \dots, R_m
CPU cycles, memory space, I/O devices
 - Each resource type R_i has *1 or more* instances
- Each process utilizes a resource as follows:
- Request

The process requests the resource.

If (resource == available)

 Grant the resource

else

 Wait

- Use

- The process use the resource

- Release

- The process release the resource

Deadlock Characterization

Repeated University Question

Deadlock can arise if four conditions hold simultaneously.

● Mutual exclusion:

- Only one process at a time can use a resource. If another process requests, they need to wait.

● Hold and wait:

- A process holding at least one resource is waiting to acquire additional resources which is held by other processes

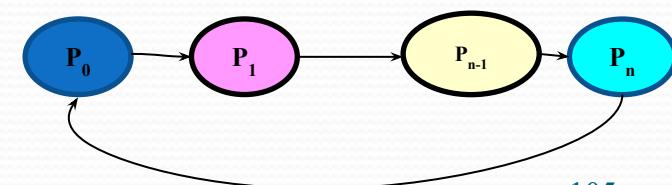
● No preemption:

- A resource can be released only voluntarily by the process holding it after that process has completed its task

● Circular wait:

- There exists a set $\{P_0, P_1, \dots, P_n\}$ of waiting processes

P_0 is waiting for a resource that is held by P_1
 P_1 is waiting for a resource that is held by P_2
 P_{n-1} is waiting for a resource that is held by P_n
 P_n is waiting for a resource that is held by P_0



Resource-Allocation Graph

Deadlocks are described in terms of directed graph called **Resource Allocation Graph**.
 Graph consists of a set of vertices V and a set of edges E .

Request edge:

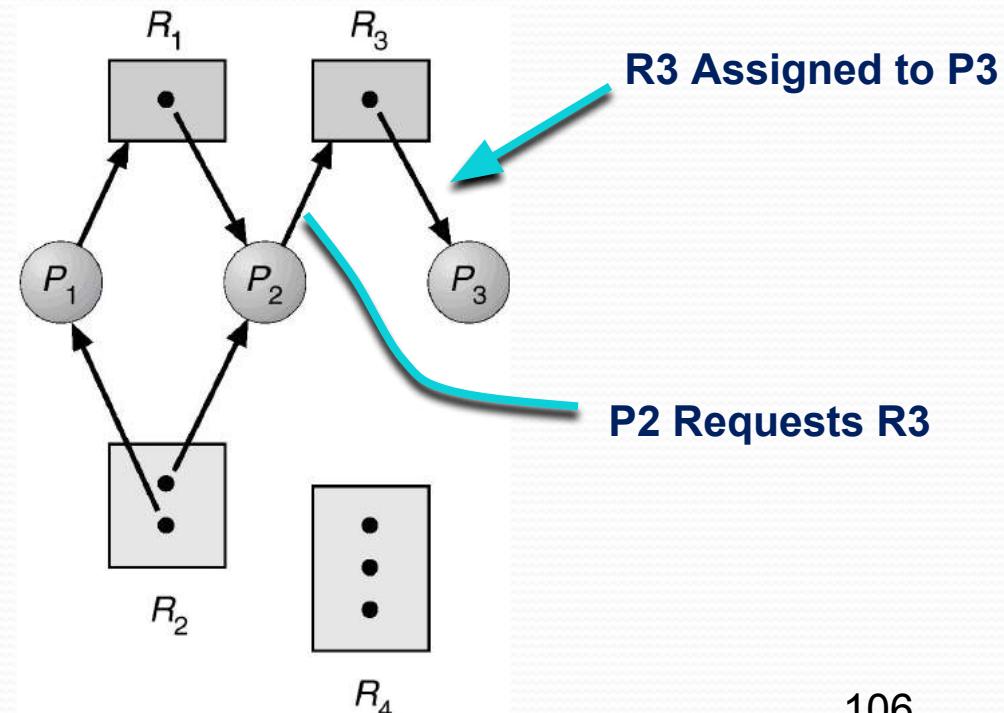
- It is a directed edge from P_1 to resource type R_j

$$P_1 \rightarrow R_j$$

Assignment edge:

- It is a directed edge from R_j to resource type P_1

$$R_j \rightarrow P_1$$



Note:

If resource type has more than 1 instance, its indicated by a dot within the rectangle.

Details

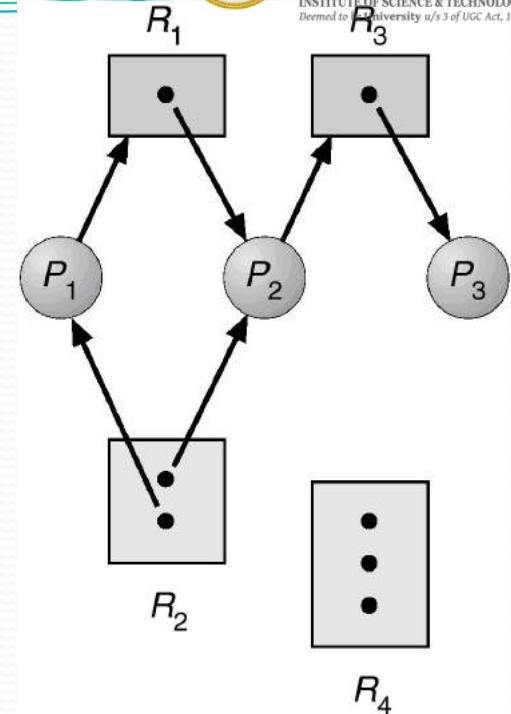
- The resource allocation graph consists of following sets:

- $P = \{ P_1, P_2, P_3 \}$
- $R = \{ R_1, R_2, R_3, R_4 \}$
- $E = \{ P_1 \sqcap R_1, P_2 \sqcap R_3, R_1 \sqcap P_2, R_2 \sqcap P_2, R_2 \sqcap P_1, R_3 \sqcap P_3 \}$

- P = Process; R = Resources; E = Edges.

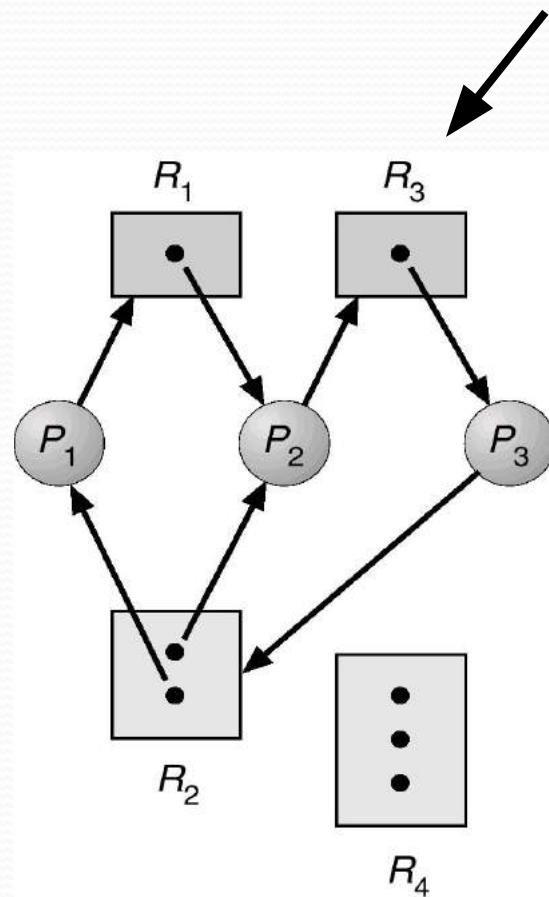
Resource Instance

- One instance of resource type R_1
- Two instances of resource type R_2
- One instance of resource type R_3
- Three instances of resource type R_4

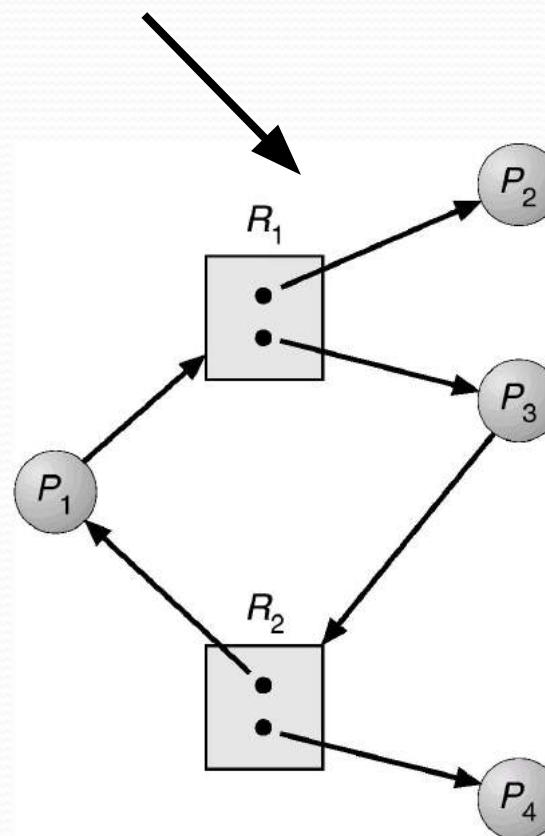


Examples

Resource allocation graph with a deadlock.



Resource allocation graph with a cycle but no deadlock.





HOW TO HANDLE DEADLOCKS ? (or)

Methods for handling deadlocks.

1 There are three methods:



Most Operating systems do this!!

Ignore Deadlocks:

2

Ensure deadlock **never** occurs using either

● **Prevention :**

- Prevent any one of the 4 conditions never happens.

● **Avoidance :**

- Allow all deadlock conditions, but calculate cycles and stop dangerous operations..

3

Allow deadlock to happen. This requires using both:

● **Detection** Know a deadlock has occurred.

● **Recovery** Regain the resources.

Deadlock Prevention

Do not allow one of the four conditions to occur.

Mutual exclusion:

- Read only files are good examples for sharable resource
 - Any number of users can access the file at the same time.
- Prevention not possible, since some devices like are non-sharable.

Hold and wait:

- Collect all resources before execution
- A sequence of resources is always collected at the beginning itself.
- Utilization is low, starvation possible.



Deadlock Prevention – Contd...

No preemption:

- If the process is holding some resources and requests another resource (that cannot be immediately allocated to it), then all the resources that the process currently holding are preempted.

Circular wait:

- $R = \{ R_1, R_2, \dots, R_m \}$ set all resource types.
- We define a function,
- For example:
 $F(\text{tape drive}) = 1$
 $F(\text{disk drive}) = 5$
 $F(\text{printer}) = 12$
- Each process requests resources in an increasing order of enumeration (ie) $F(R_j) > F(R_i)$

$$F: R \rightarrow N$$

N = natural number



Deadlock Avoidance

When we try to avoid deadlock



Utilization is less and system throughput is low

- An alternative method for avoiding deadlocks is to require additional information about how much resources are to be requested.



Safe State

NOTE: All deadlocks are unsafe, but all unsafes are NOT deadlocks.

UNSAFE

**DEADLOC
K**

SAFE

**Only with luck, the
processes avoid deadlock.**

**OS can avoid
deadlock.**



Safe State

- A system is said to be in safe state, when we allocate resources so that deadlock never occurs.
- A system is in safe state, only if there exists **safe sequence**.

Deadlock Avoidance - Example

EXAMPLE:

There exists a total of 12 resources and 3 processes.

Process	Max Needs	Allocated	Current Needs
P0	10	5	5
P1	4	2	2
P2	7	3	4

At time t0 , system is in safe state

At time t1, $\langle p_1, p_2, p_0 \rangle$ is a safe sequence.

Suppose p2 requests and is given one more resource. What happens then?



Examples

Example using one type of resource:

Initial state

Proc	Has	Max
A	3	9
B	2	4
C	2	7

Free 3 **SAFE!**

A requests 1

Proc	Has	Max
A	4	9
B	2	4
C	2	7

Free 2 **UNSAFE!**

B requests 1

Proc	Has	Max
A	3	9
B	3	4
C	2	7

Free 2 **SAFE!**

C requests 1

Proc	Has	Max
A	3	9
B	3	4
C	3	7

Free 1 **SAFE!**



Avoidance algorithms

- For a single instance of a resource type, use a Resource-allocation Graph
- For multiple instances of a resource type, use the Banker's Algorithm

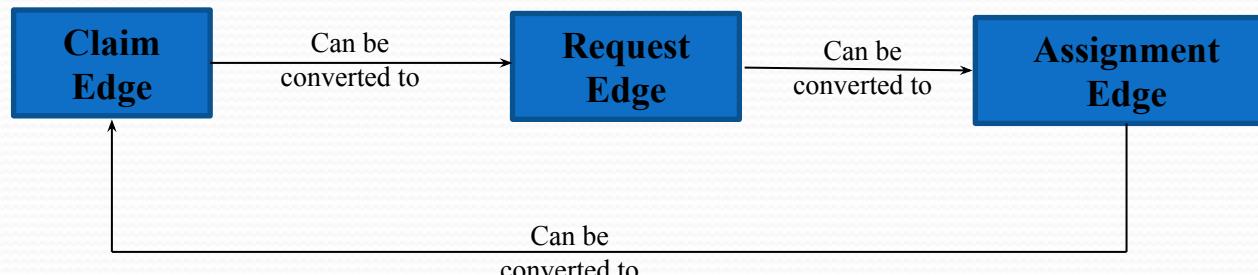


Resource-Allocation Graph

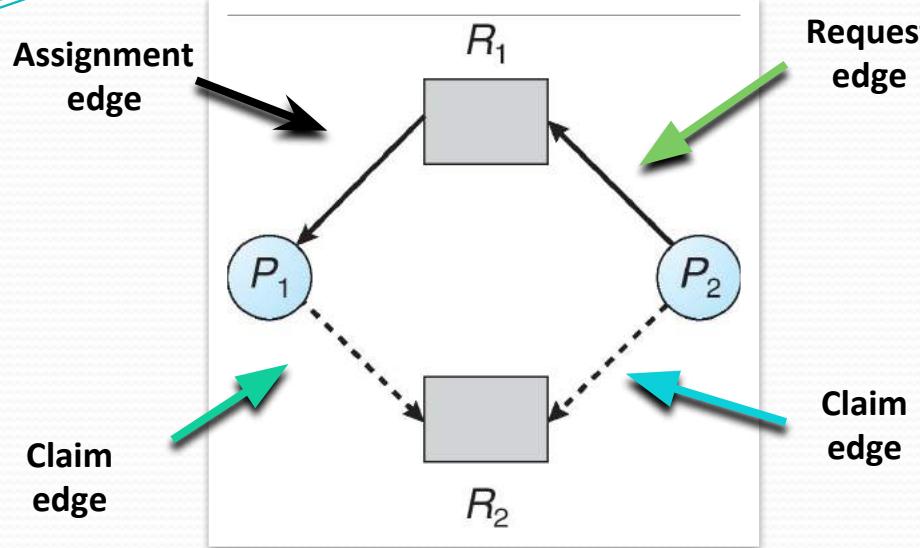
- Introduce a new kind of edge called a Claim Edge

Claim edge $P_i \xrightarrow{\text{dashed}} R_j$ indicates that process P_j may request resource R_j ;
which is represented by a dashed line.

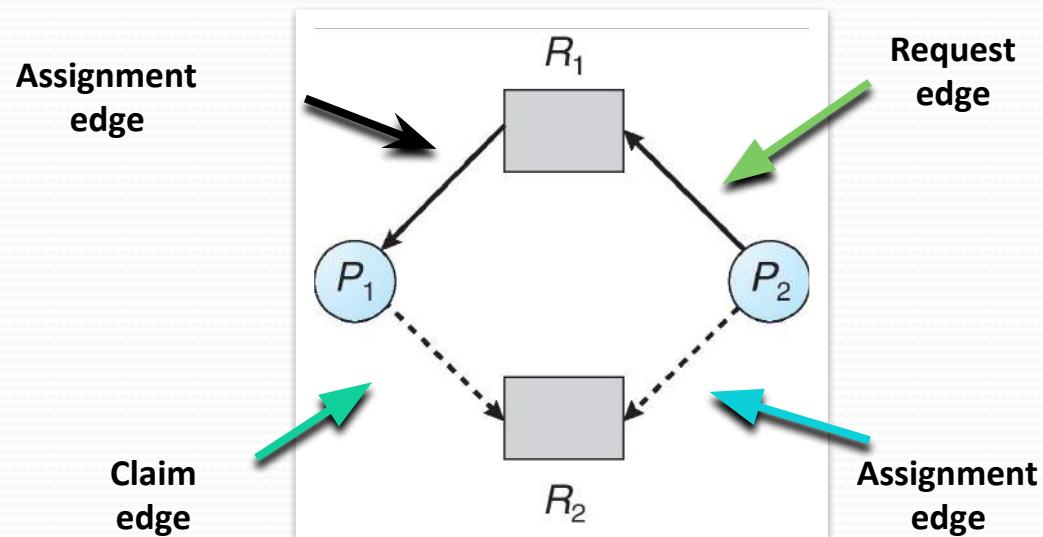
- A claim edge converts to a request edge when a process **requests** a resource
- A request edge converts to an assignment edge when the resource is **allocated** to the process
- When a resource is **released** by a process, an assignment edge reconverts to a claim edge.



Resource-Allocation Graph with Claim Edges



Unsafe State In Resource-Allocation Graph





Banker's Algorithm

- Applicable for **multiple** instances of a resource type.
 - Its less efficient than Resource-Allocation Graph

- When a process requests a resource, the system determines whether the allocation of resources will lead to safe state.
 - If it lead to safe state **allocate** resources
 - If not safe state **don't allocate** resources

Data Structures for the Banker's Algorithm

Let n = number of processes, and m = number of resources types.

- **Available**: Vector of length m . If $\text{Available}[j] = k$, there are k instances of resource type R_j available.
- **Max**: $n \times m$ matrix. If $\text{Max}[i,j] = k$, then process P_i may request at most k instances of resource type R_j .
- **Allocation**: $n \times m$ matrix. If $\text{Allocation}[i,j] = k$ then P_i is currently allocated k instances of R_j .
- **Need**: $n \times m$ matrix. If $\text{Need}[i,j] = k$, then P_i may need k more instances of R_j to complete its task.

$$\text{Need}[i,j] = \text{Max}[i,j] - \text{Allocation}[i,j]$$

Safety Algorithm

1. Let *Work* and *Finish* be vectors of length m and n , respectively.

Initialize:

$$Work = Available$$

$$Finish[i] = false \text{ for } i = 0, 1, \dots, n-1$$

2. Find an i such that both:

(a) $Finish[i] = false$

(b) $Need_i \leq Work$

If no such i exists, go to step 4

3. $Work = Work + Allocation_i$

$$Finish[i] = true$$

go to step 2

4. If $Finish[i] == \text{true}$ for all i , then the system is in a safe state

Resource-Request Algorithm for Process P_i

Request = request vector for process P_i .

If $Request_i[j] = k$ then process P_i wants k instances of resource type R_j

1. If $Request_i \leq Need_i$ go to step 2.
Otherwise error.
2. If $Request_i \leq Available$, go to step 3.
Otherwise P_i must wait, since resources are not available
3. Assume that resources are allocated:
 $Available = Available - Request;$
 $Allocation_i = Allocation_i + Request_i;$
 $Need_i = Need_i - Request_i;$



Example of Banker's Algorithm

- 5 processes P_0 through P_4 ;
3 resource types:
 A (10 instances), B (5 instances), C (7 instances)

Snapshot at time T_0 :

	<u>Allocation</u>			<u>Max</u>			<u>Available</u>		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			



Example (Cont.)

- The content of the matrix $Need$ is defined to be $Need = Max - Allocation$

Need

A B C

P_0 7 4 3

P_1 1 2 2

P_2 6 0 0

P_3 0 1 1

P_4 4 3 1

- The system is in a safe state since the sequence $\langle P_1, P_3, P_4, P_0, P_2 \rangle$ satisfies safety criteria



Example: P_1 Request (1,0,2)

- Check that Request \leq Available (ie, $(1,0,2) \leq (3,3,2)$ \Rightarrow true

	<u>Allocation</u>	<u>Need</u>	<u>Available</u>
	A B C	A B C	A B C
P_0	0 1 0	7 4 3	2 3 0
P_1		3 0 2	0 2 0
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

- Executing safety algorithm shows that sequence $< P_1, P_3, P_4, P_0, P_2 >$ satisfies safety requirement.
- Can request for (3,3,0) by P_4 be granted?
- Can request for (0,2,0) by P_0 be granted?



Deadlock Detection

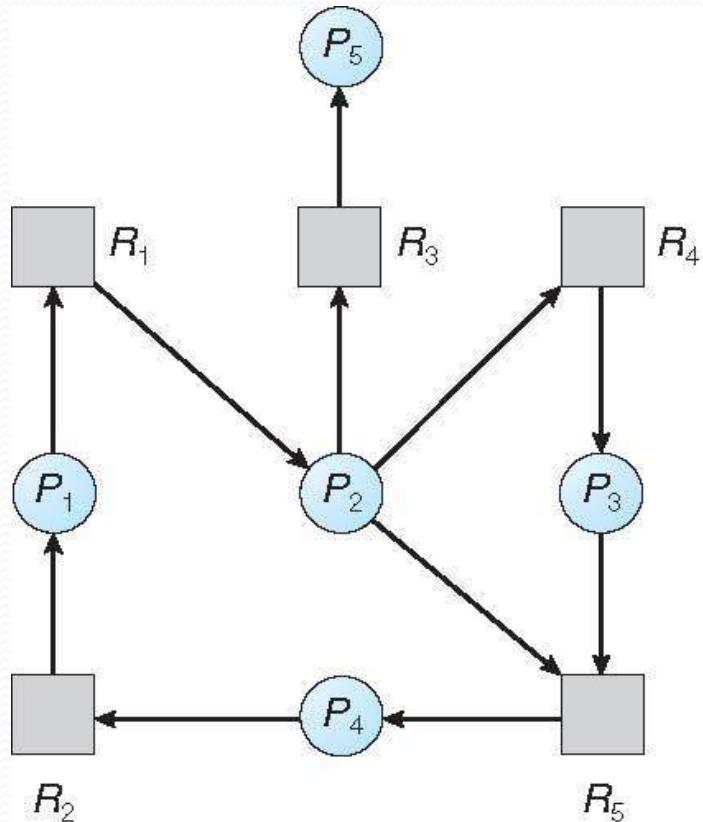
- Allow system to enter deadlock state
- Detection algorithm
- Recovery scheme



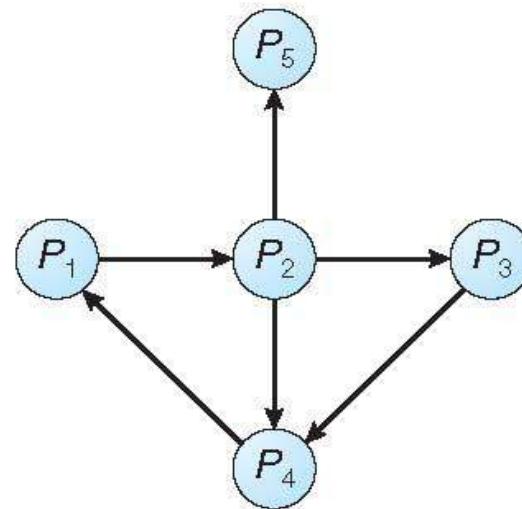
Single Instance of Each Resource Type

- Maintain **wait-for** graph
 - Nodes are processes
 - $P_i \rightarrow P_j$ if P_i is waiting for P_j
- Periodically invoke an algorithm that searches for a cycle in the graph. If there is a cycle, there exists a deadlock.
- An algorithm to detect a cycle in a graph requires an order of n^2 operations, where n is the number of vertices in the graph

Resource-Allocation Graph and Wait-for Graph



(a)



(b)

Resource-Allocation Graph

**Corresponding
wait-for graph**



Several Instances of a Resource Type

- **Available:** A vector of length m indicates the number of available resources of each type.
- **Allocation:** An $n \times m$ matrix defines the number of resources currently allocated.
- **Request:** An $n \times m$ matrix indicates the current request of each process. If $Request [i][j] = k$, then process P_i is requesting k more instances of resource type R_j .



Detection Algorithm

1. Let **Work** and **Finish** be vectors of length m and n , respectively

(a) $Work = Available$

(b) For $i = 1, 2, \dots, n$, if $Allocation_i \neq 0$, then
 $Finish[i] = \text{false}$;
otherwise, $Finish[i] = \text{true}$

2. Find an index i such that both:

(a) $Finish[i] == \text{false}$

(b) $Request_i \leq Work$

If no such i exists, go to step 4

3. $Work = Work + Allocation_i$

$Finish[i] = \text{true}$

go to step 2

4. If $Finish[i] == \text{false}$, for some i , $1 \leq i \leq n$, then the system is in deadlock state.
Moreover, P_i is also deadlocked



Example of Detection Algorithm

- Five processes P_0 through P_4 ;
- three resource types A (7 instances), B (2 instances), and C (6 instances)
- Snapshot at time T_0 :

	<u>Allocation</u>			<u>Request</u>			<u>Available</u>		
	<i>A</i>	<i>B</i>	<i>C</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>A</i>	<i>B</i>	<i>C</i>
P_0	0	1	0	0	0	0	0	0	0
P_1	2	0	0	2	0	2			
P_2	3	0	3	0	0	0			
P_3	2	1	1				1	0	0
P_4	0	0	2	0	0	2			

- Sequence $\langle P_0, P_2, P_3, P_1, P_4 \rangle$ will result in
- $Finish[i] = \text{true}$ for all i

No Deadlock



Example (Cont.)

- P_2 requests an additional instance of type C

Request

A B C

P_0 0 0 0

P_1 2 0 2

P_2 0 0 1

P_3 1 0 0

P_4 0 0 2

- State of system?

- Can reclaim resources held by process P_0 , but insufficient resources to fulfill other processes' requests
- **Deadlock exists, consisting of processes P_1 , P_2 , P_3 , and P_4**



Recovery from Deadlock

1. Process Termination

- Abort all deadlocked processes
- Abort one process at a time until the deadlock cycle is eliminated
 - In which order should we choose to abort?
 - Priority of the process
 - How long process has computed, and how much longer to completion
 - Resources the process has used
 - Resources process needs to complete
 - How many processes will need to be terminated
 - Is process interactive or batch?



Recovery from Deadlock – Contd...

2. Resource Preemption

- **Selecting a victim** – which resource or which process to be preempted? minimize cost
- **Rollback** – return to some safe state, restart process for that state ie. Rollback the process as far as necessary to break the deadlock.
- **Problem: starvation** – same process may always be picked as victim, include number of rollback in cost factor
 - Ensure that process can be picked as a victim only finite number of times.



References

Refer silberschatz, galvin “ operating system concepts” 9th edition

- CPU scheduling and policies – pg no:201-216
- Realtime and deadline –pg no: 223 to 230
- Process synchronization- pg no:253-275
- Deadlocks –pg no:311-334

(Can also refer to learning resources mentioned in the syllabus)

UNIT 3

Memory Management

Course Learning Rationale

Emphasize the importance of Memory Management concepts of an Operating system

Course Learning Outcomes

Understand the need of Memory Management functions of an Operating system

Topics

1	MEMORY MANAGEMENT: Memory Management: Logical Vs Physical address space, Swapping
	Understanding the basics of Memory management
2	Contiguous Memory allocation – Fixed and Dynamic partition
	Getting to know about Partition memory management and issues: Internal fragmentation and external fragmentation problems
3	Strategies for selecting free holes in Dynamic partition
	Understanding the allocation strategies with examples
4	Paged memory management
	Understanding the Paging technique.PMT hardware mechanism
5	Structure of Page Map Table
	Understanding the components of PMT
6	Example : Intel 32 bit and 64 –bit Architectures
	Understanding the Paging in the Intel architectures
7	Example : ARM Architectures
	Understanding the Paging with respect to ARM
8	Segmented memory management
	Understanding the users view of memory with respect to the primary memory
9	Paged segmentation Technique
	Understanding the combined scheme for efficient management- Memory Management
31-01-2021	

Basics of Memory Management

Memory : Basics

- Memory is central part for the operation of a modern computer system
- Memory consists of a large array of words or bytes, each with its own address.
- The CPU fetches instructions from memory according to the value of the program counter which may need additional loading from and storing to specific memory addresses.
- A typical instruction-execution cycle, first fetches an instruction from memory. The instruction is then decoded and may cause operands to be fetched from memory.
- After the instruction has been executed on the operands, results may be stored back in memory.

Memory : Basics (Cont.)

- The memory unit sees only a
 - Stream of memory addresses;
 - It does not know how they are generated (by the instruction counter, indexing, indirection, literal addresses, and so on) or
 - What they are for (instructions or data).
- Hence we can ignore *how* a program generates a memory address.
- We are interested only in the sequence of memory addresses generated by the running program.

Basic Hardware

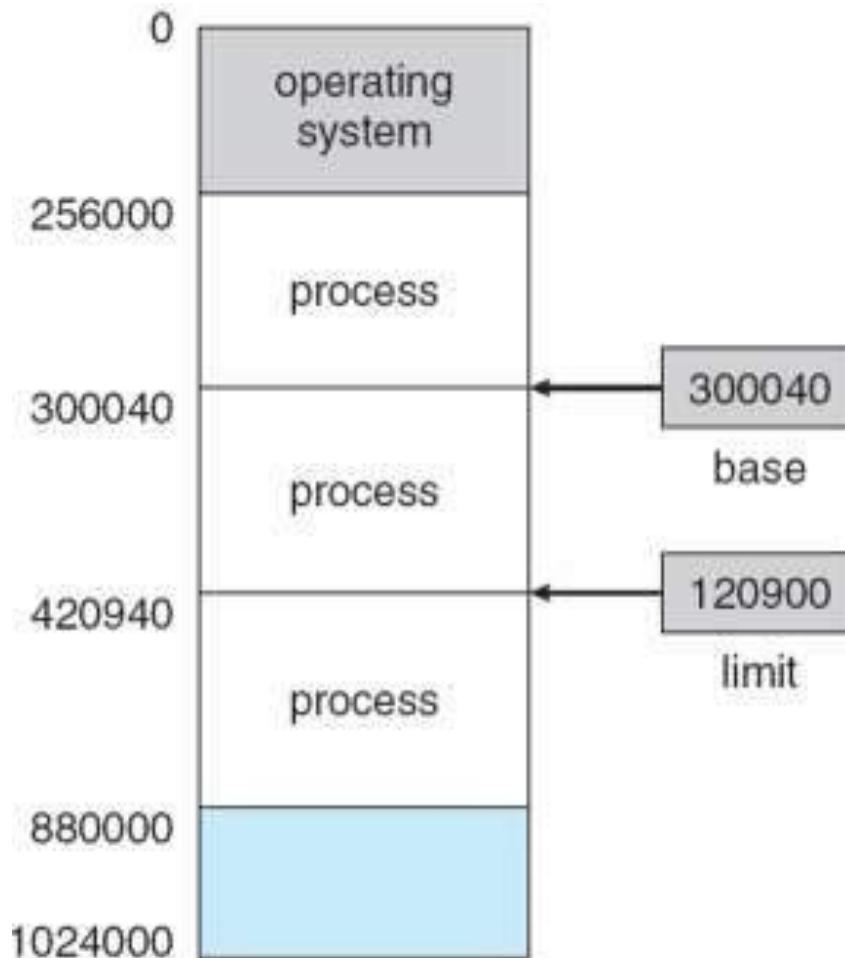
- Program must be brought (from disk) into memory and placed within a process for it to be run
- Main memory and registers are only storage CPU can access directly
- Memory unit only sees a stream of addresses + read requests, or address + data and write requests
- Register access in one CPU clock (or less)
- Main memory can take many cycles, causing a **stall Cache** sits between main memory and CPU registers
- Protection of memory required to ensure correct operation

A Base and a Limit Register

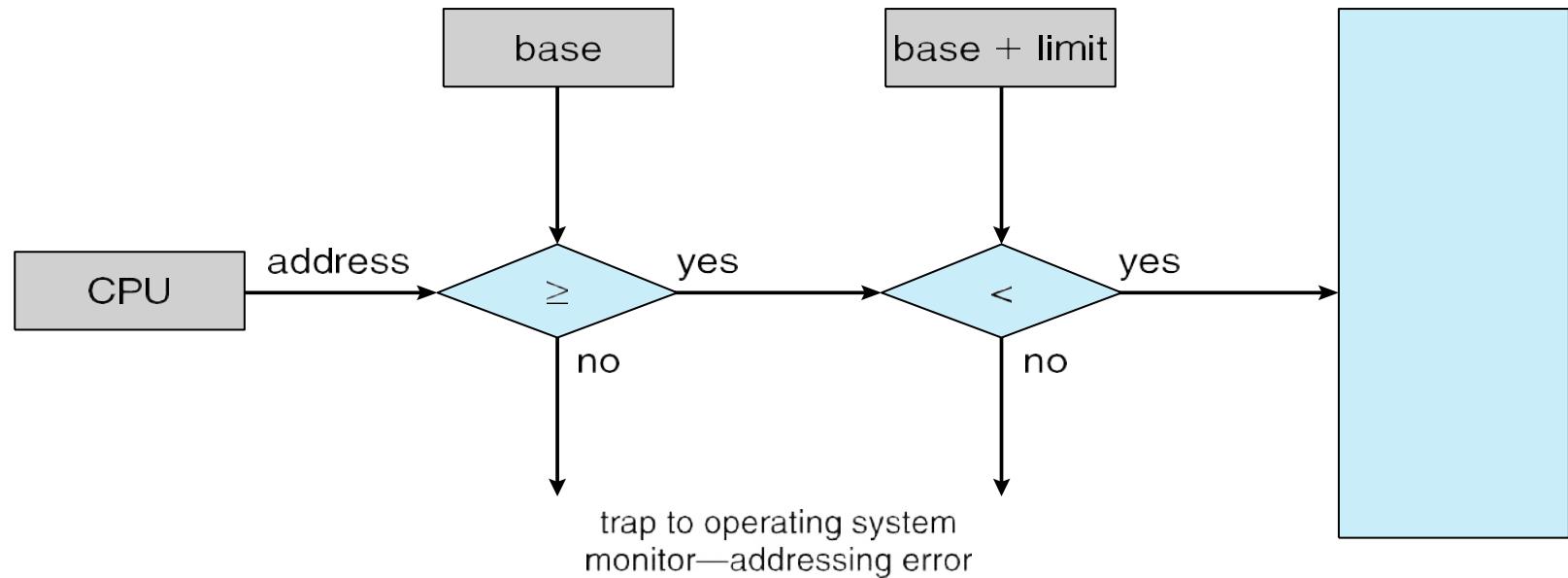
- Each process should have a separate memory space.
- To do this, we need the ability to determine the range of legal addresses that the process may access and to ensure that the process can access only these legal addresses.
- Operating system provide this protection by using two registers:
 - a base and
 - a limit
- The **base register** holds the smallest legal physical memory address
- The **limit register** specifies the size of the range

A Base and a Limit Register

- For example, if the base register holds 300040 and
- The limit register is 120900, then the program can legally access all addresses from 300040 through 420939 (inclusive).

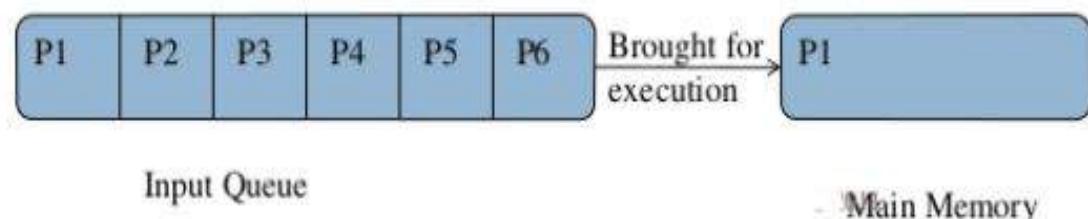


Hardware Address Protection



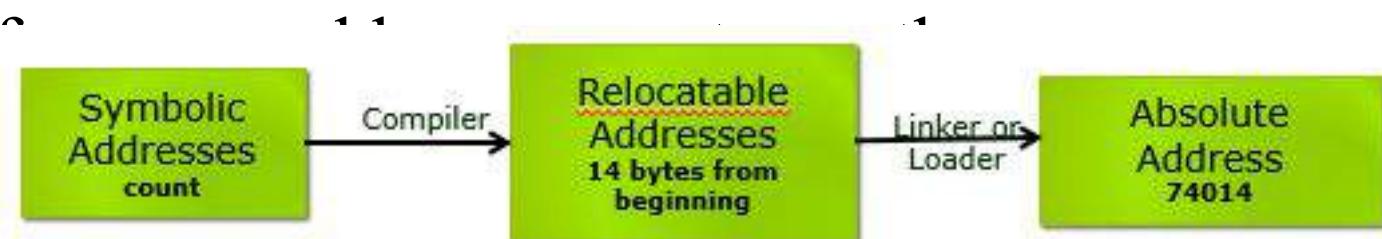
Address Binding

- Usually, a program resides on a disk as a binary executable file.
- The program to be executed must be brought into memory and placed within a process.
- Depending on the memory management in use, the process may be moved between disk and memory during its execution.
- The processes on the disk that are waiting to be brought into memory for execution form the **input queue**.



Address Binding

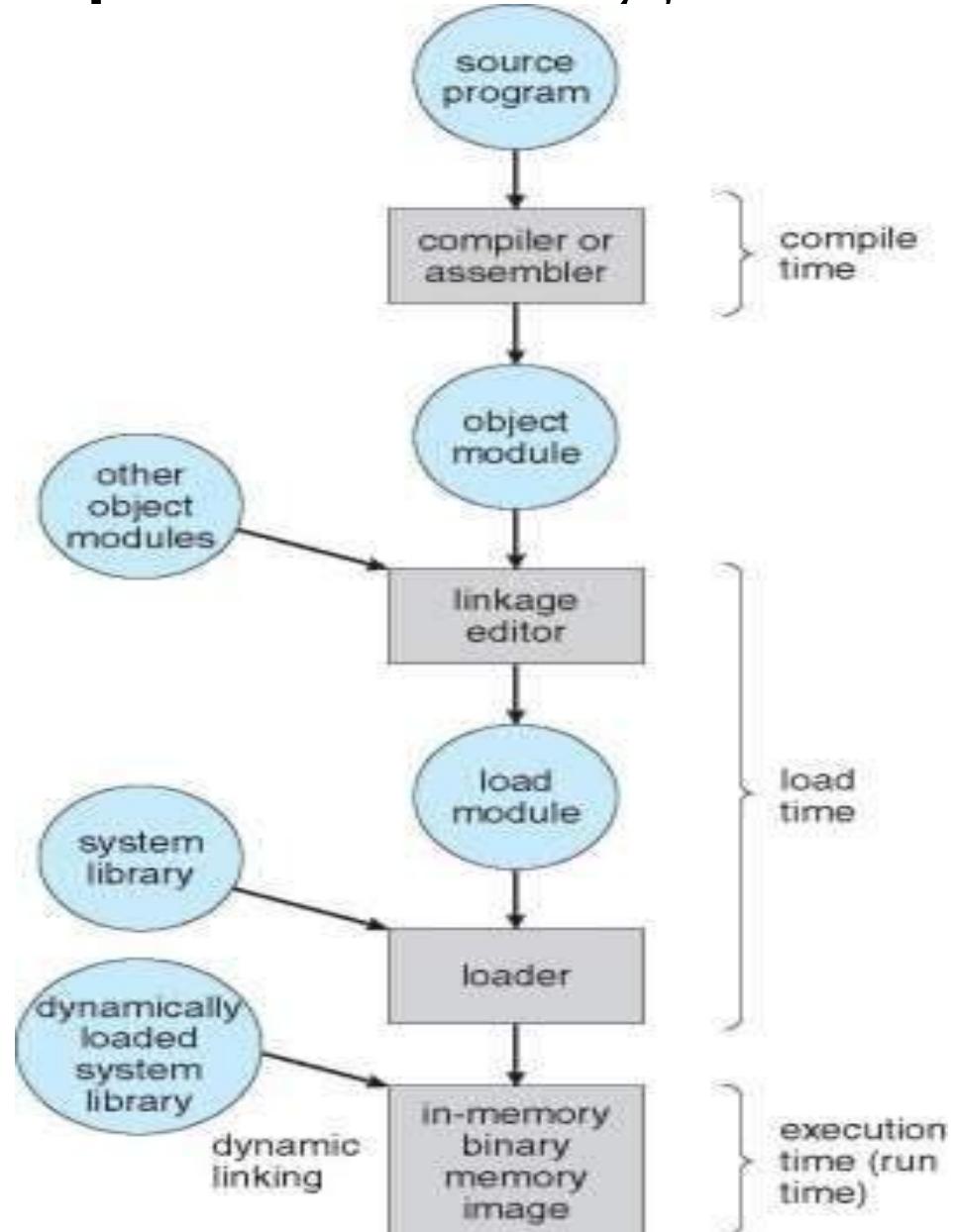
- Addresses may be represented in different ways during the execution of the program
 - Addresses in the source program are generally symbolic (such as *count*)
 - A compiler will typically **bind** these symbolic addresses to relocatable addresses (such as “14 bytes from the beginning of this module”)
- The linkage editor or loader will in turn bind the relocatable addresses to absolute addresses (such as 74014)
- Each binding is a mapping ^



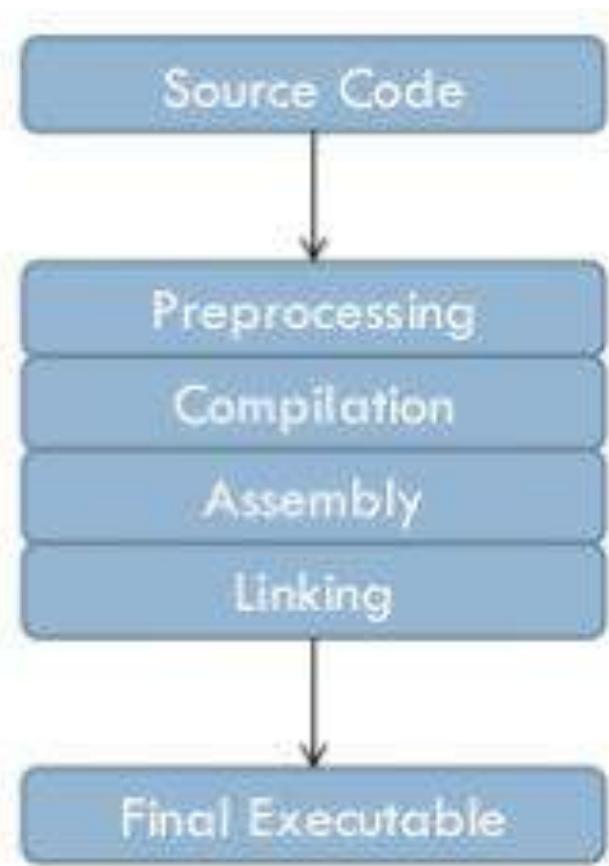
Binding of Instructions and Data to Memory

- Address binding of instructions and data to memory addresses can happen at three different stages
 - **Compile time:** If memory location known a priori, **absolute code** can be generated; must recompile code if starting location changes
 - **Load time:** Must generate **relocatable code** if memory location is not known at compile time
 - **Execution time:** Binding delayed until run time if the process can be moved during its execution from one memory segment to another
 - Need hardware support for address maps (e.g., base and limit registers)

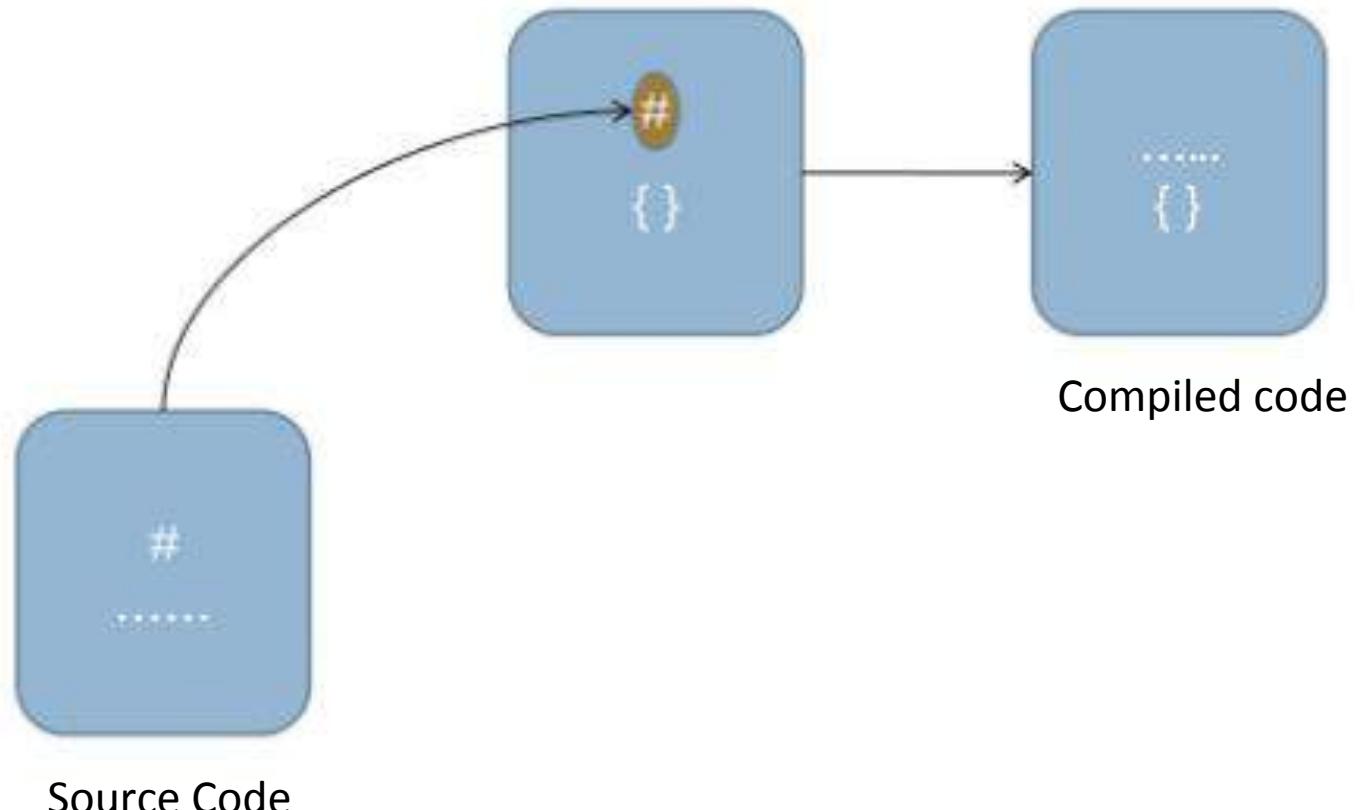
Multistep Processing of a User Program



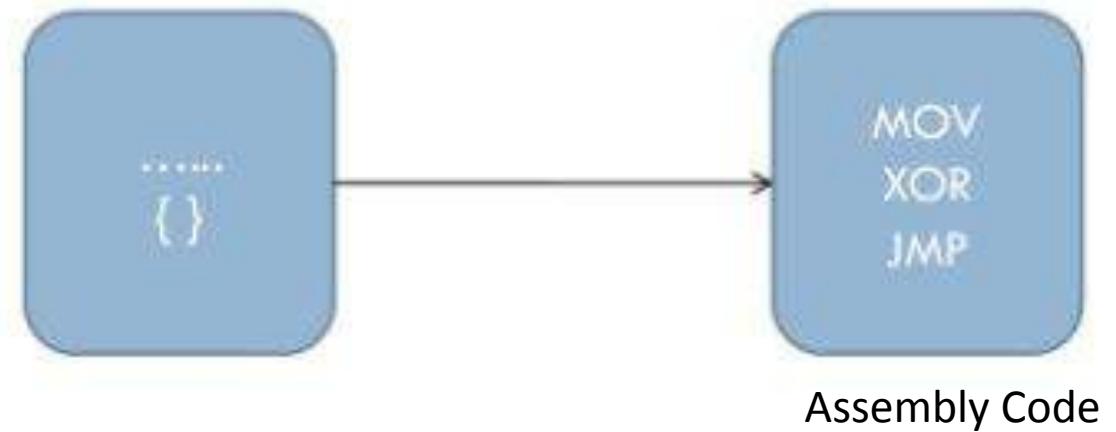
Program Execution



Compilation



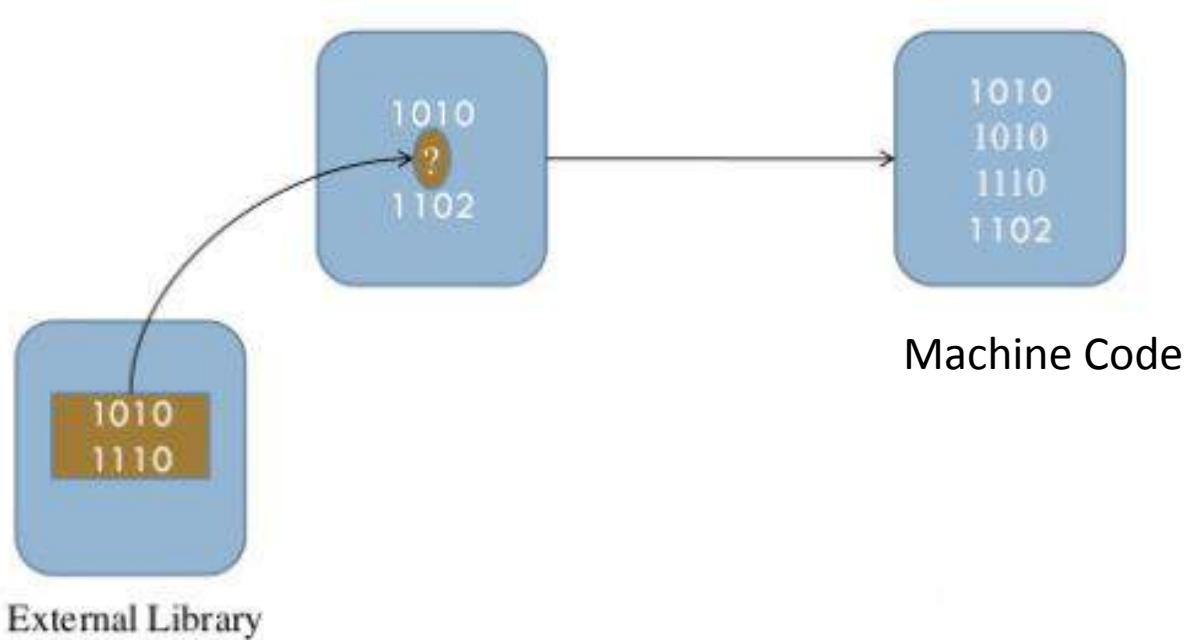
Assembler



Assembler Leaves the address of external function undefined



Linker/Loader



Logical vs Physical Address Space

- An address generated by the CPU is commonly referred to as a **logical address**
- An address seen by the memory unit—that is, the one loaded into the **memory-address register** of the memory—is commonly referred to as a **physical address**.
- The compile-time and load-time address-binding methods generate identical logical and physical addresses.
- However, the execution-time address binding scheme results in differing logical and physical addresses.

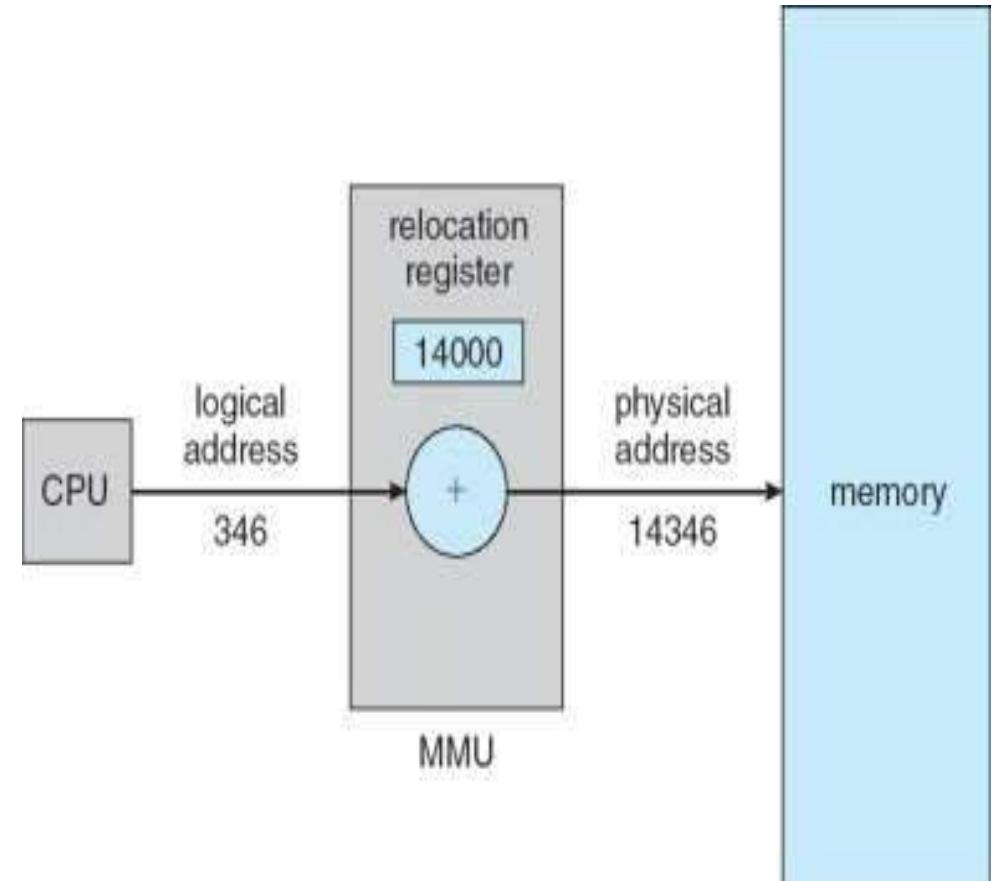


Logical vs Physical Address Space (Cont.)

- In this case, we usually refer to the logical address as a **virtual address**.
- The set of all logical addresses generated by a program is a **logical address space**;
- The set of all physical addresses corresponding to these logical addresses is a **physical address space**.
- The run-time mapping from virtual to physical addresses is done by a hardware device called the **memory-management unit (MMU)**.

Dynamic Relocation using a Relocation Register

- The base register is now called a **Relocation Register**.
- The value in the relocation register is *added* to every address generated by a user process at the time the address is sent to memory.
- For example, if the base is at 14000, then an attempt by the user to address location 0 is dynamically relocated to location 14000;
- an access to location 346 is mapped to location 14346.

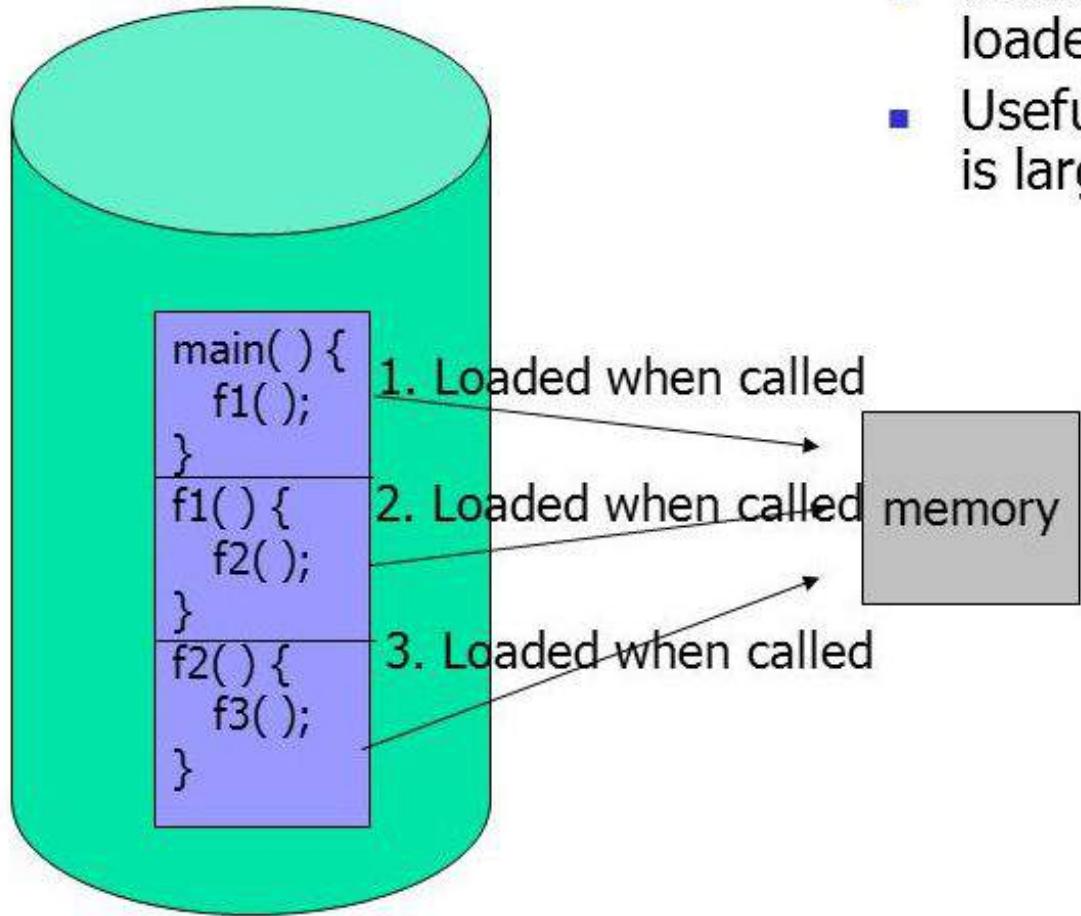


Logical vs Physical Address Space (Cont.)

- The two different types of addresses:
 - logical addresses (in the range 0 to max) and
 - physical addresses (in the range $R + 0$ to $R + max$ for a base value R)
- The user generates only logical addresses and thinks that the process runs in locations 0 to max .
- The user program generates only logical addresses and thinks that the process runs in locations 0 to max .
- However, these logical addresses must be mapped to physical addresses before they are used.

Dynamic Loading

- Unused routine is never loaded
- Useful when the code size is large

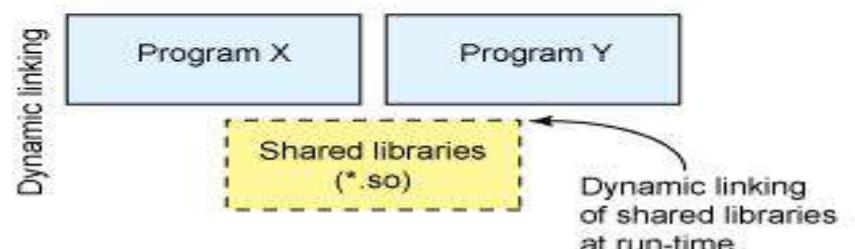
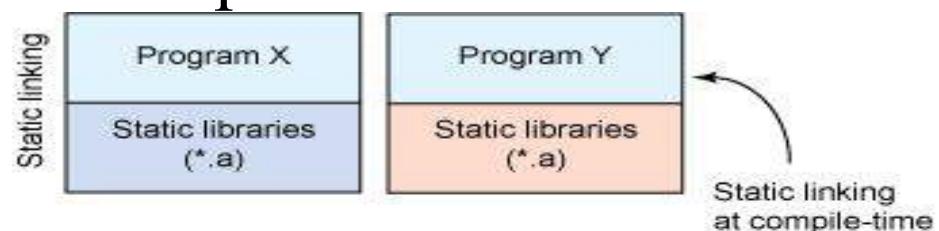


Dynamic linking

- Static linking
- System libraries and program code separately
- Some OS support only static linking

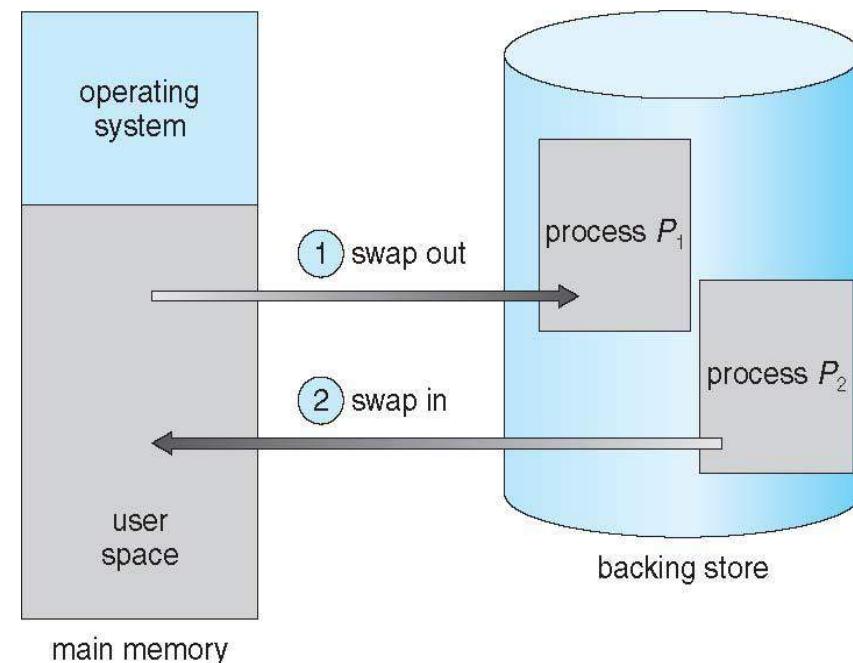
- Dynamic linking

- Linking is postponed until execution time
- Wastage of both disk space and main memory is avoided



Swapping

- A process can be **swapped** temporarily out of memory to a **backing store**, and then brought back into memory for continued execution
 - Total physical memory space of processes can exceed physical memory
 - This increases the degree of multiprogramming in a system
 - **Backing store** – Normally its hard disk



CONTIGUOUS MEMORY ALLOCATION

Contiguous Allocation

- The main memory must accommodate both the operating system and the various user processes.
- The memory is usually divided into two partitions:
 1. one for the resident operating system
 2. one for the user processes.

Contiguous Allocation (Cont.)

- Several user processes want to reside in memory at the same time.
- Then it is to be considered, how to allocate available memory to the processes that are in the input queue waiting to be brought into memory.
- In **contiguous memory allocation**, each process is contained in a single section of memory that is contiguous to the section containing the next process.

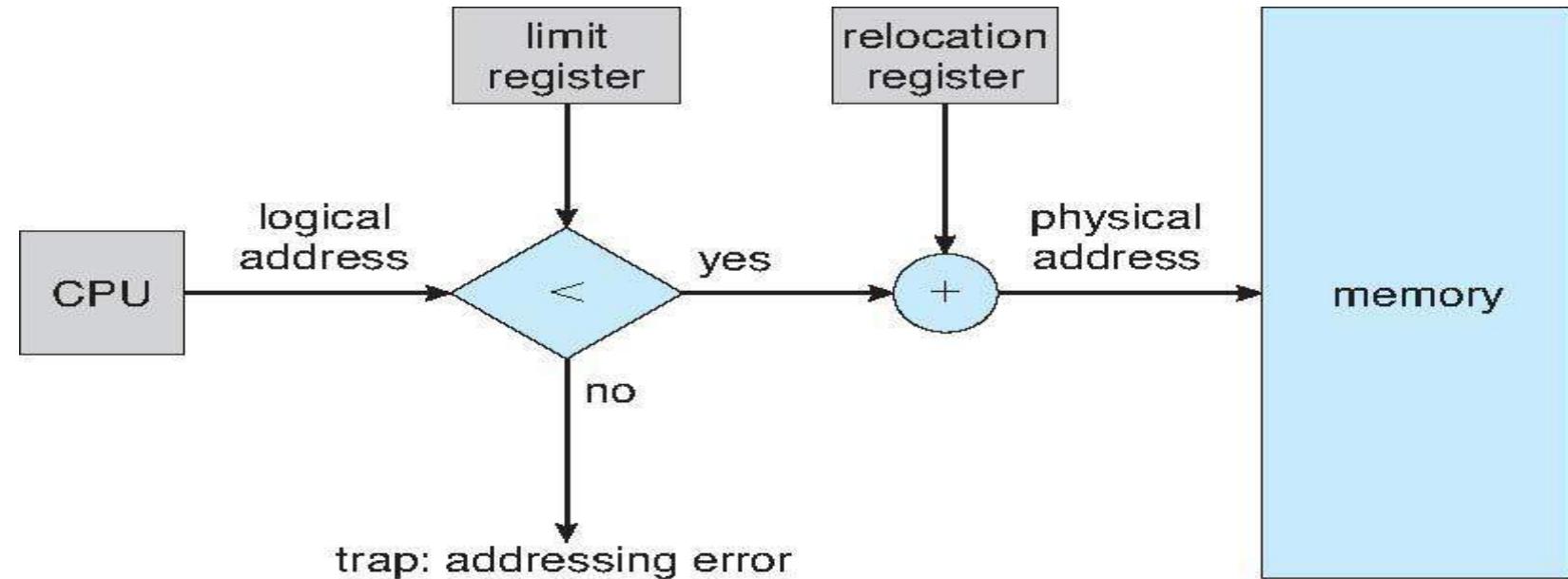
Contiguous Allocation (Cont.)

- The relocation register contains the value of the smallest physical address;
- The limit register contains the range of logical addresses (for example, relocation = 100040 and limit = 74600).
- Each logical address must fall within the range specified by the limit register.
- The MMU maps the logical address dynamically by adding the value in the relocation register.

Contiguous Allocation (Cont.)

- This mapped address is sent to memory. When the CPU scheduler selects a process for execution, the dispatcher loads the relocation and limit registers with the correct values as part of the context switch.
- Because every address generated by a CPU is checked against these registers, we can protect both the operating system and the other users programs and data from being modified by this running process.

Hardware Support for Relocation and Limit Registers



Contiguous Allocation (Cont.)

- The relocation-register scheme provides an effective way to allow the operating system's size to change dynamically. This flexibility is desirable in many situations.
- For example, the operating system contains code and buffer space for device drivers.

Contiguous Allocation (Cont.)

- If a device driver (or other operating-system service) is not commonly used, we do not want to keep the code and data in memory, as we might be able to use that space for other purposes.
- Such code is sometimes called **transient** operating-system code; it comes and goes as needed. Thus, using this code changes the size of the operating system during program execution.

Contiguous Allocation Technique

- There are two popular techniques used for contiguous memory allocation

□ Fixed Partitioning

□ Variable Partitioning

Fixed Partitioning

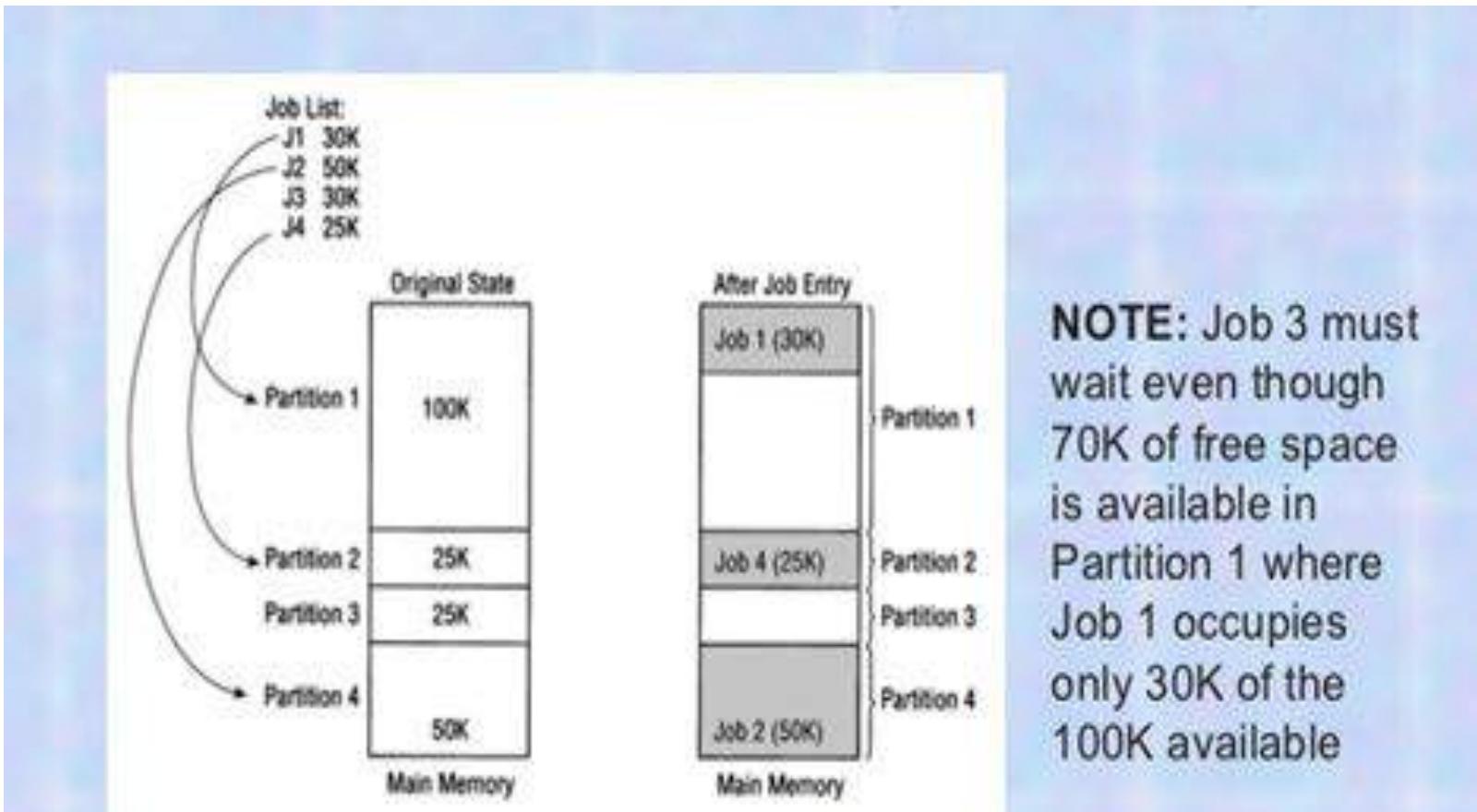
- Static partitioning is a fixed partitioning scheme.
- In this partitioning, **number of partitions** (non-overlapping) in RAM are **fixed** but **size of each partition may or may not be same.**
- As it is **contiguous** allocation, hence no spanning is allowed.
- Here partition are made before execution or during system configure.
- Each partition is allowed to store only one process.
- These partitions are allocated to the processes as they arrive.

Fixed Partitioning (Cont.)

- The degree of multiprogramming is bound by the number of partitions.
- In this **multiple partition method**, when a partition is free, a process is selected from the input queue and is loaded into the free partition.
- When the process terminates, the partition becomes available for another process.

Fixed Partitioning (Cont.)

Example



Disadvantages of Fixed Partitioning

- **Limitation on the Size of the Process:** – Sometimes, when the size of the process is larger than the maximum partition size, then we cannot load that process into the memory. So, this is the main disadvantage of the fixed partition.
- **Degree of Multiprogramming is Less:** – We can understand from the degree of multiprogramming that it means at the same time, the maximum number of processes we can load into the memory. In Fixed Partitioning, the size of the partition is fixed, and we cannot vary it according to the process size; therefore, in fixed partitioning, the degree of multiprogramming is less and fixed.
- **Internal Fragmentation**

Let us first discuss about what is fragmentation and then discuss about Internal Fragmentation

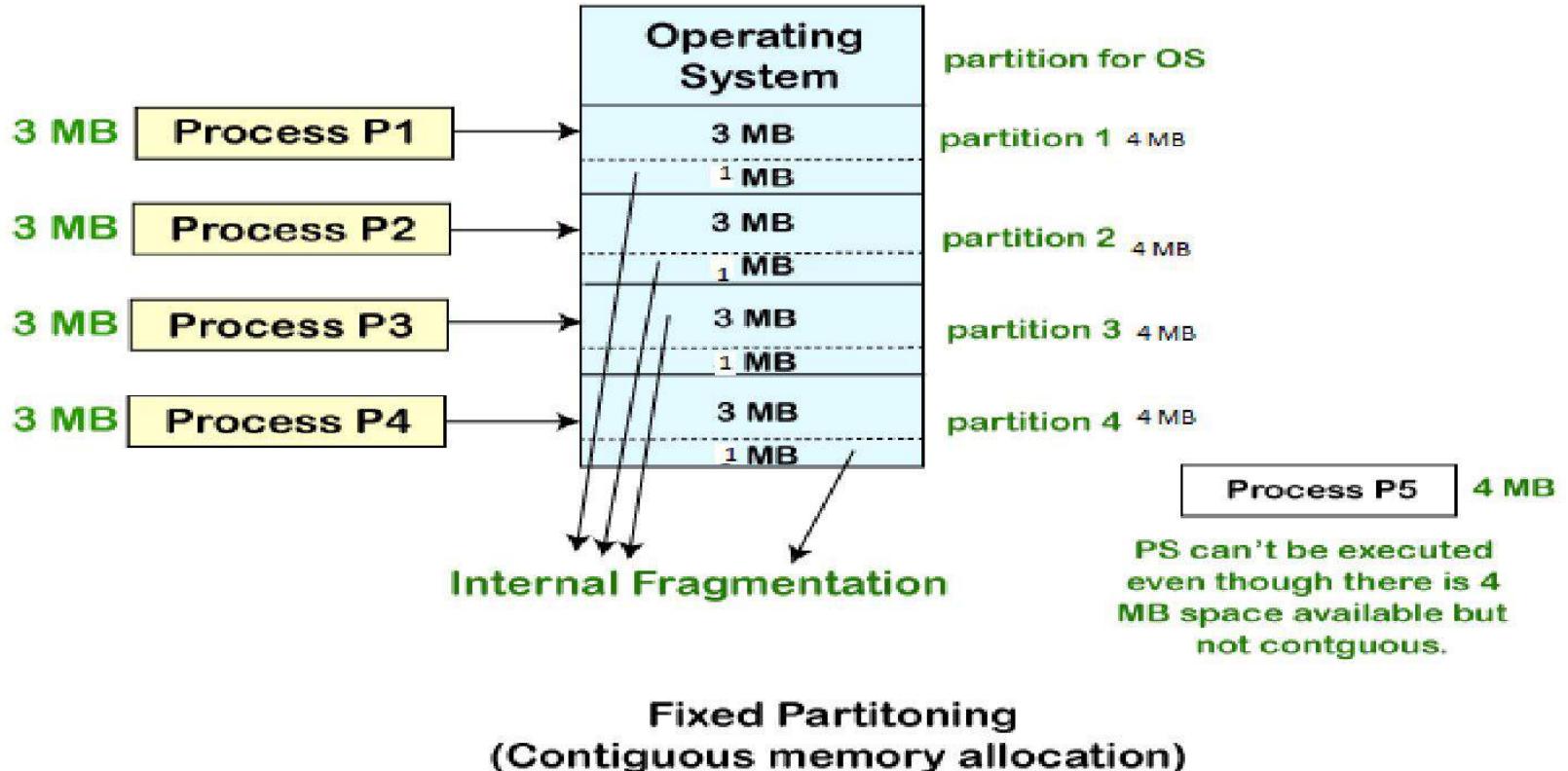
Fragmentation

- Fragmentation is an unwanted problem where the memory blocks cannot be allocated to the processes due to their small size and the blocks remain unused.
- It can also be understood as when the processes are loaded and removed from the memory they create free space or hole in the memory and these small blocks cannot be allocated to new upcoming processes and results in inefficient use of memory.
- Basically, there are two types of fragmentation:
 - Internal Fragmentation
 - External Fragmentation

Internal Fragmentation

- It occurs when the space is left inside the partition after allocating the partition to a process.
- This space is called as internally fragmented space.
- This space can not be allocated to any other process.
- This is because only Fixed (static) partitioning allows to store only one process in each partition.
- Internal Fragmentation occurs only in Fixed (static) partitioning .

Internal Fragmentation (Cont.)



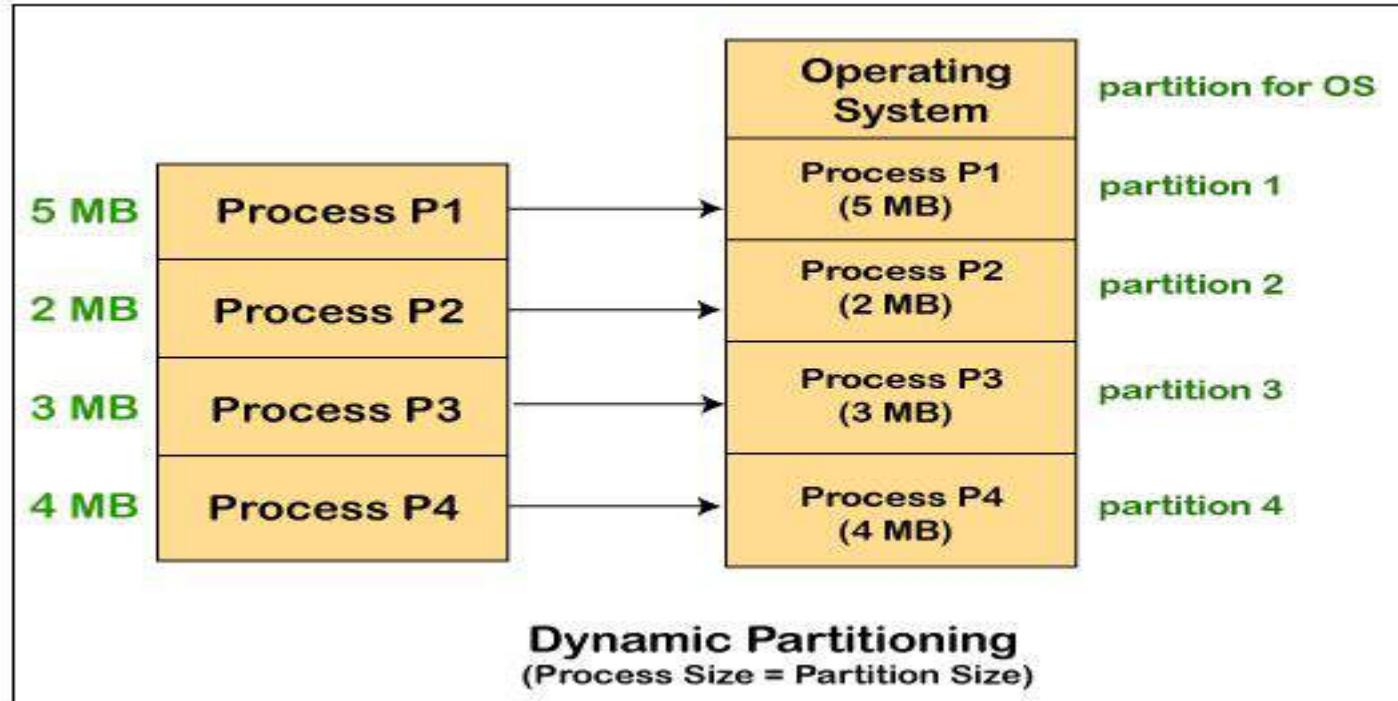
Solution to Internal Fragmentation

- This problem is occurring because we have fixed the sizes of the memory blocks. This problem can be removed if we use dynamic partitioning for allocating space to the process.
- In dynamic partitioning, the process is allocated only that much amount of space which is required by the process. So, there is no internal fragmentation.

Variable - Partitioning

- In the variable-partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied.
- Initially, all memory is available for user processes and is considered one large block of available memory, a hole.
- Eventually, memory contains a set of holes of various sizes.
- As processes enter the system, they are put into an input queue.

Variable – Partitioning (Cont.)



Variable – Partitioning (Cont.)

- The operating system takes into account the memory requirements of each process and the amount of available memory space in determining which processes are allocated memory.
- When a process is allocated space, it is loaded into memory, and it can then compete for CPU time.
- When a process terminates, it releases its memory, which the operating system may then fill with another process from the input queue.

Variable – Partitioning (Cont.)

- At any given time, then, we have a list of available block sizes and an input queue. The operating system can order the input queue according to a scheduling algorithm. Memory is allocated to processes until, finally, the memory requirements of the next process cannot be satisfied — that is, no available block of memory (or hole) is large enough to hold that process.

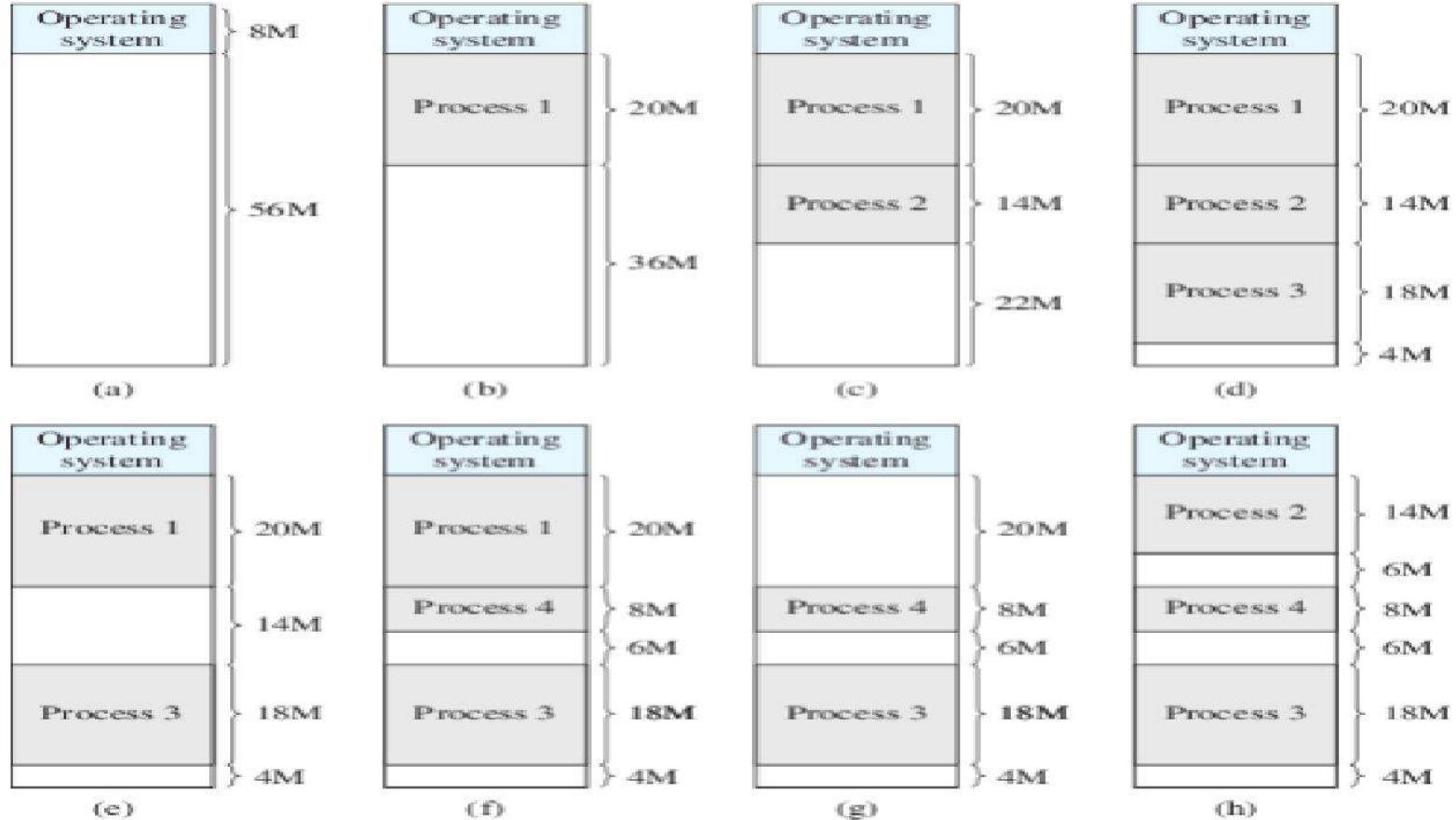
Variable – Partitioning (Cont.)

- The operating system can then wait until a large enough block is available, or it can skip down the input queue to see whether the smaller memory requirements of some other process can be met.
- In general, as mentioned, the memory blocks available comprise a set of holes of various sizes scattered throughout memory. When a process arrives and needs memory, the system searches the set for a hole that is large enough for this process.

Variable – Partitioning (Cont.)

- If the hole is too large, it is split into two parts. One part is allocated to the arriving process; the other is returned to the set of holes.
- When a process terminates, it releases its block of memory, which is then placed back in the set of holes.
- If the new hole is adjacent to other holes, these adjacent holes are merged to form one larger hole.
- At this point, the system may need to check whether there are processes waiting for memory and whether this newly freed and recombined memory could satisfy the demands of any of these waiting processes.

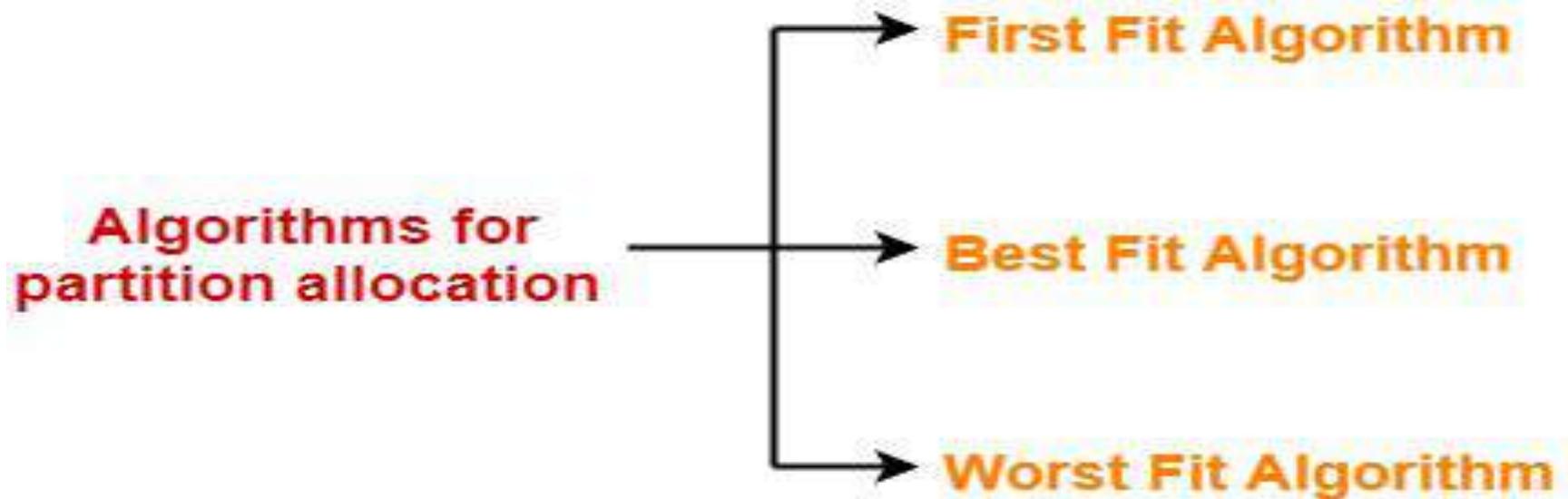
Variable – Partitioning (Cont.)



Variable – Partitioning (Cont.)

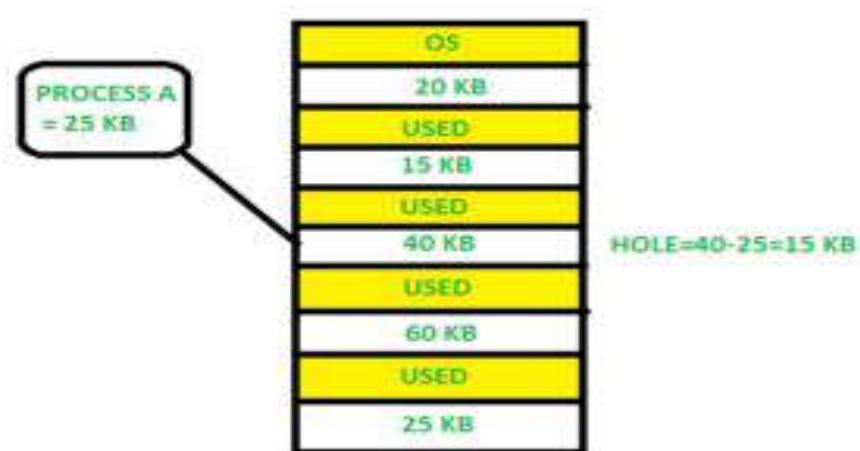
- This procedure is a particular instance of the general **dynamic storage-allocation problem**, which concerns how to satisfy a request of size n from a list of free holes.
- There are many solutions to this problem. The **first-fit**, **best-fit**, and **worst-fit** strategies are the ones most commonly used to select a free hole from the set of available holes.

Partition Allocation Algorithm



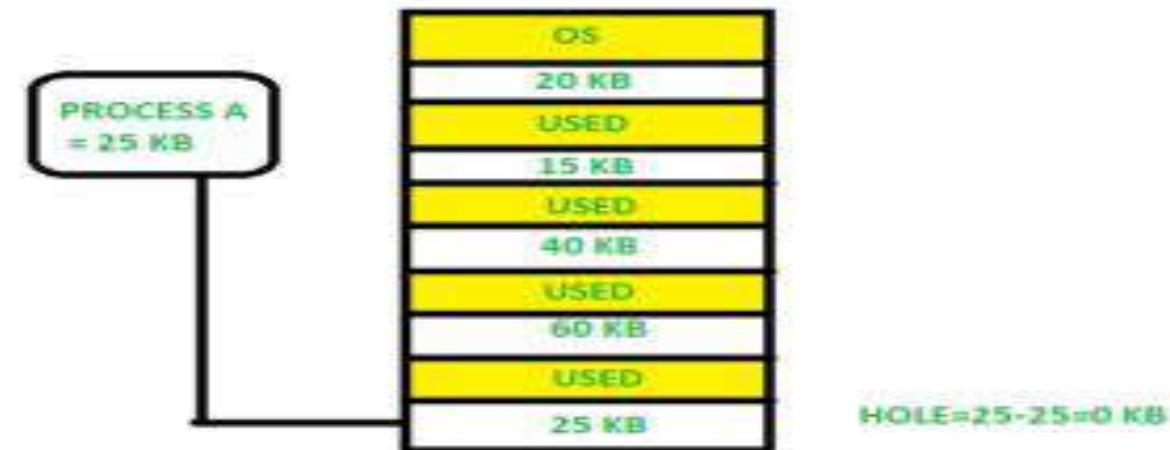
First Fit

- Allocate the first hole that is big enough.
- Searching can start either at the beginning of the set of holes or at the location where the previous first-fit search ended.
- We can stop searching as soon as we find a free hole that is large enough.



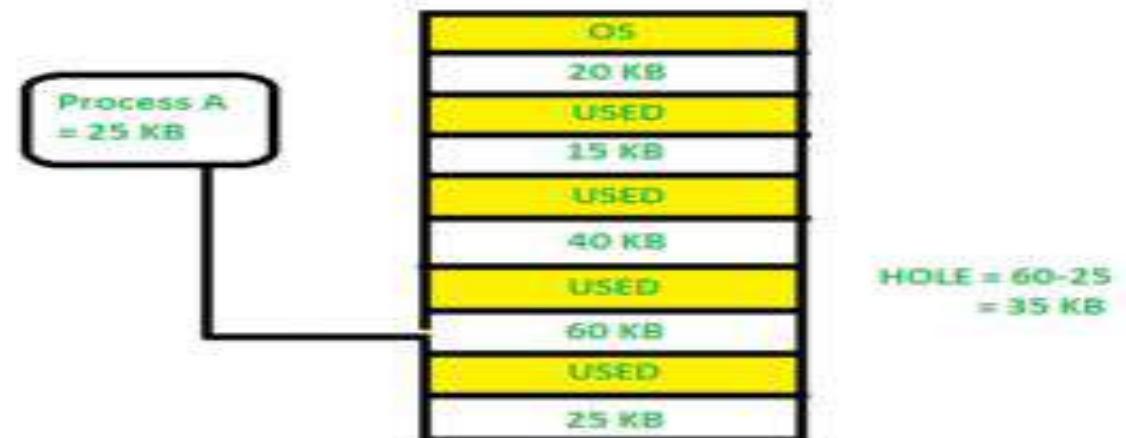
Best Fit

- Allocate the smallest hole that is big enough.
- We must search the entire list, unless the list is ordered by size.
- This strategy produces the smallest leftover hole.



Worst Fit

- Allocate the largest hole.
- Again, we must search the entire list, unless it is sorted by size.
- This strategy produces the largest leftover hole which may be useful for the other process



Problem to Solve

Question

Given five memory partitions of 100Kb, 500Kb, 200Kb, 300Kb, 600Kb (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of 212 Kb, 417 Kb, 112 Kb, and 426 Kb (in order)? Which algorithm makes the most efficient use of memory?

Steps to Solve – First Fit

Memory partitions: 100Kb, 500Kb, 200Kb, 300Kb, 600Kb



Processes: 212 Kb, 417 Kb, 112 Kb, and 426 Kb

First Fit

212 Kb is put in 500 Kb partition



417 Kb is put in 600 Kb partition



112 Kb is put in 288 Kb partition (new partition 288 Kb = 500 Kb - 212 Kb)



426 Kb must wait

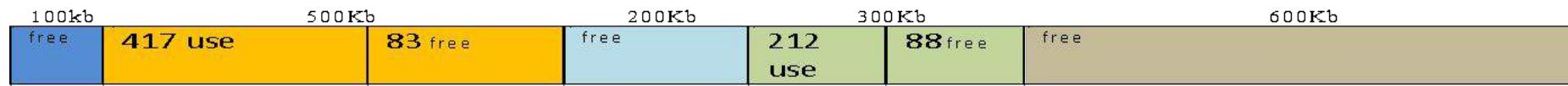
Steps to Solve – Best Fit

Best Fit

212 Kb is put in 300Kb partition



417Kb is put in 500Kb partition



112Kb is put in 200Kb partition



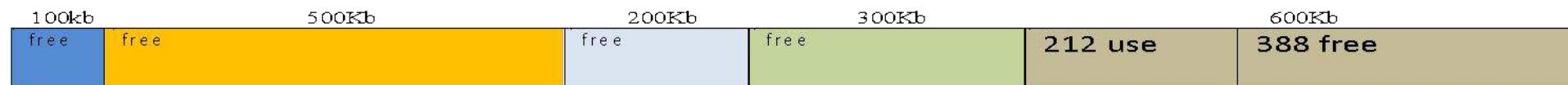
426Kb is put in 600Kb partition



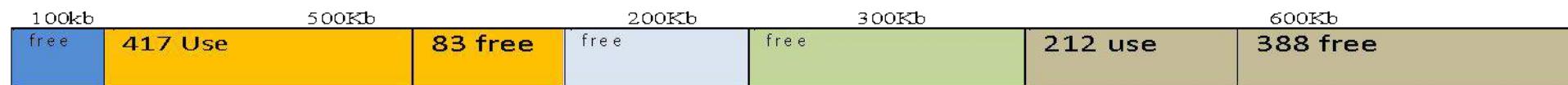
Steps to Solve – Worst Fit

Worst Fit

212Kb is put in 600Kb partition



417Kb is put in 500Kb partition



112Kb is put in 388Kb partition



426Kb must wait

Disadvantages of Variable - Partitioning

- Complex Memory Allocation
- External Fragmentation

Complex Memory Allocation

- The task of allocation and deallocation is tough because the size of the partition varies whenever the partition is allocated to the new process. The operating system has to keep track of each of the partitions.
- So, due to the difficulty of allocation and deallocation in the dynamic memory allocation, and every time we have to change the size of the partition; therefore, it is tough for the operating system to handle everything.

External Fragmentation

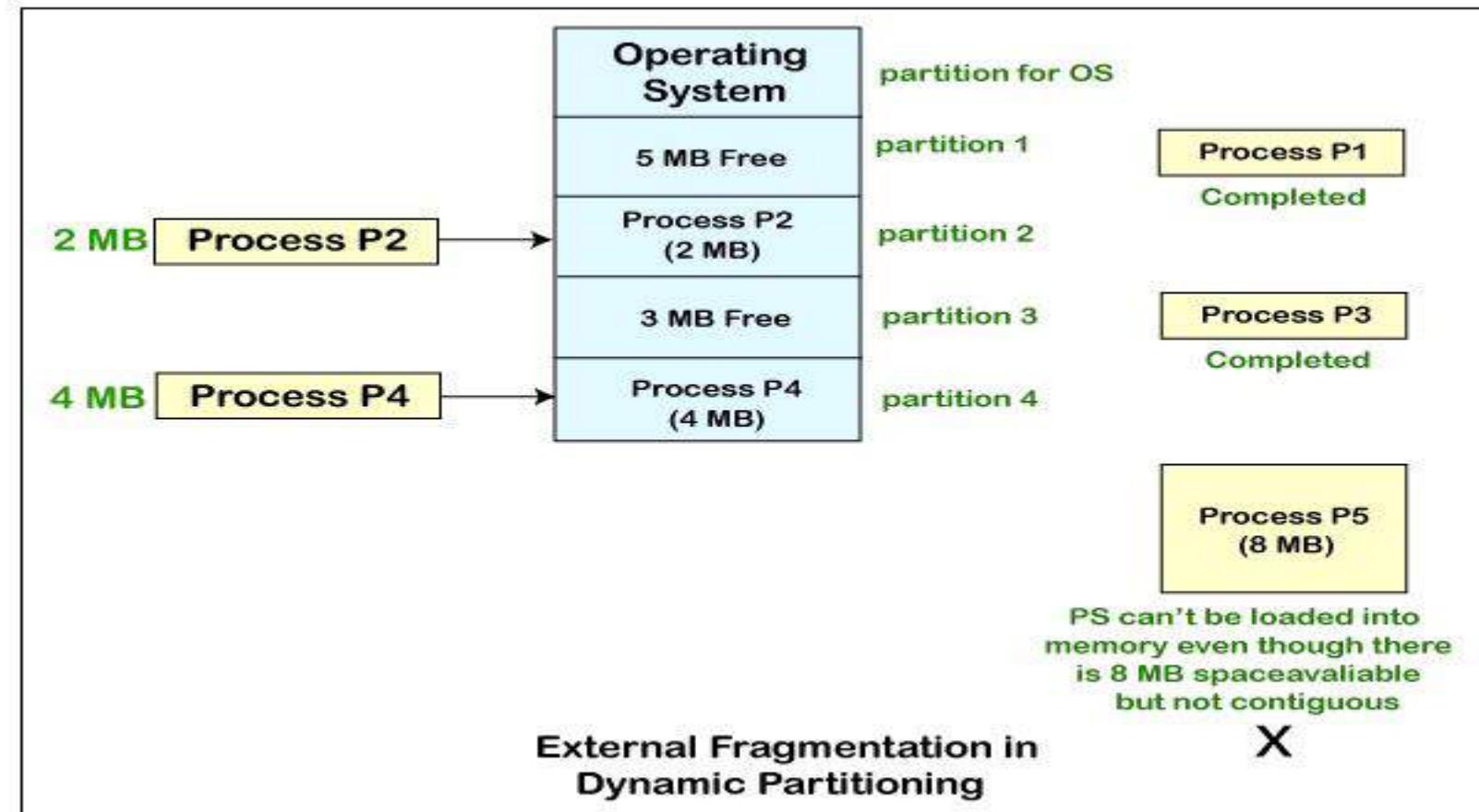
- It occurs when the total amount of empty space required to store the process is available in the main memory.
- But because the space is not contiguous, so the process can not be stored.

External Fragmentation (Cont.)

- Consider, three processes P1 (1MB), P2 (3MB), and P3 (1MB), and we want to load the processes in the various partitions of the main memory.
- Now the processes P1 and P3 are completed, and space that is assigned to the process P1 and P3 is freed. Now we have partitions of 1 MB each, which are unused and present in the main memory, but we cannot use this space to load the process of 2 MB in the memory because space is not contiguous.
- The rule says that we can load the process into the memory only when it is contiguously residing in the main memory. So, if we want to avoid external fragmentation, then we have to change this rule.

External Fragmentation (Cont.)

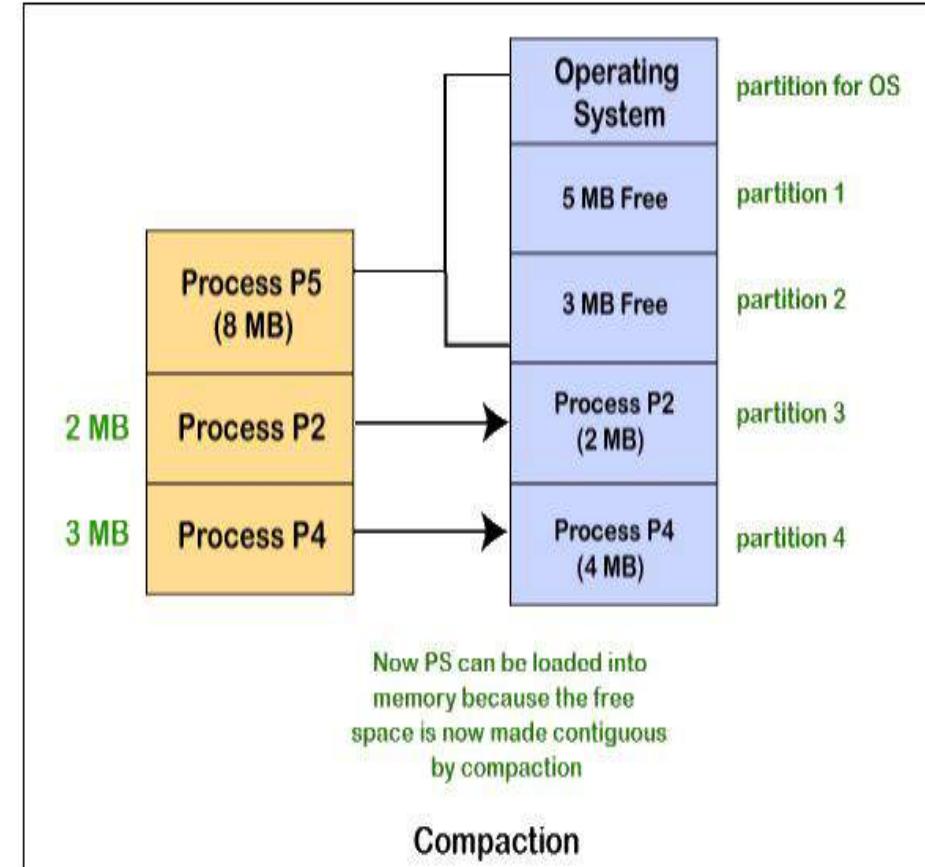
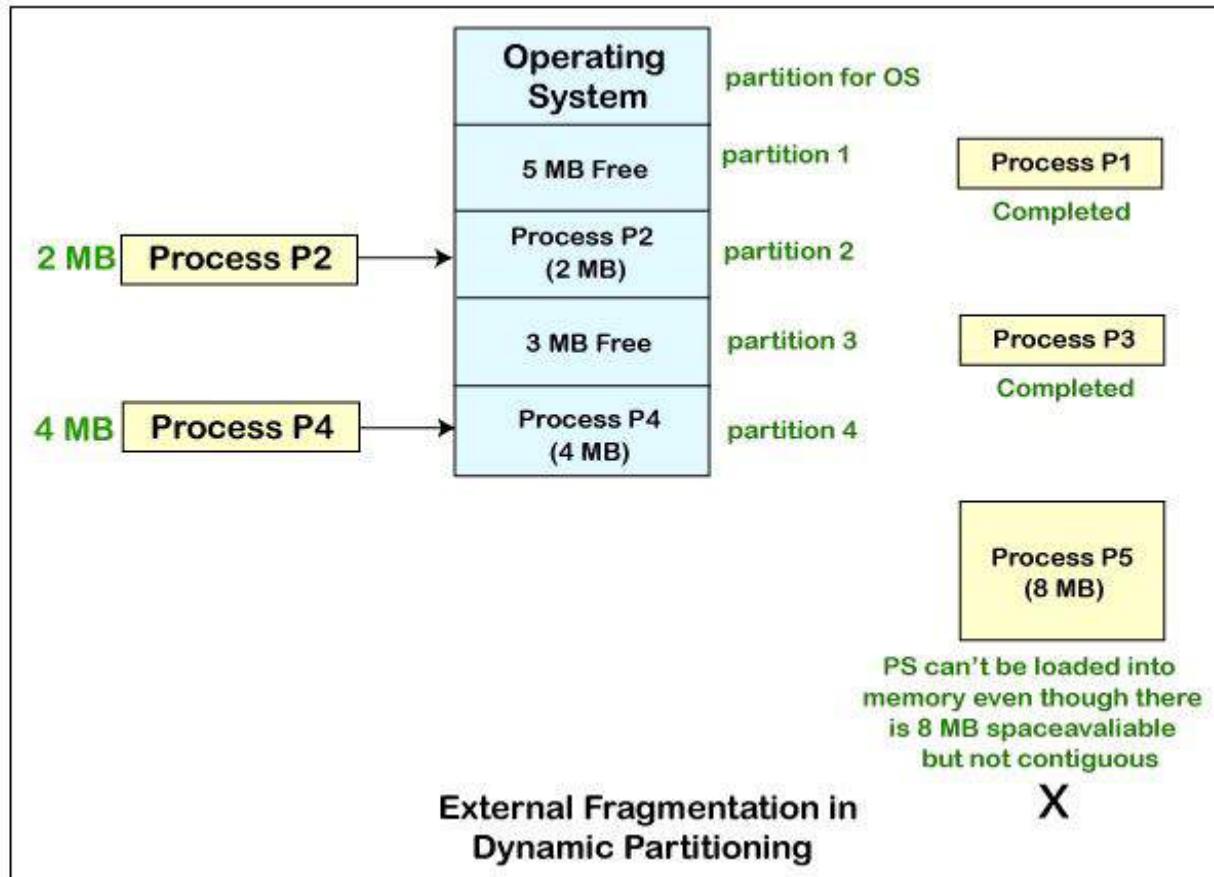
- Example



Solution to External Fragmentation

- This problem is occurring because we are **allocating memory continuously** to the processes. So, if we remove this condition external fragmentation can be reduced.
- One of the way to remove external fragmentation is **compaction**. When dynamic partitioning is used for memory allocation then external fragmentation can be reduced by **merging all the free memory** together in one large block.

Solution to External Fragmentation (Cont.)



Solution to External Fragmentation (Cont.)

- This technique is also called **defragmentation**. This larger block of memory is then used for allocating space according to the needs of the new processes.

Problem with Compaction

- Due to compaction, the system efficiency is decreased because we need to move all the free spaces from one place to other.
- In this way, the more amount of time is wasted, and the CPU remains idle all the time. Instead of that, with the help of compaction, we can avoid external fragmentation, but this will make the system inefficient.

Solution to External Fragmentation (Cont.)

- Another way, **paging & segmentation** (non-contiguous memory allocation techniques) where memory is allocated non-contiguously to the processes.

Fixed vs Variable Partitioning

S.NO.	Fixed partitioning	Variable partitioning
1.	In multi-programming with fixed partitioning the main memory is divided into fixed sized partitions.	In multi-programming with variable partitioning the main memory is not divided into fixed sized partitions.
2.	Only one process can be placed in a partition.	In variable partitioning, the process is allocated a chunk of free memory.
3.	It does not utilize the main memory effectively.	It utilizes the main memory effectively.
4.	There is presence of internal fragmentation and external fragmentation.	There is external fragmentation.
5.	Degree of multi-programming is less.	Degree of multi-programming is higher.
6.	It is more easier to implement.	It is less easier to implement.
7.	There is limitation on size of process.	There is no limitation on size of process.

Internal vs External Fragmentation

INTERNAL FRAGMENTATION	EXTERNAL FRAGMENTATION
Memory allocated to a process may be slightly larger than the requested memory. The difference between these two numbers is internal fragmentation.	External fragmentation exists when there is enough total memory space to satisfy a request but available spaces are not contiguous.
First-fit and best-fit memory allocation does not suffer from internal fragmentation.	First-fit and best-fit memory allocation suffers from external fragmentation.
Systems with fixed-sized allocation units, such as the single partitions scheme and paging suffer from internal fragmentation.	Systems with variable-sized allocation units, such as the multiple partitions scheme and segmentation suffer from external fragmentation.

Non Contiguous Allocation

- Non-contiguous memory allocation is a memory allocation technique.
- It allows to store parts of a single process in a non-contiguous fashion.
- Thus, different parts of the same process can be stored at different places in the main memory.

Techniques

Non-Contiguous Memory Allocation Techniques



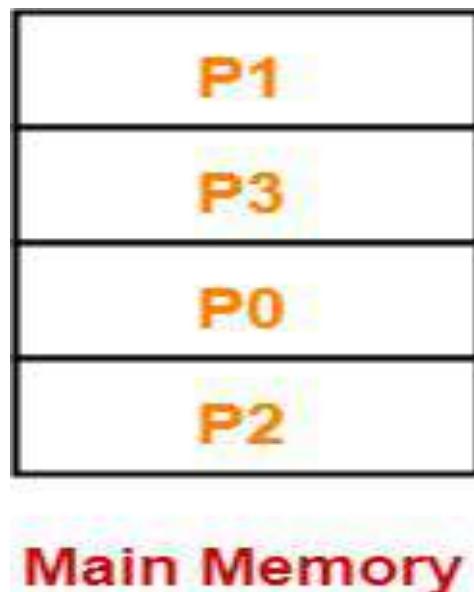
PAGING

Paging

- Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory.
- Paging permits the physical address space of a process to be non – contiguous.
- The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.

Example

- Consider a process is divided into 4 pages P_0, P_1, P_2 and P_3 .



- In the above example, the process P is divided into 4 sections namely P₀, P₁, P₂, P₃.
- All the four sections are mapped in the main memory in a non – contiguous fashion.

Paging

- In the paging mechanism two addresses are involved
 - Logical Address

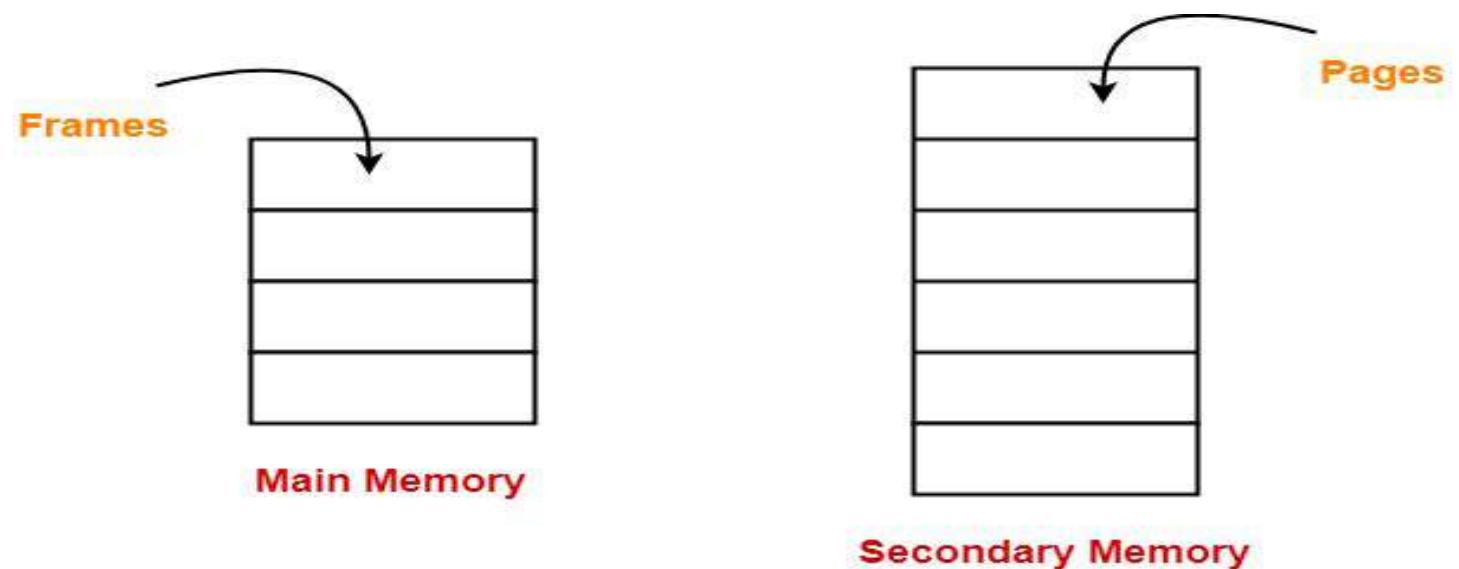
CPU always generates a logical address.
 - Physical Address

A physical address is needed to access the main memory.

Logical Address and Physical Address

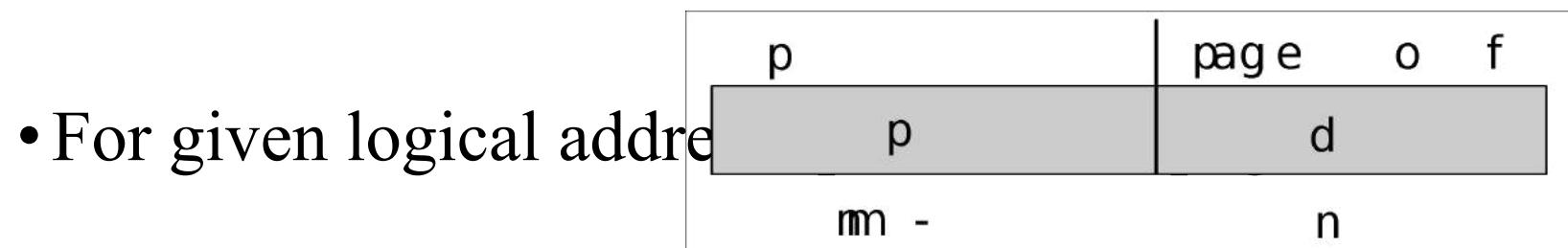
- **Logical Address or Virtual Address :**
An address generated by the CPU
- **Logical Address Space or Virtual Address Space:**
The set of all logical addresses generated by a program
- **Physical Address:**
An address actually available on memory unit
- **Physical Address Space :**
The set of all physical addresses corresponding to the logical addresses

- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
- The Logical address Space is also divided into fixed-size blocks, called **pages**.
- Page Size = Frame Size



Logical Address

- Address generated by CPU is divided into:
 - **Page number (p)** – used as an index into a **page table** which contains base address of each page in physical memory
 - **Page offset (d)** – combined with base address to define the physical memory address that is sent to the memory unit



- For given logical address

Physical Address

- Physical Address is divided into

Frame number(f): Number of bits required to represent the frame of Physical Address Space or Frame number.

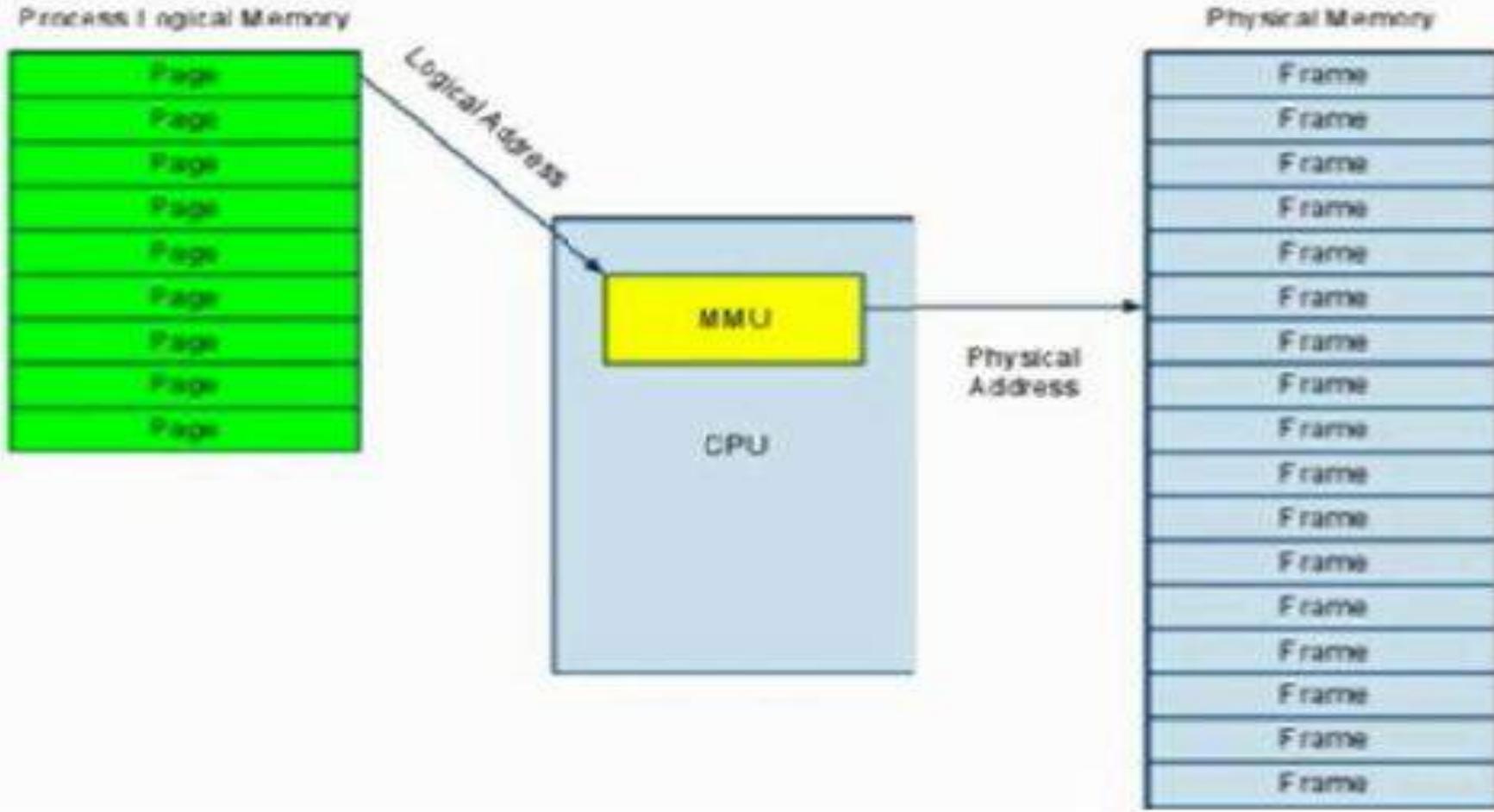
Frame offset(d): Number of bits required to represent particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

- If **Logical Address** = 31 bit, then **Logical Address Space** = 2^{31} words = 2 G words (1 G = 2^{30})
- If **Logical Address Space** = 128 M words = $2^7 * 2^{20}$ words, then **Logical Address** = $\log_2 2^{27} = 27$ bits
- If **Physical Address** = 22 bit, then **Physical Address Space** = 2^{22} words = 4 M words (1 M = 2^{20})
- If **Physical Address Space** = 16 M words = $2^4 * 2^{20}$ words, then **Physical Address** = $\log_2 2^{24} = 24$ bits

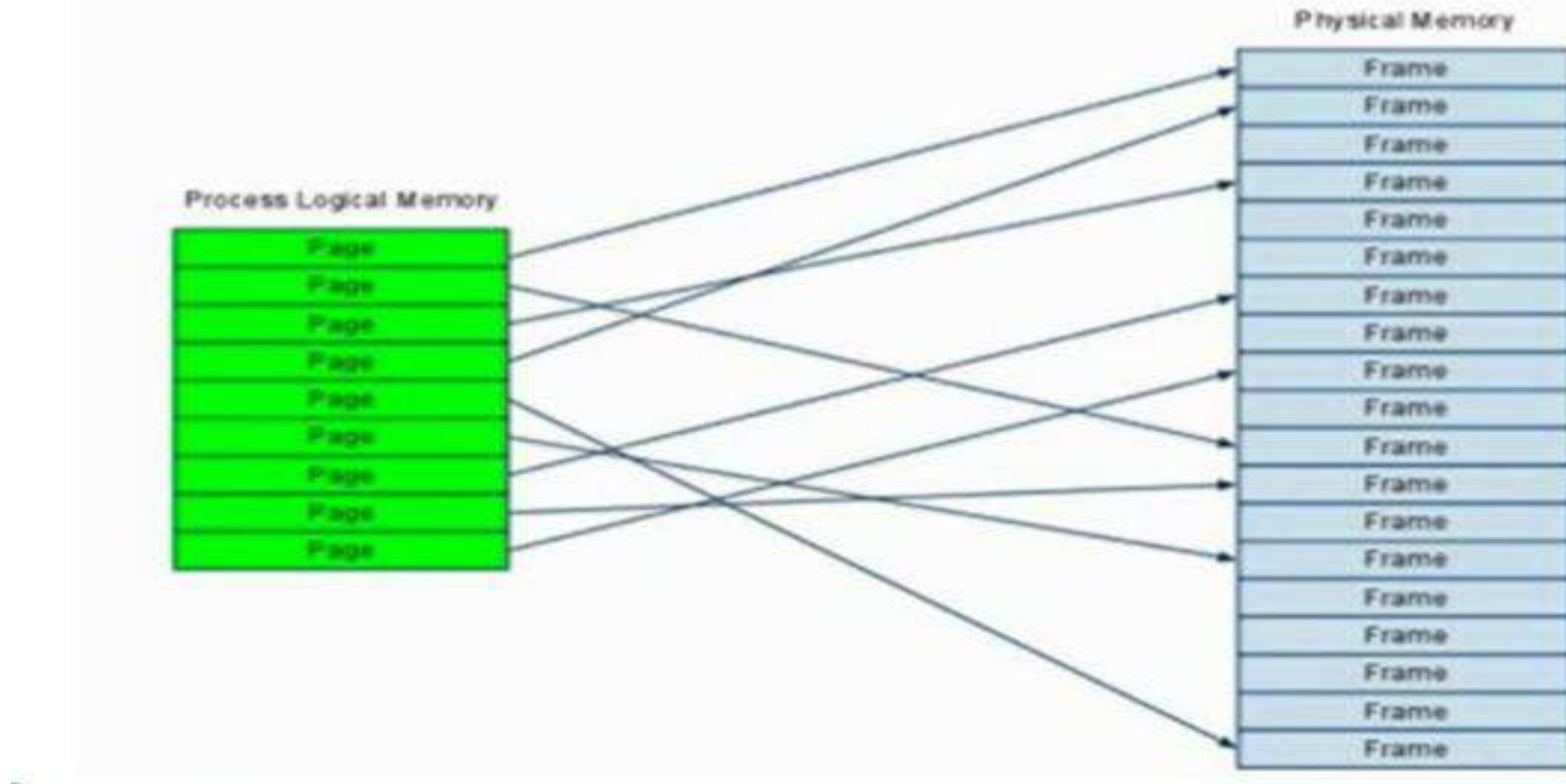
Memory Management

- The Mapping between these two address is done to execute the process using the paging technique.
- The Hardware device that at run time maps virtual address to physical address is done by the Memory Management Unit (MMU)

Page Translation



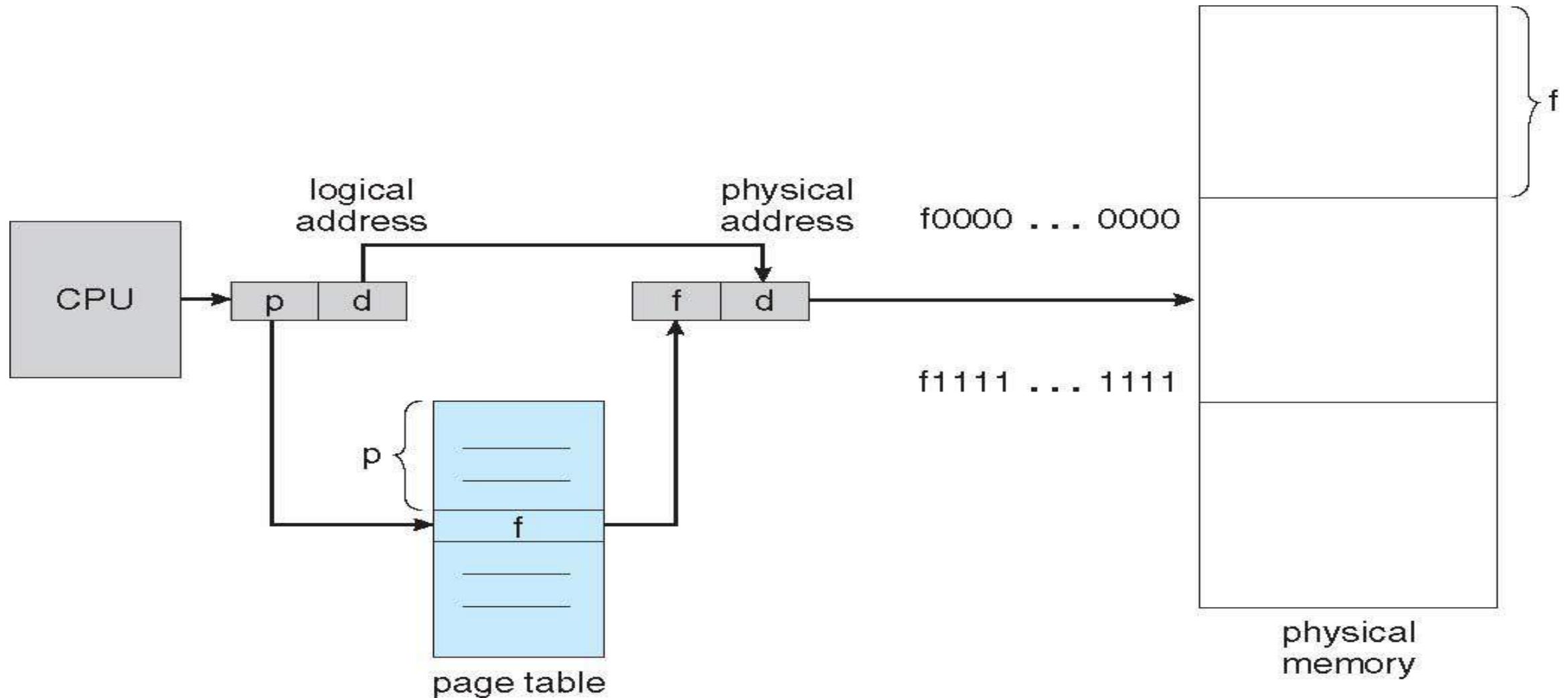
Address Mapping



Paging Hardware

- How does system perform translation? Simplest solution: use a page table.
- Page table is a linear array indexed by virtual page number that gives the physical page frame that contains that page through the look up process.

Paging Hardware



Paging Hardware

LOOK UP PROCESS

Step 1 :Extract page number.

Step 2 : Extract offset.

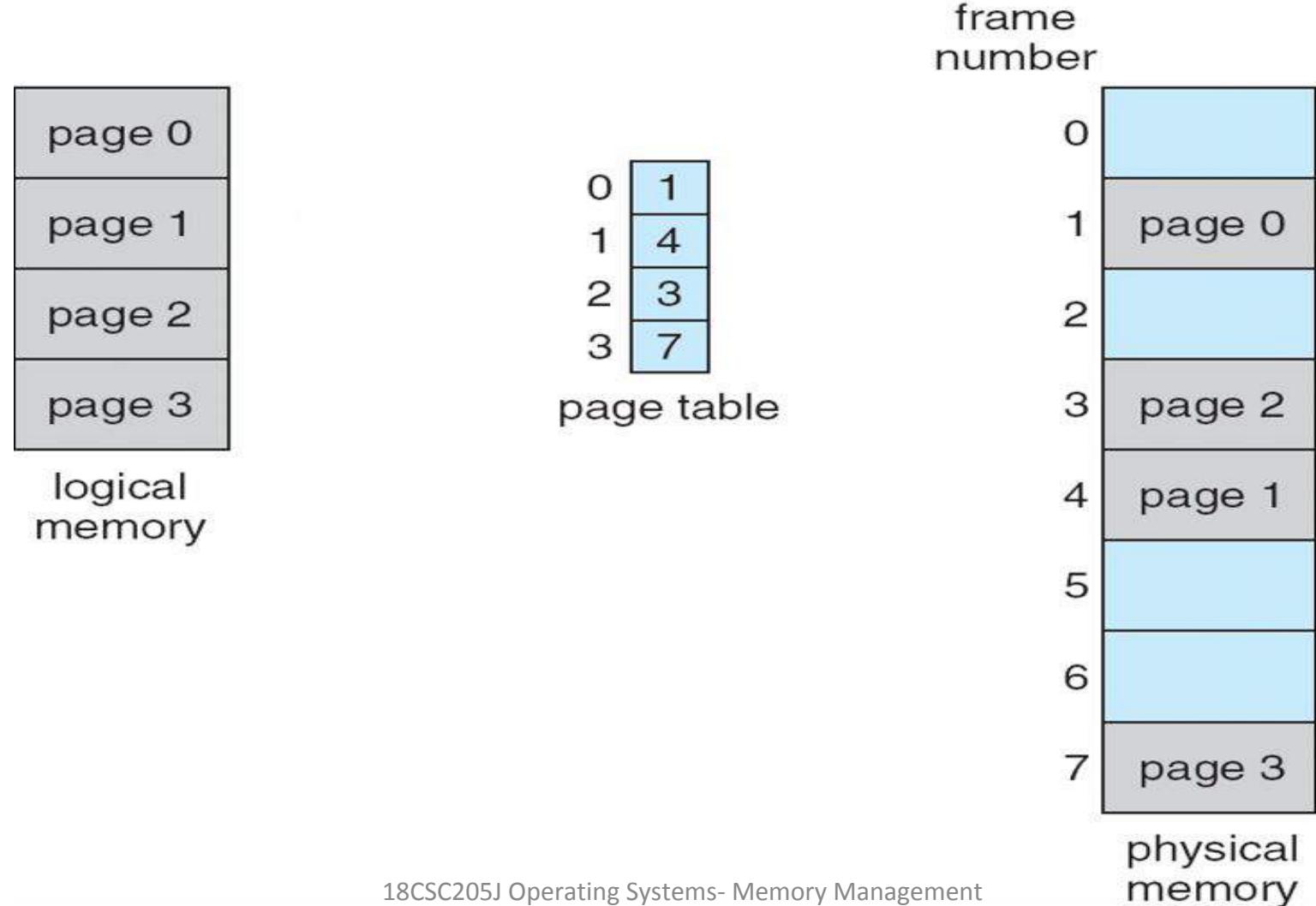
Step 3 : Check that page number is within address space of process.

Step 4 : Look up page number in page table.

Step 5 : Add offset to resulting physical page number

Step 6 : Access memory location.

Paging Model of Logical and Physical Memory



- In the example a process is divided into four pages as page0, page1, page 2 and page3.
- All the four pages are mapped to the frames in the physical memory through the page table.
- The page table maps the page number to its corresponding frame number in the physical memory.
- page0 is mapped to frame1 , page1 is mapped to frame 4, page2 is mapped to frame 3 and page4 is mapped to frame 7.

Paging Example

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

logical memory

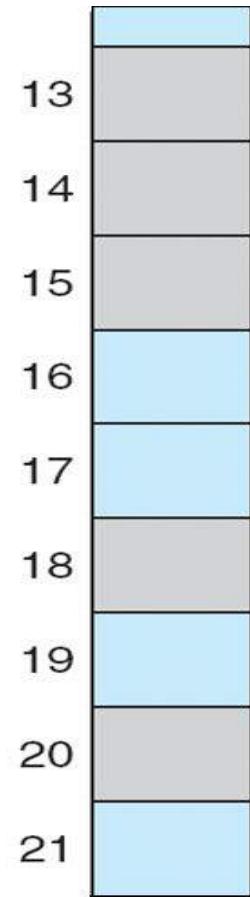
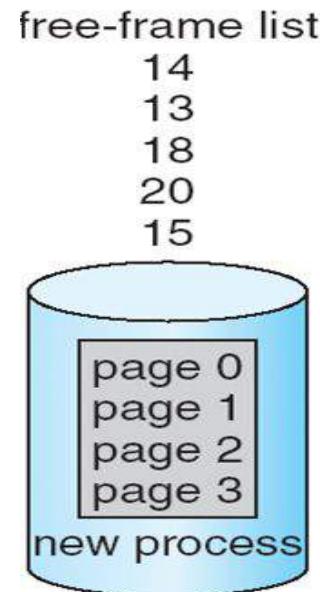
0	5
1	6
2	1
3	2

page table

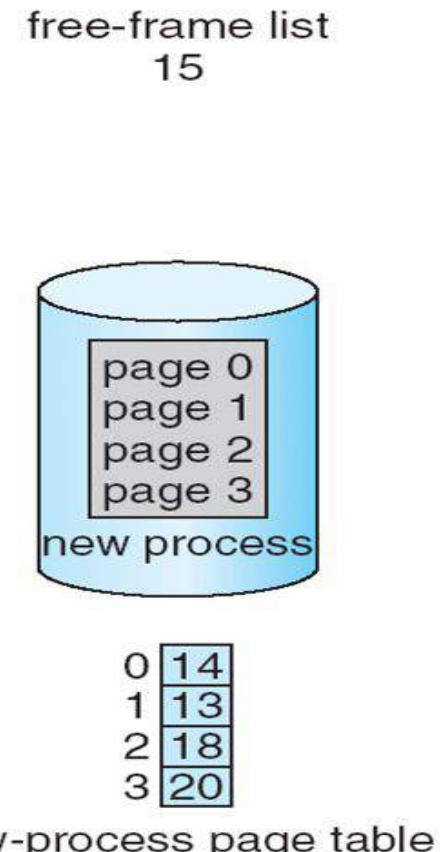
0	
4	i j k l
8	m n o p
12	
16	
20	a b c d
24	e f g h
28	

physical memory

Free Frame List



(a)



(b)



Free Frame List

The example

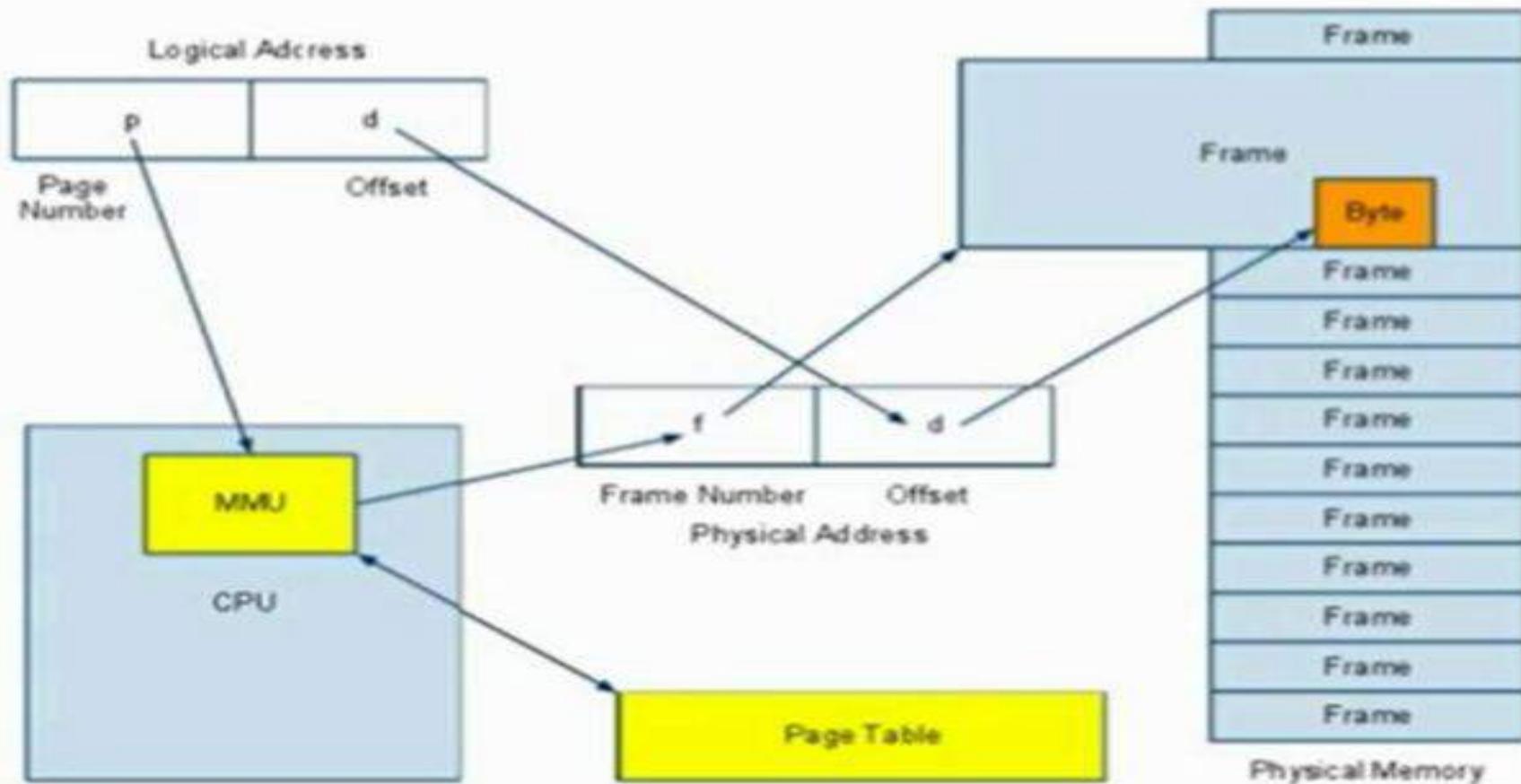
Fig (a) shows a process divided into 4 pages and the list of free frames available in the physical memory.

Fig(b) shows the mapping of these pages to the free frames available and the updated list of free frames available in the physical memory.

Page Table

- Page table is kept in main memory
- **Page-table base register (PTBR)** points to the page table
- **Page-table length register (PTLR)** indicates size of the page table

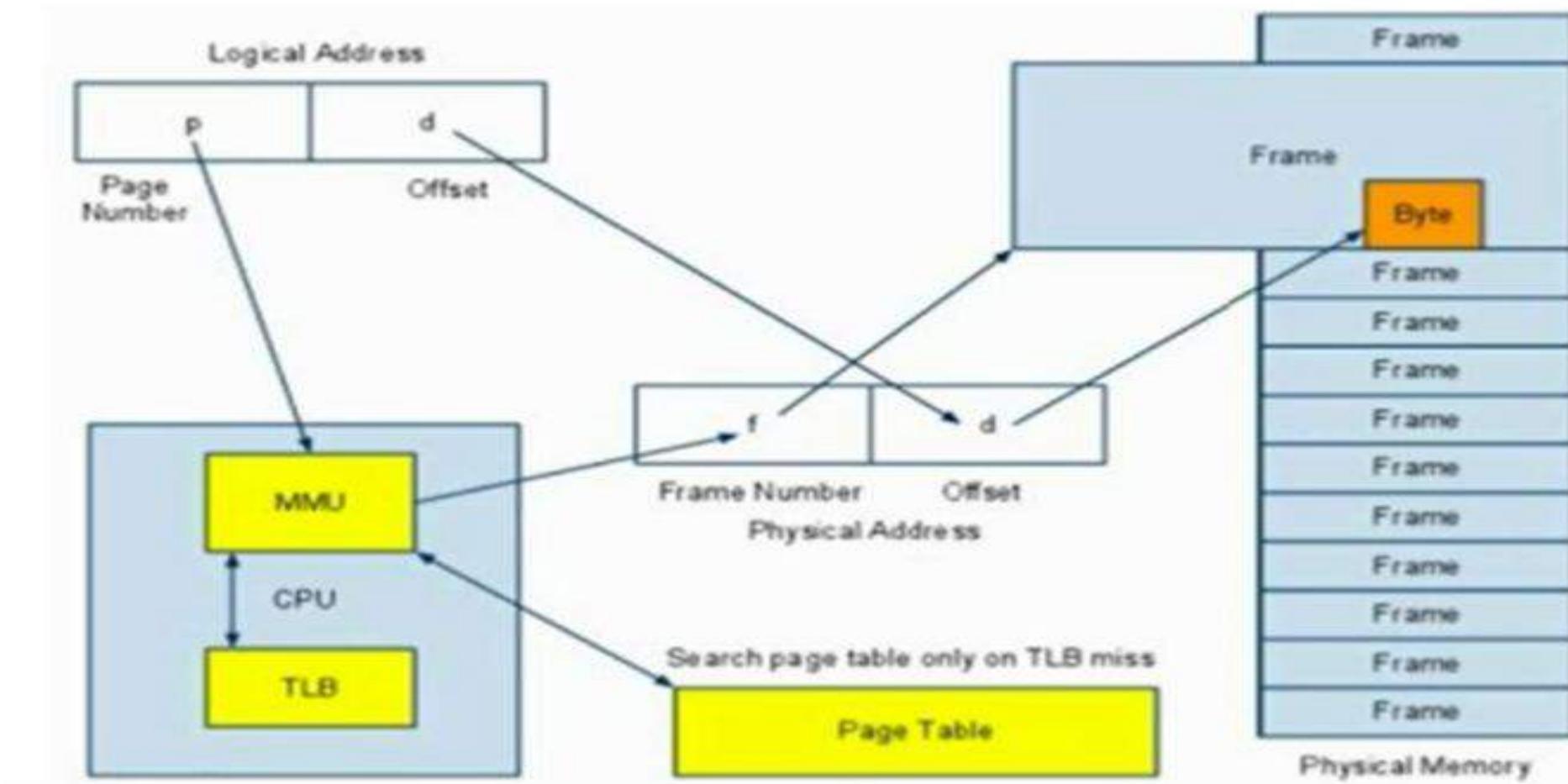
MMU Address Transaction



Problem with paging

- In this scheme every data/instruction access requires two memory accesses
 - One for the page table and one for the data / instruction
- The two memory access problem can be solved by the use of a special fast-lookup hardware cache called **associative memory or translation look-aside buffers (TLBs)**

TLB-Assisted Transaction



Translation Lookaside Buffer (TLB)

- Speed up the lookup problem with a cache. Store most recent page lookup values in TLB.

Procedure of TLB

Step 1 :Extract page number.

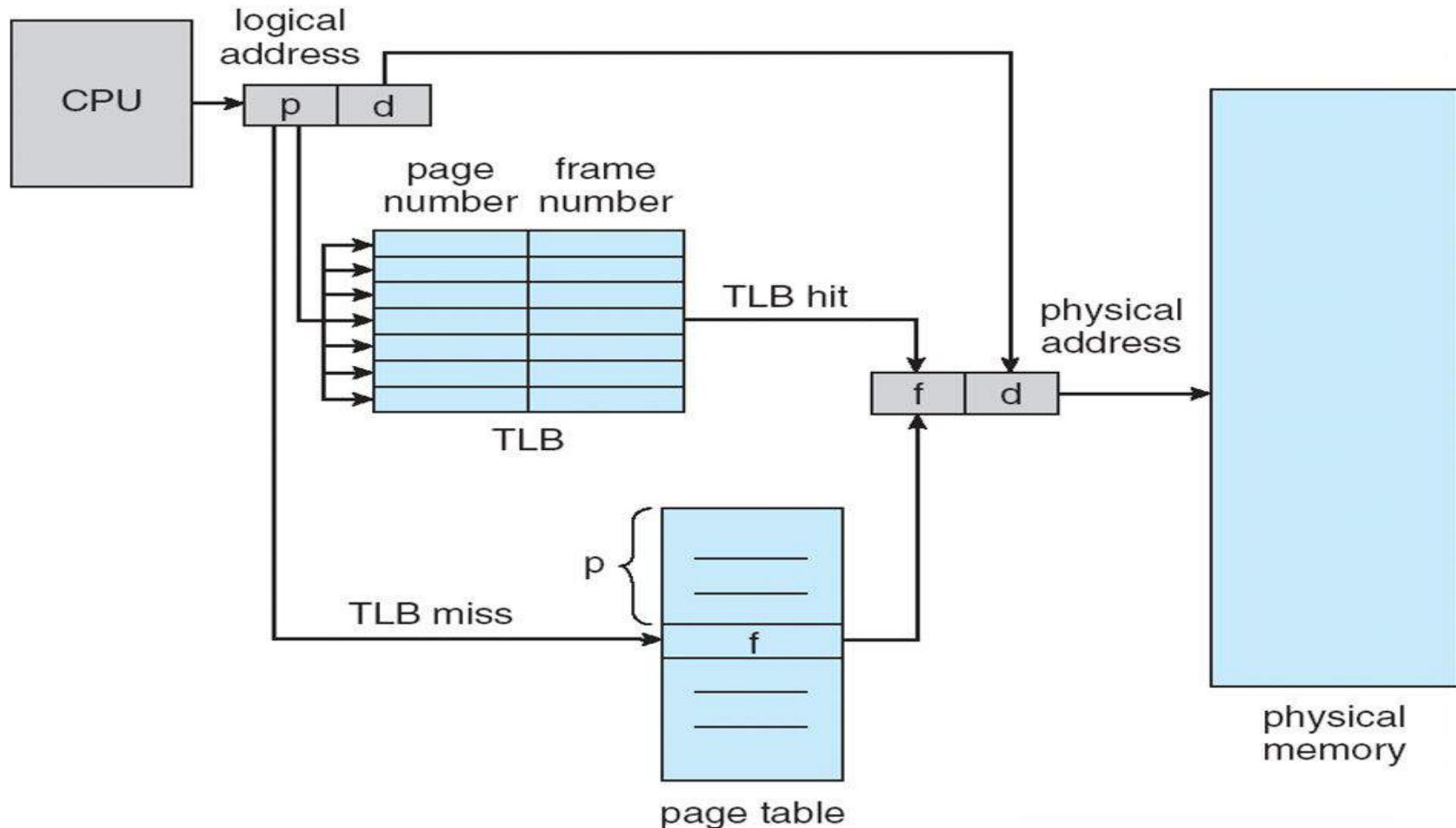
Step 2 : Extract offset.

Step 3 : Look up page number in TLB.

Step 4 : If there, add offset to physical page number and access memory location.

Step 5 : Otherwise, trap to OS. OS performs check, looks up physical page number, and loads translation into TLB. Restarts the instruction.

Paging with TLB



- If the page number is not in the TLB (TLB miss) a memory reference to the page table must be made.
- In addition, we add the page number and frame number into TLB
- If the TLB already full, the OS have to must select one for replacement
- Some TLBs allow entries to be **wire down**, meaning that they cannot be removed from the TLB, for example kernel codes
- The percentage of times that a particular page number is found in the TLN is called **hit ratio**

- Fixed size allocation of physical memory in page frames dramatically simplifies allocation algorithm.
- OS can just keep track of free and used pages and allocate free pages when a process needs memory.
- There is no fragmentation of physical memory into smaller and smaller allocatable chunks.

Effective Access Time

- Associative Lookup = ε time unit
 - Can be < 10% of memory access time
- Hit ratio = α
 - Hit ratio – percentage of times that a page number is found in the associative registers; ratio related to number of associative registers
- Consider $\alpha = 80\%$, $\varepsilon = 20\text{ns}$ for TLB search, 100ns for memory access
- **Effective Access Time (EAT)**

$$\begin{aligned} \text{EAT} &= (1 + \varepsilon) \alpha + (2 + \varepsilon)(1 - \alpha) \\ &= 2 + \varepsilon - \alpha \end{aligned}$$
- Consider $\alpha = 80\%$, $\varepsilon = 20\text{ns}$ for TLB search, 100ns for memory access
 - $\text{EAT} = 0.80 \times 100 + 0.20 \times 200 = 120\text{ns}$

Memory Protection

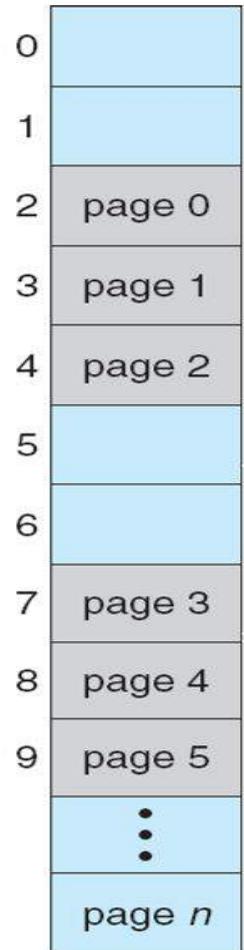
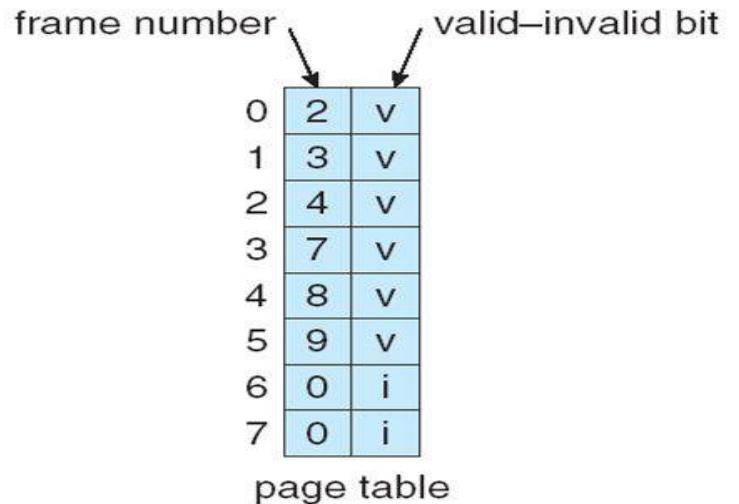
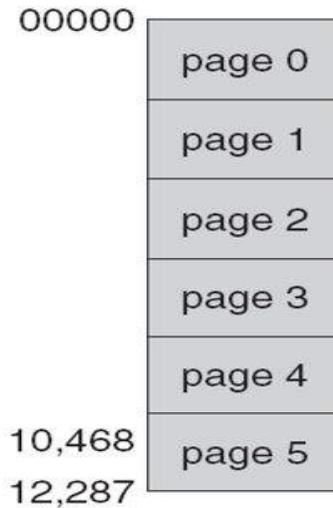
The process has to be protected for the following reasons:

- Preventing one process from reading or writing another process' memory.
- Preventing one process from reading another process' memory.
- Preventing a process from reading or writing some of its own memory.
- Preventing a process from reading some of its own memory.

Memory Protection

- Memory protection is implemented by associating protection bit with each frame to indicate if read-only or read-write access is allowed
 - Can also add more bits to indicate page execute-only, and so on
- **Valid-invalid** bit attached to each entry in the page table:
 - “valid” indicates that the associated page is in the process’ logical address space, and is thus a legal page
 - “invalid” indicates that the page is not in the process’ logical address space
 - Or use **page-table length register (PTLR)**
- Any violations result in a trap to the kernel

Valid – Invalid Bit



- Pages that are not loaded into memory are marked as invalid in the page table, using the invalid bit.
- In the above example Page0 to page5 is present in the memory .so , the corresponding valid/Invalid bit is marked as V(Valid).
- The other pages which are not loaded such as page6 and page7 are marked as I(Invalid)

PageFault

- if a page is needed that was not originally loaded up, then a *page fault trap* is generated
- Page fault is handled by the following steps:

Step 1: The memory address requested is first checked, to make sure it was a valid memory request.

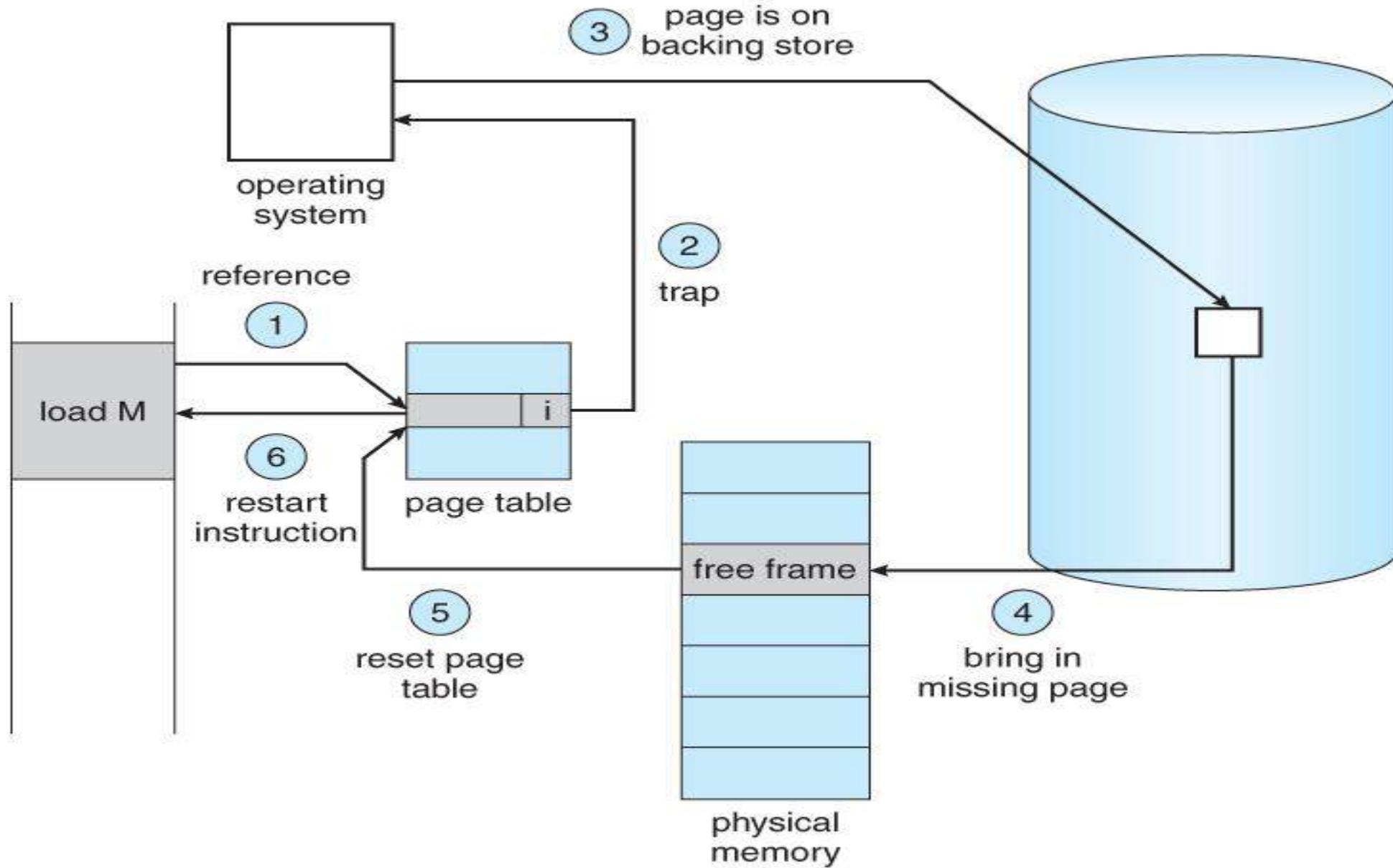
Step 2: If the reference was invalid, the process is terminated.
Otherwise, the page must be paged in.

Step 3: A free frame is located, possibly from a free-frame list.

Step 4: A disk operation is scheduled to bring in the necessary page from disk.

Step 5: When the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to indicate that this is now a valid page reference.

Step 6: The instruction that caused the page fault must now be restarted from the beginning



Shared Pages

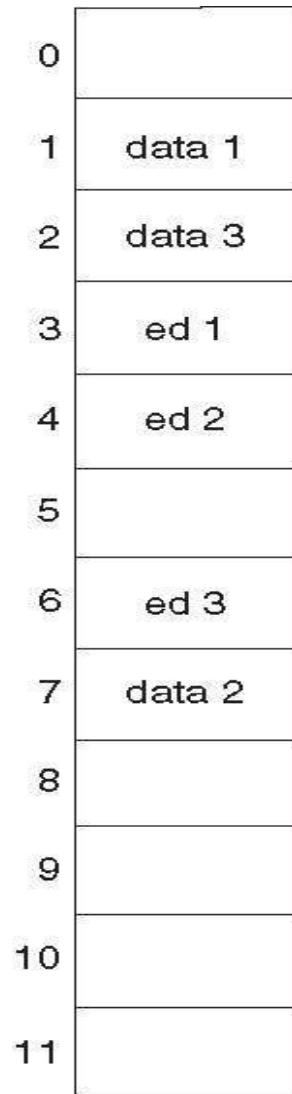
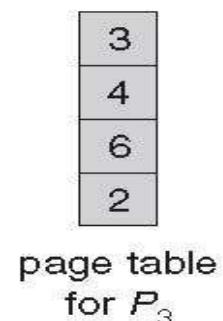
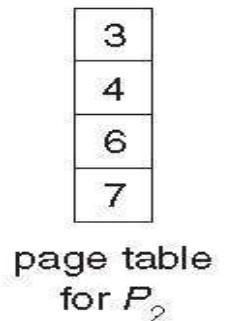
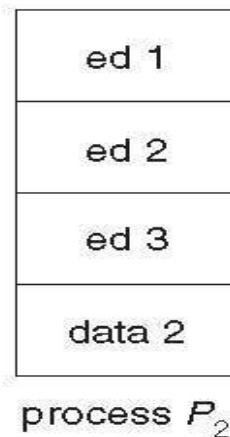
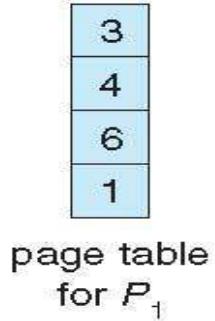
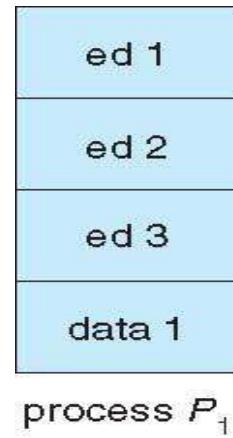
- **Shared code**

- One copy of read-only (**reentrant**) code shared among processes (i.e., text editors, compilers, window systems)
- Similar to multiple threads sharing the same process space
- Also useful for interprocess communication if sharing of read-write pages is allowed

- **Private code and data**

- Each process keeps a separate copy of the code and data
- The pages for the private code and data can appear anywhere in the logical address space

Shared Pages



Shared Pages

- In the example figure, three pages ed1,ed2 and ed3 are shred by three processes Process1 , Process2, Process3.
- The page is loaded only once and the corresponding frame number is updated in the page table of the three processes.

STRUCTURE OF A PAGE TABLE

Structure of the Page Table

- Memory structures for paging can get huge using straight-forward methods
 - Consider a 32-bit logical address space as on modern computers
 - Page size of 4 KB (2^{12})
 - Page table would have 1 million entries ($2^{32} / 2^{12}$)
 - If each entry is 4 bytes -> 4 MB of physical address space / memory for page table alone
 - That amount of memory used to cost a lot
 - Don't want to allocate that contiguously in main memory
- Hierarchical Paging
- Hashed Page Tables
- Inverted Page Tables

Two-Level Paging Example

- Simple solution to this problem is to divide the page table into smaller pieces We can accomplish this division in several ways
- A logical address (on 32-bit machine with 1K page size) is divided into:
 - a page number consisting of 22 bits
 - a page offset consisting of 10 bits

page number	page offset
p_1	p_2
12	10
	10

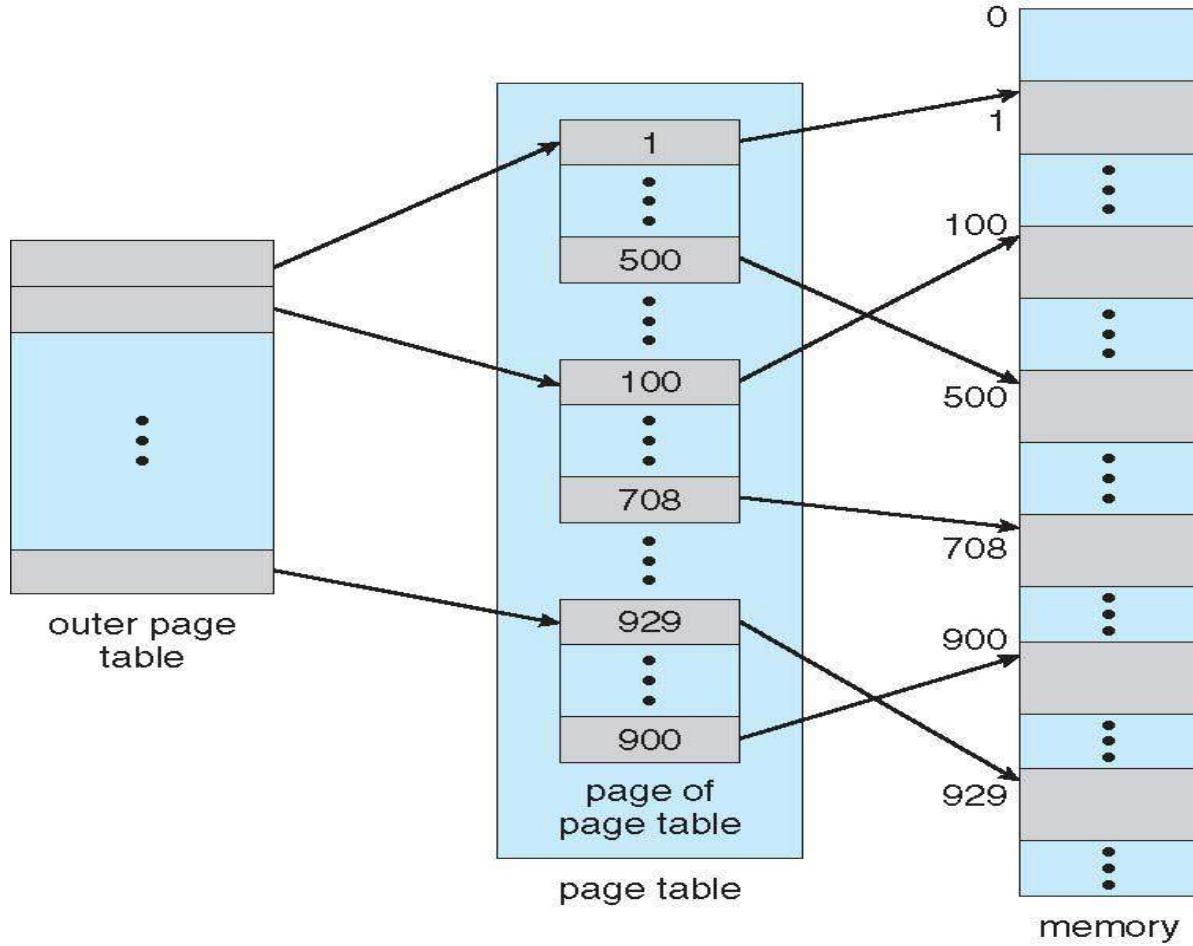
- Since the page table is paged, the page number is further divided into:
 - a 12-bit page number
 - a 10-bit page offset

- Thus, a logical address is as follows:
- where p_1 is an index into the outer page table, and p_2 is the displacement within the page of the inner page table

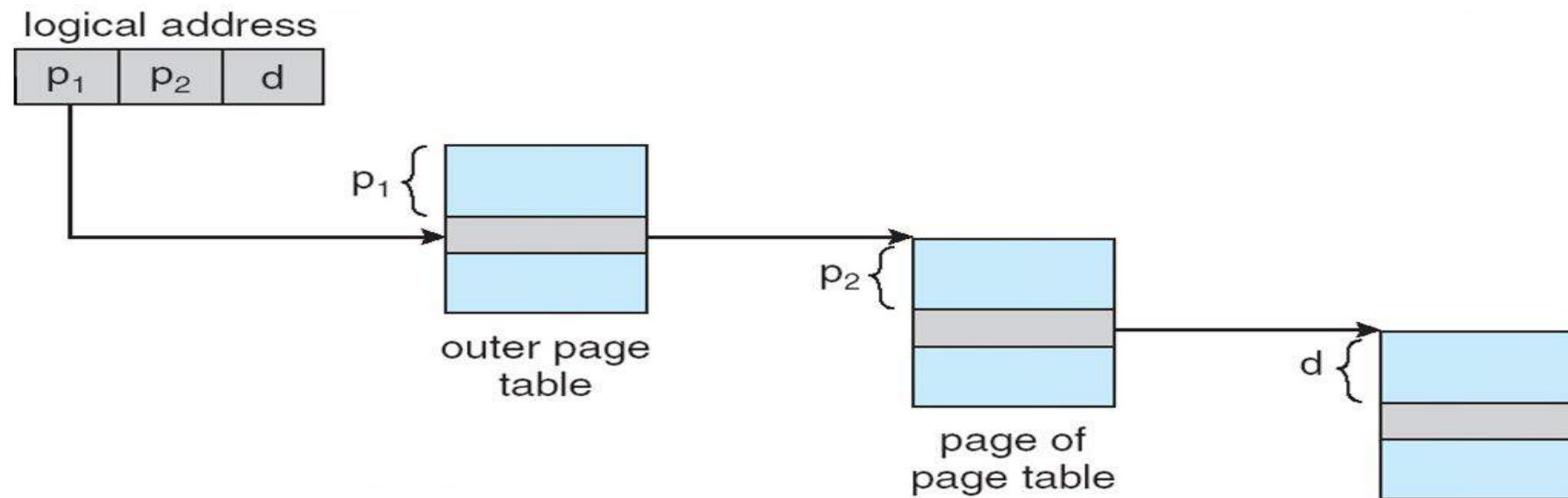
Hierarchical Page Tables

- Break up the logical address space into multiple page tables
- A simple technique is a two-level page table
- We then page the page table

Two-Level Page-Table Scheme



Address-Translation Scheme for two level 32 bit paging architecture



64-bit Logical Address Space

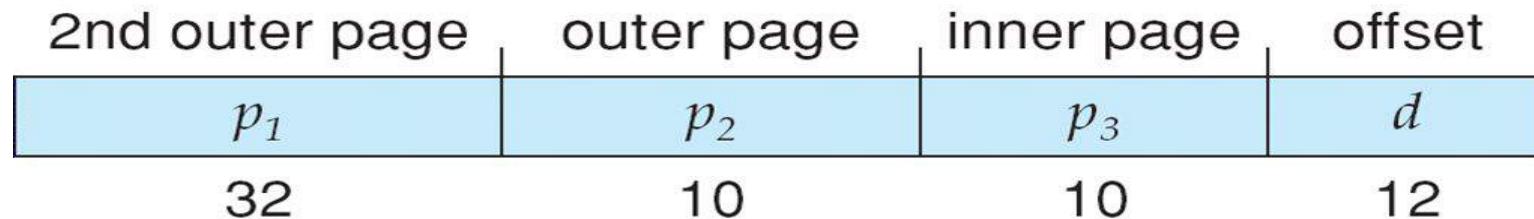
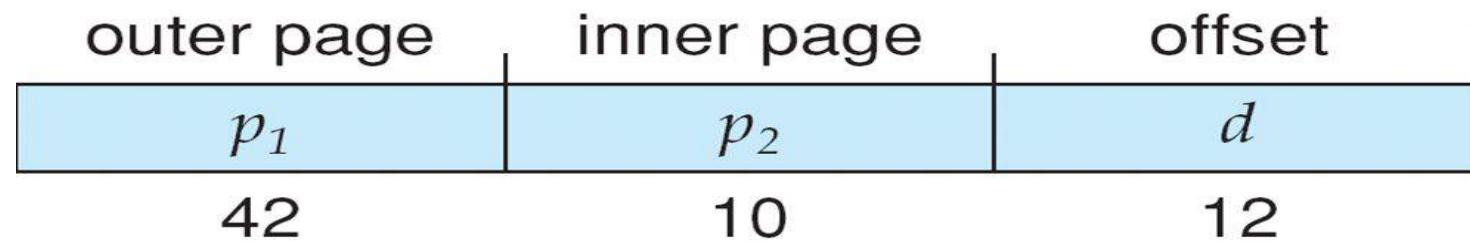
- For a system with a 64-bit logical address space, a two-level paging scheme is no longer appropriate
- If page size is 4 KB (2^{12})
 - Then page table has 2^{52} entries
 - If two level scheme, inner page tables could be 2^{10} 4-byte entries
 - Address would look like

outer page	inner page	page offset
P_1	P_2	d
42	10	12

- Outer page table has 2^{42} entries or 2^{44} bytes
- One solution is to add a 2nd outer page table
- But in the following example the 2nd outer page table is still 2^{34} bytes in size
 - And possibly 4 memory access to get to one physical memory location

- We can divide the outer page table in various ways. For example, we can page the outer page table, giving us a three-level paging scheme. Suppose that the outer page table is made up of standard-size pages (2¹⁰ entries, or 2¹² bytes). In this case, a 64-bit address space is still daunting
- This keeps growing that is the reason hierarchical page tables are generally considered inappropriate

Three-level Paging Scheme



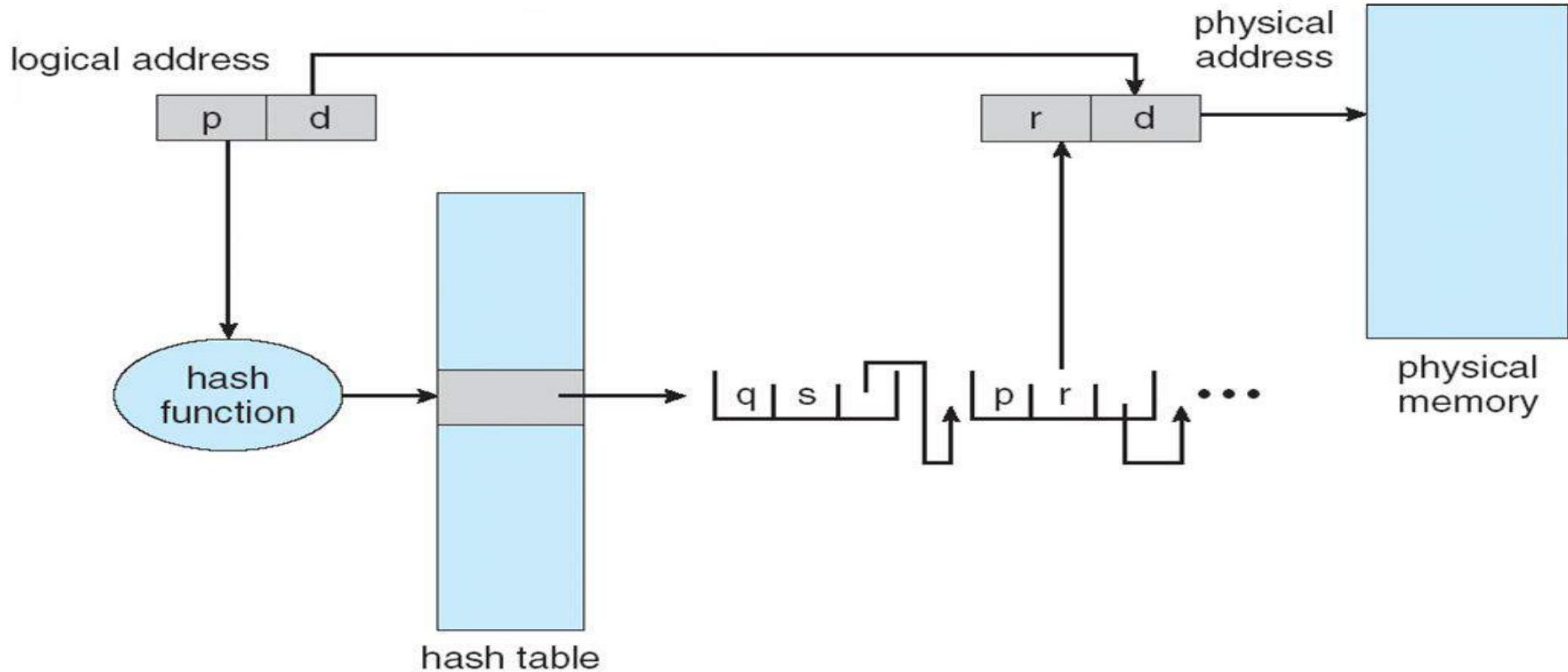
Hashed Page Tables

- One approach for handling address spaces larger than 32 bits is to use a hashed page table, with the hash value being the virtual page number
- Common in address spaces > 32 bits
- The virtual page number is hashed into a page table
 - This page table contains a chain of elements hashing to the same location
- Each element contains (1) the virtual page number (2) the value of the mapped page frame (3) a pointer to the next element

Hashed Page Tables (Cont.)

- Virtual page numbers are compared in this chain searching for a match
 - If a match is found, the corresponding physical frame is extracted
- Variation for 64-bit addresses is **clustered page tables**
 - Similar to hashed but each entry refers to several pages (such as 16) rather than 1
 - Especially useful for **sparse** address spaces (where memory references are non-contiguous and scattered)

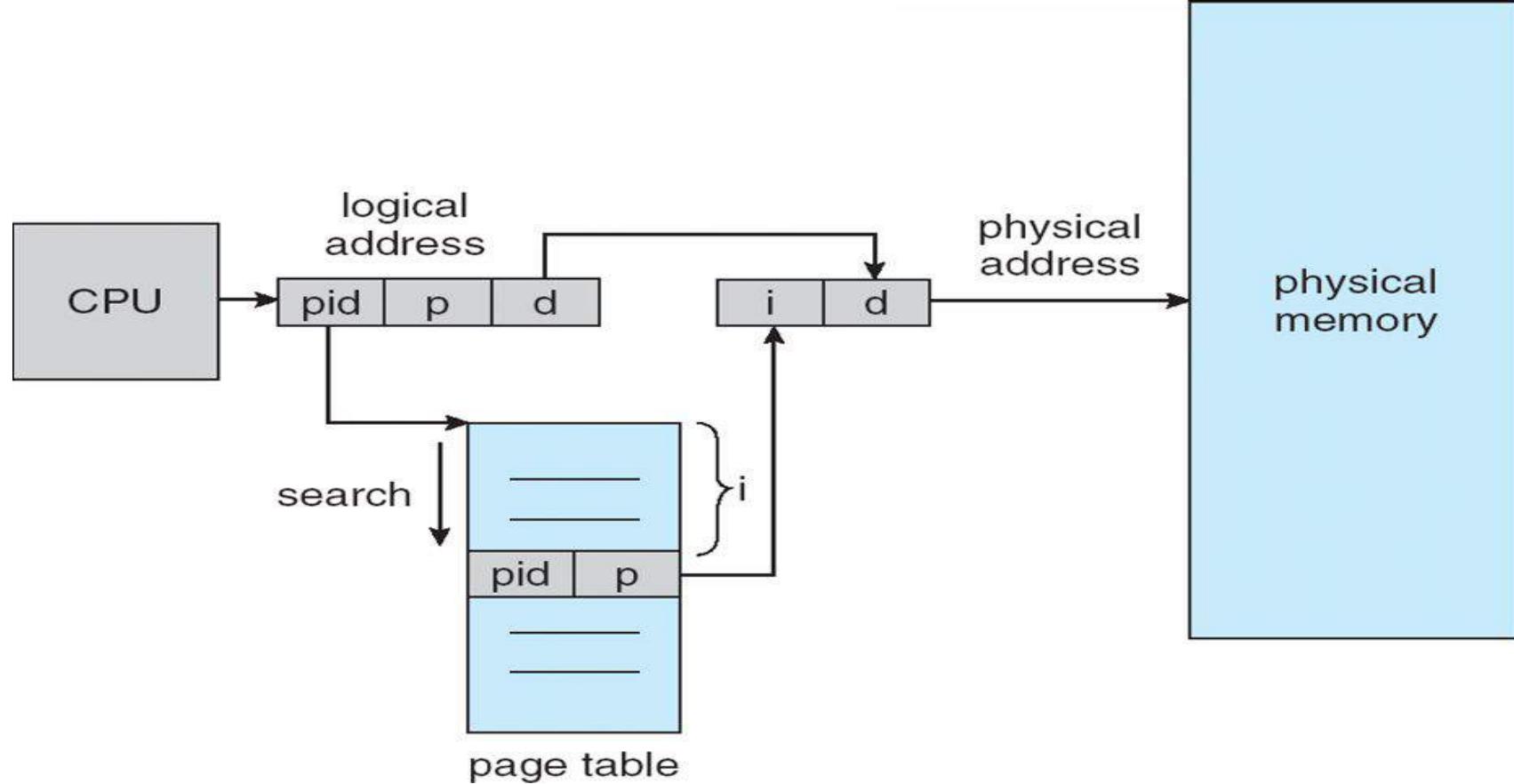
Hashed Page Table



Inverted Page Table

- Rather than each process having a page table and keeping track of all possible logical pages, track all physical pages
- One entry for each real page of memory
- Entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page

Inverted Page Table Architecture

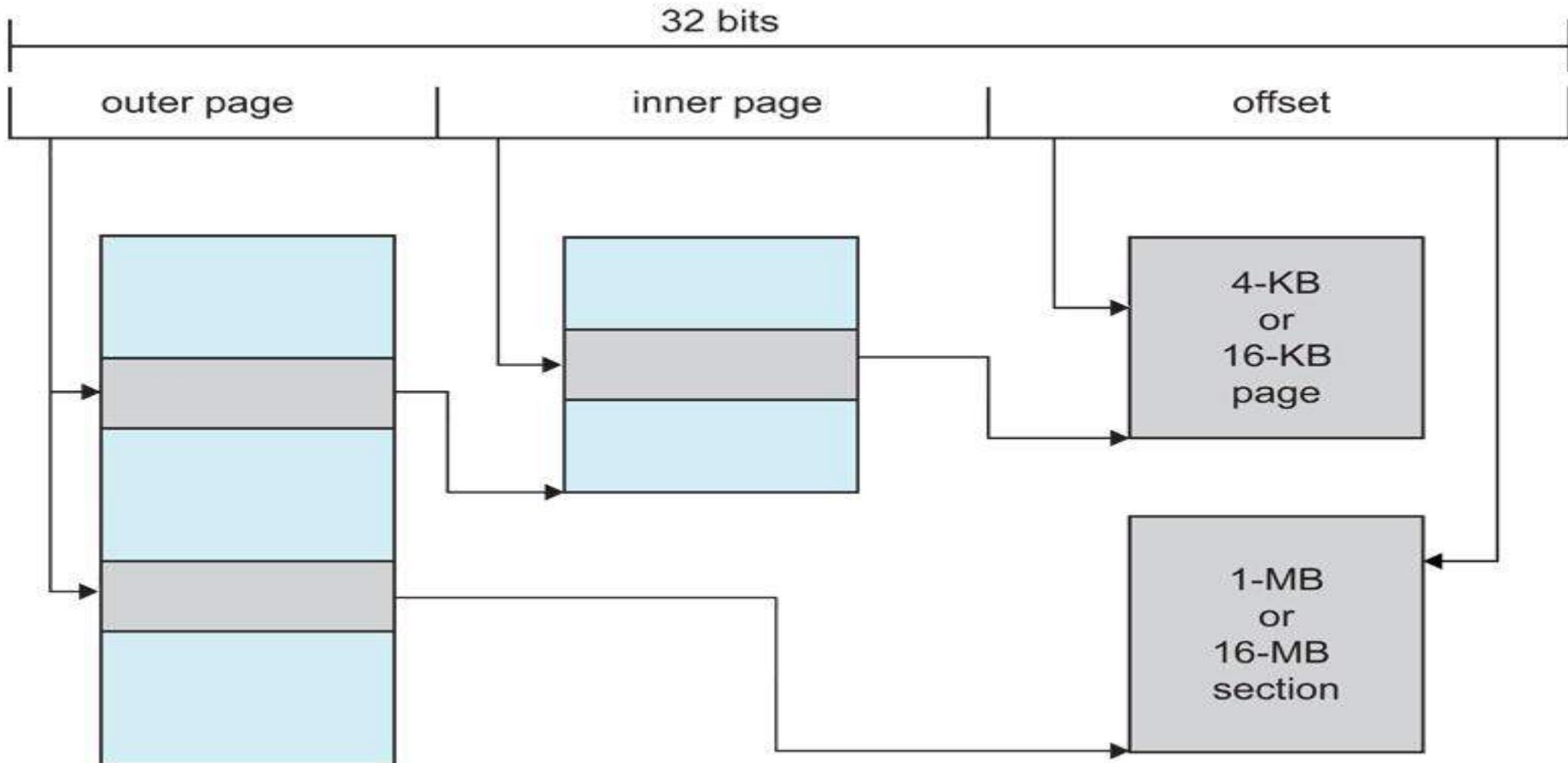


Paging - ARM

- The paging scheme used by ARM processors uses a 2 level page table.
- The first level page table has 16kB size and each entry covers a region of 1MB (sections).
- The second level page tables have 1kB size and each entry covers a region of 4KB (pages).
- The TLB also supports 16MB sections and 64kB pages

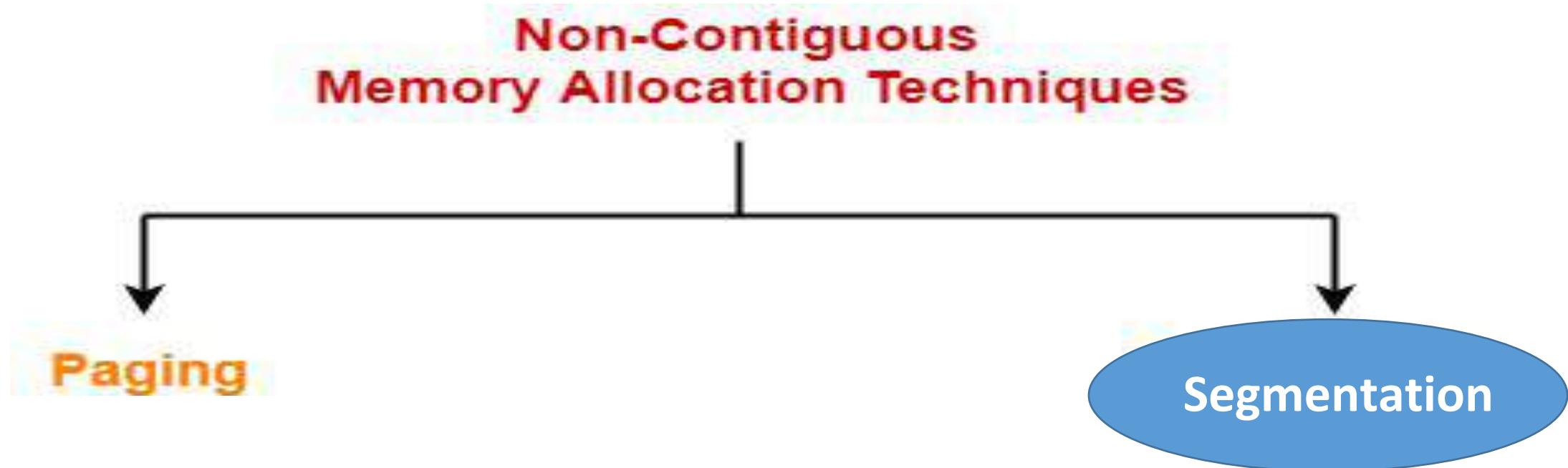
Paging - ARM

- On ARMv4 and ARMv5 pages might also be divided into what the manual calls 'subpages'
- Example : for a 4KB page, 1KB subpages, and for a 64KB page, 16KB subpages.
- The ARM processor supports 2 independent page tables.
- The first page table can be abbreviated from the full 16kB
- The second page table is always 16KB in size and is used for addresses beyond the end of the first page table.



SEGMENTATION

Non-Contiguous Memory Allocation



Segmentation

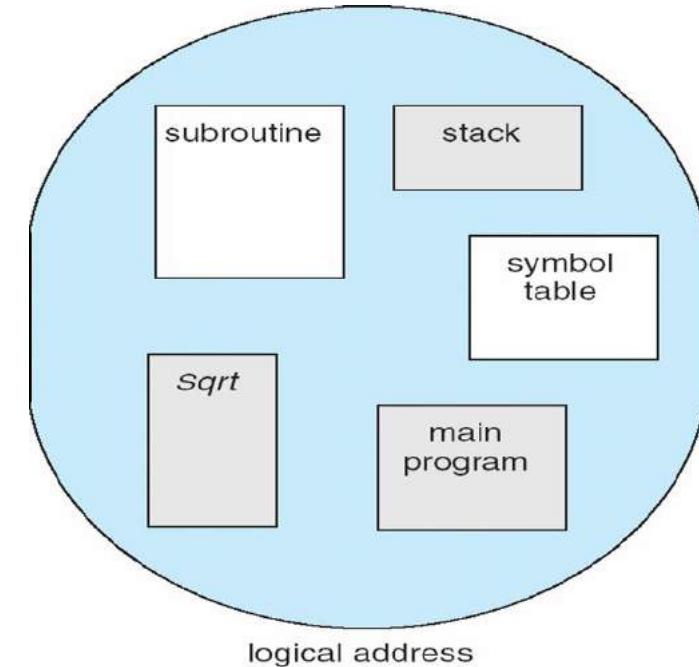
- Memory-management scheme that supports user view of memory.
- Segmentation overcomes the problem of dealing with memory in terms of its physical properties which is inconvenient to both the operating system and the programmer.
- The system have more freedom to manage memory.
- The programmer would get an experience to work in more natural programming environment.

Program and Segment

- A program is a collection of segments in which segment is a logical unit which consists of the following :
 - main program
 - procedure
 - Function
 - Method
 - Object
 - local variables, global variables
 - common block
 - Stack
 - symbol table
 - arrays

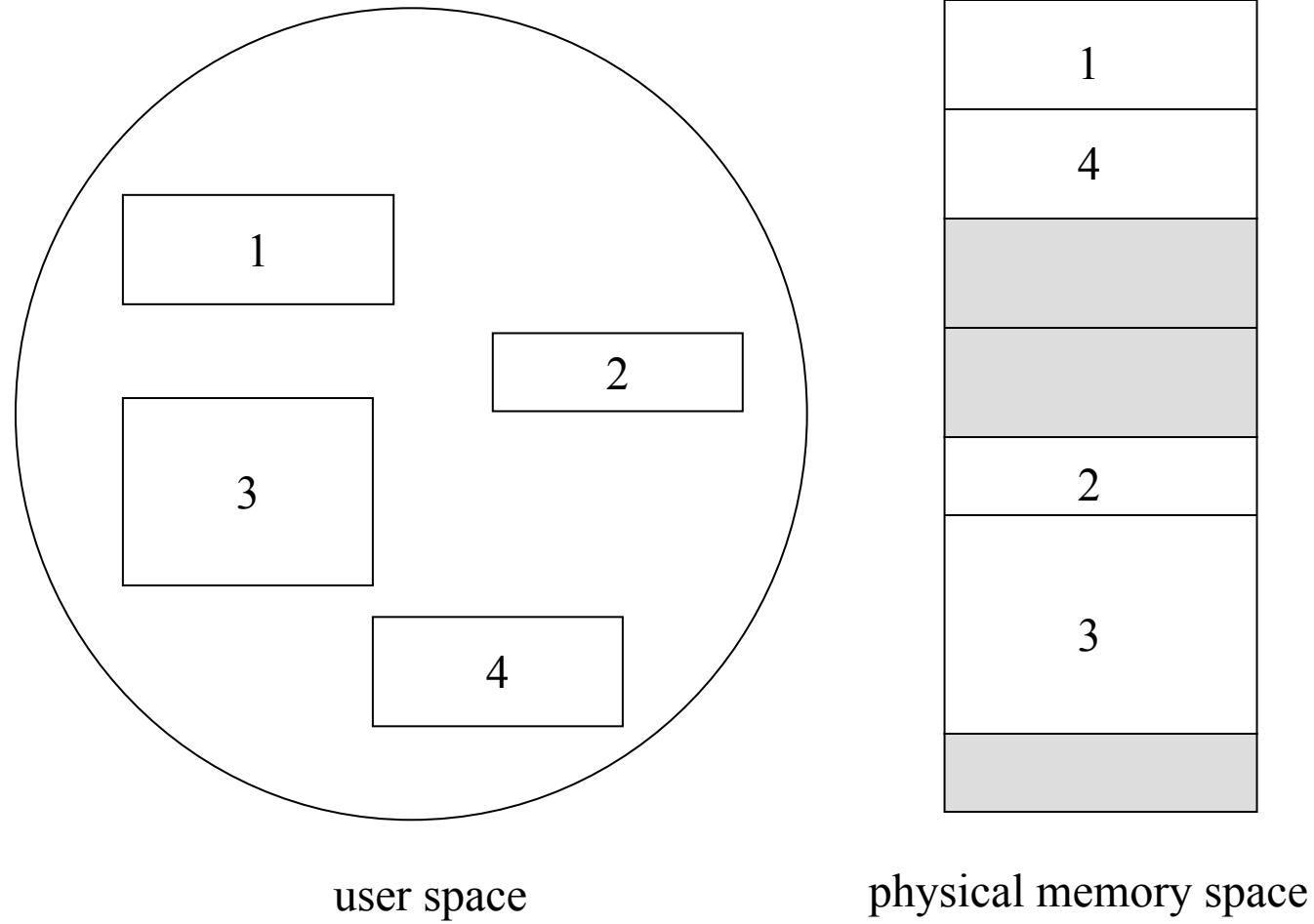
User's View of a Program

- Most programmers prefer to view memory as a collection of variable-sized segments with no necessary ordering among the segments.
- From programmer point of view , a program will appear as a main program with a set of methods, procedures, or functions. Program also includes the following data structures:
 - Objects
 - Arrays
 - Stacks and
 - Variables
- Modules or data elements is referred by name.
- The programmer talks about “the stack,” “the math library,” and “the main program” without caring what addresses in memory these elements occupy.
- Programmer is not concerned with whether the stack is stored before or after the Sqrt() function.



Logical View of Segmentation

- In the figure, both user space and physical memory space are given.
- Segments vary in length, and the length of each is intrinsically defined by its purpose in the program.
- Segments are numbered as 1,2,3 and 4.
- In Physical memory space, Segments are allocated in different parts of the memory
- For example,
 Elements within a segment are identified by their offset from the beginning of the segment: the first statement of the program, the seventh stack frame entry in the stack and the fifth instruction of the Sqrt()



Example – Compilation of C program

- Normally, when a program is compiled, the compiler automatically constructs segments reflecting the input program.
- A “C” compiler might create separate segments for the following:
 - The code
 - Global variables
 - The heap, from which memory is allocated
 - The stacks used by each thread
 - The standard C library
- Libraries that are linked in during compile time might be assigned separate segments.
- The loader would take all these segments and assign them segment numbers.

Segmentation Architecture

- Logical address consists of a two tuple:
<segment-number, offset>
- A logical address consists of two parts: a **segment number, s** , and an **offset** into that segment, d .
- The **segment number** is used as an index to the segment table.
- The **offset d** of the logical address must be between 0 and the segment limit.
- If it is not, a **trap** is sent to the operating system (logical addressing attempt beyond end of segment).
- When an offset d is legal, it is added to the segment base to produce the address in physical memory of the desired byte.

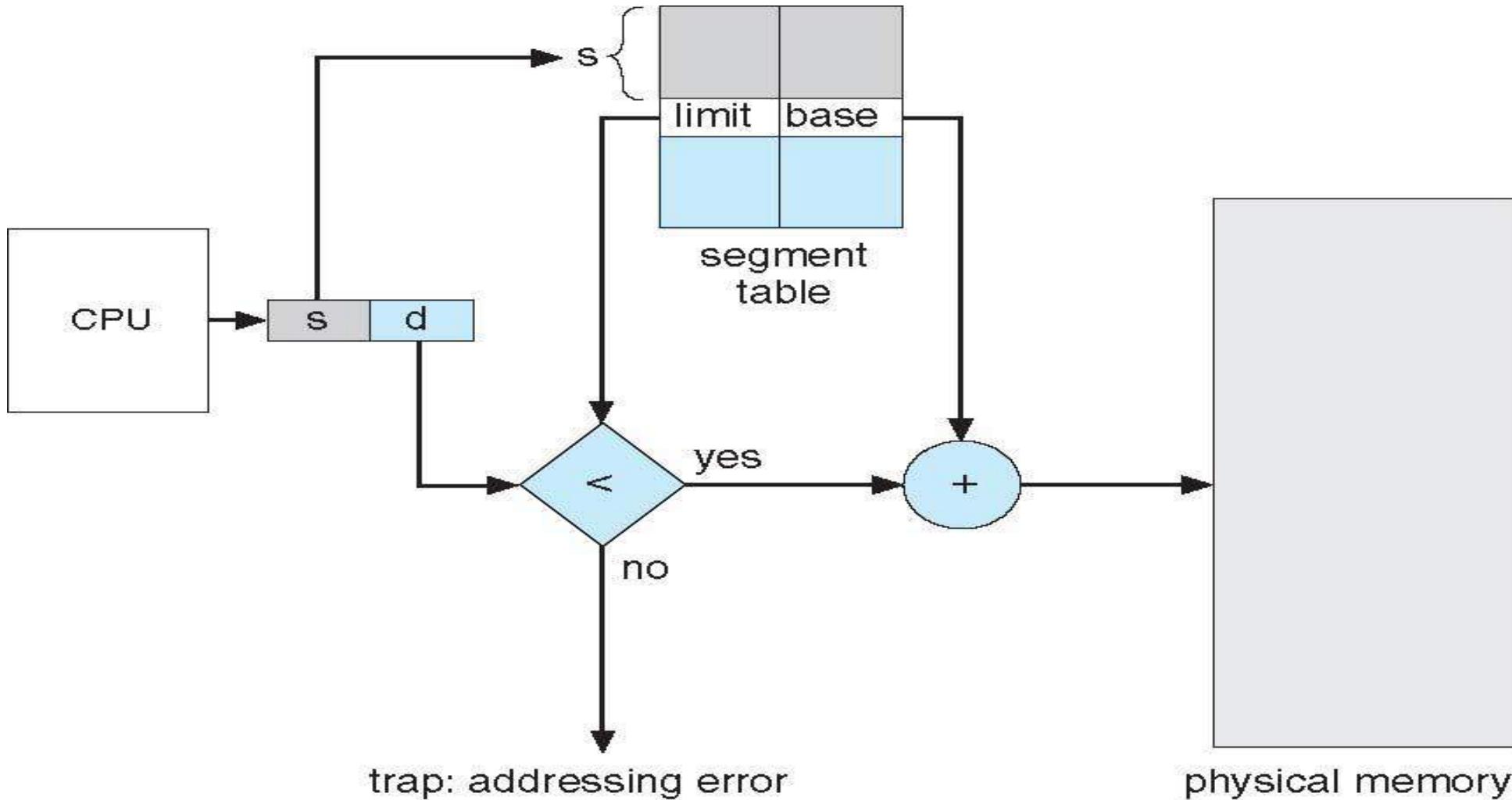
Segmentation Architecture (Cont.)

- **Segment table** – maps two-dimensional physical addresses; each table entry has:
 - **base** – contains the starting physical address where the segments reside in memory
 - **limit** – specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location in memory
- **Segment-table length register (STLR)** indicates number of segments used by a program;
segment number **s** is legal if **s < STLR**

Segmentation Architecture (Cont.)

- Protection
 - With each entry in segment table associate:
 - validation bit = 0 \Rightarrow illegal segment
 - read/write/execute privileges
- Protection bits associated with segments with code sharing occurs at segment level
- Since segments varies in length, memory allocation is a dynamic

Segmentation Hardware



Segmentation Example

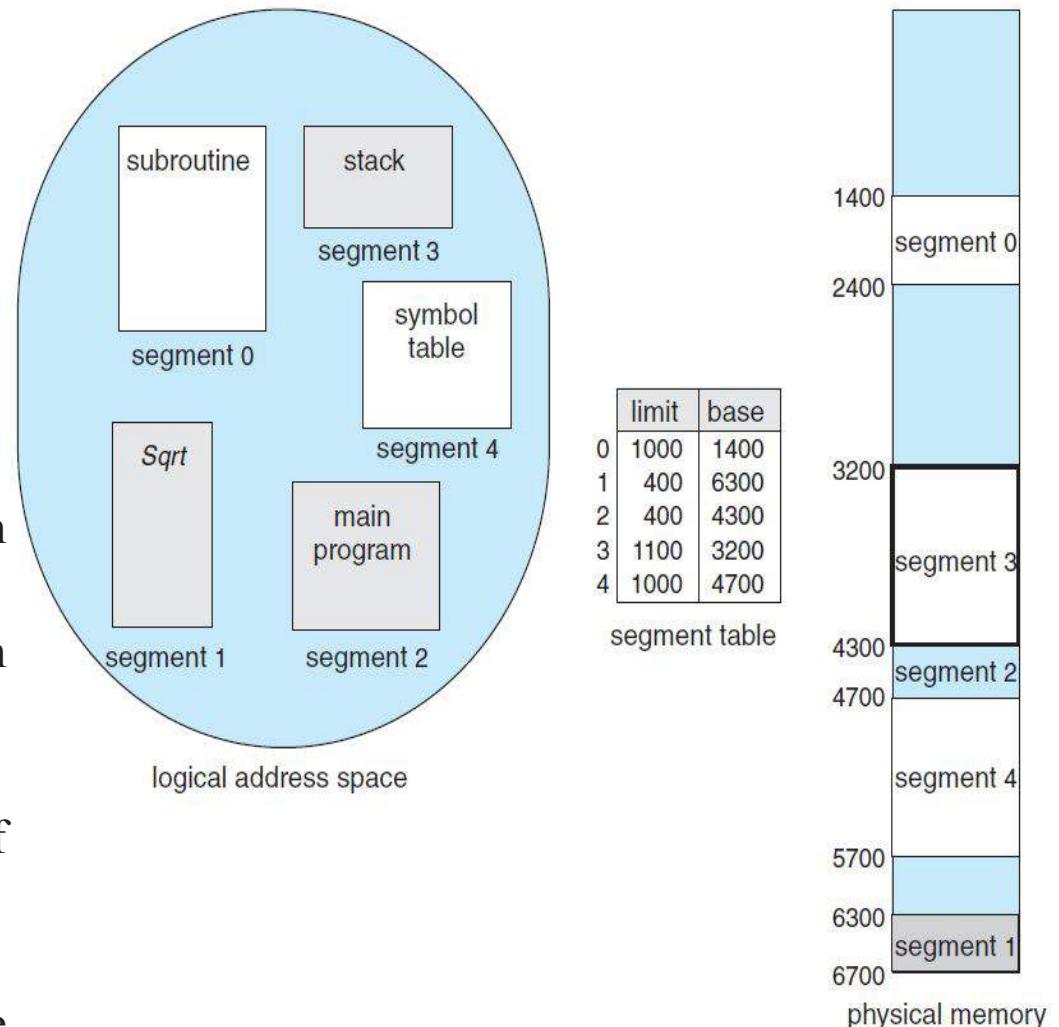
- There are Five Segments numbered from 0 through 4.
- The segments are stored in physical memory as shown in the diagram.
- The segment table has a separate entry for each segment, giving the beginning address of the segment in physical memory (or base) and the length of that segment (or limit).

For example, segment 2 is 400 bytes long and begins at location 4300.

A reference to byte 53 of segment 2 is mapped onto location $4300 + 53 = 4353$.

A reference to segment 3, byte 852, is mapped to 3200 (the base of segment 3) + 852 = 4052.

A reference to byte 1222 of segment 0 would result in a **trap** to the operating system, as this segment is only 1,000 bytes long.



Paged Segmentation

- Paging + Segmentation
- Takes advantage of both paging and segmentation
- Implementation: Treat each segment independently. Each segment has a page table
- Logical address now consists of 3 parts:
 - Segment number
 - Page number
 - Offset

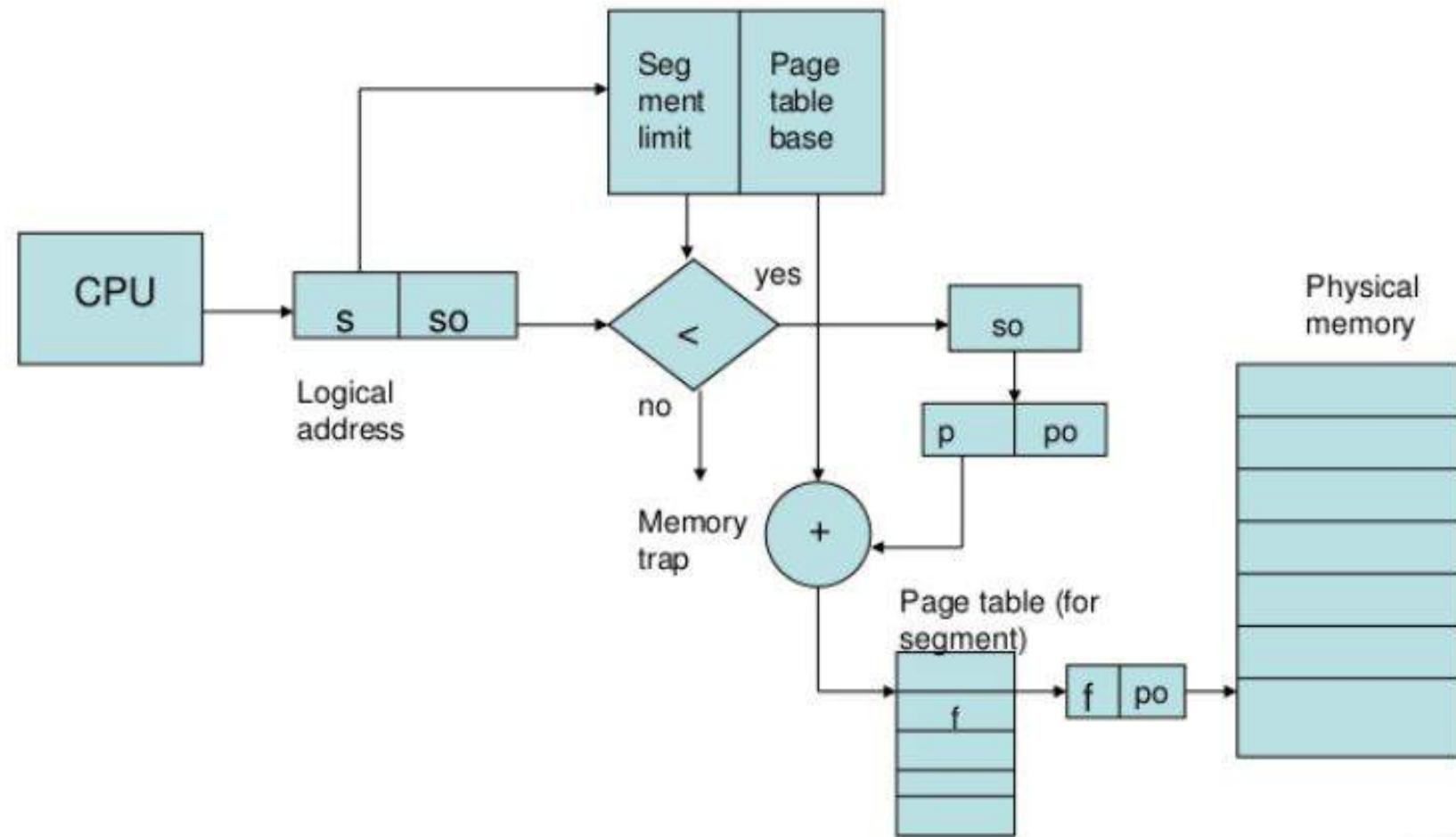
Paged Segmentation Implementation

- Segment table has base address for page table
- Look up <page number, offset> in page table
- Get physical address in return
- Examples: MULTICS and Intel 386

Combined Paging and Segmentation Scheme

- In a combined paging/segmentation system, a user's address is broken up into a number of segments.
- Each segment is broken up into a number of fixed-sized pages which are equal in length to the main memory frame

Address Translation

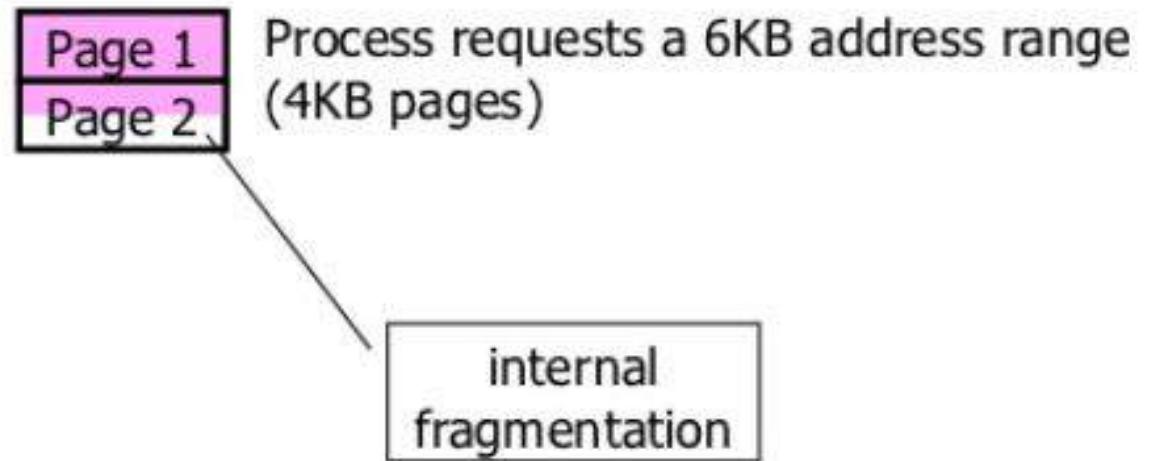


Advantages of Combined Scheme

- Reduces memory usage as opposed to pure paging
 - Page table size limited by segment size
 - Segment table has only one entry per actual segment
- Share individual pages by copying page table entries
- Share whole segments by sharing segment table entries, which is the same as sharing the page table for that segment
- Most advantages of paging still hold
 - Simplifies memory allocation
 - Eliminates external fragmentation
- This system combines the efficiency in paging with protection and sharing capabilities of segmentation

Disadvantage

- Internal fragmentation still exists

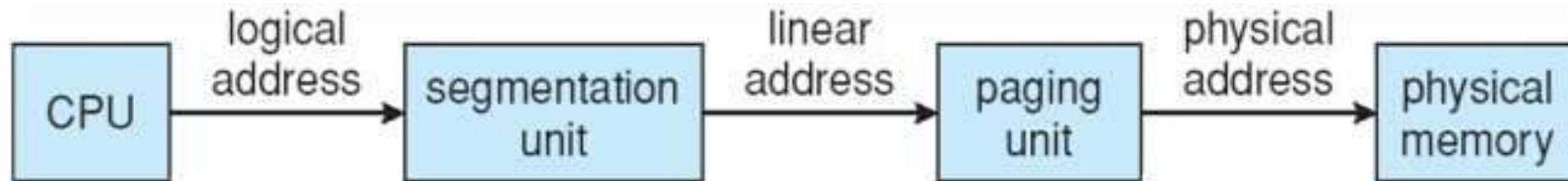


Example: The Intel 32 and 64-bit Architectures

- Dominant industry chips
- Pentium CPUs are 32-bit and called IA-32 architecture
- Current Intel CPUs are 64-bit and called IA-64 architecture
- Many variations in the chips, cover the main ideas here

Example: The Intel IA-32 Architecture

- Memory management is divided as – Segmentation and Paging
- CPU generates logical address
 - Selector given to segmentation unit
 - Which produces linear addresses
 - Linear address given to paging unit
 - Which generates physical address in main memory
 - Paging units form equivalent of MMU
 - Pages sizes can be 4 KB or 4 MB



The Intel IA-32 Segmentation

- Supports both segmentation and paging
 - Each segment can be 4 GB
 - Up to 16 K segments per process
 - Divided into two partitions
 - First partition of up to 8 K segments are private to process (kept in **local descriptor table (LDT)**)
 - Second partition of up to 8K segments shared among all processes (kept in **global descriptor table (GDT)**)

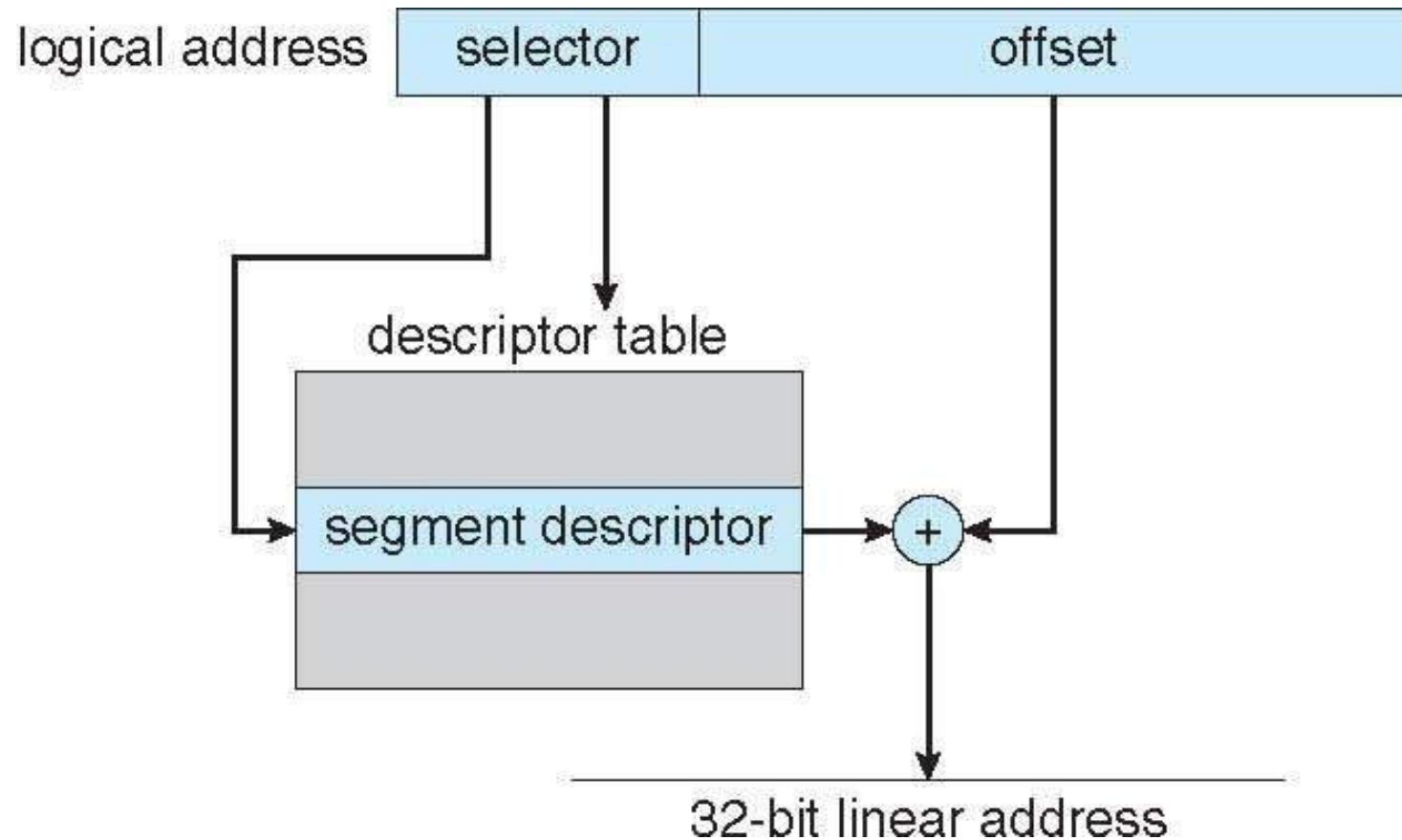
Intel IA-32 Segmentation

- The logical address is a pair (selector, offset), where the selector is a 16-bit number:



- s* designates segment number
- g* indicates if segment is GDT or LDT
- p* deals with protection
- Offset is a 32-bit number and it specifies the location of the byte within segment

Intel IA-32 Segmentation

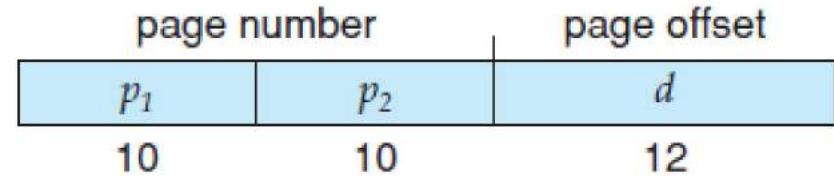


Intel IA-32 Segmentation

- There are six segment registers – six segments can be addressed at a time
- There are six 8-byte microprogram registers to hold LDT or GDT descriptors
- Linear address is 32-bits long
 - Base and limit of segment is used to generate linear address
 - Limit is used to check address validity
 - Invalid address results in a trap
 - Value of base with offset results in 32-bit address

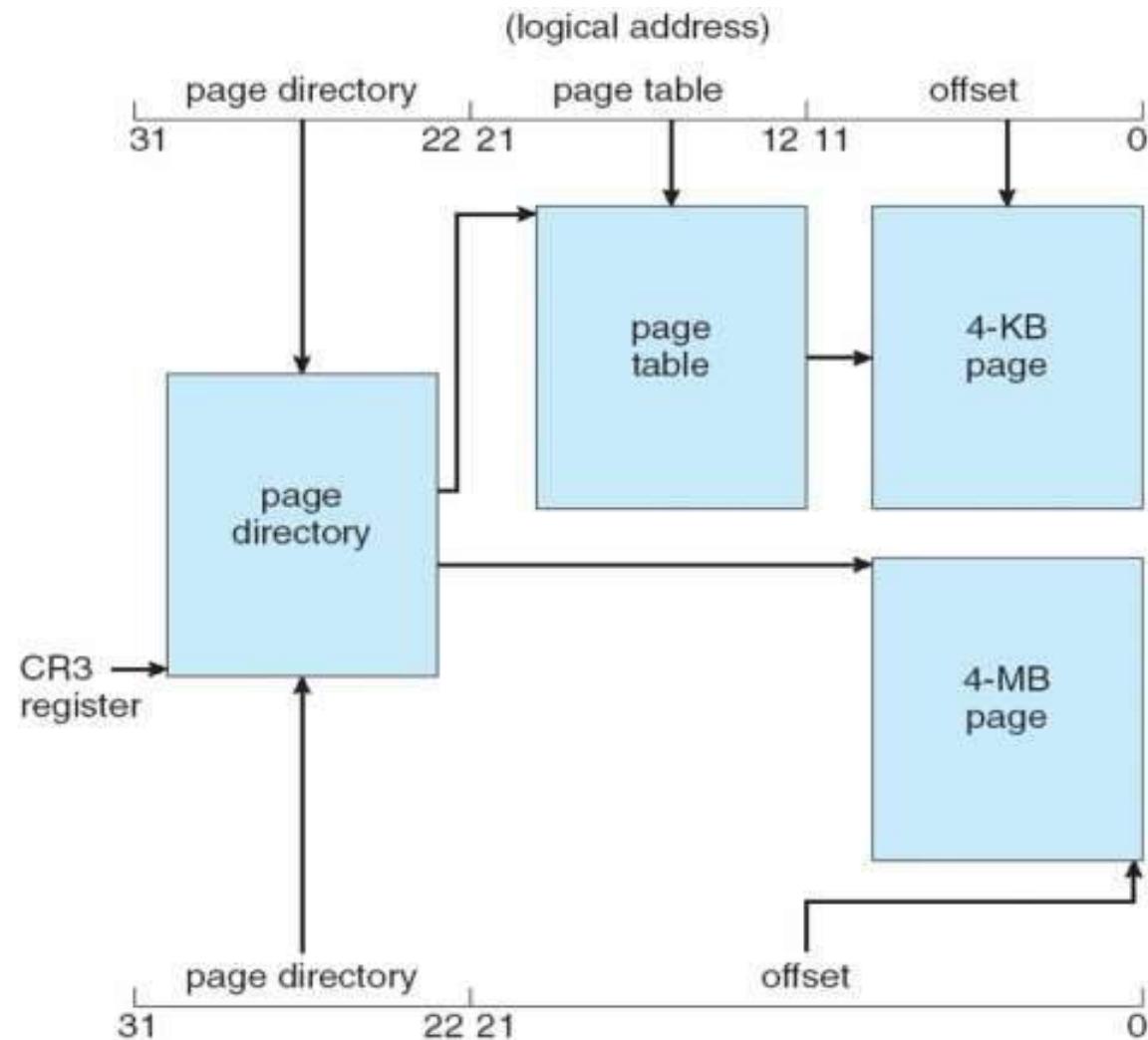
IA-32 Paging

- Pages sizes of 4KB or 4MB is allowed in IA-32
- 4KB pages use two-level paging scheme



- The 10 higher order bits reference the **page directory** – outermost page table
- The innermost 10 bits refer the inner page table
- Lower order 12 bits refer the offset in the page
- 1 bit in page directory is Page-Size flag
 - If set, indicates 4MB page rather than the default 4KB page

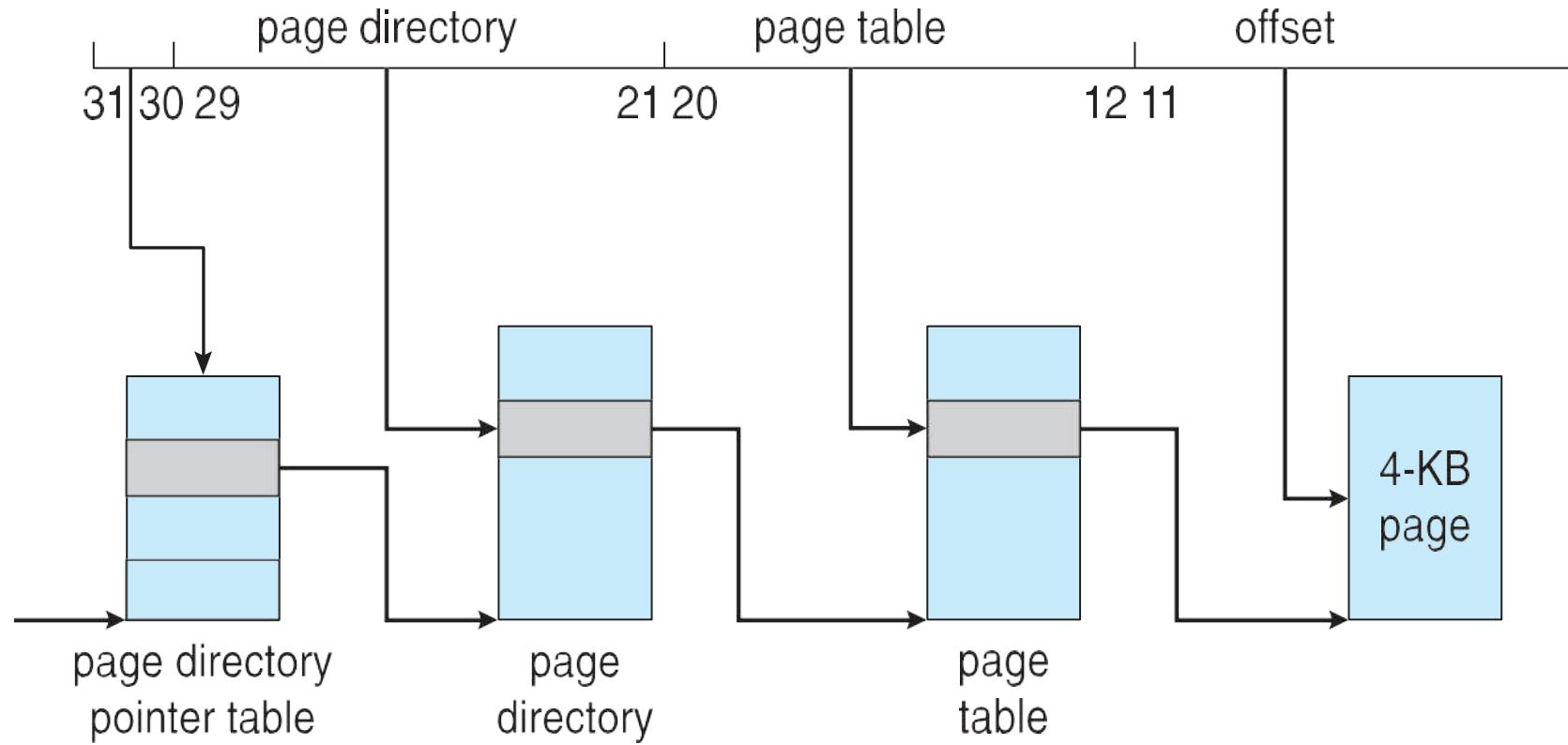
Intel IA-32 Paging Architecture



Intel IA-32 Page Address Extensions

- 32-bit address limits led Intel to create page address extension (PAE), allowing 32-bit apps access to more than 4GB of memory space
- Paging went to a 3-level scheme
- Top two bits refer to a page directory pointer table
- Page-directory and page-table entries moved to 64-bits in size
- Net effect is increasing address space to 36 bits – 64GB of physical memory
- Linux and MAC support PAE
- But, 32-bit versions of Windows desktop operating systems still provide support for only 4 GB of physical memory, even if PAE is enabled.

Intel IA-32 Page Address Extensions



Intel x86-64

- Current generation Intel x86 architecture
 - Support for larger physical and logical address spaces
 - 64 bit-systems can address 2^{64} of addressable memory (16 exabytes)
 - In practice, only implement 48 bit addressing
 - Page sizes of 4 KB, 2 MB, 1 GB
 - Four levels of paging hierarchy
 - Can also use PAE so virtual addresses are 48 bits and physical addresses are 52 bits
- | | | | | | |
|--------|---------|---------------------------------|-------------------|---------------|--------|
| unused | level 4 | page directory
pointer table | page
directory | page
table | offset |
|--------|---------|---------------------------------|-------------------|---------------|--------|

Reference

- Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Operating systems, 9th ed., John Wiley & Sons, 2013



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,
CHENNAI.**

18CSC205J-Operating Systems

Unit- IV



18CSC205J Operating Systems

Unit IV

Course Learning Rationale

- Emphasize the importance of Memory Management concepts of an Operating system
- Realize the significance of Device Management part of an Operating system

Course Learning Outcomes

- Understand the need of Memory Management functions of an Operating system
- Find the significance of Device management role of an Operating system

Learning Resources

- Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Operating systems, 9th ed., John Wiley & Sons, 2013
- William Stallings, Operating Systems-Internals and Design Principles, 7th ed., Prentice Hall, 2012
- Andrew S.Tanenbaum, Herbert Bos, Modern Operating systems, 4th ed., Pearson, 2015
- Bryant O'Hallaxn, Computer systems- A Programmer "s Perspective, Pearson, 2015

Contents

- Virtual Memory– Background
- Understanding the need of demand Paging
- Virtual Memory – Basic concepts – page fault handling
- Understanding, how an OS handles the page faults
- Performance of Demand paging
- Understanding the relationship of effective access time and the page fault rate
- Copy-on write,
- Understanding the need for Copy-on write
- Page replacement Mechanisms: FIFO, Optimal, LRU and LRU approximation Techniques
- Understanding the Pros and cons of the page replacement techniques
- Counting based page replacement and Page Buffering Algorithms
- To know on additional Techniques available for page replacement strategies
- Allocation of Frames - Global Vs. Local Allocation
- Understanding the root cause of the Thrashing
- Thrashing, Causes of Thrashing
- Understanding the Thrashing
- Working set Model
- Understanding the working set model for controlling the Working set Model

Virtual Memory

Virtual memory is a technique that allows the execution of processes that are not completely in Main memory.

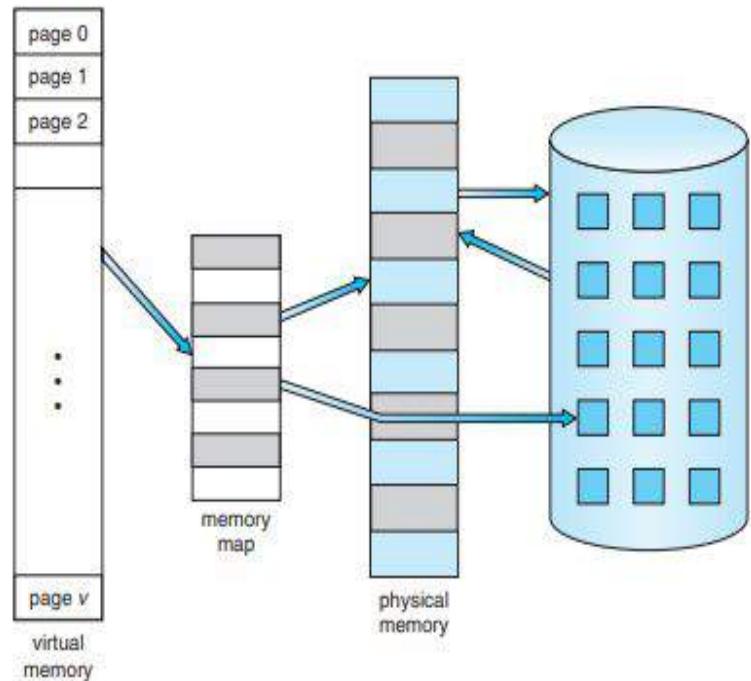
Virtual memory – involves the separation of logical memory as perceived by users from physical memory.

Need for virtual memory

- Programs often have code to handle unusual error conditions. Since these errors seldom, if ever, occur in practice, this code is almost never executed.
- Arrays, lists, and tables are often allocated more memory than they actually need. An array may be declared 100 by 100 elements, even though it is seldom larger than 10 by 10 elements.
- An assembler symbol table may have room for 3,000 symbols, although the average program has less than 200 symbols. Certain options and features of a program may be used rarely. Even in those cases where the entire program is needed, it may not all

Virtual Memory

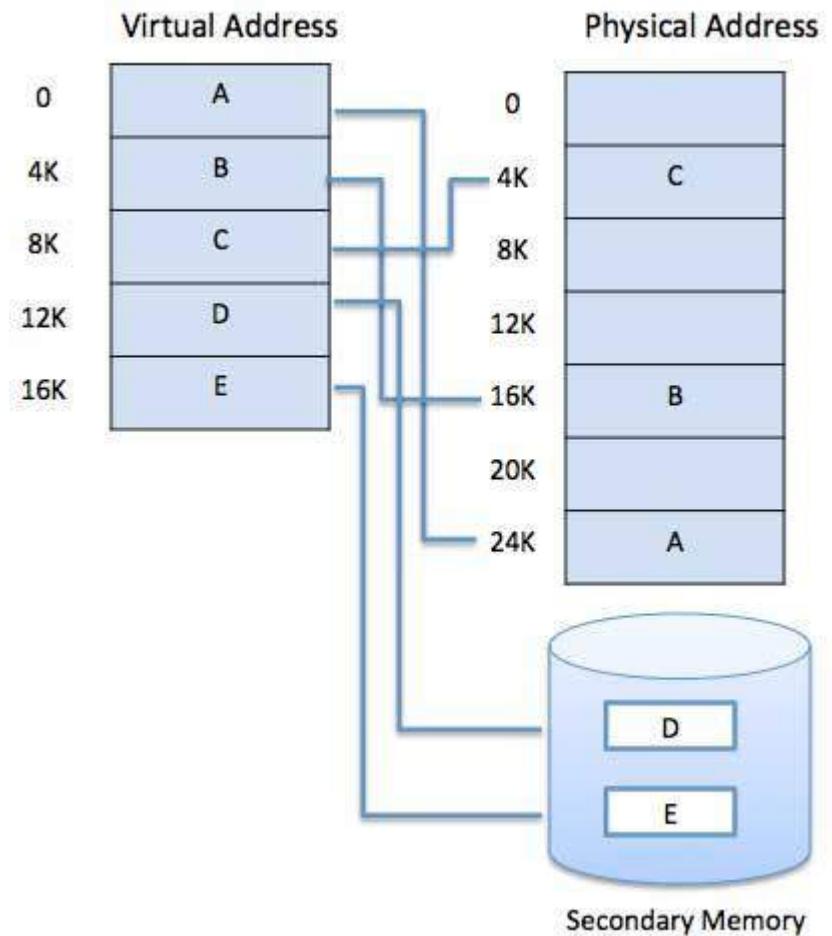
- One major advantage of this scheme is that programs can be larger than physical memory.
- Further, virtual memory abstracts main memory into an extremely large, uniform array of storage, separating logical memory as viewed by the user from physical memory.
- This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available.



- Virtual memory makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available.
- This technique frees programmers from the concerns of memory-storage limitations.
- Virtual memory also allows processes to share files easily and to implement shared memory.
- In addition, it provides an efficient mechanism for process creation.

Virtual Memory

- A program would no longer be constrained by the amount of physical memory that is available.
- Users would be able to write programs for an extremely large virtual address space, simplifying the programming task.
- Because each user program could take less physical memory.
- More programs could be run at the same time, with a corresponding increase in CPU utilization and throughput.
- With no increase in response time or turnaround time.

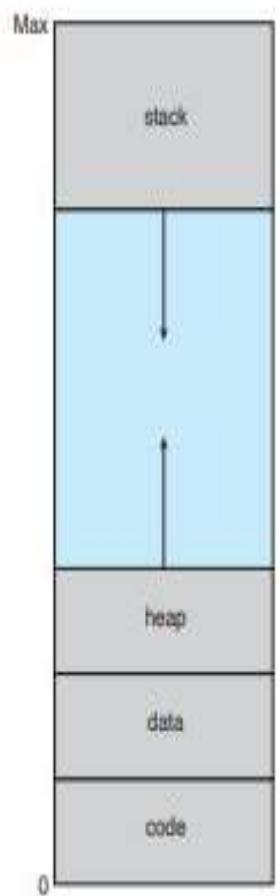


Virtual address space

- The virtual address space of a process refers to the logical (or virtual) view of how a process is stored in memory.
- The virtual address space for a process is the set of virtual memory addresses that it can use.
- The address space for each process is private and cannot be accessed by other processes unless it is shared.
- A virtual address does not represent the actual physical location of an object in memory.
- Instead, the system maintains a page table for each process, which is an internal data structure used to translate virtual addresses into their corresponding physical addresses.
- Each time a thread references an address, the system translates the virtual address to a physical address.

Virtual address space

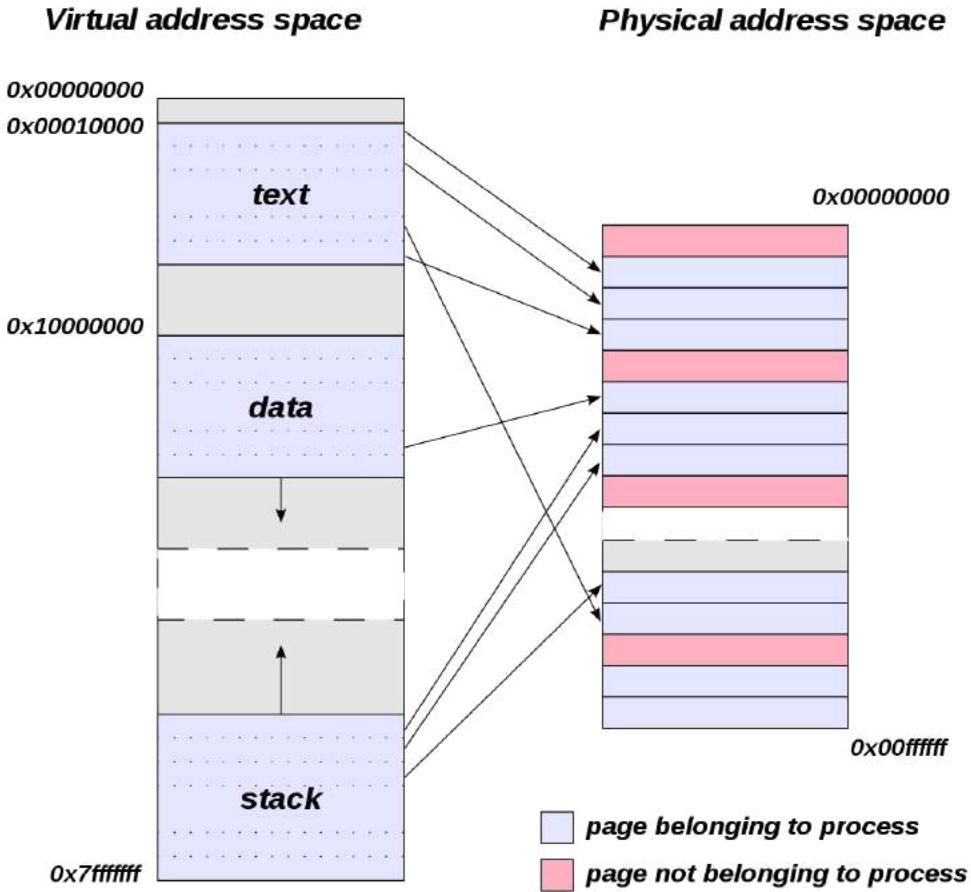
- In this view, a process begins at a certain logical address—say, address 0—and exists in contiguous memory, as shown in Figure.
- Here it is allowed the heap to grow upward in memory as it is used for dynamic memory allocation.
- Allows the stack to grow downward in memory through successive function calls.
- The large blank space (or hole) between the heap and the stack is part of the virtual address space but will require actual physical pages only if the heap or stack grows.
- Virtual address spaces that include holes are known as sparse address spaces.
- Using a sparse address space is beneficial because the holes can be filled as the stack or heap segments grow or if we wish to dynamically link libraries (or possibly other shared objects) during program execution.



Virtual address space

Virtual Address Space vs Physical Address Space

- Address uniquely identifies a location in the memory.
- The logical address is a virtual address and can be viewed by the user.
- The user can't view the physical address directly.
- The logical address is used like a reference, to access the physical address.
- **Logical address** is generated by CPU during a program execution whereas, the **physical address** refers to a location in the memory unit.



Shared Library Using Virtual Memory

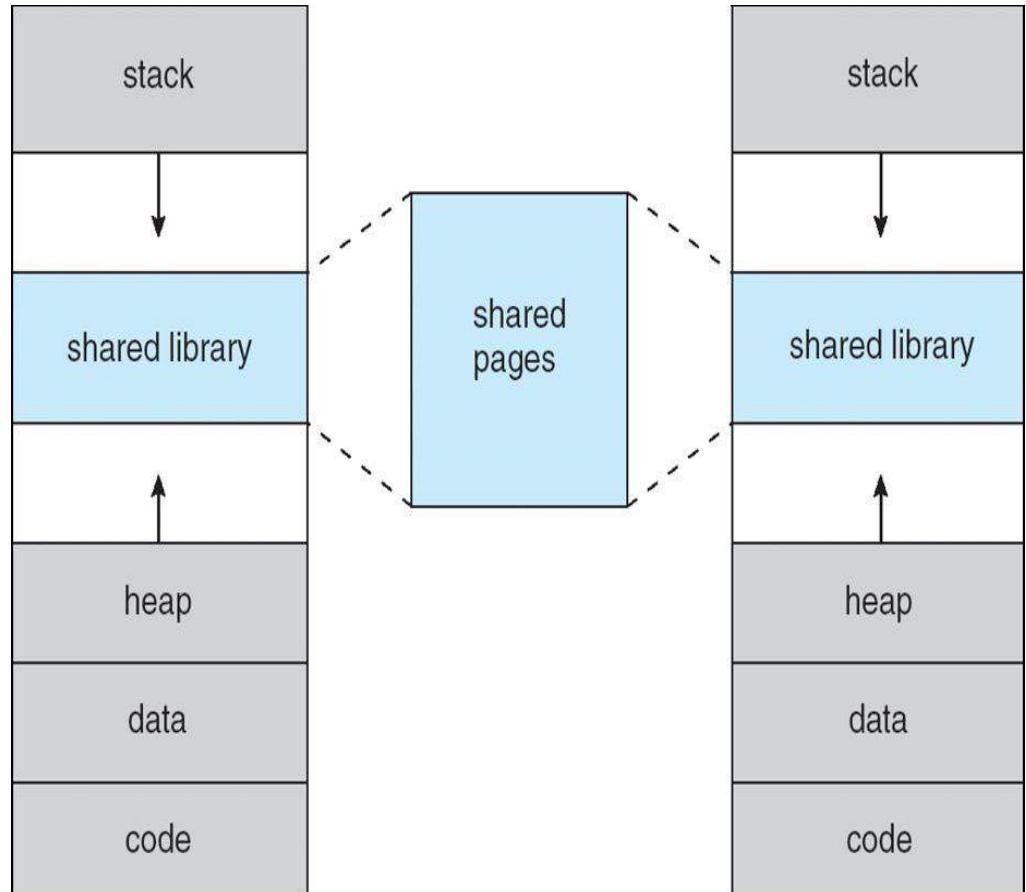
- In addition to separating logical memory from physical memory, virtual memory allows files and memory to be shared by two or more processes through page sharing.

This leads to the following benefits:

- System libraries can be shared by several processes through mapping of the shared object into a virtual address space.
- Although each process considers the libraries to be part of its virtual address space, the actual pages where the libraries reside in physical memory are shared by all the processes.
- Typically, a library is mapped read-only into the space of each process that is linked with it, Similarly, processes can share memory.
- Two or more processes can communicate through the use of shared memory.
- Pages can be shared during process creation with the fork() system call, thus speeding up process creation.

Shared Library Using Virtual Memory

- Virtual memory allows one process to create a region of memory that it can share with another process.
- Processes sharing this region consider it part of their virtual address space, yet the actual physical pages of memory are shared, much as is illustrated in Figure.



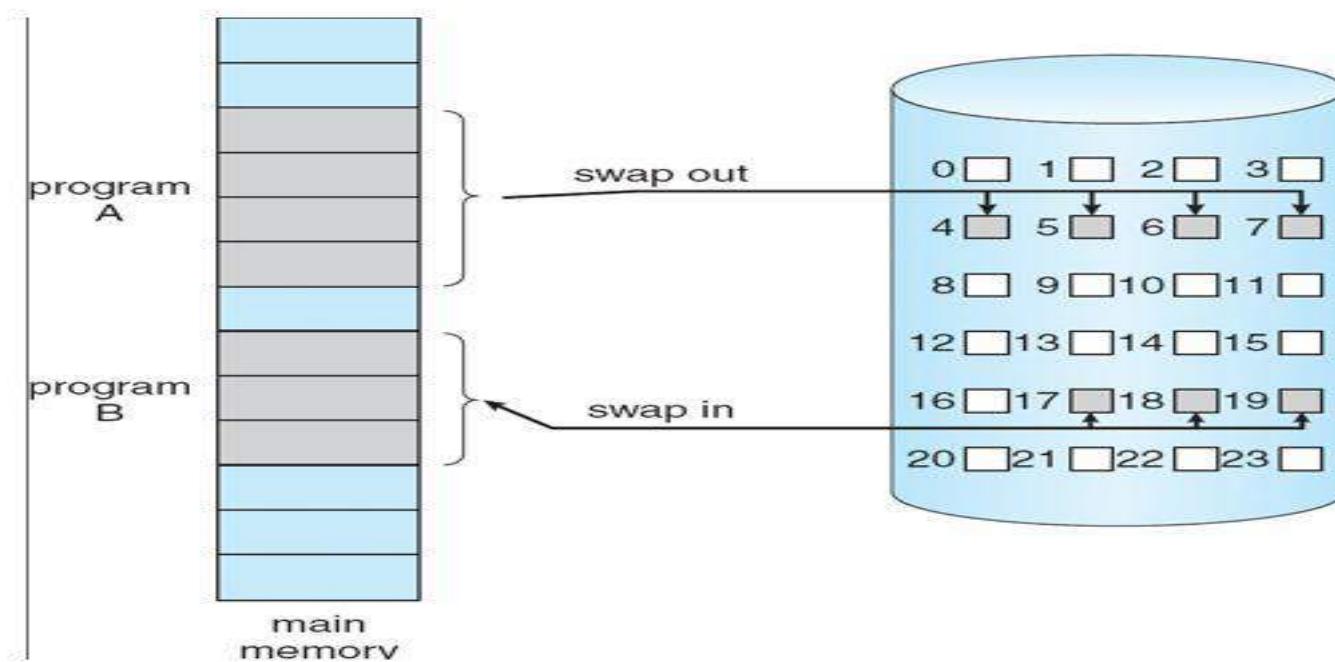
Demand Paging – Basic Concepts

How an executable program might be loaded from disk into memory?

- Load the entire program in physical memory at program execution time.
 - problem of this method, we may not initially need the entire program in memory
- An alternate approach to resolve this problem to load pages only as they are needed. This method called as **Demand Paging**.
- Demand paging commonly used in virtual memory systems.
- Demand paged virtual memory, pages are loaded only when they are needed during program execution.

Demand Paging

- A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance.



Demand Paging

- In the context of a demand-paging system, use of the term “swapper” is technically incorrect.
- Because **swapper manipulates entire process into memory**, whereas a pager is concerned with the individual pages of a process.
- We thus use “pager,” rather than “swapper,” in connection with demand paging.
- **Lazy swapper** – will not swap the entire process into memory, A lazy swapper never swaps a page into memory unless that page will be needed.

Valid-Invalid Bit

- Need some form of hardware support to distinguish between the pages that are in memory and the pages that are on the disk.
- The **valid–invalid bit** scheme can be used for this purpose.
- With each page table entry a valid–invalid bit is associated ($v \Rightarrow$ in-memory – **memory resident**, $i \Rightarrow$ not-in-memory)
- Initially valid–invalid bit is set to **i** on all entries.
- Example of a page table snapshot:

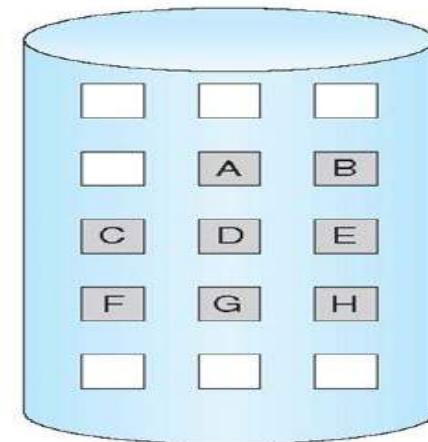
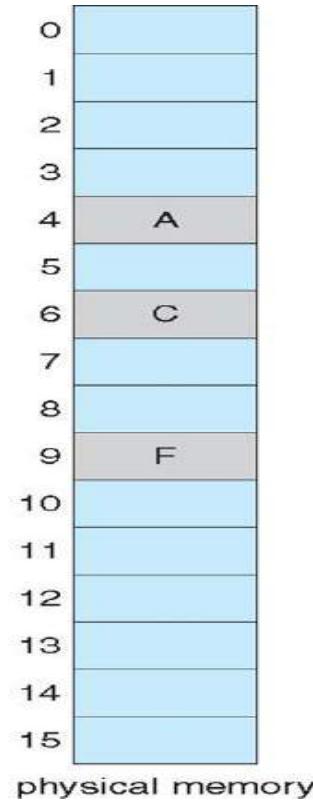
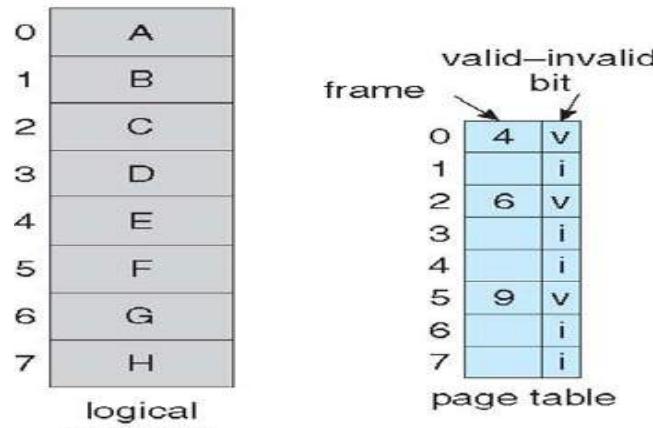
Frame #	valid-invalid bit
	v
	v
	v
	i
...	
	i
	i

page table

Page Table

When Some Pages Are Not in Main Memory

- **Page fault** - if the process tries to access a page that was not brought into memory.
- Operating system's failure to bring the desired page into memory.

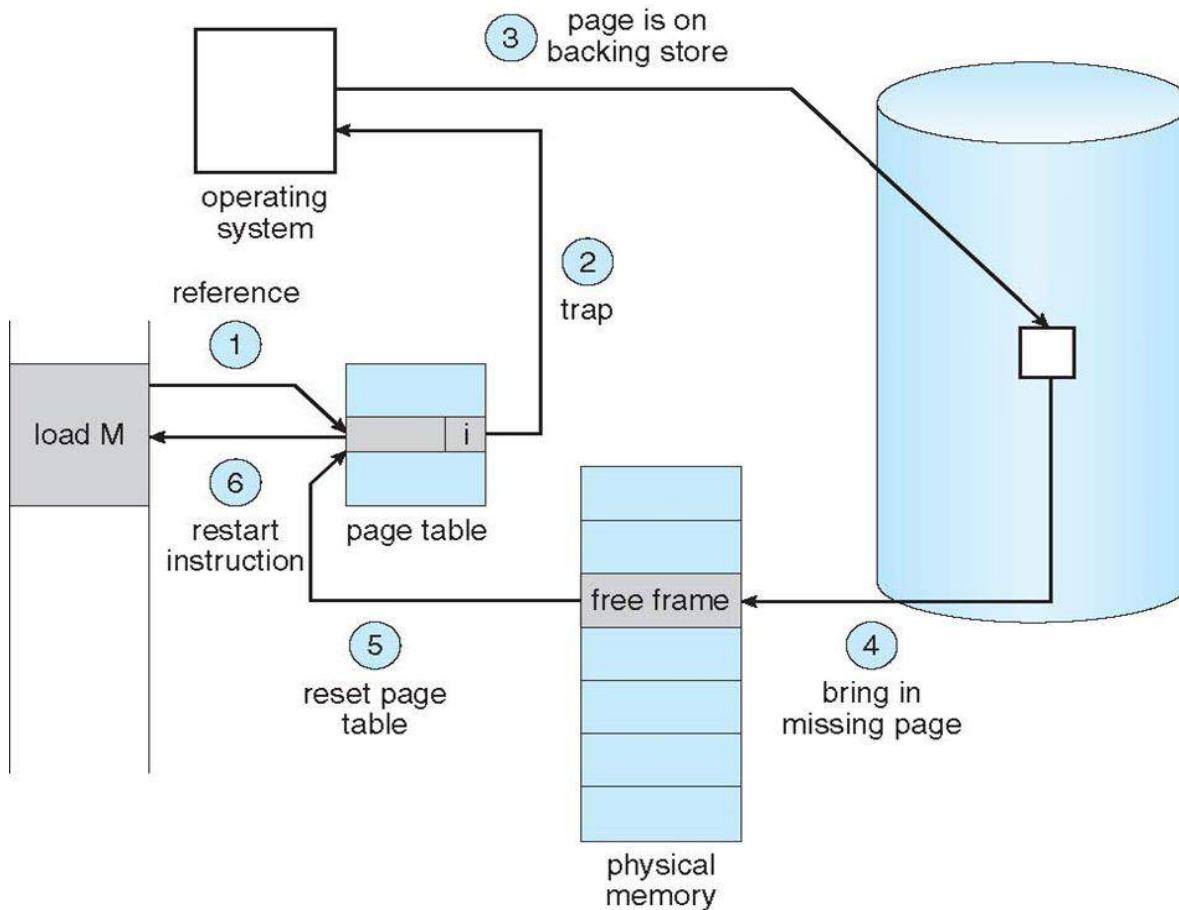


Page Fault

The procedure for handling page fault

1. Check an internal table (usually kept with the process control block) whether the reference was a valid or an invalid memory access.
2. If the reference
 1. **Invalid**, we terminate the process.
 2. **Valid** but we have not yet brought in that page, we now page it in.
3. Find a free frame
4. Schedule a disk operation to read the desired page into the newly allocated frame.
5. When the disk read is complete, modify the internal table kept with the process and the page table to indicate that the page is now in memory.
6. We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.

Steps in Handling a Page Fault



Aspects of Demand Paging

- Extreme case – start process with *no* pages in memory
 - OS sets instruction pointer to first instruction of process, non-memory-resident -> page fault
 - After this page is brought into memory, the process continues to execute, faulting as necessary until every page that it needs is in memory.
 - At that point, it can execute with no more faults.
 - This scheme is **pure demand paging**: never bring a page into memory until it is required.
- The hardware to support demand paging
 - **Page table** - This table has the ability to mark an entry invalid through a valid–invalid bit or a special value of protection bits.
 - **Secondary memory** - This memory holds those pages that are not present in main memory.
 - The secondary memory is usually a high-speed disk. It is known as the **swap device**

Performance of Demand Paging

- Demand paging can significantly **affect the performance** of a computer system.
- **Memory-access time** denoted **ma**, ranges from 10 to 200 nanoseconds. If there is no page faults, the **effective access time** is equal to the memory access time.
- If page fault occurs, we must first read the relevant page from disk and then access the desired word.
- Let **p** be the probability of a page fault ($0 \leq p \leq 1$).
- We would expect **p to be close to zero**, expect to have only a few page faults.
- The effective access time is then
 - effective access time = $(1 - p) \times ma + p \times \text{page fault time.}$**
- To compute the effective access time, we must know how much time is needed to service a page fault.

Performance of Demand Paging

A page fault causes the following sequence to occur:

1. Trap to the operating system
2. Save the user registers and process state
3. Determine that the interrupt was a page fault
4. Check that the page reference was legal and determine the location of the page on the disk
5. Issue a read from the disk to a free frame:
 1. Wait in a queue for this device until the read request is serviced
 2. Wait for the device seek and/or latency time
 3. Begin the transfer of the page to a free frame
6. While waiting, allocate the CPU to some other user
7. Receive an interrupt from the disk I/O subsystem (I/O completed)
8. Save the registers and process state for the other user
9. Determine that the interrupt was from the disk
10. Correct the page table and other tables to show page is now in memory
11. Wait for the CPU to be allocated to this process again
12. Restore the user registers, process state, and new page table, and then resume the interrupted instruction

Performance of Demand Paging

- Not all 12 steps are necessary in every case.
- When the CPU is allocated to another process while waiting I/O occurs.
 - Allows **multiprogramming** to maintain CPU utilization but **requires additional time** to resume the page-fault service routine when the I/O transfer is complete.
- In any case, we are faced with three major components of the page-fault service time:
 - Service the page-fault interrupt.
 - Read in the page.
 - Restart the process.

Performance of Demand Paging

- The first and third tasks can be reduced, with careful coding to several hundred instructions. These tasks may take from 1 to 100 microseconds each.
- Probably, The page-switch time - 8 milliseconds.
 - A typical hard disk has an average latency of 3 milliseconds,
 - A seek of 5 milliseconds
- Remember also that we are looking at only the device-service time.
- If a queue of processes is waiting for the device, we have to **add device-queueing time**

Performance of Demand Paging

For example,

Average page-fault service time - 8 milliseconds

memory access time - 200 nanoseconds

The effective access time?

$$\text{effective access time} = (1 - p) \times \text{ma} + p \times \text{page fault time}.$$

$$\text{effective access time} = (1 - p) \times (200) + p (8 \text{ milliseconds})$$

$$= (1 - p) \times 200 + p \times 8,000,000$$

$$= 200 + 7,999,800 \times p.$$

Let **p** be the probability of a page fault ($0 \leq p \leq 1$)

- We see, then, that the effective access time is **directly proportional** to the **page-fault rate**.

Performance of Demand Paging

- Swap space I/O faster than file system I/O even if on the same device
 - Swap allocated in larger chunks, less management needed than file system
- Copy entire process image to swap space at process load time
 - Then page in and out of swap space
 - Used in older BSD Unix
- Demand page in from program binary on disk, but discard rather than paging out when freeing frame
 - Used in Solaris and current BSD
 - Still need to write to swap space
 - Pages not associated with a file (like stack and heap) – anonymous memory
 - Pages modified in memory but not yet written back to the file system
- Mobile systems
 - Typically don't support swapping
 - Instead, demand page from file system and reclaim read-only pages (such as code)

Demand Paging

Advantages

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

Disadvantages

- Number of tables and the amount of processor over head for handling page interrupts are greater than in the case of the simple paged management techniques.

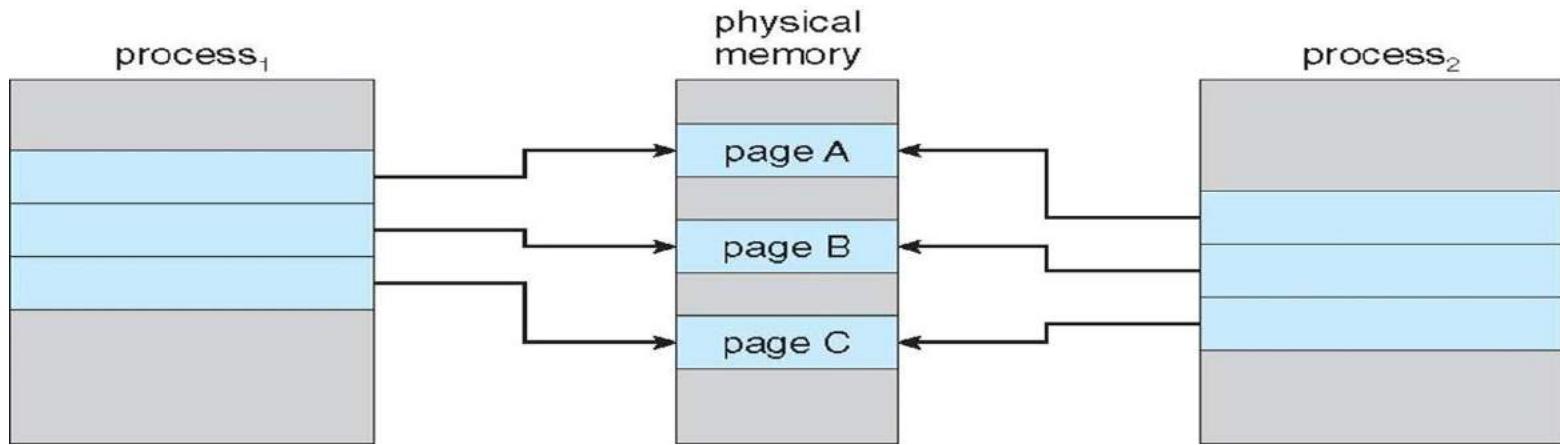
Copy-on-Write

- The fork() system call creates a child process that is a duplicate of its parent. Creating a **copy of the parent's address space for the child**, duplicating the pages belonging to the parent.
- Child processes invoke the exec() system call immediately after creation, the copying of the parent's address space
- **Copy-on-Write (COW)** allows both parent and child processes to initially *share* the same pages in memory
 - If either process modifies a shared page, only then is the page copied
- COW allows more efficient process creation as only modified pages are copied

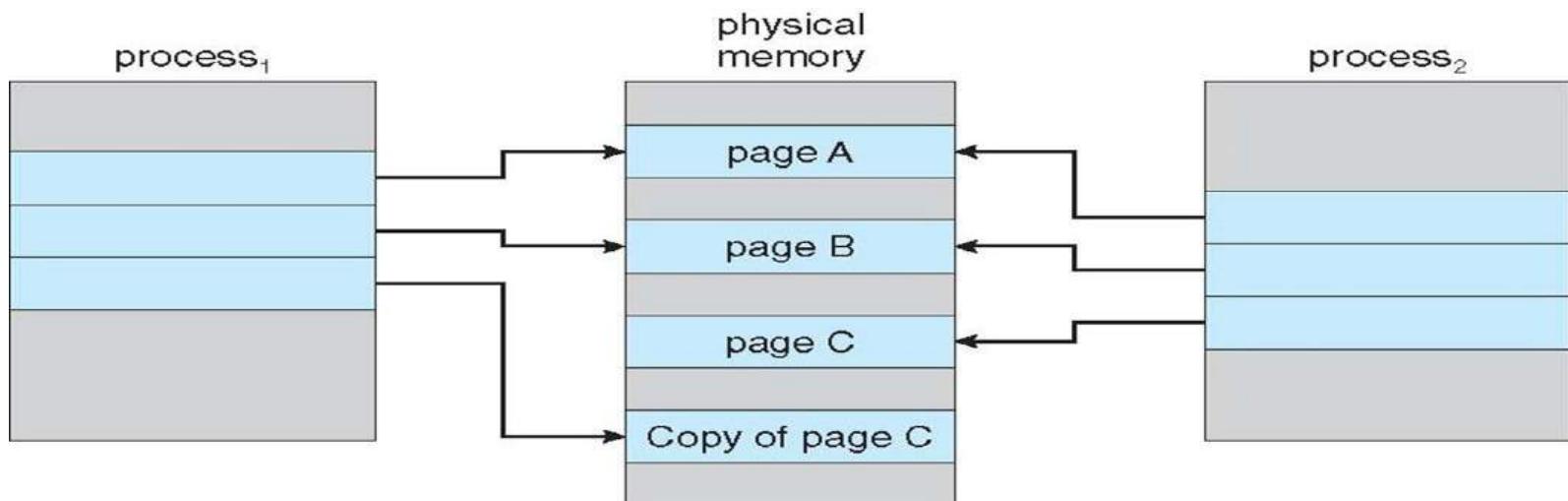
Copy-on-Write

- The child process attempts to modify a page containing portions of the stack, with the pages set to be copy-on-write.
- The operating system will create a copy of this page, mapping it to the address space of the child process.
- The child process modify its copied page and **not the page belonging to the parent process.**
- Obviously, when the copy-on-write technique is used, only the pages that are modified by either process are copied; **all unmodified pages can be shared by the parent and child processes.**
- The following figure shows copy on write on page C

Before Process 1 Modifies Page C



After Process 1 Modifies Page C



Copy-on-Write

- In general, free pages are allocated from a **pool of zero-fill-on-demand pages**
 - Pool should always have free frames for fast demand page execution
 - Don't want to have to free a frame as well as other processing on page fault
 - Why zero-out a page before allocating it?
- vfork() variation on fork() system call has parent suspend and child using copy-on-write address space of parent
 - Designed to have child call exec()
 - Very efficient

Need for Copy-on-Write

- **Copy-on-write or CoW** is a technique to efficiently copy data resources in a computer system.
- If a unit of data is copied but not modified, the "copy" can exist as a reference to the original data.
- Only when the copied data is modified is a copy created, and new bytes are actually written.
- Copy-on-write is closely related to data deduplication.
- Whereas data deduplication analyzes chunks or blocks of data, copy-on-write applies to entire files or allocated units of memory.

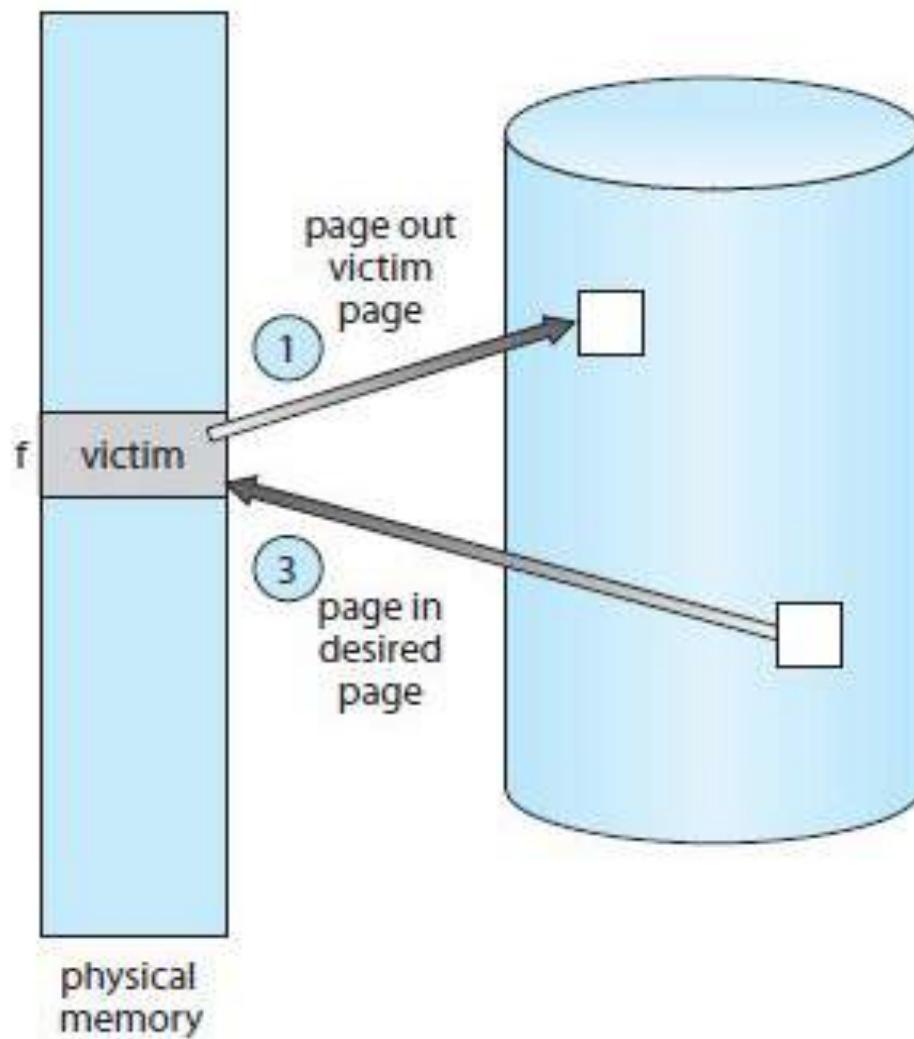
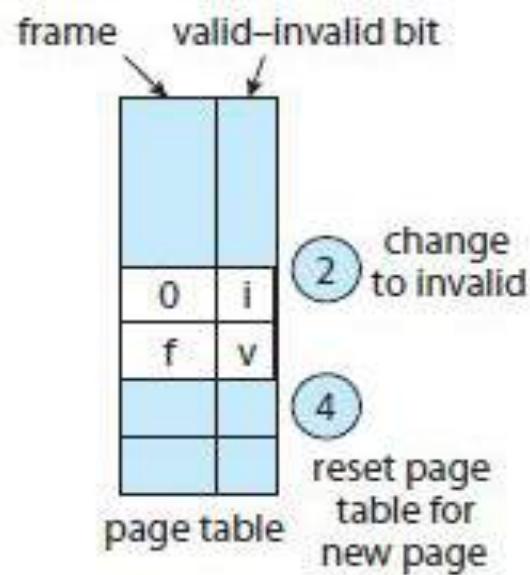
Page Replacement Algorithm

- Page replacement is a process of swapping out an existing page from the frame of a main memory and replacing it with the required page.
- Reason for Page Replacement
 - Page fault
 - A page fault occurs when a page referenced by the CPU is not found in the main memory.
 - The only solution for the page fault is, the required page has to be brought from the secondary memory into the main memory.
 - This is done by replacing the already occupied page in the main memory with the desired page.

Steps in Page Replacement

- Identify the location of the desired page on the disk.
- If the page fault occurs, identify a free frame
- If there is a free frame then locate the desired page into the free frame
- If there is no free frame then use a page-replacement algorithm to select a victim frame that is to be replaced.
 - Now write the victim page to the disk then update the page and frame tables.
 - Read the desired page into the newly freed frame then change the page and frame tables.
 - Continue the process when there is a page fault occurs again

Steps in Page Replacement



Page Replacement Algorithms

- Page replacement algorithms help to decide which page must be swapped out from the main memory to create a room for the incoming page.
- Various page replacement algorithms are
 - First In First Out (FIFO) page replacement algorithm
 - Optimal page replacement algorithm
 - Least Recently Used (LRU) page replacement algorithm
 - LRU Approximation page replacement algorithm
 - Additional Reference Bits Algorithm
 - Second Chance / Clock Algorithm
 - Enhanced Second Chance Algorithm
 - Counting Based page replacement algorithm
 - Least Frequently Used (LRU) page replacement algorithm
 - Most Frequently Used (MRU) page replacement algorithm
 - Page Buffering algorithm

FIFO Page Replacement Algorithm

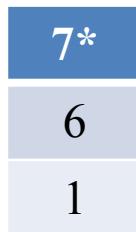
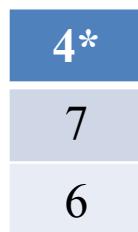
- As the name suggests, this algorithm works on the principle of “**First in First out**“.
- It replaces the oldest page that has been present in the main memory for the longest time.
- It is implemented by keeping track of all the pages in a queue.
- **Steps in FIFO**
 - Bring the page into the queue
 - If page is available then do
 - Else there is a page fault
 - If there is a free frame, locate the desired page which causes the page fault
 - If there is no free frame, identify the victim frame to be replaced
 - The page which comes first into the queue will be chosen as the victim frame.
 - Now this victim frame will be replaced with the desired page and the queue is updated for further process.

FIFO Page replacement algorithms

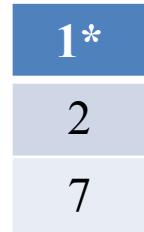
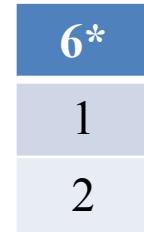
FIFO Page Replacement Algorithm

Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	6	7	7
Frame 2		7	7	7	7	7	7	2	2	2
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Miss	Hit

Number of Page Faults in FIFO = 6



Queue after second replacement

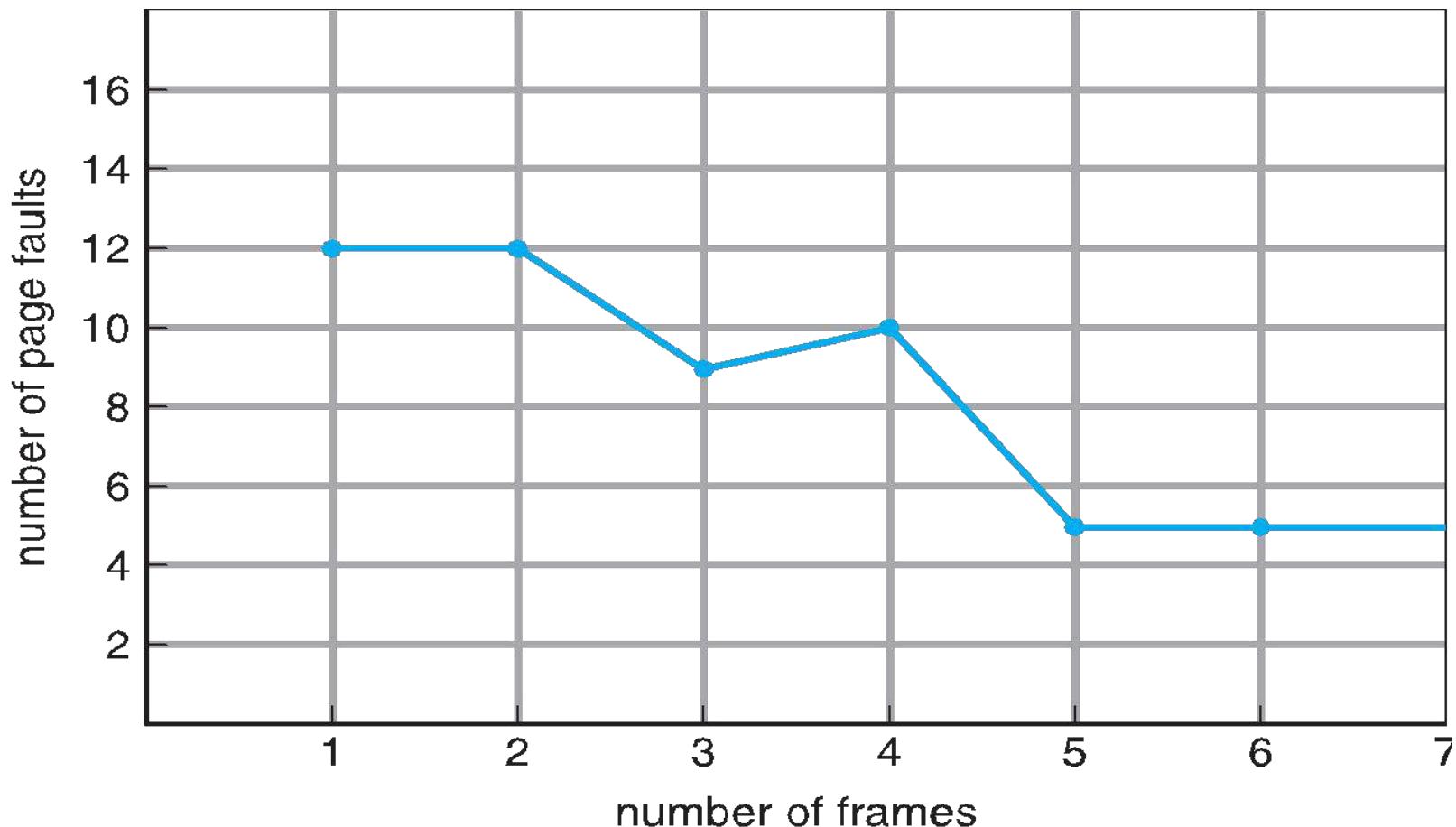


Initial Queue Queue after first replacement

Queue after third replacement

* Indicates the top element of the queue or the element which comes first into the queue

FIFO Illustrating Belady's Anomaly



If number of page frames increases page fault also increases. This is called as Belady's Anomaly

Optimal Page Replacement Algorithm

- This algorithm replaces the page that will not be referred by the CPU in future for the longest time.
- It is practically difficult to implement this algorithm.
- This is because the pages that will not be used in future for the longest time cannot be predicted.
- However, it is the best known algorithm and gives the least number of page faults.
- Hence, it is used as a performance measure criterion for other algorithms.

Optimal Page Replacement Algorithm

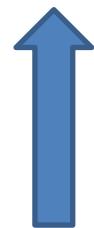
- Steps in the Working Process (This is only for understanding purpose)
 - Create required number of columns based on the reference strings used
 - A pointer is used which moves from MSB to LSB of the columns
 - The pointer identifies the future reference of the strings
 - If the page fault occurs, identify the free frame
 - If there is a free frame, locate the desired page into the free frame
 - If there is no free frames, choose a victim frame that is to replaced with the desired page
 - The victim frame is chosen by the pointer, where the pointer chooses a victim frame which will be referred very later in the future
 - Now the desired page is located in the main memory which caused the page fault

Optimal Page Replacement Algorithm

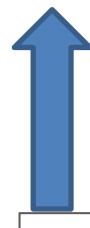
Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	2	2	2
Frame 2		7	7	7	7	7	7	7	7	7
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Hit	Hit

Number of Page Faults in Optimal Page Replacement Algorithm = 5

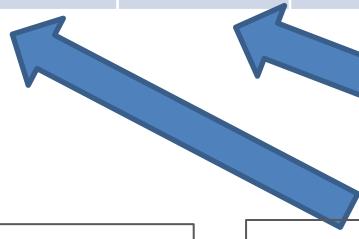
1	2	3	4	5	6	7	8	9	10
4	7	6	1	7	6	1	2	7	2



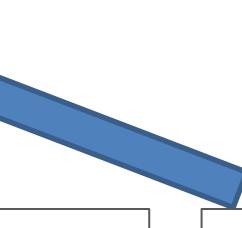
4 is not referred in the future



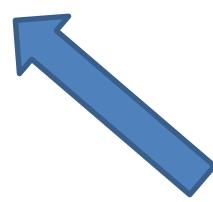
7 is again referred in 5th and 9th position in the future



6 is again referred in 6th position in the future



1 is again referred in 7th position in the future



2 is again referred in 10th position in the future

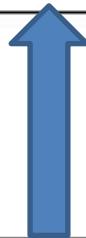
LRU Page Replacement Algorithm

- As the name suggests, this algorithm works on the principle of “**Least Recently Used**“.
- It replaces the page that has not been referred by the CPU for the longest time.
- Use past knowledge rather than future
- Additionally LRU can be implemented using counter and stack

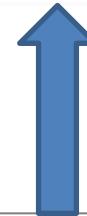
LRU Page Replacement Algorithm

Positions	1	2	3	4	5	6	7	8	9	10
Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	6	7	7
Frame 2		7	7	7	7	7	7	2	2	2
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Miss	Hit

Number of Page Faults in LRU = 6



4 is referred in the 1st position before, so it is replaced



7 is referred in the 2nd position before, so it is replaced



6 is referred in the 3rd position before, so it is replaced

LRU IMPLEMENTATION

Counter implementation

- Every page entry has a counter
- every time page is referenced through this entry
- When a page needs to be changed, look at the counters to find smallest value Search through table

LRU IMPLEMENTATION

Stack implementation

- Keep a stack of page numbers in a double link form
- If Page referenced
 - move it to the top
 - requires 6 pointers to be changed
- But each update more expensive
- No search for replacement
- LRU and OPT are cases of stack algorithms that don't have Belady's Anomaly

Stack to Record Most Recent Page References

reference string

4 7 0 7 1 0 1 2 1 2 7 1 2

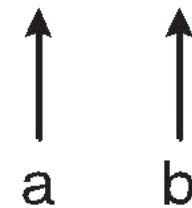


stack
before
a



stack
after
b

a b



Two black arrows point upwards from the reference string to the stack. The first arrow points to the '7' at index 11, and the second arrow points to the '2' at index 12. These indices correspond to the positions of the new page '7' and the page '2' being promoted to the stack, respectively.

LRU Approximation Algorithms

- The reference bit for a page is set by the hardware whenever that page is referenced (either a read or a write to any byte in the page).
- Reference bits are associated with each entry in the page table.
 - With each page associate a bit, initially as ‘0’
 - When page is referenced, set the bit to ‘1’
 - Now replace the page whose bit is ‘0’ if bit ‘1’ exists
 - Order of the page is not known but the used pages are known

LRU Approximation – Additional Reference Bits Algorithm

- Limitations in using single reference bit had been solved by providing the order of the pages
- Additional reference bits are ‘1’ byte in capacity
- Already existing reference bits are shifted to right in the additional reference bits table
- Page with the lowest number is the least recently used which will be chosen for replacement
- FIFO can be used at the time of tie

- Initially all the bits are set to ‘0’

	Reference Bit	Additional Bits						
Page 0	0	0	0	0	0	0	0	0
Page 1	0	0	0	0	0	0	0	0
Page 2	0	0	0	0	0	0	0	0
Page 3	0	0	0	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0

- Pages 1 and 3 are referred

	Reference Bit	Additional Bits						
Page 0	0	0	0	0	0	0	0	0
Page 1	1	0	0	0	0	0	0	0
Page 2	0	0	0	0	0	0	0	0
Page 3	1	0	0	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0

- The referred bit is copied to the MSB of the additional bits and reset the reference bit to ‘0’

	Reference Bit	Additional Bits							
Page 0	0	0	0	0	0	0	0	0	0
Page 1	0	1	0	0	0	0	0	0	0
Page 2	0	0	0	0	0	0	0	0	0
Page 3	0	1	0	0	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0	0

- Now pages 1 and 2 are referred. Then the bits which are present already in the MSB of the Additional bits table is shifted to right one place

	Reference Bit	Additional Bits						
Page 0		0	0	0	0	0	0	0
Page 1	1	0	1	0	0	0	0	0
Page 2	1	0	0	0	0	0	0	0
Page 3	0	0	1	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0

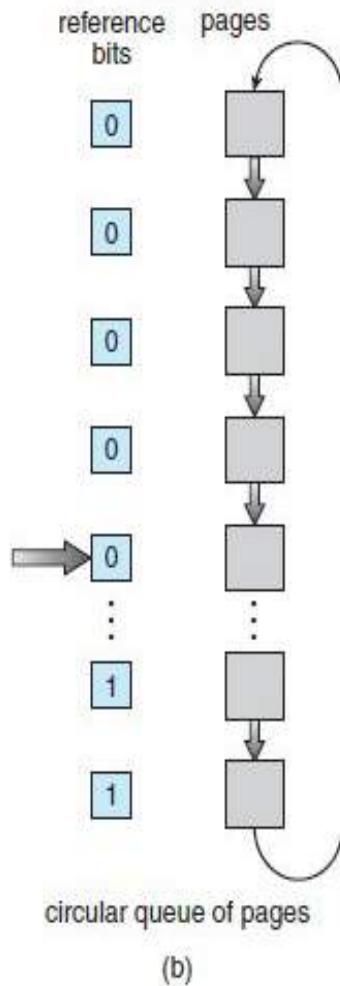
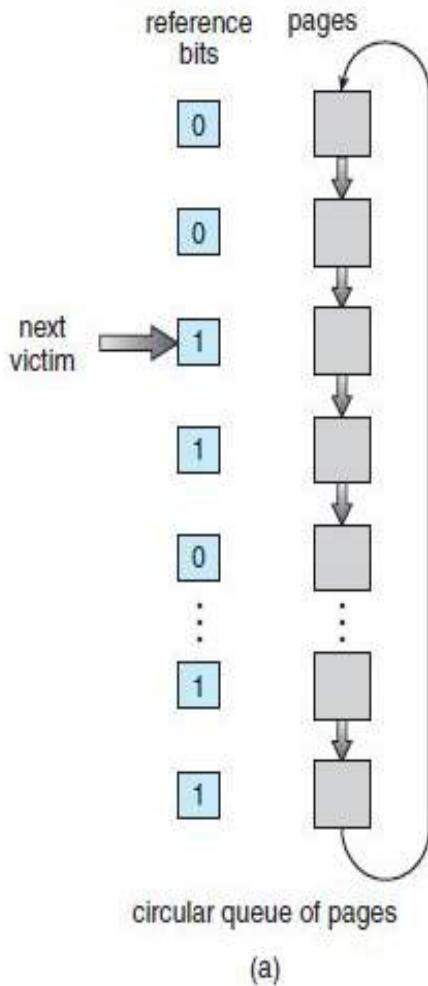
- Again the process continues by copying the referred bits to the MSB of Additional bits table

	Reference Bit	Additional Bits						
Page 0		0	0	0	0	0	0	0
Page 1	0	1	1	0	0	0	0	0
Page 2	0	1	0	0	0	0	0	0
Page 3	0	0	1	0	0	0	0	0
Page 4	0	0	0	0	0	0	0	0

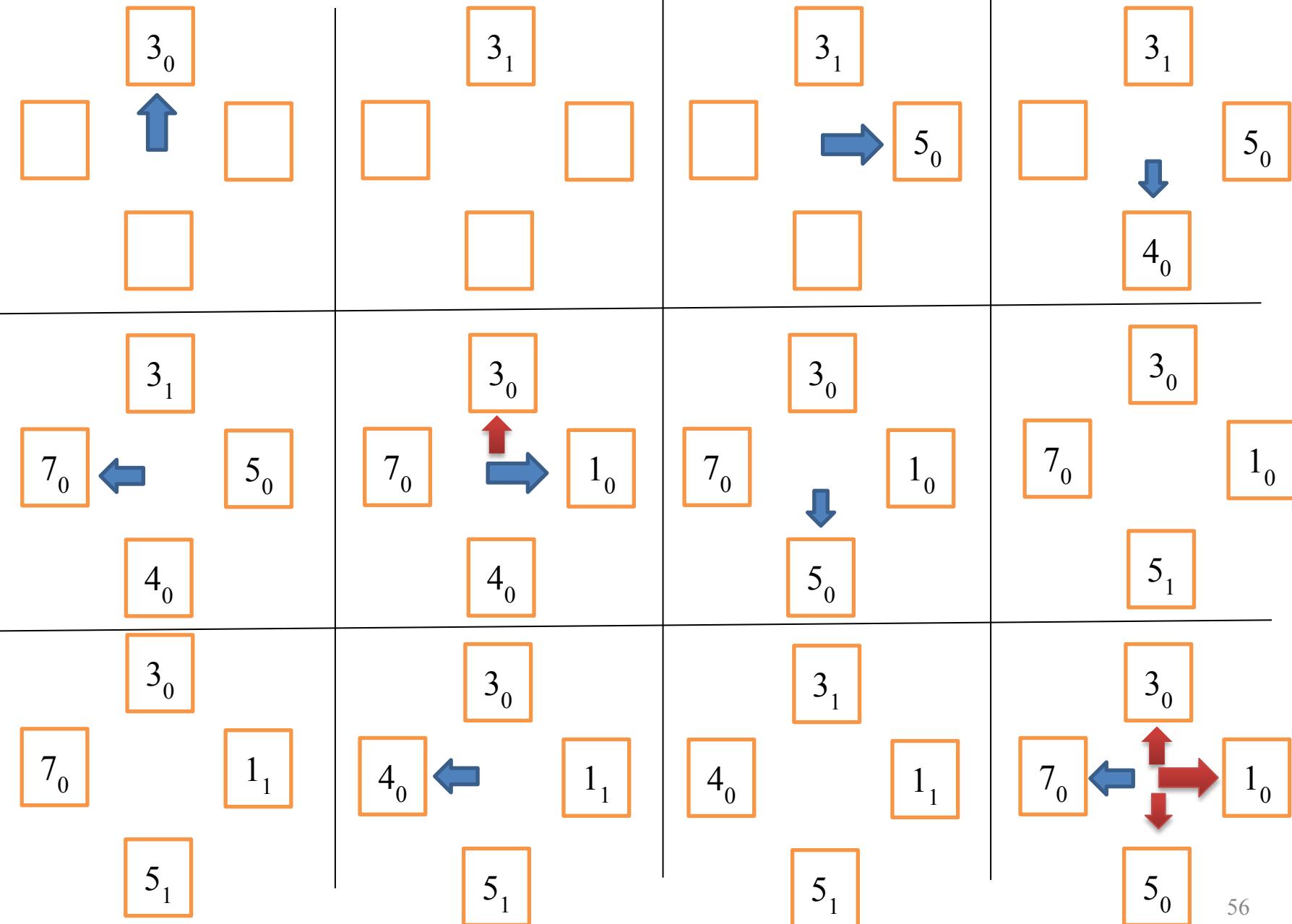
LRU Approximation – Second Chance/Clock Algorithm

- By default FIFO is used
- Page is replaced based on reference bit
- If the reference bit is ‘0’ then the page is considered as not recently used and taken for replacement
- If the reference bit is ‘1’ then the page is considered as recently used and it is given a second chance to stay in the memory
- Now its reference bit is reset to ‘0’

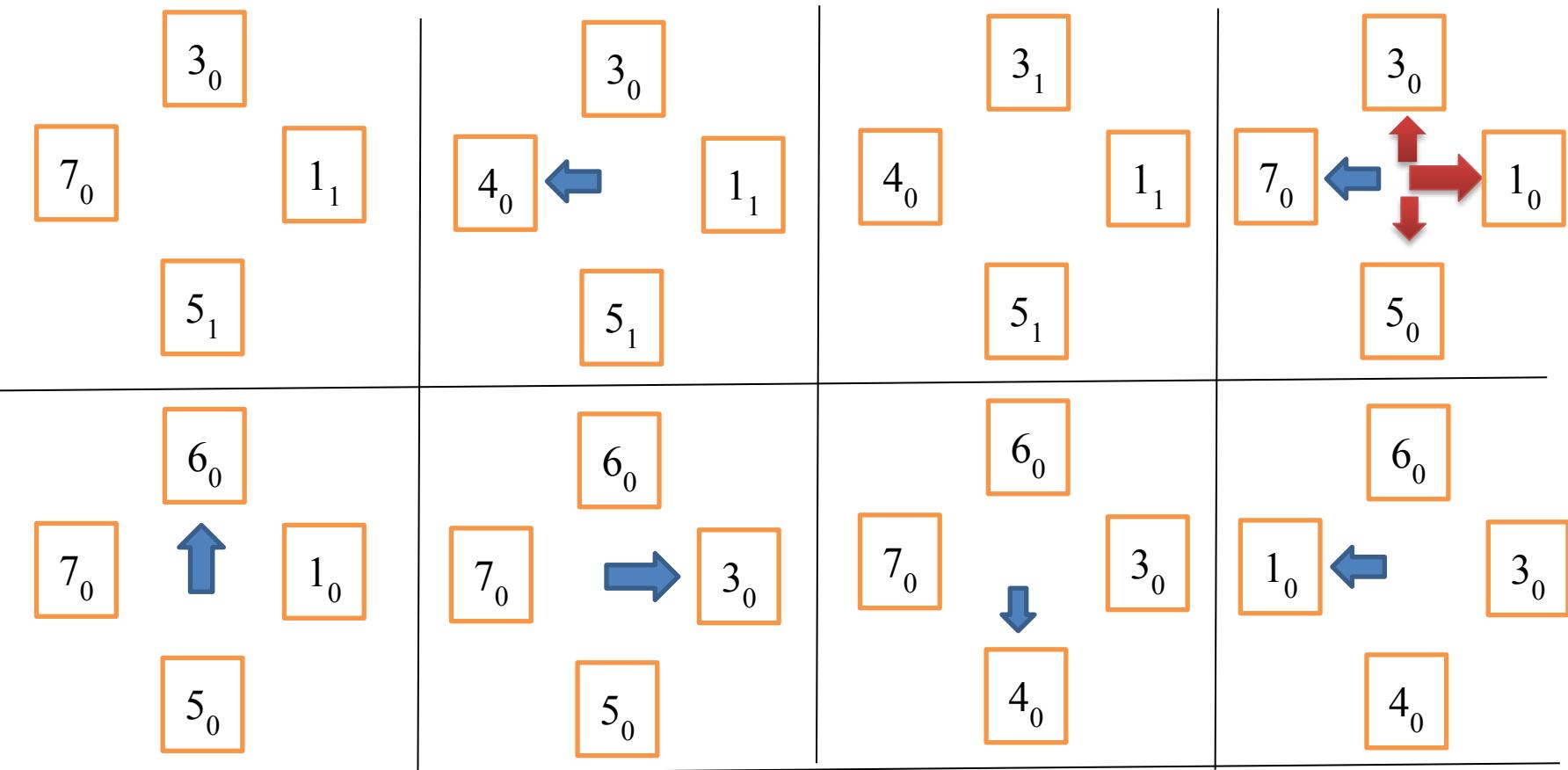
- Fig (a) denotes next victim as page containing reference bits as 1, Fig (b) denotes that second chance is given to the referred pages and points the next victim as page containing reference bit as ‘0’
- At the mean time the reference bit is reset to ‘0’ for the page who got the second chance



• reference strings - 3 3 5 4 7 1 5 5 1 4 3 7 6 3 4 1



- reference strings - 3 3 5 4 7 1 5 5 1 4 3 7 6 3 4 1



- Number of page hits – 4
- Number of page faults – 12

LRU Approximation – Enhanced Second Chance Algorithm

- Use both the reference bit and modify bit for the replacement
- (0,0) – neither recently used or modified – best to replace
- (0,1) – not recently used but modified – page needs to be written before replacement
- (1,0) – recently used but clean – probably will be used again soon
- (1,1) – recently used and modified – may be used again, has to be written to the disk
- Out of these four classes, the page which falls under (0,0) will be replaced first, then (0,1) will be chosen for replacement, next (1,0) and finally (1,1) will be chosen for replacement

Understanding the Pros and cons of the page replacement techniques

FIFO Page Replacement Algorithm

Advantages

- Easy to understand and execute

Disadvantages

- It is not very effective
- System needs to keep track of each frame
- Page fault increases when increasing the page frames, undergoes Belady's anomaly

Understanding the Pros and cons of the page replacement techniques

Optimal Page Replacement Algorithm

Advantages

- Lowest page fault rate
- Never suffers from Belady's anomaly
- Twice as good as FIFO

Disadvantages

- Difficult to implement
- It needs forecast i.e. Future knowledge

Understanding the Pros and cons of the page replacement techniques

LRU Page Replacement Algorithms

Advantages

- It is amenable to full statistical analysis
- Never suffers from Belady's anomaly
- Easy to identify the faulty page that is not needed to long time

Disadvantages

- It has more complexity
- Need additional Data Structure and high hardware support

COUNTING ALGORITHMS

- It keeps count of number of references (frequencies) made to each page in a reference string
- Whenever a page comes in its frequency increases
- Whenever a page leaves the memory its frequency is reset
- **Least frequently used page replacement**
 - Page with least number of frequency is replaced first
 - Assumption – Heavily used page will be referred again
 - When there is a tie in frequencies, then use FIFO

COUNTING ALGORITHMS

- **Most Frequently used Page replacement algorithm**
- A page which has maximum number of frequencies in the counter will be replaced first
- Assumption – The page just brought in the memory is yet to use again
- If there is a tie in frequencies use FIFO for replacement

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

0	2	0	1	0	3	2	1	2	3



Counter value of each page	Page 0 : 0	Page 1 : 0	Page 2 : 0	Page 3 : 0

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

0	2	0	1	0	3	2	1	2	3
0									

F									
---	--	--	--	--	--	--	--	--	--

Counter value of each page	Page 0 : 1	Page 1 : 0	Page 2 : 0	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

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

F	F								
---	---	--	--	--	--	--	--	--	--

Counter value of each page	Page 0 : 1	Page 1 : 0	Page 2 : 1	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

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

F	F	H							
---	---	---	--	--	--	--	--	--	--

Counter value of each page	Page 0 : 2	Page 1 : 0	Page 2 : 1	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

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

F	F	H	F						
---	---	---	---	--	--	--	--	--	--

Counter value of each page	Page 0 : 2	Page 1 : 1	Page 2 : 1	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

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

F	F	H	F	H					
---	---	---	---	---	--	--	--	--	--

Counter value of each page	Page 0 : 3	Page 1 : 1	Page 2 : 1	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

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

F	F	H	F	H	F				
---	---	---	---	---	---	--	--	--	--

Counter value of each page	Page 0 : 3	Page 1 : 1	Page 2 : 0	Page 3 : 1
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

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

F	F	H	F	H	F	F			
---	---	---	---	---	---	---	--	--	--

Counter value of each page	Page 0 : 3	Page 1 : 0	Page 2 : 1	Page 3 : 1
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

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

F	F	H	F	H	F	F	F		
---	---	---	---	---	---	---	---	--	--

Counter value of each page	Page 0 : 3	Page 1 : 1	Page 2 : 1	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

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

F	F	H	F	H	F	F	F	H	
---	---	---	---	---	---	---	---	---	--

Counter value of each page	Page 0 : 3	Page 1 : 1	Page 2 : 2	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform LFU

0	2	0	1	0	3	2	1	2	3
0	0	0	0	0	0	0	0	0	0
	2	2	2	2	3	3	1	1	3
			1	1	1	2	2	2	2
F	F	H	F	H	F	F	F	H	F

Counter value of each page	Page 0 : 3	Page 1 : 0	Page 2 : 2	Page 3 : 1
----------------------------	------------	------------	------------	------------

- Number of page hits – 3
- Number of page faults – 7

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

0	2	0	1	0	3	2	1	2	3

--	--	--	--	--	--	--	--	--	--

Counter value of each page	Page 0 : 0	Page 1 : 0	Page 2 : 0	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

0	2	0	1	0	3	2	1	2	3
0									

F									
---	--	--	--	--	--	--	--	--	--

Counter value of each page	Page 0 : 1	Page 1 : 0	Page 2 : 0	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

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

F	F								
---	---	--	--	--	--	--	--	--	--

Counter value of each page	Page 0 : 1	Page 1 : 0	Page 2 : 1	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

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

F	F	H							
---	---	---	--	--	--	--	--	--	--

Counter value of each page	Page 0 : 2	Page 1 : 0	Page 2 : 1	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

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

F	F	H	F						
---	---	---	---	--	--	--	--	--	--

Counter value of each page	Page 0 : 2	Page 1 : 1	Page 2 : 1	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

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

F	F	H	F	H					
---	---	---	---	---	--	--	--	--	--

Counter value of each page	Page 0 : 3	Page 1 : 1	Page 2 : 1	Page 3 : 0
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

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

F	F	H	F	H	F				
---	---	---	---	---	---	--	--	--	--

Counter value of each page	Page 0 : 0	Page 1 : 1	Page 2 : 1	Page 3 : 1
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

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

F	F	H	F	H	F	H			
---	---	---	---	---	---	---	--	--	--

Counter value of each page	Page 0 : 0	Page 1 : 1	Page 2 : 2	Page 3 : 1
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

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

F	F	H	F	H	F	H	H		
---	---	---	---	---	---	---	---	--	--

Counter value of each page	Page 0 : 0	Page 1 : 2	Page 2 : 2	Page 3 : 1
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

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

F	F	H	F	H	F	H	H	H	
---	---	---	---	---	---	---	---	---	--

Counter value of each page	Page 0 : 0	Page 1 : 2	Page 2 : 3	Page 3 : 1
----------------------------	------------	------------	------------	------------

Reference String – 0, 2, 0, 1, 0, 3, 2, 1, 2, 3

Total Frames allocated – 3

Perform MFU

0	2	0	1	0	3	2	1	2	3
0	0	0	0	0	3	3	3	3	3
	2	2	2	2	2	2	2	2	2
			1	1	1	1	1	1	1
F	F	H	F	H	F	H	H	H	H

F	F	H	F	H	F	H	H	H	H

Counter value of each page	Page 0 : 0	Page 1 : 2	Page 2 : 3	Page 3 : 2

- Number of page hits – 6
- Number of page faults – 4

Page-Buffering Algorithms

- Used in addition to page replacement algorithms
- System keeps a pool of free frames
- When a page fault occurs
 - Victim frame is chosen
 - but requires some time to choose the victim frame
 - At that time desired page is read into a free frame which is maintained by the system
 - By doing this the desired page can start doing its work without waiting for the swapping process of the victim frame in the main memory
 - Victim frame is written out later
 - Finally Victim frame is added to the pool of free frames

Page-Buffering Algorithms

- Another idea is to maintain a list of modified pages
- Whenever paging device is idle, a modified page is selected and written to the disk
 - When there is read/write operation at that time paging device does this job
- Modify bit is then reset
- Increases the probability that a page is clean when chosen for replacement
 - Used to identify the clean pages very quickly, because after performing this operation maximum modified pages will be already written to the disk
 - Makes the replacement very easy

Page-Buffering Algorithms

- Yet Another idea is to keep a pool of free frames
- Remember which page was in each frame
- If the same page is needed again, it is taken from the pool of free frames and used
 - But the page should not be modified for reusing
 - No I/O operation is required
- If the required page is not in the pool of free frames, another free frame is selected
- The new page is read into free frame

Allocation of Frames

- How do we allocate the fixed amount of free memory among the various process?
- Example: If we have 93 free frames and 2 processes, How many frames does each process get?
- Consider a single-user system with 128 KB of memory composed of pages 1 KB in size.
- This system has 128 frames.
 - The operating system may take 35 KB of memory and
 - Leaving remaining 93 frames for the user process.
- Other possible strategy: the user process is allocated with any free frame.

Demand paging Vs. Pure Demand paging

Demand paging

- In demand paging, a page is not loaded into main memory until it is needed.
- In Demand paging follows that pages should only be brought into memory if the executing process demands them.

Pure Demand paging

- In pure demand paging, even a single page is not loaded into memory initially. Hence pure demand paging causes a page fault. Page fault, the situation in which the page is not available whenever a processor needs to execute it.
- while in pure demand paging swapping, where all memory for a process is swapped from secondary storage to main memory during the process startup.

Allocation of Frames – Pure Demand Paging

Under Pure demand paging:

- All 93 frames - initially kept on free-frame list.
- When a user process started execution, it would generate a sequence of page faults.
- There is not page fault up to 93 frames.
- When the free-frame list was exhausted then a page-replacement algorithm would be called to select one of the 93 in-memory pages to be replaced with the 94th, and so on.
- When the process terminated, the 93 frames would once again be placed on the free-frame list.

Allocation of Frames

- Each process needs minimum number of frames
 - Allocating minimum number of frames impacts on the performance.
 - If the number of frames allocated to each process decreases, the page-fault rate increases, which slow downs the execution of process
 - Also when a page fault occurs before completion of an executing then the instruction must be restarted.
- It is necessary to hold all the different pages that any single instruction can reference.
- Example: IBM 370 – 6 pages to handle storage to storage (SS) MOVE instruction:
 - instruction is 6 bytes, might span 2 pages
 - 2 pages to handle *from* one storage area
 - 2 pages to handle *to* another storage area

Allocation schemes

- Minimum number of frames per process is defined by the architecture.
- Maximum number is defined by the amount of available physical memory in the system
- In between is still left with significant choice of frame allocation.
- Two major allocation schemes
 - Fixed allocation
 - Equal Allocation
 - Proportional Allocation
 - Priority allocation

Fixed Allocation

- **Equal allocation** – For example, if there are 100 frames (after allocating frames for the OS) and 5 processes, give each process 20 frames
 - Keep some as free frame buffer pool
- **Proportional allocation** – Allocate according to the size of process
 - Dynamic as degree of multiprogramming, process sizes change

Consider a system with a 1-KB frame size. If a small student process of 10 KB and an interactive database of 127 KB are the only two processes running in a system with 62 free frames, it does not make much sense to give each process 31 frames. The student process does not need more than 10 frames, so the other 21 are, strictly speaking, wasted

- s_i = size of process p_i
- $S = \sum s_i$
- m = total number of frames
- a_i = allocation for $p_i = \frac{s_i}{S} \times m$

$$\begin{aligned}
 m &= 64 \\
 s_1 &= 10 \\
 s_2 &= 127 \\
 a_1 &= \frac{10}{137} \times 62 \approx 4 \\
 a_2 &= \frac{127}{137} \times 62 \approx 57
 \end{aligned}$$

Fixed Allocation – Contd.

- In equal and proportional allocation
- If the multiprogramming level is increased, each process will lose some frames for the new process.
- if the multiprogramming level decreases, the frames that were allocated from departed process are spread to the remaining processes.

Priority Allocation

- Suppose if we want to give the high-priority process with more memory to speed up its execution than the low-priority processes.
- Use a proportional allocation scheme using priorities rather than size
- If process P_i generates a page fault,
 - select for replacement one of its frames
 - select for replacement a frame from a process with lower priority number

Frame Allocation - Another factor

- Multiple processes competing for frames, we can classify page-replacement algorithms into two broad categories namely Global and Local.

Global replacement

- Process selects a replacement frame from the set of all frames; one process can take a frame from another
 - But then process execution time can vary greatly
 - But greater throughput so more common

Local replacement

- Each process selects from only its own set of allocated frames
 - More consistent per-process performance
 - But possibly underutilized memory

Global vs. Local Allocation – Contd.

- If, we allow high-priority processes to select frames from low-priority processes for replacement
 - Global replacement, approach allows a high-priority process to increase its frame allocation at the expense of a low-priority process
 - Local replacement, the number of frames allocated to a process does not change.

Non-Uniform Memory Access (NUMA)

- So far all memory accessed equally
- Many systems are **NUMA** – speed of access to memory varies
 - Consider system boards containing CPUs and memory, interconnected over a system bus
- Optimal performance comes from allocating memory “close to” the CPU on which the thread is scheduled
 - And modifying the scheduler to schedule the thread on the same system board when possible
- Solved by Solaris by creating **lgroups**
 - Structure to track CPU / Memory low latency groups
 - Used my schedule and pager
 - When possible schedule all threads of a process and allocate all memory for that process within the lgroup

Root cause of the Thrashing

- Thrashing is caused by under allocation of the minimum number of pages required by a process, forcing it to continuously page fault.
- The system can detect thrashing by evaluating the level of CPU utilization as compared to the level of multiprogramming.
- It can be eliminated by reducing the level of multiprogramming.

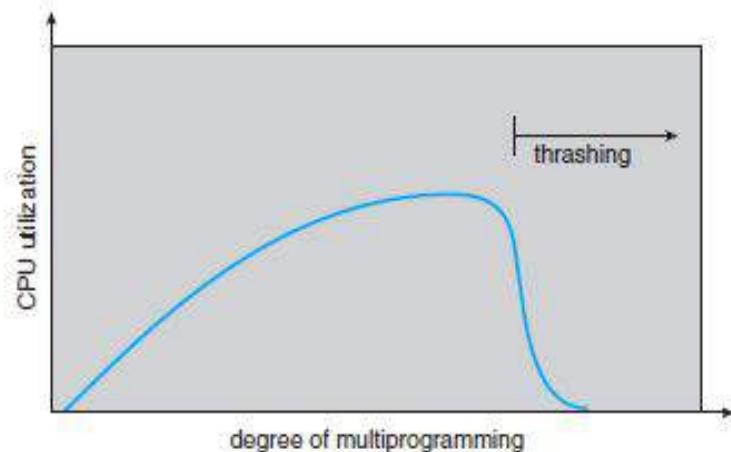
Thrashing

- A process is busy with swapping pages in and out. This high paging activity is called Thrashing.
- A process is in thrashing if it is spending more time on paging rather than executing.

Cause of Thrashing

If a process does not have “enough” pages, the page-fault rate is very high. This leads to:

- low CPU utilization.
- operating system thinks that it needs to increase the degree of multiprogramming.
- another process added to the system.



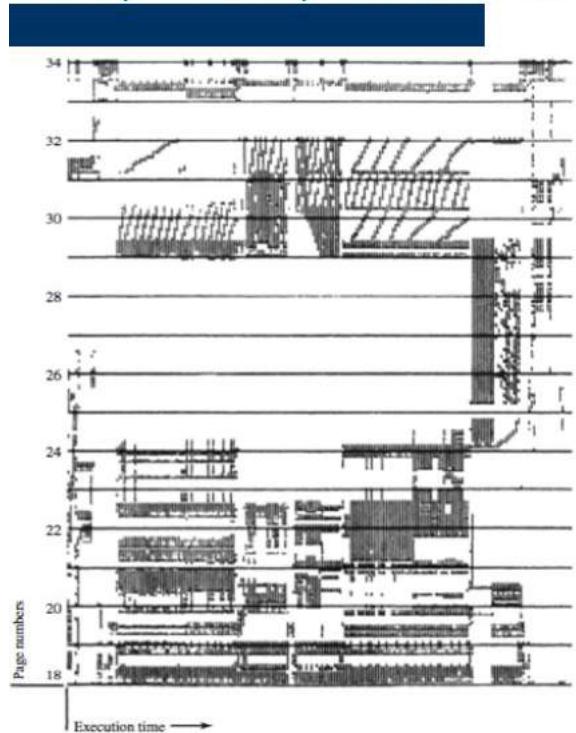
Demand Paging and Thrashing

- Why does demand paging work?
 - To prevent thrashing, we must provide a process with as many frames as it needs.
- But how do we know how many frames it “needs”?
- Locality model - working-set strategy starts by looking at how many frames a process is actually using. This approach defines the locality model of process execution

Demand Paging and Thrashing - Contd

- Process migrates from one locality to another (Figure)
- Localities may overlap
- Example:
 - Function is called, that defines a new locality.
 - In this, memory references are made to the instructions of the function call, its local variables, and a subset of the global variables. When we exit the function, the process leaves this locality, since the local variables and instructions of the function are no longer in active use.

Locality in a Memory-Reference Pattern



Demand Paging and Thrashing – Cont.

- Why does thrashing occur?
 - Σ size of locality > total memory size
 - Limit effects by using local or priority page replacement
 - Local replacement algorithms - if one process starts thrashing it cannot steal frames from another process and cause thrashing latter.
 - However if processes are thrashing, they will be in the queue for the paging device most of the time.
 - The average service time for a page fault will increase, due to the longer queue for the paging device.
 - Thus the effective access time will increase even for a process that is not thrashing.

Models to Avoid Thrashing

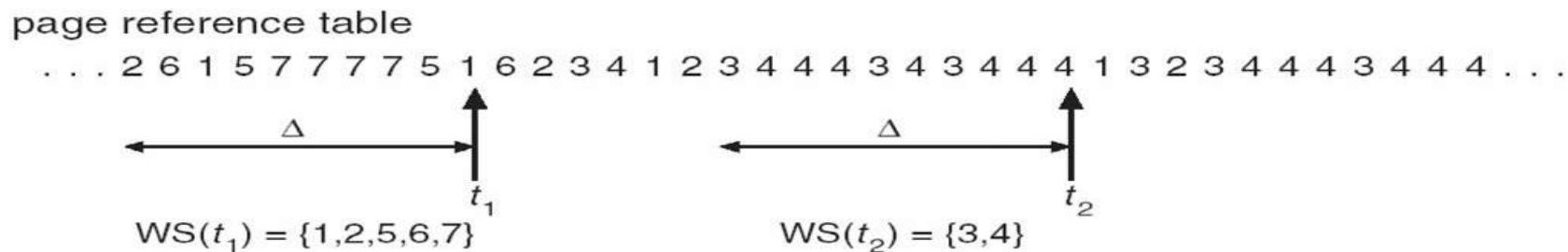
- There are 2 Models to Avoid Thrashing
 - Working – set Model
 - Works based on Locality of Reference
 - Page Fault Frequency Model
 - Based on Upper and Lower Page Fault Rate

Working Set Model

- It is based on the assumption of locality.
- Δ ≡ working-set window ≡ a fixed number of page references
Example: 10,000 instruction
- The idea is to examine the most recent Δ page references.
- WSS_i (working set of Process P_i) = total number of pages referenced in the most recent Δ (varies in time)
 - if Δ too small will not encompass entire locality.
 - if Δ too large will encompass several localities.
 - if $\Delta = \infty \Rightarrow$ will encompass entire program.
- $D = \sum WSS_i \equiv$ total demand frames
- if $D > m \Rightarrow$ Thrashing
 - Policy if $D > m$, then suspend one of the processes.
 - if $D < m \Rightarrow$ No Thrashing

Working Set Model – Contd.

- The sequence of memory references shown in Figure,
- if $\Delta = 10$ memory references,
- Then the working set at time
 - t_1 is $\{1, 2, 5, 6, 7\}$
 - t_2 , the working set has changed to $\{3, 4\}$.



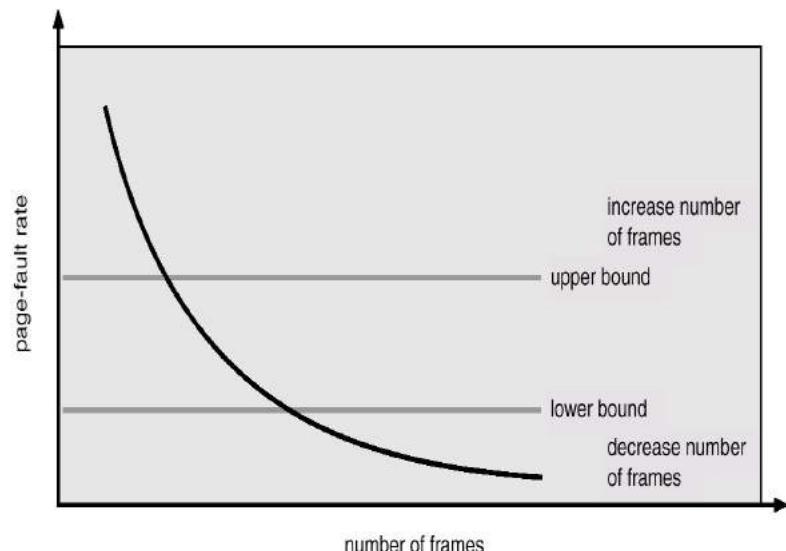
The **working set model** states that a process can be in RAM if and only if all of the pages that it is currently using (often approximated by the most recently used pages) can be in RAM.

Keeping Track of the Working Set

- The working set prevents thrashing while keeping the degree of multiprogramming as high as possible, in order to optimize the CPU utilization.
- Difficulty: keeping track of the working set
 - Approximate with interval timer + a reference bit
- Example: $\Delta = 10,000$
 - Timer interrupts after every 5000 time units.
 - Keep in memory 2 bits for each page.
 - Whenever a timer interrupts copy and sets the values of all reference bits to 0.
 - If one of the bits in memory = 1 \Rightarrow page in working set.
- Why is this not completely accurate?
- Improvement = 10 bits and interrupt every 1000 time units.

Page-Fault Frequency

- More direct approach than Working Set model.
 - Where working set seems a clumsy way to control thrashing.
- If page fault rate is too high, then the process need more frames.
- Whereas page fault is too low, then the process may have too many frames.
- Establish “acceptable” **page-fault frequency (PFF)** rate and use local replacement policy
 - If actual rate too low, process loses frame
 - If actual rate too high, process gains frame



Page-Fault Frequency – Contd.

- As with the working-set strategy, we may have to swap out a process.
- If the page-fault rate increases and no free frames are available, we must select some process and swap it out to backing store.
- The freed frames are then distributed to processes with high page-fault rates

18CSC205J – Operating Systems – Unit V

STORAGE MANAGEMENT AND FILE MANAGEMENT

Course Learning Rationale and Course Learning Outcomes

Course Learning Rationale

- Realize the significance of Device management part of an Operating System
- Comprehend the need of File management functions of an Operating System

Course Learning Outcomes

- Find the significance of device management role of an Operating system
- Recognize the essentials of File Management of an Operating system

Contents

- STORAGE MANAGEMENT :
- UNDERSTANDING THE BASICS IN STORAGE MANAGEMENT
- DISK SCHEDULING
- UNDERSTANDING THE VARIOUS SCHEDULING WITH RESPECT TO THE DISK
- FILE SYSTEM INTERFACE:
- FILE CONCEPT
- FILE ACCESS METHODS
- UNDERSTANDING THE FILE BASICS
- FILE SHARING AND PROTECTION
- EMPHASIS THE NEED FOR THE FILE SHARING AND ITS PROTECTION
- FILE SYSTEM IMPLEMENTATION :
- BASIC FILE SYSTEM STRUCTURE
- FILE SYSTEM IMPLEMENTATION

Mass-Storage Systems

- Overview of Mass Storage Structure
 - Hard disk
 - Magnetic tape
 - Storage Array
 - Storage Area Networks
 - Network Attached Storage

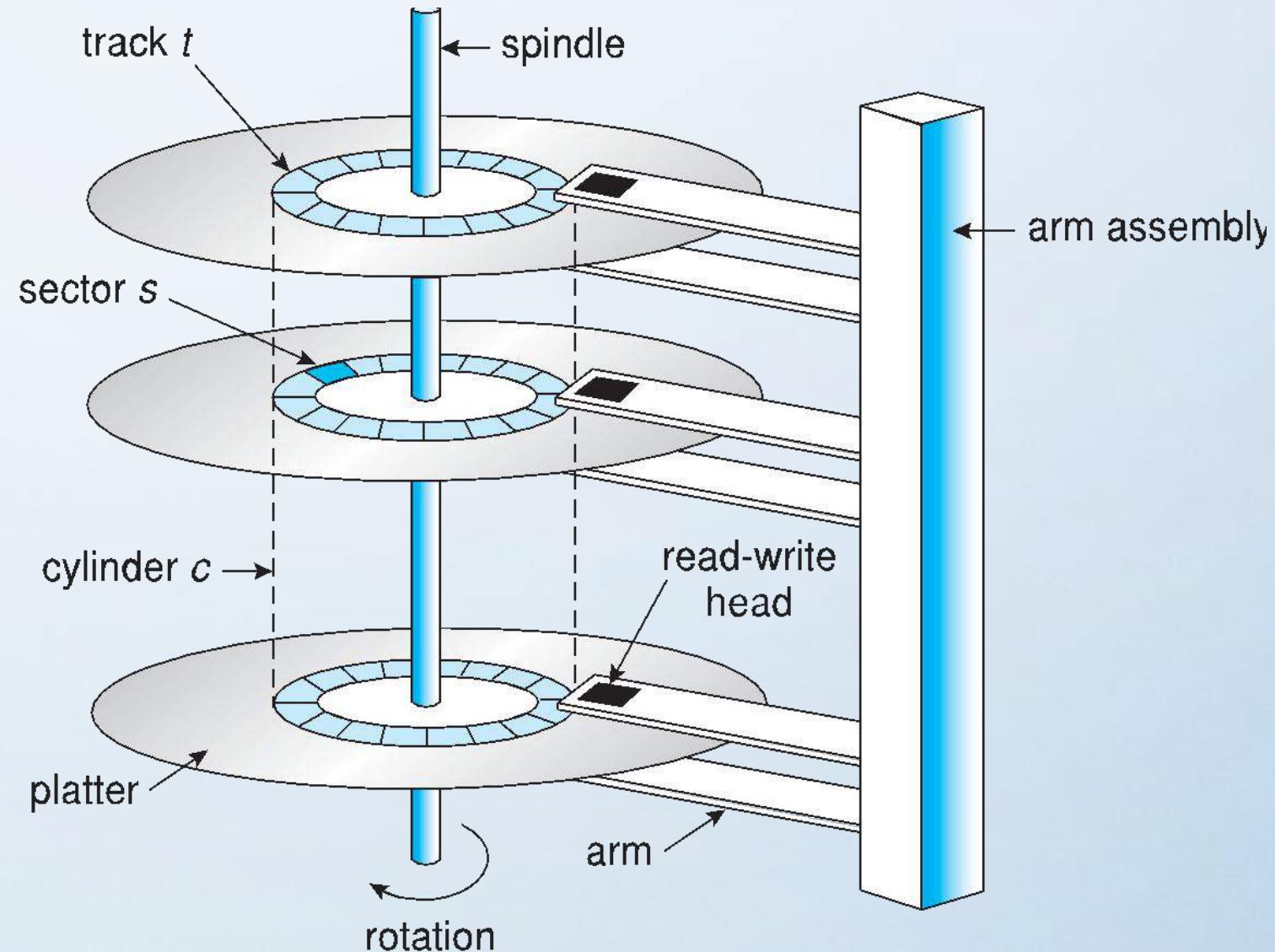
Objectives

- To describe the physical structure of secondary storage devices and its effects on the uses of the devices
- To explain the performance characteristics of mass-storage devices
- To evaluate disk scheduling algorithms

Overview of Mass Storage Structure

- **Magnetic disks** provide bulk of secondary storage of modern computers
 - Drives rotate at 60 to 250 times per second
 - **Transfer rate** is rate at which data flow between drive and computer
 - **Positioning time (random-access time)** is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
 - **Head crash** results from disk head making contact with the disk surface -- That's bad
- Disks can be removable
- Drive attached to computer via **I/O bus**
 - Busses vary, including **EIDE, ATA, SATA, USB, Fibre Channel, SCSI, SAS, Firewire**
 - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

Moving-head Disk Mechanism



Hard Disks

- Platters range from .85" to 14" (historically)
 - Commonly 3.5", 2.5", and 1.8"
- Range from 30GB to 3TB per drive
- Performance
 - Transfer Rate – theoretical – 6 Gb/sec
 - Effective Transfer Rate – real – 1Gb/sec
 - Seek time from 3ms to 12ms – 9ms common for desktop drives
 - Average seek time measured or calculated based on 1/3 of tracks
 - Latency based on spindle speed
 - $1 / (\text{RPM} / 60) = 60 / \text{RPM}$
 - Average latency = $\frac{1}{2}$ latency

Hard Disks

- Platters range from .85" to 14" (historically)
 - Commonly 3.5", 2.5", and 1.8"
- Range from 30GB to 3TB per drive
- Performance
 - Transfer Rate – theoretical – 6 Gb/sec
 - Effective Transfer Rate – real – 1Gb/sec
 - Seek time from 3ms to 12ms – 9ms common for desktop drives
 - Average seek time measured or calculated based on 1/3 of tracks
 - Latency based on spindle speed
 - $1 / (\text{RPM} / 60) = 60 / \text{RPM}$
 - Average latency = $\frac{1}{2}$ latency



Terms to be considered for calculating Hard Disk Performance

- 1. **Seek time-** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- 2. **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- 3. **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- 4. **Disk Access Time:** Disk Access = Seek time + Rotational latency + Transfer time

Hard Disk Performance(contd...)

- To transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead calculate average I/O time or Disk access time of 4KB block.

Spindle [rpm]	Average latency [ms]
4200	7.14
5400	5.56
7200	4.17
10000	3
15000	2

Disk Access time = Seek time + Rotational latency + Transfer time

Given 1. Avg Seek time = 5ms, 2. Avg Rotational Latency time=4.17ms 3. Controller overhead = 0.1ms 4. transfer rate or time=1GB/sec

Calculate transfer time or transfer rate of 4KB = $4/1024 * 1024$ per sec

$$=3.8146e-6 \text{ or } 3.814 \times 10^{-6}$$

$$\begin{aligned} \text{To convert into ms} &= 3.814 \times 10^{-6} \times 10^3 \\ &= 3.814 \times 10^{-3} \\ &= 0.003814 \text{ms} \end{aligned}$$

$$\begin{aligned} \text{Avg I/O time or Avg Disk access time} &= 5\text{ms} + 4.17\text{ms} + 0.1\text{ms} + 0.003814 \\ &= 9.27381\text{ms} \\ &= 9.3\text{ms} \end{aligned}$$

The First Commercial Disk Drive



- 1956
- IBM RAMDAC computer included the IBM Model 350 disk storage system
- 5M (7 bit) characters
- 50 x 24" platters
- Access time = < 1 second

Solid-State Disks

- Nonvolatile memory used like a hard drive
 - Many technology variations
- Can be more reliable than HDDs
- More expensive per MB
- Maybe have shorter life span
- Less capacity
- But much faster
- Busses can be too slow -> connect directly to PCI for example
- No moving parts, so no seek time or rotational latency

Magnetic Tape

- Was early secondary-storage medium
 - Evolved from open spools to cartridges
- Relatively permanent and holds large quantities of data
- Access time slow
- Random access ~1000 times slower than disk
- Mainly used for backup, storage of infrequently-used data, transfer medium between systems
- Kept in spool and wound or rewound past read-write head
- Once data under head, transfer rates comparable to disk
 - 140MB/sec and greater
- 200GB to 1.5TB typical storage
- Common technologies are LTO-{3,4,5} and T10000

Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
 - Low-level formatting creates **logical blocks** on physical media
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
 - Sector 0 is the first sector of the first track on the outermost cylinder
 - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
 - Logical to physical address should be easy
 - Except for bad sectors
 - Non-constant # of sectors per track via constant angular velocity

Disk Attachment

- Host-attached storage accessed through I/O ports talking to I/O busses
- SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** requests operation and **SCSI targets** perform tasks
 - Each target can have up to 8 **logical units** (disks attached to device controller)
- FC is high-speed serial architecture
 - Can be switched fabric with 24-bit address space – the basis of **storage area networks (SANs)** in which many hosts attach to many storage units
- I/O directed to bus ID, device ID, logical unit (LUN)

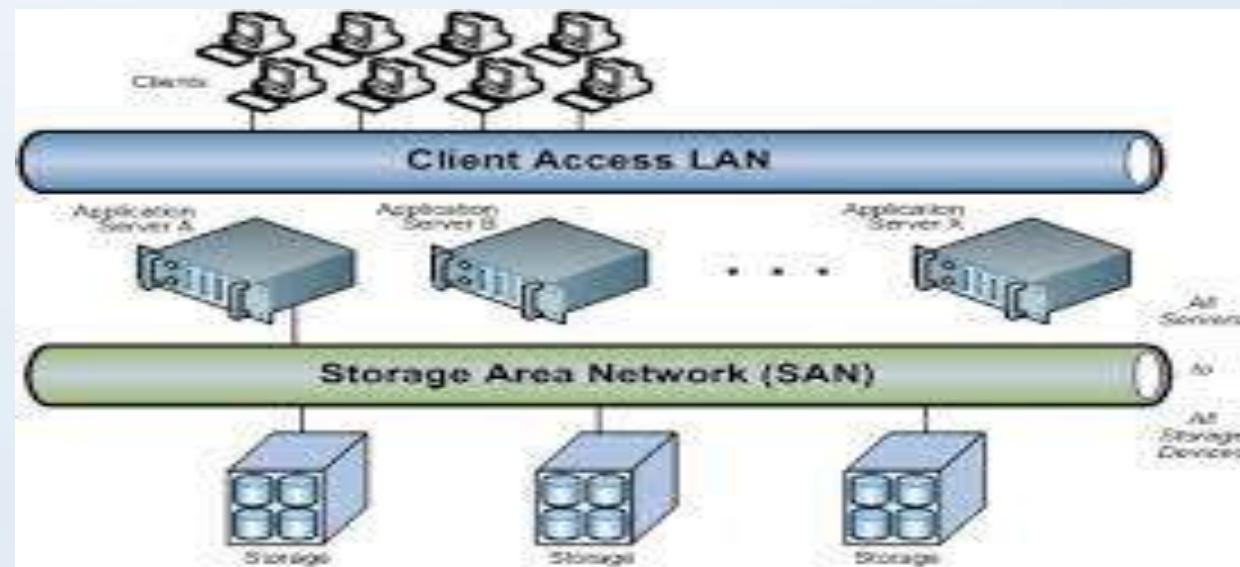
Storage Array

- Can just attach disks, or arrays of disks
- Storage Array has controller(s), provides features to attached host(s)
 - Ports to connect hosts to array
 - Memory, controlling software (sometimes NVRAM, etc)
 - A few to thousands of disks
 - RAID, hot spares, hot swap (discussed later)
 - Shared storage -> more efficiency
 - Features found in some file systems
 - Snapshots, clones, thin provisioning, replication, deduplication, etc



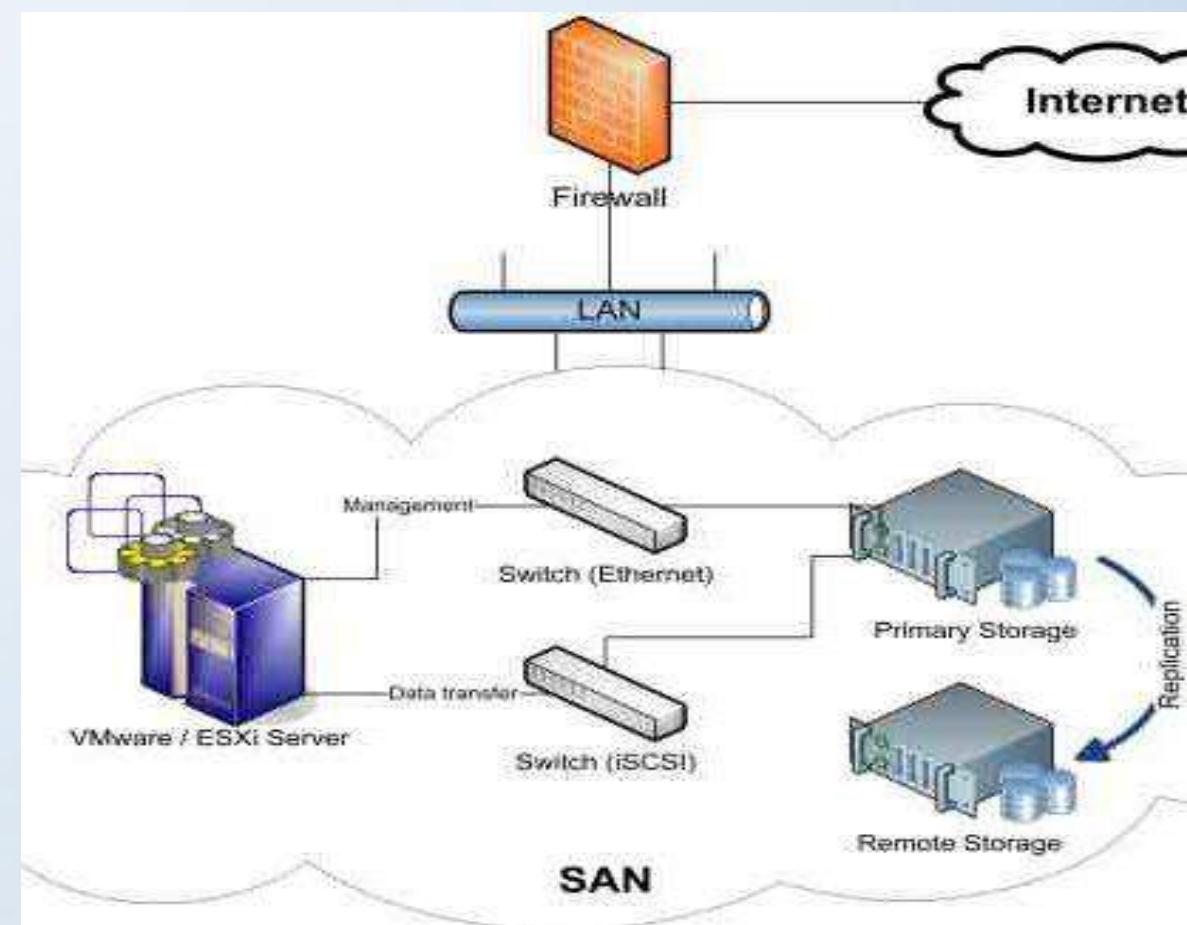
Storage Area Network

- Common in large storage environments
- Multiple hosts attached to multiple storage arrays - flexible



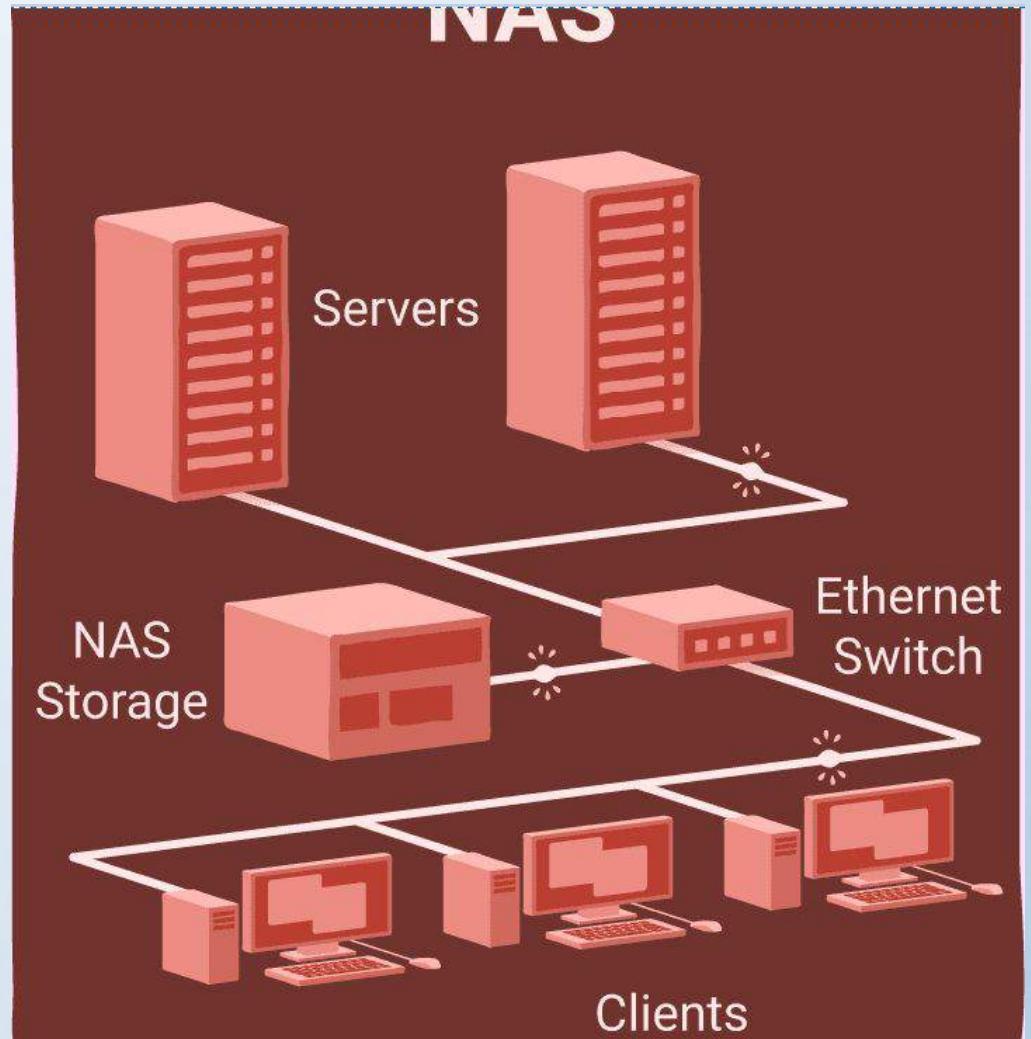
Storage Area Network (Cont.)

- SAN is one or more storage arrays
 - Connected to one or more Fibre Channel switches
- Hosts also attach to the switches
- Storage made available via **LUN Masking** from specific arrays to specific servers
- Easy to add or remove storage, add new host and allocate it storage
 - Over low-latency Fibre Channel fabric



Network-Attached Storage

- Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)
 - Remotely attaching to file systems
- NFS and CIFS are common protocols
- Implemented via remote procedure calls (RPCs) between host and storage over typically TCP or UDP on IP network
- **iSCSI** protocol uses IP network to carry the SCSI protocol
 - Remotely attaching to devices (blocks)



Difference between SAN Vs NAS

SAN

- ✓ Local network that uses Fibre Channel to connect several data storage devices.
- ✓ Transitioning from Fibre Channel to the same IP-based approach of NAS.

NAS

- A dedicated hardware device that connects to a local area network, usually through an Ethernet connection.
- Many NAS devices now offer performance capacities once reserved for SAN.

•Disk Scheduling

- Introduction to Disk Scheduling
- Why Disk Scheduling is Important
- Disk Scheduling Algorithms
 - FCFS
 - SSTF
 - SCAN
 - C-SCAN
 - LOOK
 - C-LOOK
- Disk Management

Disk Scheduling

What is Disk Scheduling?

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Minimize seek time
- Seek time \approx seek distance
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer
- There are many sources of disk I/O request
 - OS
 - System processes
 - Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue

Why Disk Scheduling is Important?

- Disk scheduling is important because:
- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

Disk Scheduling (Cont.)

- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying “depth”)
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters

Disk Scheduling Algorithms

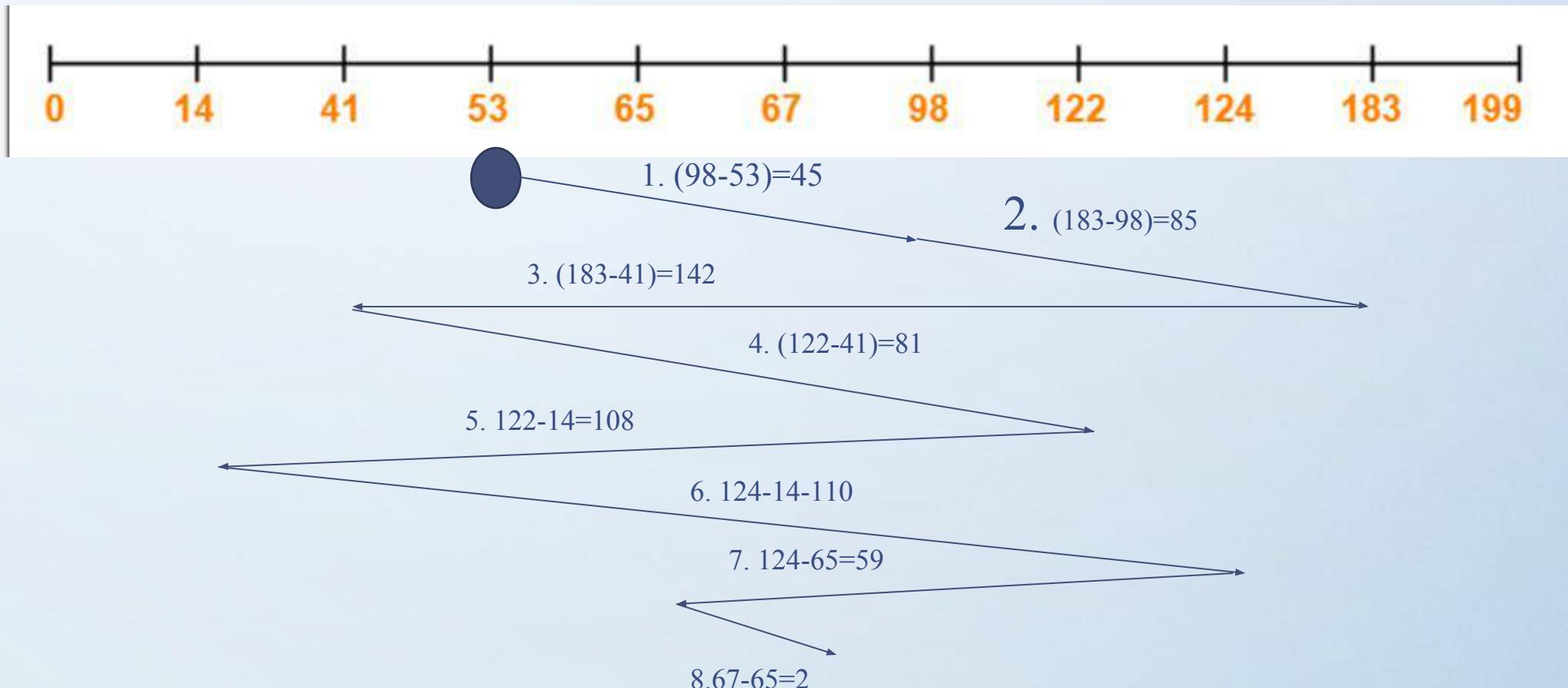
- Disk Scheduling Algorithms
 - FCFS
 - SSTF
 - SCAN
 - C-SCAN
 - LOOK
 - C-LOOK

Disk Scheduling Algorithm - Example

- Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The FCFS,SSTF,SCAN,C-SCAN,LOOK, and C-LOOK scheduling algorithm is used. The head is initially at cylinder number 53. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.

1. FCFS(First Come First Serve)

- FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.
- head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.
- 98, 183, 41, 122, 14, 124, 65, 67



FCFS(contd..)

- Total head movements using FCFS= $[(98 - 53) + (183 - 98) + (183 - 41) + (122 - 41) + (122 - 14) + (124 - 14) + (124 - 65) + (67 - 65)]$
- $= 45 + 85 + 142 + 81 + 108 + 110 + 59 + 2$
- $= 632$

Advantages:

- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

- Does not try to optimize seek time
- May not provide the best possible service

2. SSTF(Shortest Seek Time First)

- Shortest Seek Time First (SSTF) selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

Advantages:

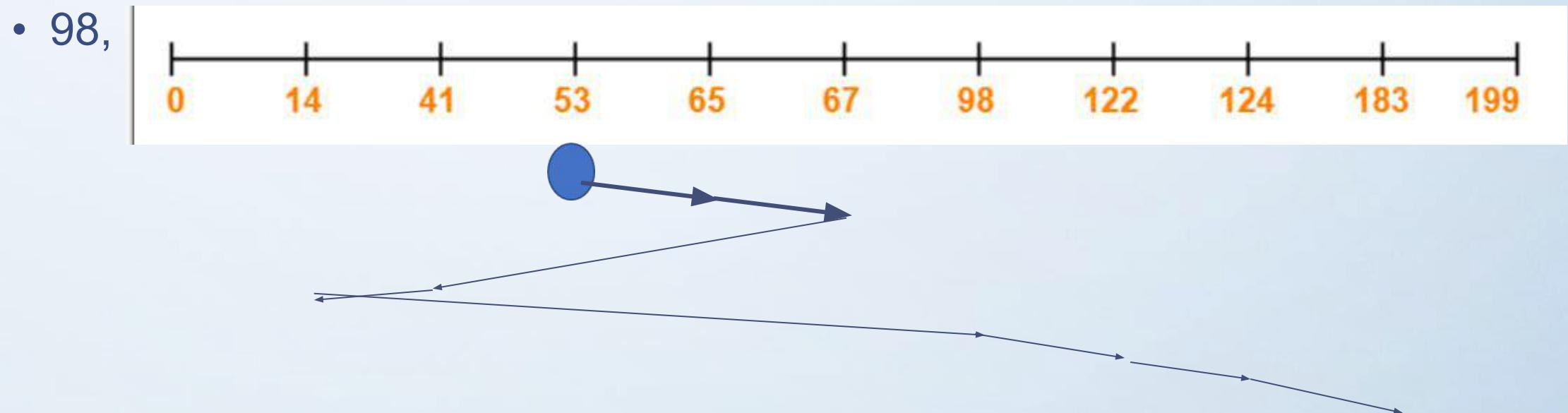
- Average Response Time decreases
- Throughput increases

Disadvantages:

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favours only some requests

SSTF(contd...)

- head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.



Total head movement using
$$\text{SSTF} = (65-53) + (67-65) + (67-41) + (41-14) + (98-14) + (122-98) + (124-122) + (183-124) = 12 + 2 + 26 + 27 + 84 + 24 + 2 + 59 = 236$$

3. SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- SCAN algorithm Sometimes called the elevator algorithm
- As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Advantages:

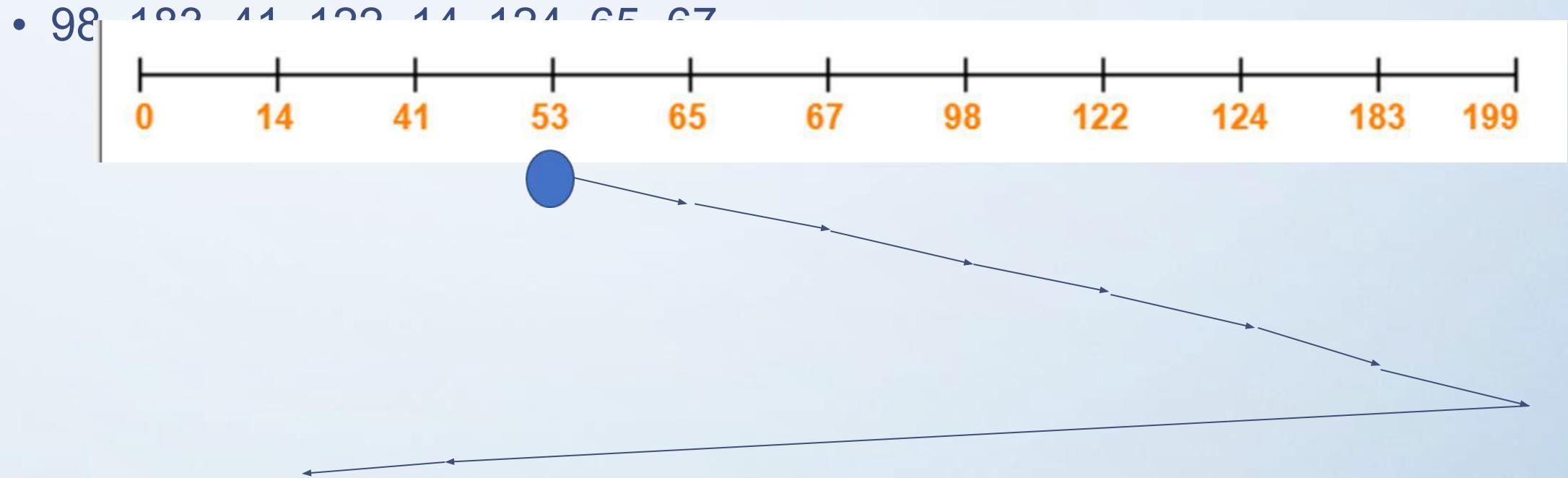
- High throughput
- Low variance of response time
- Average response time

Disadvantages:

- Long waiting time for requests for locations just visited by disk arm

SCAN or Elevator((1st Solution)

- head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.

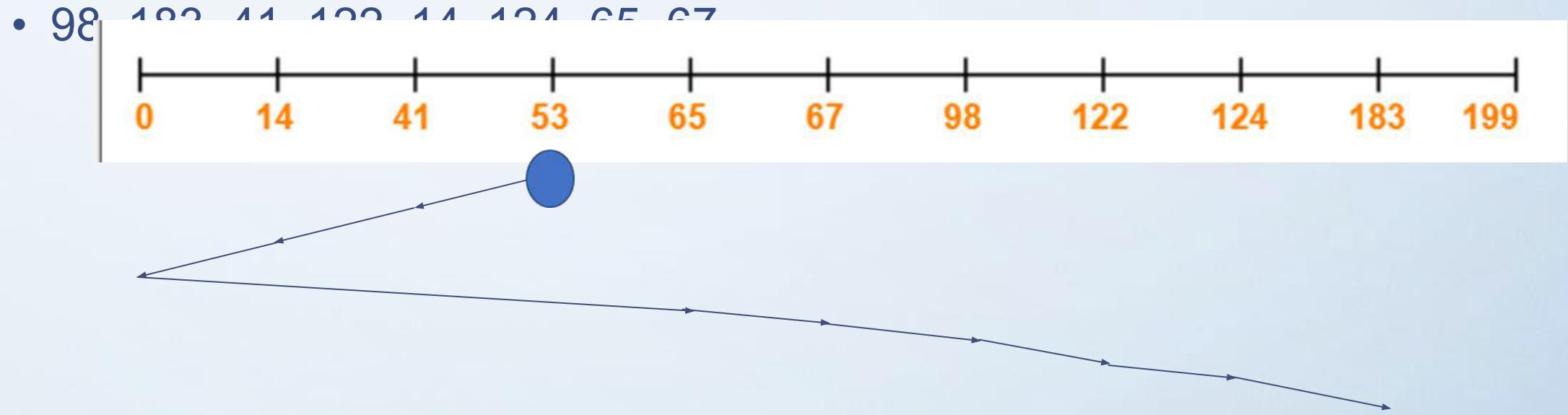


Total Head Movement using

$$\begin{aligned}
 \text{SCAN} = & [(65-53)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(199-183)+(199- \\
 & 41)+ \\
 & (41-14)] \\
 = & 12+2+31+24+2+59+16+158+27=331
 \end{aligned}$$

SCAN or Elevator(2nd Solution)- Best Choice

- head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.



Total Head Movement using SCAN=[(53-41)+(41-14)+(14-0)+(65-0)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)]
 $=12+27+14+65+2+31+24+59$
 $= 234$

4. C-SCAN

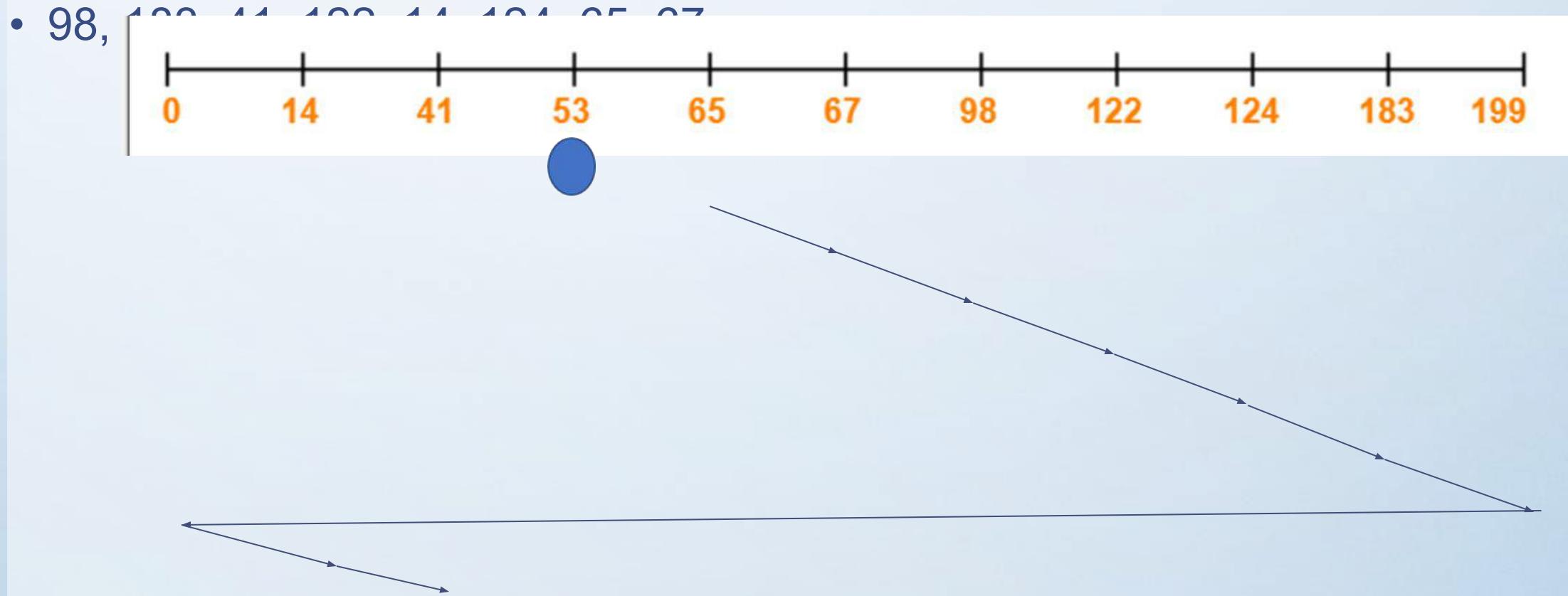
- The head moves from one end of the disk to the other, servicing requests as it goes
 - When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one
- Provides a more uniform wait time than SCAN

Advantages:

- Provides more uniform wait time compared to SCAN

C-SCAN(contd...)

- head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.



Total Head Movement using

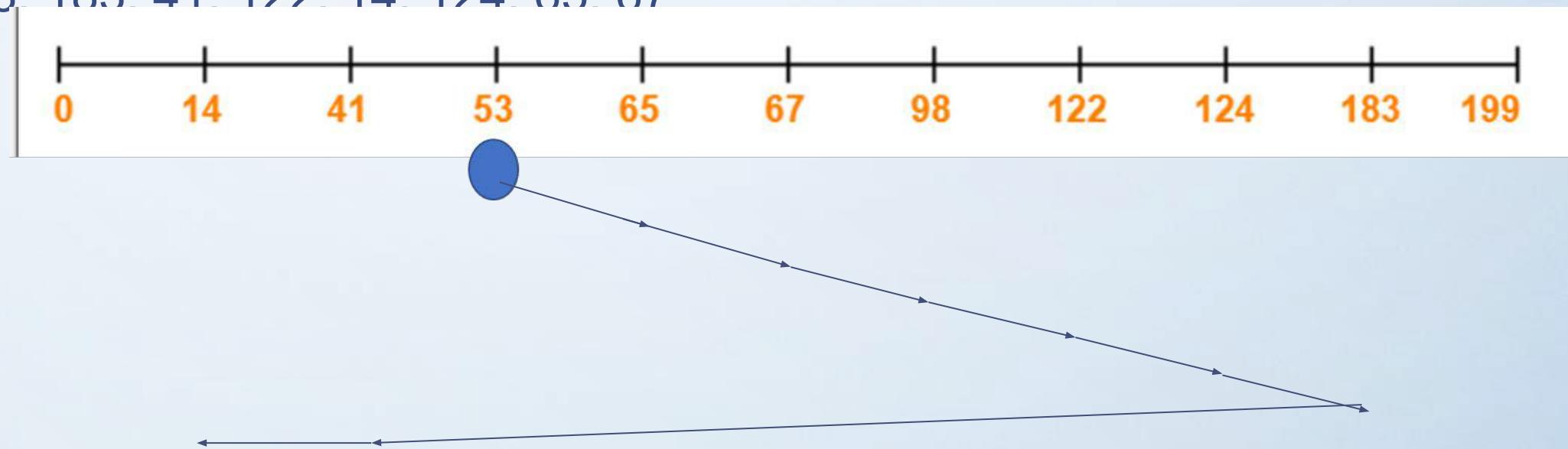
$$\begin{aligned}
 \text{C-SCAN} &= [(65-53)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(199-183)+(199-0)+(14-0)+(41-14)] \\
 &= 12+2+31+24+2+59+16+199+14+27 = 386
 \end{aligned}$$

5. LOOK

- It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

LOOK(Contd...)

- head is initially at cylinder number 53. The cylinders are numbered from 0 to 199.
- 98 183 41 122 14 124 65 67



Total Head Movement using

$$\text{look} = [(65-53)+(67-65)+(98-67)+(122-98)+(124-122)+\\(183-124)+(183-41)+(41-14)]\\=12+2+31+24+2+59+142+27)=299$$

6. C-LOOK

- LOOK a version of SCAN, C-LOOK a version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk
- In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

C- LOOK(contd...)

- head is initially at cylinder number 53. The cylinders are numbered from 0



Total Head Movement using

$$\text{C-LOOK} = [(65-53)+(67-65)+(98-67)+(122-98)+(124-122)+(183-124)+(183-14)+(41-14)]$$

$$= 12+2+31+24+2+59+169+27=326$$

No.of Head movements of All Algorithms

- Total head movements
- 1.FCFS=632
- 2.SSTF=232
- 3.SCAN=331or 234
- 4. C-SCAN=386
- 5.LOOK=299
- 6.C-SCAN=326

Selection of a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
 - Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
 - And metadata layout
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary
- **Either SSTF or LOOK is a reasonable choice for the default algorithm**
- What about rotational latency?
 - Difficult for OS to calculate
- How does disk-based queueing effect OS queue ordering efforts?

Disk Management

- **Low-level formatting**, or **physical formatting** — Dividing a disk into sectors that the disk controller can read and write
 - Each sector can hold header information, plus data, plus error correction code (**ECC**)
 - Usually 512 bytes of data but can be selectable
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk
 - **Partition** the disk into one or more groups of cylinders, each treated as a logical disk
 - **Logical formatting** or “making a file system”
 - To increase efficiency most file systems group blocks into **clusters**
 - Disk I/O done in blocks
 - File I/O done in clusters

Disk Management (Cont.)

- **Raw disk** access for apps that want to do their own block management, keep OS out of the way (databases for example)
- Boot block initializes system
 - The bootstrap is stored in ROM
 - **Bootstrap loader** program stored in boot blocks of boot partition
- Methods such as **sector sparing** used to handle bad blocks

File System Interface

- File Concept
- File Access Methods
- File Sharing and Protection

Objective of File Interface

- To explain the function of file systems
- To describe the interfaces to file systems
- To discuss file-system design tradeoffs, including access methods, file sharing, file locking, and structures

File Concept

File Concept

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.
- **Files** are the most important mechanism for storing data permanently on mass-storage devices. *Permanently* means that the data is not lost when the machine is switched off. Files can contain:
 - data in a format that can be interpreted by programs, but not easily by humans (*binary files*);
 - alphanumeric characters, codified in a standard way (e.g., using ASCII or Unicode), and directly readable by a human user (*text files*). Text files are normally organized in a sequence of lines, each containing a sequence of characters and ending with a special character (usually the *newline character*). Consider, for example, a Java program stored in a file on the hard-disk. In this unit we will deal only with text files.
 - Each file is characterized by a *name* and a *directory* in which the file is placed (one may consider the whole path that allows one to find the file on the hard-disk as part of the name of the file).

File Attributes

- **Name:** It is the only information stored in a human-readable form.
- **Identifier:** Every file is identified by a unique tag number within a file system known as an identifier.
- **Location:** Points to file location on device.
- **Type:** This attribute is required for systems that support various types of files.
- **Size.** Attribute used to display the current file size.
- **Protection.** This attribute assigns and controls the access rights of reading, writing, and executing the file.
- **Time, date and security:** It is used for protection, security, and also used for monitoring

File Operations

1.Create

- Creation of the file is the most important operation on the file. Different types of files are created by different methods for example text editors are used to create a text file, word processors are used to create a word file and Image editors are used to create the image files.

2.Write

- Writing the file is different from creating the file. The OS maintains a write pointer for every file which points to the position in the file from which, the data needs to be written.

3.Read

- Every file is opened in three different modes : Read, Write and append. A Read pointer is maintained by the OS, pointing to the position up to which, the data has been read.

4.Re-position

- Re-positioning is simply moving the file pointers forward or backward depending upon the user's requirement. It is also called as seeking.

5.Delete

- Deleting the file will not only delete all the data stored inside the file, It also deletes all the attributes of the file. The space which is allocated to the file will now become available and can be allocated to the other files.

6.Truncate

- Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file get replaced.

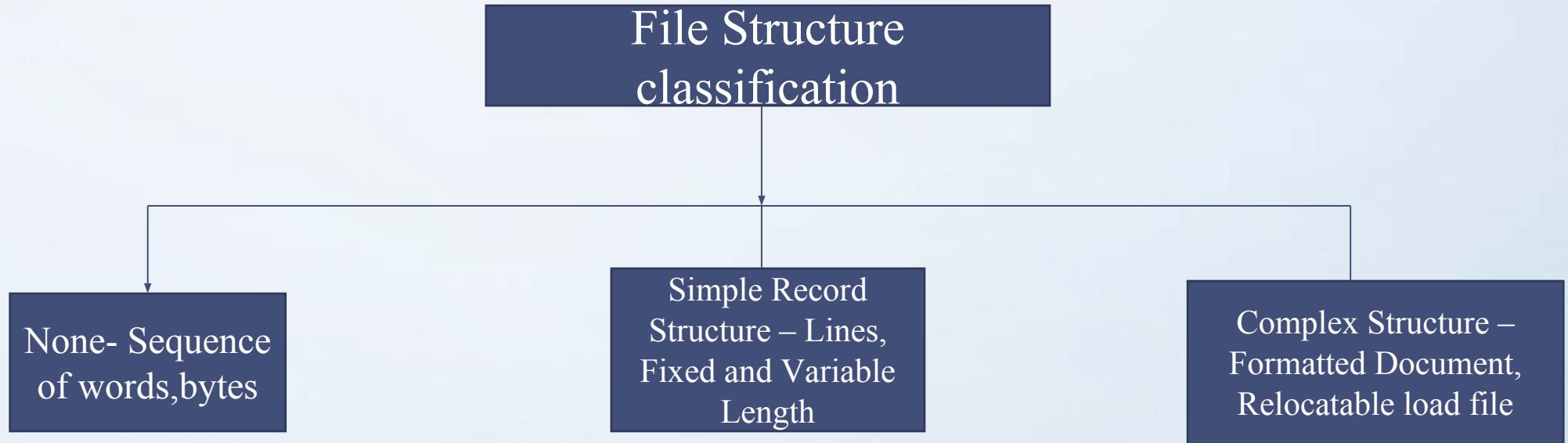
File Types – Names, Extension

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

File Structure

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

File Structure



File Access Methods

File Access
Methods

Sequential

Direct

Hashed

File Access Methods

1. **Sequential Access Method:** – Sequential Access Method is one of the simplest file access methods. Most of the OS (operating system) uses a sequential access method to access the file. In this method, word by word, the operating system reads the file, and we have a pointer that points to the file's base address. If we need to read the file's first word, then there is a pointer that offers the word which we want to read and increments the value of the word by 1, and till the end of the file, this process will continue.
 - The Modern world system offers the facility of index access and direct access to file. But the sequential access method is one of the most used methods because more files like text files, audio files, and the video files require to be sequentially accessed.
 - **Key Points:**
 1. Sequential Access Method is reasonable for tape.
 2. In the sequential access method, if we use read command, then the pointer is moved ahead by 1.
 3. If the write command is used, then the memory will be allocated, and at the end of the file, a pointer will be moved.

File Access Methods

2. Direct Access Method: - Direct Access Method is also called the Relative access method. In the Database System, we mostly use the Direct Access Method. In most of the situations, there is a need for information in the filtered form from the database. And in that case, the speed of sequential access may be low and not efficient.

- Let us assume that each storage block stores 4 records, and we already know that the block which we require is stored in the 10th block. In this situation, the sequential access method is not suitable. Since if we use this, then to access the record which is needed, this method will traverse all the blocks.
- So, in this situation, the Direct access method gives a more satisfying result, else the OS (operating system) needs to perform a few complex jobs like defining the block number, which is required. It is mostly implemented in the application of the database.

File Access Methods

3. Index Access Method: – Index Access Method is another essential method of file accessing. In this method, for a file an index is created, and the index is just like an index which is at the back of the book. The index includes the pointer to the different blocks. If we want to find a record in the file, then first, we search in the index, and after that, with the help of the pointer, we can directly access the file.

- In the Index Access method, we can search fast in the large database, and also easy. But in this, the method need some additional space to store the value of the index in the memory.

- **Key Points:**

1. On top of the sequential access method, the index access method is built.
2. The Index access method can control the pointer with the help of the index.



SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

File Sharing



Objective of File sharing

- For multiple users-File sharing, file naming and file protection is a challenging and important task.
- In order to handle this, file access control and protection need to be done.
- If a user wants to share a file with other user, proper protection schemes need to be followed.
- Whenever a file is shared for various users on distributed systems , it must be shared across distributed systems.
- In order to share and distribute files, Network File System (NFS) is used.
- For single user system ,the system need to maintain many files and directory attributes.



File sharing-Multiple Users

- Many file systems evolved with the concept of file (or directory) **owner** (or **user**) and **group**.
- The owner is the user who can change attributes and grant access and who has the most control over the file.
- The group attribute defines a subset of users who can share access to the file.
- For example, the owner of a file on a UNIX system can issue all operations on a file, while members of the file's group can execute one subset of those operations, and all other users can execute another subset of operations. Exactly which operations can be executed by group members and other users is definable by the file's owner



File sharing-Remote File Systems

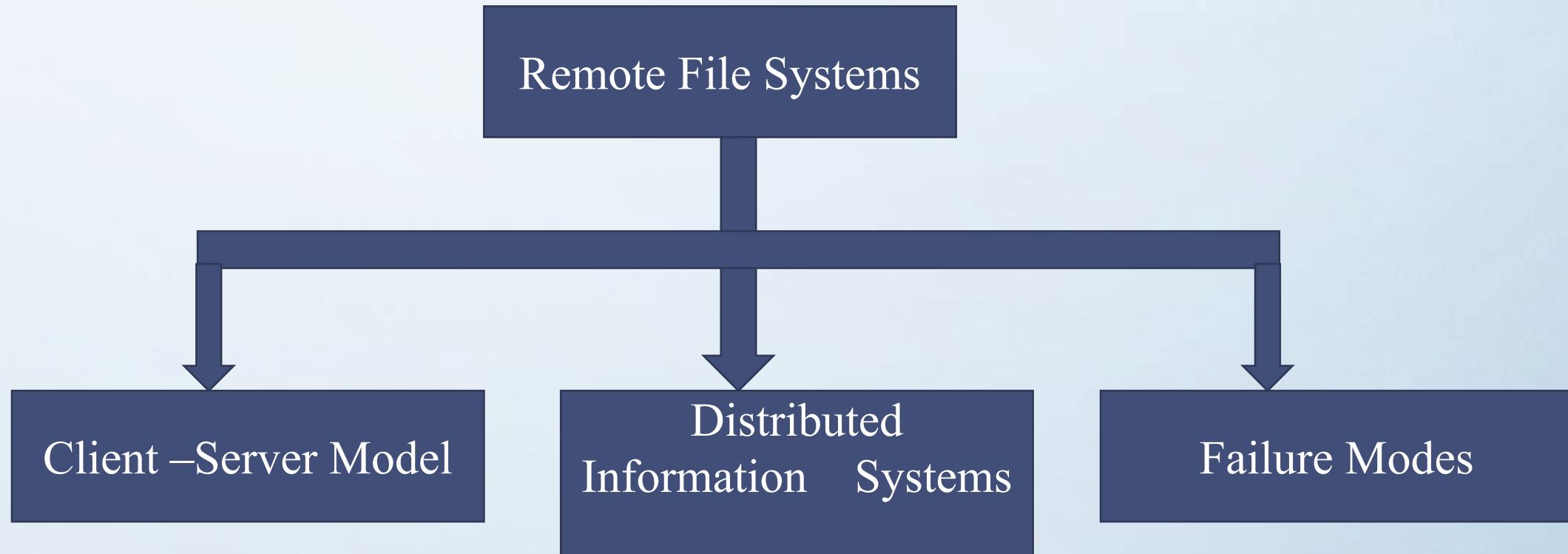
- The communication among remote computers uses the concept called Networking that allows the sharing of resources spread across a campus or even around the world.
- The first implemented method for file sharing involves manually transferring files between machines via programs like ftp.
- The second major method for file sharing uses a **distributed file system (DFS)** in which remote directories are visible from a local machine.
- The third method for file sharing , uses **WorldWide Web**, is a reversion to the first.
- A browser is needed to gain access to the remote files, and separate operations (essentially a wrapper for ftp) are used to transfer files.
- Increasingly, cloud computing is being used for file sharing as well.

File sharing-Remote File Systems

- ftp is used for both anonymous and authenticated access.
- **Anonymous access** allows a user to transfer files without having an account on the remote system.
- TheWorldWideWeb uses anonymous file exchange almost exclusively.
- DFS involves a much tighter integration between the machine that is accessing the remote files and the machine providing the files.



File sharing-Remote File Systems



File sharing-Remote File Systems- Client Server Model



- The machine containing the files is the **server**, and the machine seeking access to the files is the **client**.
- The server can serve multiple clients, and a client can use multiple servers, depending on the implementation details of a given client–server facility.
- Example:
 - **NFS** is standard UNIX client-server file sharing protocol
 - **CIFS** is standard Windows protocol
 - Standard operating system file calls are translated into remote calls



File sharing -Distributed Information Systems

- Distributed Information Systems (**distributed naming services**) such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing
- Examples:
- Other distributed information systems provide ***user name/password/user ID/group ID*** space for a distributed facility.
- UNIX systems have employed a wide variety of distributed information methods.
- Sun Microsystems introduced **yellow pages** (since renamed **network information service**, or **NIS**), and most of the industry adopted its use.

File Sharing -Failure Modes

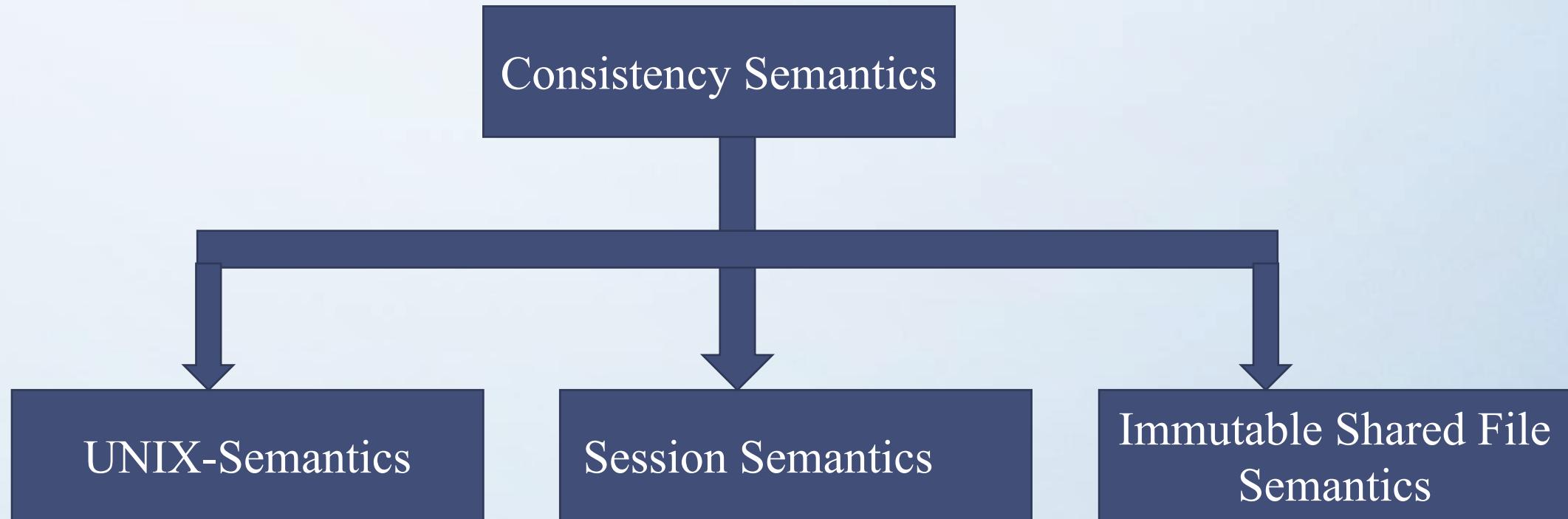
- All file systems have failure modes
 - For example corruption of directory structures or other non-user data, called **metadata**
- Remote file systems add new failure modes, due to network failure, server failure
- Recovery from failure can involve **state information** about status of each remote request
- **Stateless** protocols such as NFS v3 include all information in each request, allowing easy recovery but less security

File Sharing-Consistency Semantics



- **Consistency semantics** represent an important criterion for evaluating any file system that supports file sharing.
- These semantics specify how multiple users of a system are to access a shared file simultaneously.
- In particular, they specify when modifications of data by one user will be observable by other users.
- These semantics are typically implemented as code with the file system.
- Consistency semantics are directly related to the process synchronization algorithms

File sharing - Consistency Semantic



File Sharing-Consistency Semantics

Unix Semantics



- The UNIX file system uses the following consistency semantics:
 - Writes to an open file by a user are visible immediately to other users who have this file open.
 - One mode of sharing allows users to share the pointer of current location into the file. Thus, the advancing of the pointer by one user affects all sharing users. Here, a file has a single image that interleaves all accesses, regardless of their origin.

File Sharing-Consistency Semantics-Session Semantics



- The Andrew file system (OpenAFS) uses the following consistency semantics:
 - Writes to an open file by a user are not visible immediately to other users that have the same file open.
 - Once a file is closed, the changes made to it are visible only in sessions starting later. Already open instances of the file do not reflect these changes.
- According to these semantics, a file may be associated temporarily with several (possibly different) images at the same time.

File Sharing-Consistency Semantics-Immutable-Shared-Files Ser



- A unique approach is that of **immutable shared files**.
- Once a file is declared as shared by its creator, it cannot be modified. An immutable file has two key properties: its name may not be reused, and its contents may not be altered.
- Thus, the name of an immutable file signifies that the contents of the file are fixed.
- The implementation of these semantics in a distributed system

File Protection-Objective

- When information is stored in a computer system, we want to keep it safe from physical damage (the issue of reliability) and improper access (the issue of protection).
- Reliability is generally provided by duplicate copies of files.
- Many computers have systems programs that automatically (or through computer-operator intervention) copy disk files to tape at regular intervals (once per day or week or month) to maintain a copy should a file system be accidentally destroyed.
- File systems can be damaged by hardware problems (such as errors in reading or writing), power surges or failures, head crashes, dirt, temperature extremes, and vandalism.
- Files may be deleted accidentally. Bugs in the file-system software can also cause file contents to be lost



Protection

- Protection can be provided in many ways. Protection mechanisms provide controlled access by limiting the types of file access that can be made.
 - **Read.** Read from the file.
 - **Write.** Write or rewrite the file.
 - **Execute.** Load the file into memory and execute it.
 - **Append.** Write new information at the end of the file.
 - **Delete.** Delete the file and free its space for possible reuse.
 - **List.** List the name and attributes of the file.
 - Other operations, such as renaming, copying, and editing the file, may also be controlled.

Protection-Access Control

- The most general scheme to implement identity dependent access is to associate with each file and directory an **access-control list (ACL)** specifying user names and the types of access allowed for each user.
- When a user requests access to a particular file, the operating system checks the access list associated with that file.
- If that user is listed for the requested access, the access is allowed. Otherwise, a protection violation occurs, and the user job is denied access to the file.



Protection-Access Control

- Mode of access: read(4 bits), write(2 bits), execute(1 bit)
- Three classes of users on Unix / Linux(Default Permission)

RWX

a) **owner access** 7 ⇒ 1 1 1

RWX

b) **group access** 6 ⇒ 1 1 0

RWX

c) **public access** 1 ⇒ 0 0 1

To change Permissions the following commands are used :

- Chmod: Change Mode(For files)
- Chown: Change Owner(For Files)
- Chgroup: Change group permission(For files)

Protection-Access Control-Control List Management

- Example:

Windows users typically manage access-control lists via the GUI. Figure shows a file-permission window on Windows 7 NTFS file system. In this example, user “guest” is specifically denied access to the file ListPanel.java



Unix File Permission Example

- While using **ls -l** command, it displays various information related to file permission as follows :

```
$ ls -l /home/amrood
-rwxr-xr-- 1 amrood    users 1024 Nov 2 00:10 myfile
drwxr-xr-- 1 amrood    users 1024 Nov 2 00:10 mydir
```

- The first column represents different access modes, i.e., the permission associated with a file or a directory.
- The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) –
- The first three characters (2-4) represent the permissions for the file's owner. For example, **-rwxr-xr--** represents that the owner has read (r), write (w) and execute (x) permission.
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example, **-rwxr-xr--** represents that the group has read (r) and execute (x) permission, but no write permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example, **-rwxr-xr--** represents that there is **read (r)** only permission.

File System Implementation

File System Implementation

- File System Structure
- Directory Implementation
- File Allocation Methods
- Swap Space Management

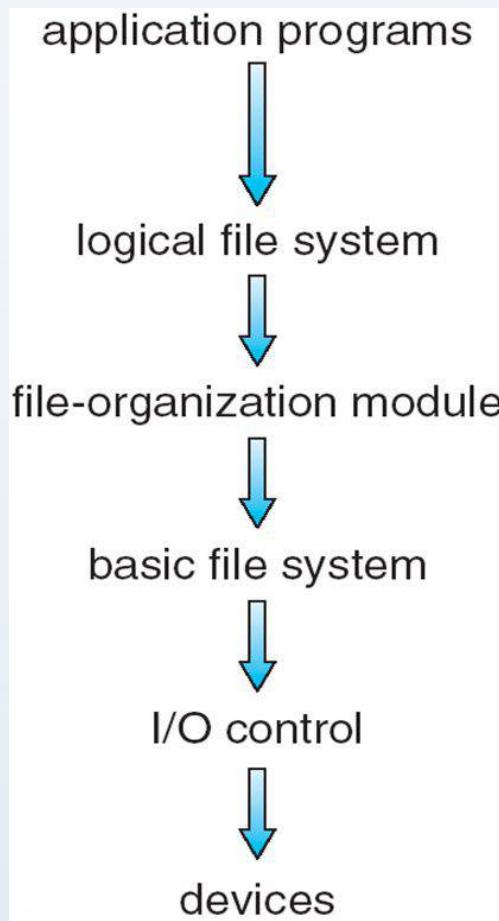
Objectives of File System Implementation

- It helps to understand the file system structure
- It helps to apprehend about how the files and folders are organized in the directory structure and its classification
- To describe the implementation of remote file systems
- To discuss block allocation and free-block algorithms and trade-offs
- To describe the details of implementing local file systems and directory structures

File-System Structure

- File structure
 - Logical storage unit
 - Collection of related information
- **File system** resides on secondary storage (disks)
 - Provided user interface to storage, mapping logical to physical
 - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
- Disk provides in-place rewrite and random access
 - I/O transfers performed in **blocks** of **sectors** (usually 512 bytes)
- **File control block** – storage structure consisting of information about a file
- **Device driver** controls the physical device
- File system organized into layers

Layered File System



- **application program** asks for a file, the first request is directed to the logical file system.
- The **logical file system** contains the **Meta data information** of the file and directory structure and provides **protection** also. It translates file name into file number, file handle, location by maintaining file control blocks (inodes in UNIX)
- **The logical blocks of files need to be mapped to physical blocks in harddisk.** This mapping is done by **File organization module**
- The basic file system - issues the commands to I/O control in order to fetch those blocks. **Basic file system** given command like “retrieve block 123” translates to device driver.
- I/O controls contain the codes by using which it can access hard disk.
- These codes are known as **device drivers** Given commands like “read drive1, cylinder 72, track 2, sector 10, into memory location 1060” outputs low-level hardware specific commands to hardware controller

File System Implementation

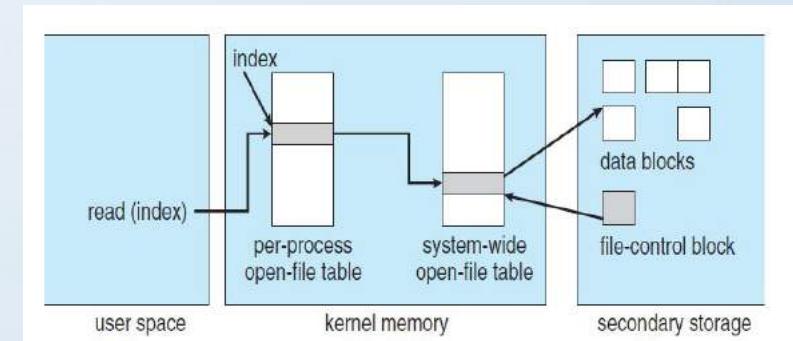
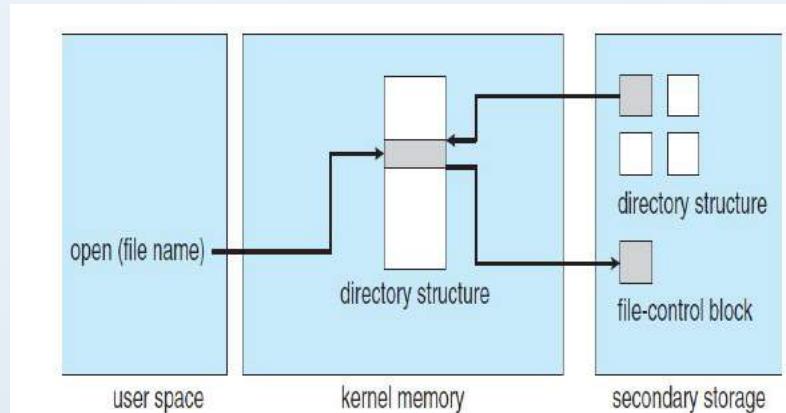
- **on-disk and in-memory structures**
- **On Disk Data Structures**
- Boot Control Block - Boot Control Block contains all the information which is needed **to boot an operating system from that volume**. It is called boot block in UNIX file system. In NTFS, it is called the partition boot sector.
- Volume Control Block - Volume control block all the information regarding that volume such as number of blocks, size of each block, partition table, pointers to free blocks and free FCB blocks. In UNIX file system, it is known as super block. In NTFS, this information is stored inside master file table.
- Directory Structure (per file system) - A directory structure

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

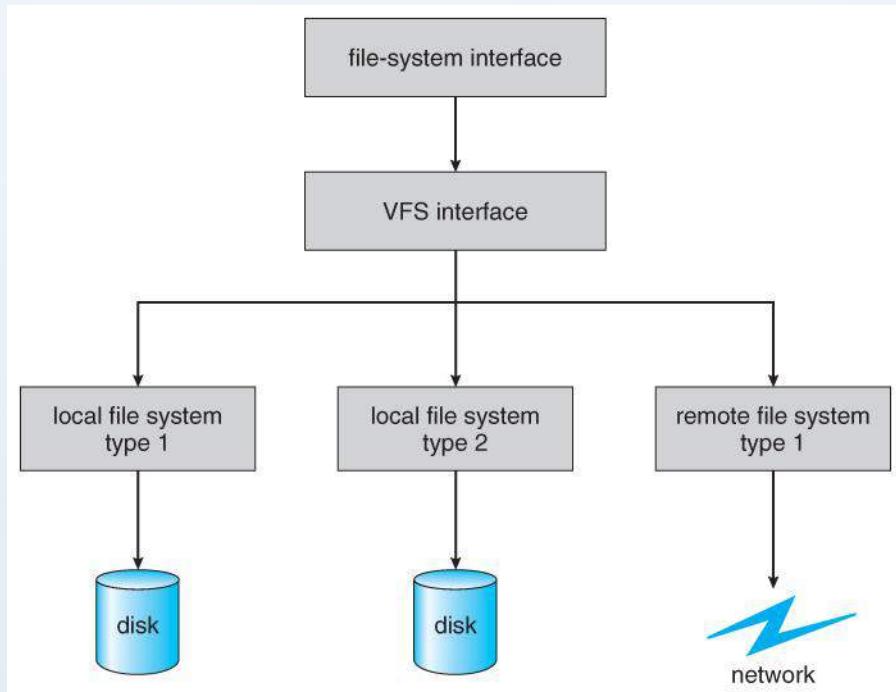
File control Block

In-Memory File System Structures

- The in-memory data structures are used for file system management as well as performance improvement via caching. This information is loaded on the mount time and discarded on ejection.



Virtual File Systems



- *Virtual File Systems, VFS*, provide a common interface to multiple different filesystem types. In addition, it provides for a unique identifier (*vnode*) for files across the entire space, including across all filesystems of different types. (UNIX inodes are unique only across a single filesystem, and certainly do not carry across networked file systems.)
- The VFS in Linux is based upon four key object types:
 - The inode object, representing an individual file
 - The file object, representing an open file.
 - The superblock object, representing a filesystem.
 - The dentry object, representing a directory entry.

Directory Implementation

Directories need to be fast to search, insert, and delete, with a minimum of wasted disk space.

Linear List

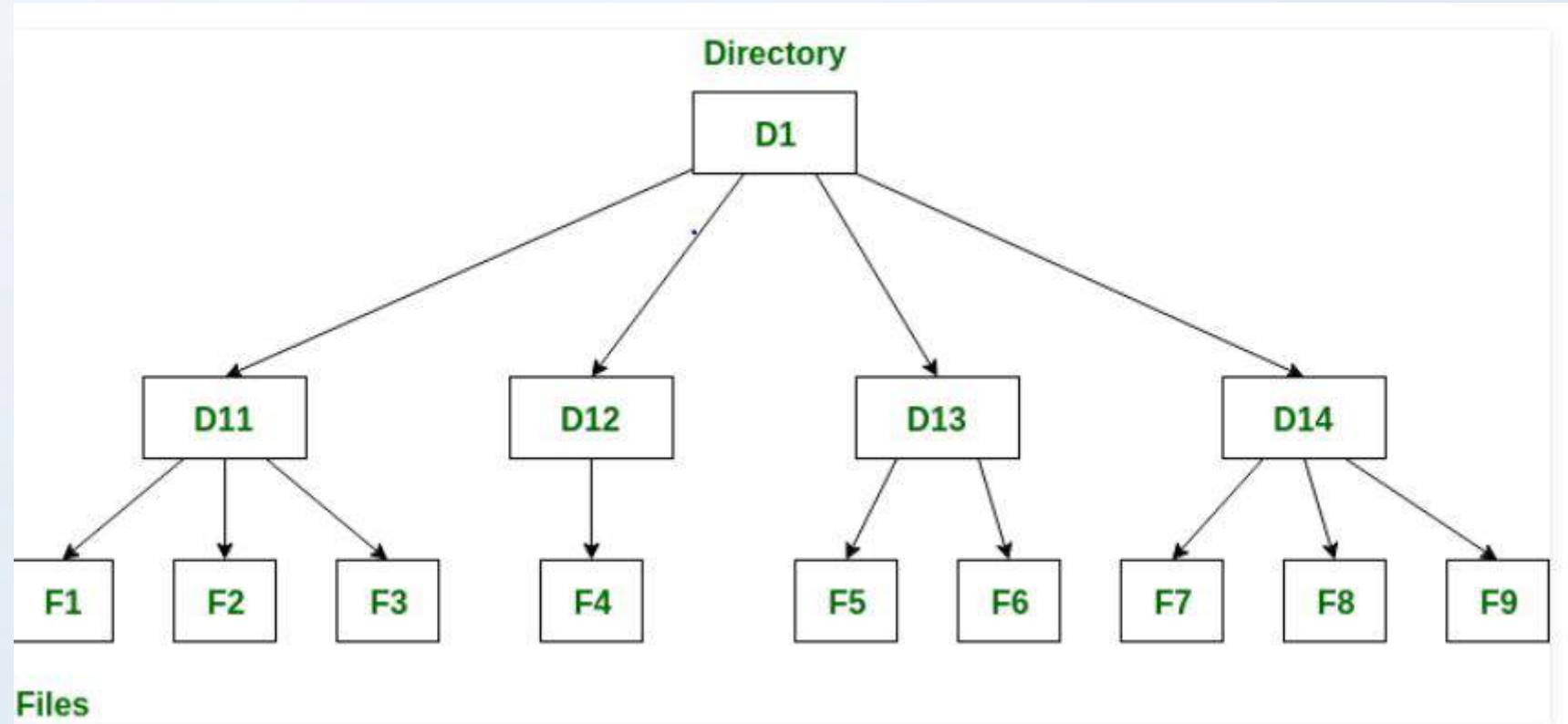
- A linear list is the simplest and easiest directory structure to set up, but it does have some drawbacks.
- Finding a file (or verifying one does not already exist upon creation) requires a linear search.
- Deletions can be done by moving all entries, flagging an entry as deleted, or by moving the last entry into the newly vacant position.
- Sorting the list makes searches faster, at the expense of more complex insertions and deletions.
- A linked list makes insertions and deletions into a sorted list easier, with overhead for the links.
- More complex data structures, such as B-trees, could also be considered.

Hash Table

- A hash table can also be used to speed up searches.
- Hash tables are generally implemented *in addition to* a linear or other structure

Directory Structure

- A **directory** is a container that is used to contain folders and file. It organizes files and folders into a hierarchical manner.



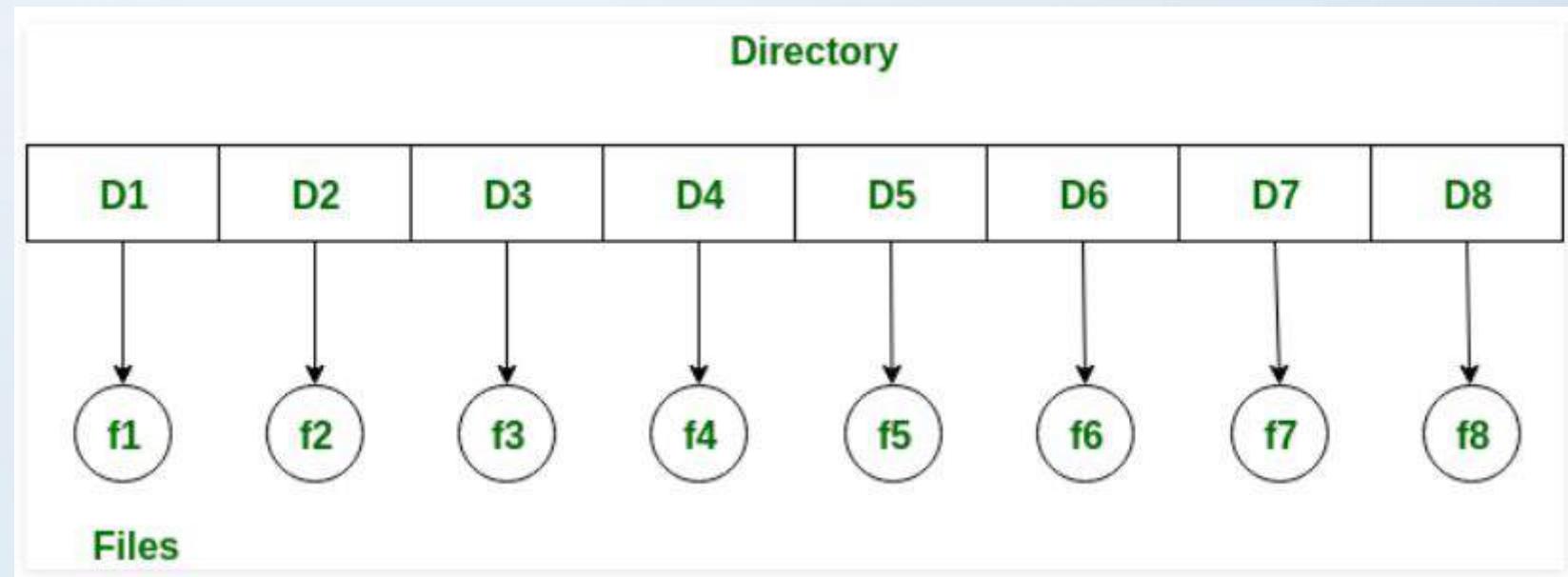
Directory Structure(Contd...)

- There are several logical structures of a directory, these are given below.

- 1. Single-level directory**
- 2. Two-level directory**
- 3. Tree-structured directory**
- 4. Acyclic graph directory**
- 5. General graph directory structure**

1. Single level Directory Structure

- Single level directory is simplest directory structure. In it all files are contained in same directory which make it easy to support and understand.
- A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have the unique name . if two users call their dataset test, then the unique name rule violated.



Single level Directory Structure(Contd...)

Advantages:

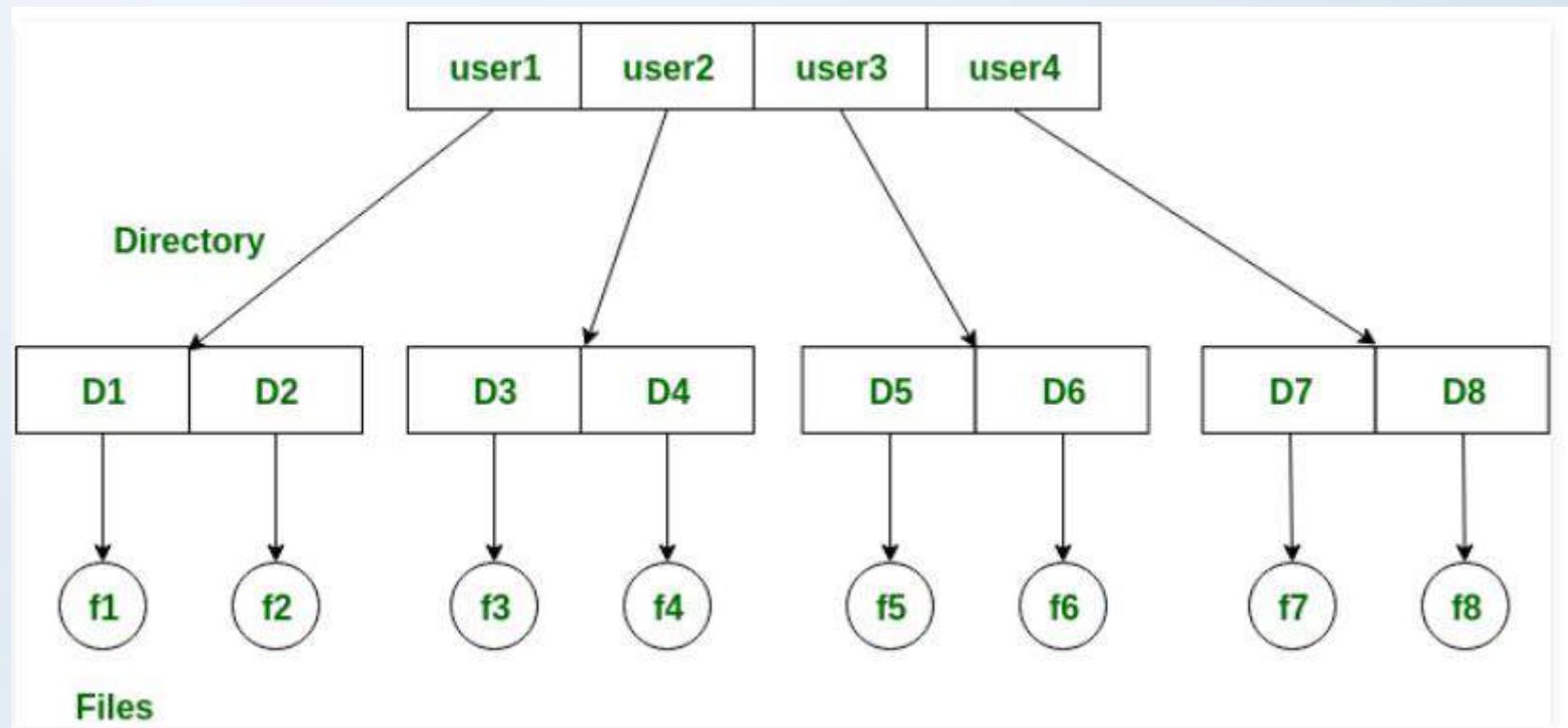
- Since it is a single directory, so its implementation is very easy.
- If the files are smaller in size, searching will become faster.
- The operations like file creation, searching, deletion, updating are very easy in such a directory structure.

Disadvantages:

- There may chance of name collision because two files can not have the same name.
- Searching will become time taking if the directory is large.
- In this can not group the same type of files together.

2. Two-level directory

- As we have seen, a single level directory often leads to confusion of files names among different users. the solution to this problem is to create a separate directory for each user.
- In the two-level directory structure, each user has their own *user files directory (UFD)*. The UFDs have similar structures, but each lists only the files of a single user. system's *master file directory (MFD)* is searched whenever a new user id=s logged in. The MFD is indexed by username or account number, and each entry points to the UFD for that user.



Two-level directory (contd...)

Advantages:

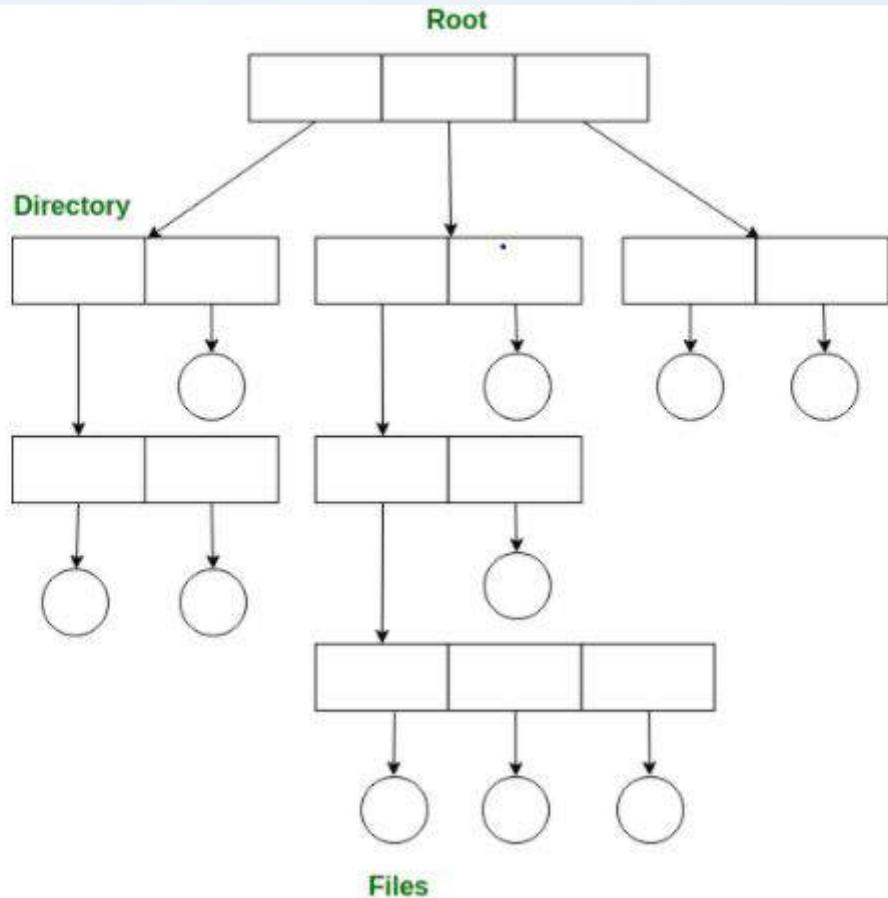
- We can give full path like /User-name/directory-name/.
- Different users can have same directory as well as file name.
- Searching of files become more easy due to path name and user-grouping.

Disadvantages:

- A user is not allowed to share files with other users.
- Still it not very scalable, two files of the same type cannot be grouped together in the same user

3. Tree-structured directory

- Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height.
- This generalization allows the user to create their own subdirectories and to organize their files according to their needs.



Tree-structured directory(contd...)

A tree structure is the most common directory structure. The tree has a root directory, and every file in the system have a unique path.

Advantages:

1. Very generalize, since full path name can be given.
2. Very scalable, the probability of name collision is less.
3. Searching becomes very easy, we can use both absolute path as well as relative

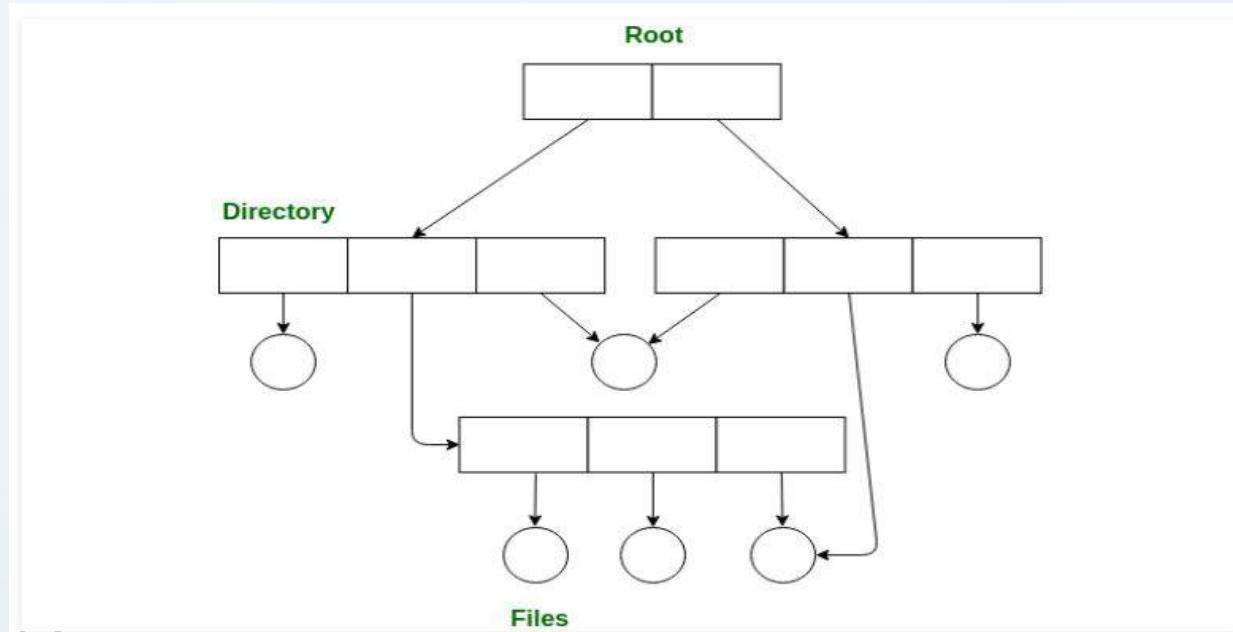
Disadvantages:

1. Every file does not fit into the hierarchical model, files may be saved into multiple directories.
 2. We can not share files.
 3. It is inefficient, because accessing a file may go under multiple directories.
-

4. Acyclic graph directory

- An acyclic graph is a graph with no cycle and allows to share subdirectories and files. The same file or subdirectories may be in two different directories. It is a natural generalization of the tree-structured directory.
- It is used in the situation like when two programmers are working on a joint project and they need to access files. The associated files are stored in a subdirectory, separating them from other projects and files of other programmers, since they are working on a joint project so they want the subdirectories to be into their own directories. The common subdirectories should be shared. So here we use Acyclic directories.
- It is the point to note that shared file is not the same as copy file . If any programmer makes some changes in the subdirectory it will reflect in both subdirectories.

Acyclic graph directory(contd...)



Advantages:

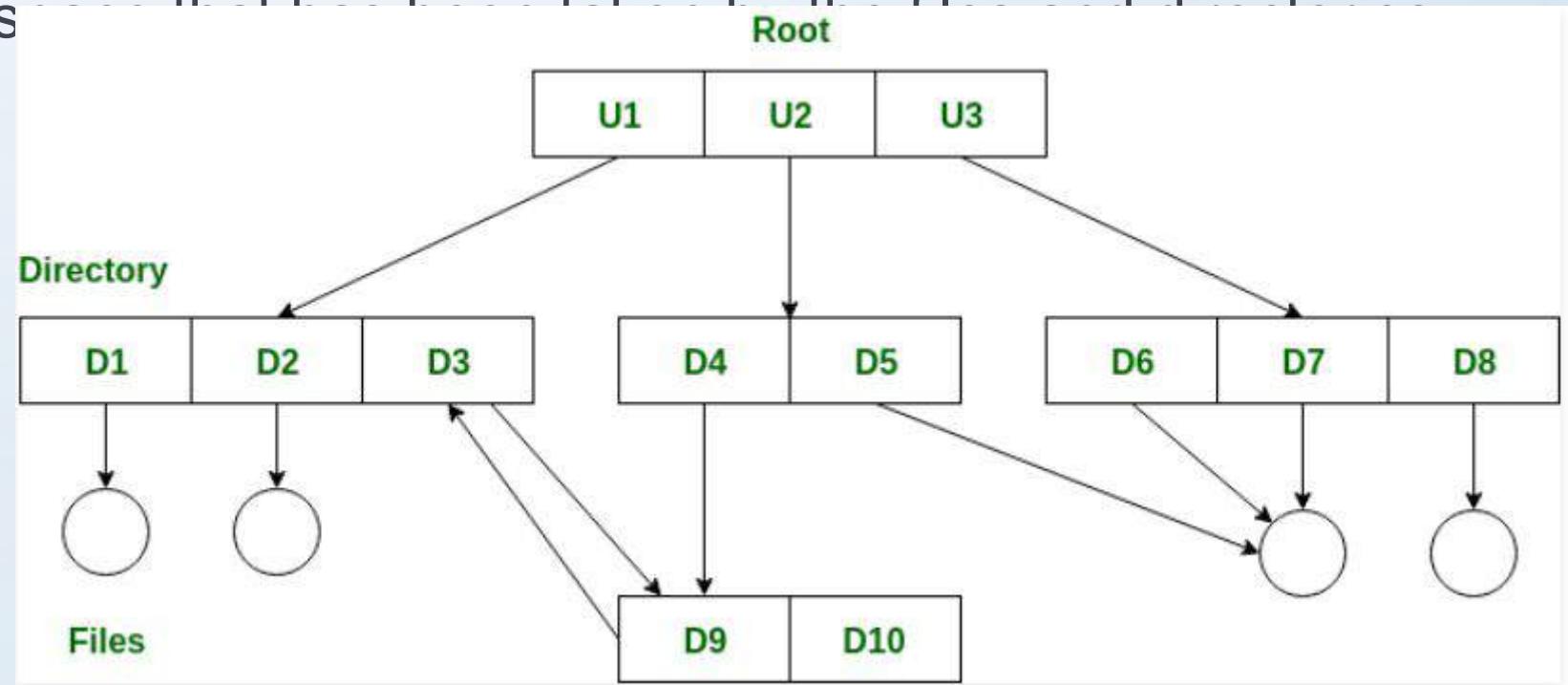
- We can share files.
- Searching is easy due to different-different paths.

Disadvantages:

- We share the files via linking, in case of deleting it may create the problem,
- If the link is softlink then after deleting the file we left with a dangling pointer.
- In case of hardlink, to delete a file we have to delete all the reference associated with it.

5. General graph directory structure

- In general graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.
- The main problem with this kind of directory structure is to calculate total size or s



General graph directory structure(contd...)

Advantages:

- It allows cycles.
- It is more flexible than other directories structure.

Disadvantages:

- It is more costly than others.
- It needs garbage collection.

Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
- Contiguous allocation
- Linked allocation
- Indexed allocation

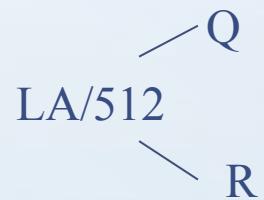
Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk
- Simple – only starting location (block #) and length (number of blocks) are required
- Random access
- Wasteful of space (dynamic storage-allocation problem)
- Files cannot grow

Contiguous Allocation

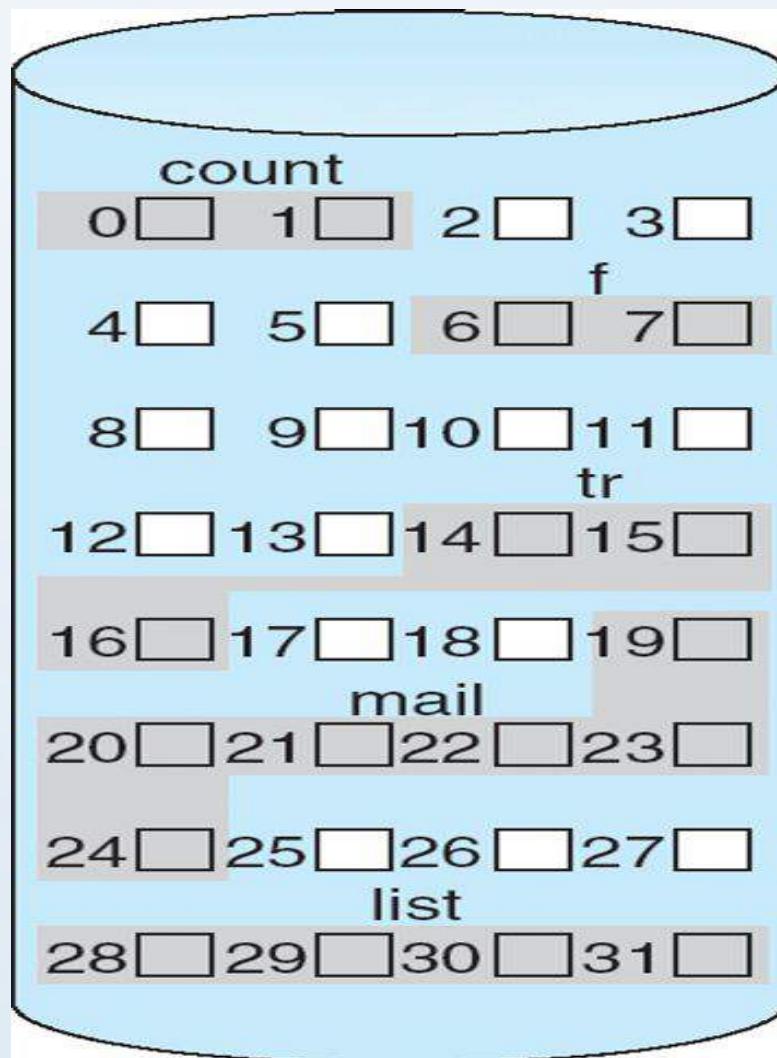
- Mapping from logical to physical
 - Block to be accessed = ! + starting address
 - Displacement into block = R

LA/512



```
graph TD; LA[LA/512] --> Q[Q]; LA --> R[R]
```

Contiguous Allocation of Disk Space



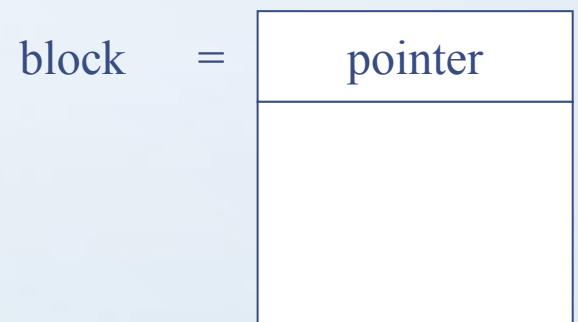
directory		
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Extent-Based Systems

- Many newer file systems (I.e. Veritas File System) use a modified contiguous allocation scheme
- Extent-based file systems allocate disk blocks in extents
- An **extent** is a contiguous block of disks
 - Extents are allocated for file allocation
 - A file consists of one or more extents

Linked Allocation

Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.



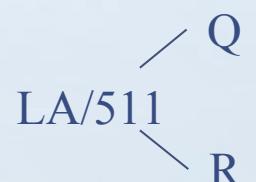
Linked Allocation (Cont.)

- Simple – need only starting address
- Free-space management system – no waste of space
- No random access
- Mapping

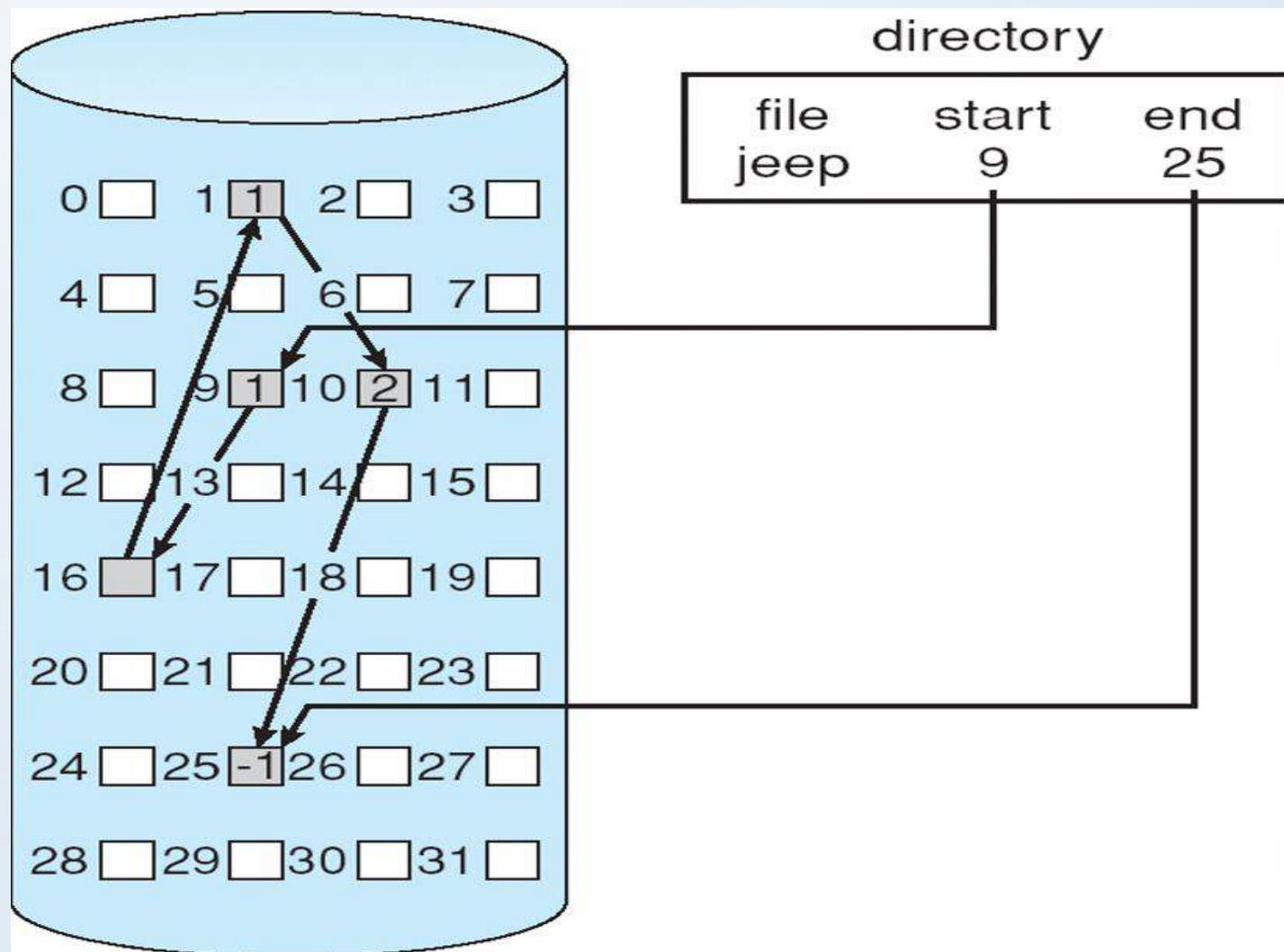
Block to be accessed is the Qth block in the linked chain of blocks representing the file.

Displacement into block = R + 1

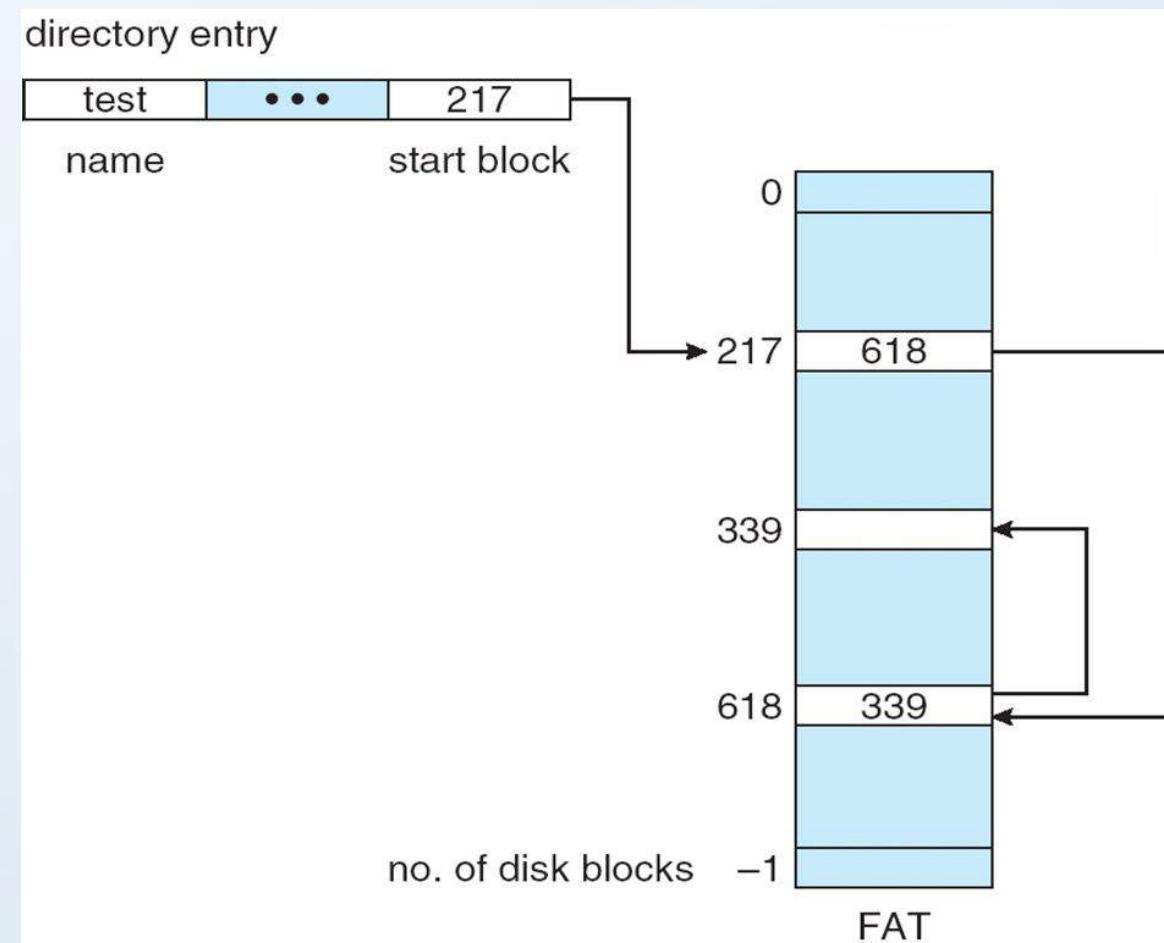
File-allocation table (FAT) – disk-space allocation used by MS-DOS and OS/2.



Linked Allocation

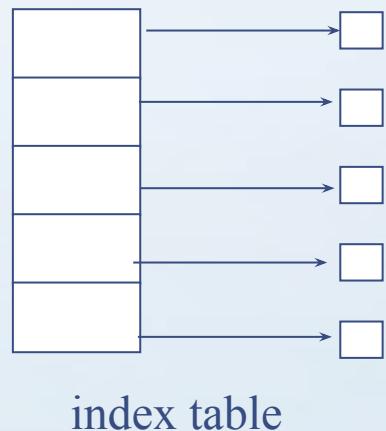


File-Allocation Table

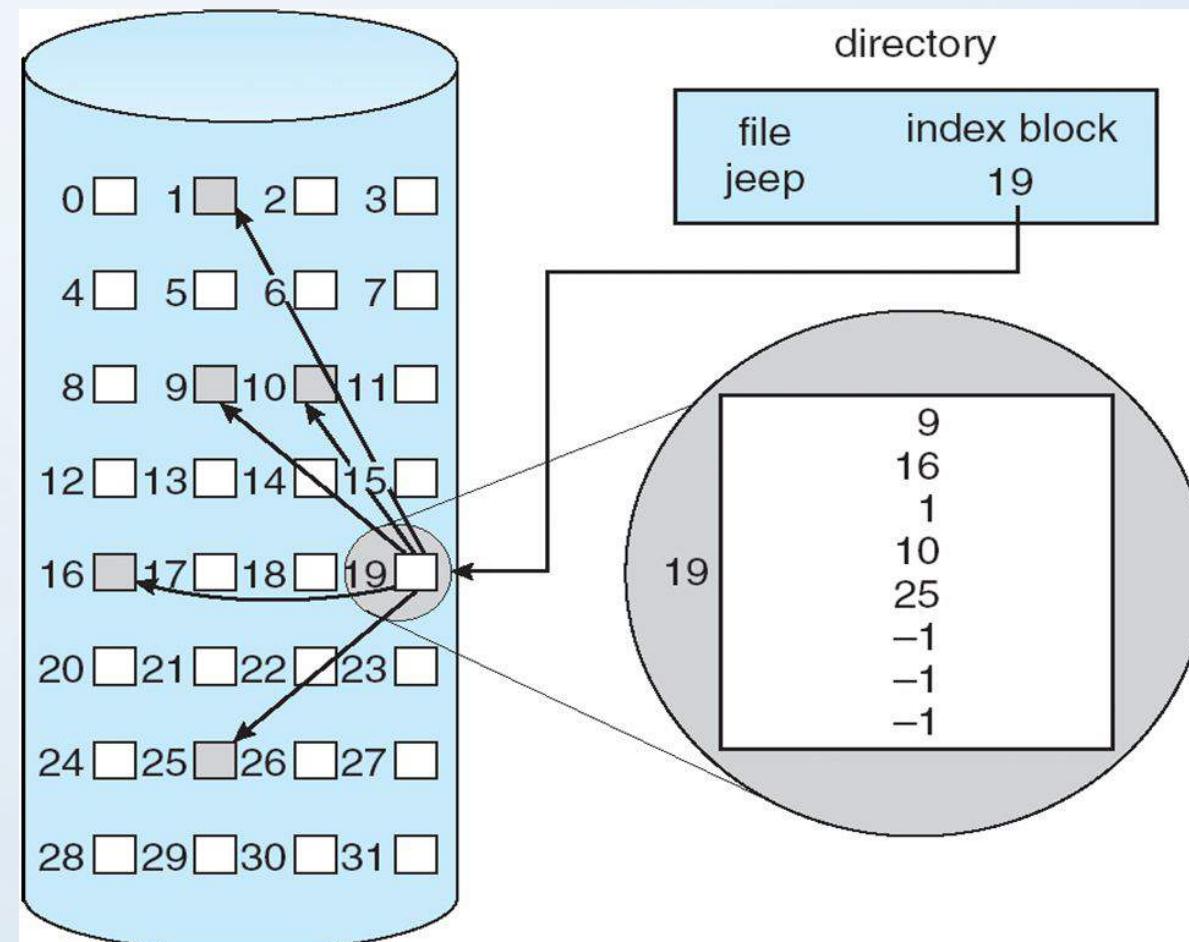


Indexed Allocation

- Brings all pointers together into the **index block**
- Logical view



Example of Indexed Allocation



Indexed Allocation (Cont.)

- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block
- Mapping from logical to physical in a file of maximum size of 256K words and block size of 512 words. We need only 1 block for index table

Q = displacement into index table

R = displacement into block

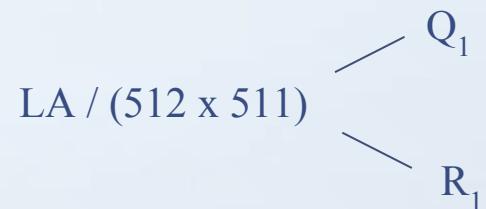


Indexed Allocation – Mapping (Cont.)

- Mapping from logical to physical in a file of unbounded length (block size of 512 words)
- Linked scheme – Link blocks of index table (no limit on size)

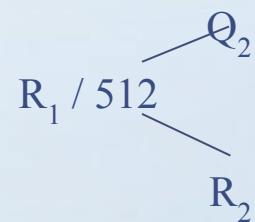
Q_1 = block of index table

R_1 is used as follows:



Q_2 = displacement into block of index table

R_2 displacement into block of file:

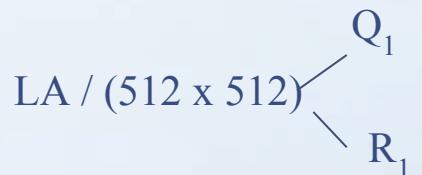


Indexed Allocation – Mapping (Cont.)

- Two-level index (maximum file size is 512^3)

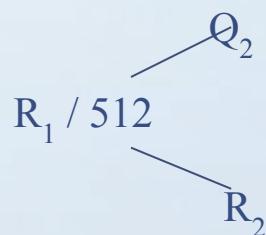
Q_1 = displacement into outer-index

R_1 is used as follows:

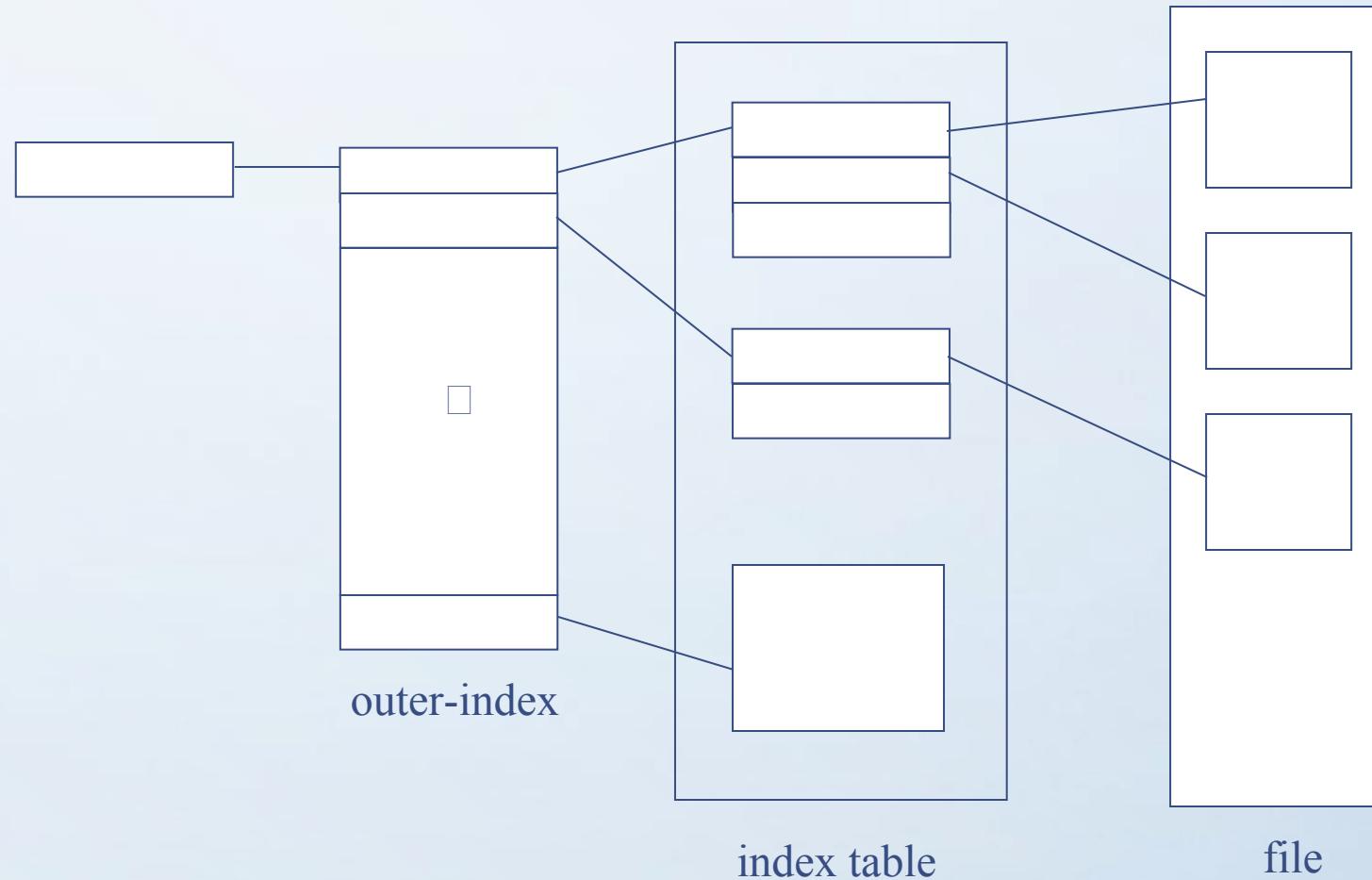


Q_2 = displacement into block of index table

R_2 displacement into block of file:



Indexed Allocation – Mapping (Cont.)



Worksheet

Problem:1

Consider a file currently consisting of 100 blocks. Assume that the file-control block (and the index block, in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one block, the following conditions hold. In the contiguous-allocation case, assume that there is no room to grow at the beginning but there is room to grow at the end. Also assume that the block information to be added is stored in memory.

- a. The block is added at the beginning.
- b. The block is added in the middle.
- c. The block is added at the end.
- d. The block is removed from the beginning.
- e. The block is removed from the middle.

Answer

	<u>Contiguous</u>	<u>Linked</u>	<u>Indexed</u>
a.	201	1	1
b.	, 101	52	1
c.	1	3	1
d.	198	1	0
e.	98	52	0
f.	0	100	0

Problem:2

File of 101 blocks, file positions already in memory, and block to add already in memory. Every directory or index operation is done in memory. There is room on the disk after the file but not before. How many operations to...

- 1.Add a block at the beginning
- 2.Add a block after the 51st block
- 3.Add a block at the end
- 4.Remove the beginning block
- 5.Remove the 51st block
- 6.Remove the end block

Free-Space Management

Bit vector (n blocks)



$\text{bit}[i]$ 0 \Rightarrow $\text{block}[i]$ free
= 1 \Rightarrow $\text{block}[i]$ occupied

(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit

Free-Space Management (Cont.)

- Bit map requires extra space

- Example:

block size = 2^{12} bytes

disk size = 2^{30} bytes (1 gigabyte)

$$n = 2^{30}/2^{12} = 2^{18} \text{ bits (or } 32\text{K bytes)}$$

- Easy to get contiguous files
- Linked list (free list)
 - Cannot get contiguous space easily
 - No waste of space
- Grouping
- Counting

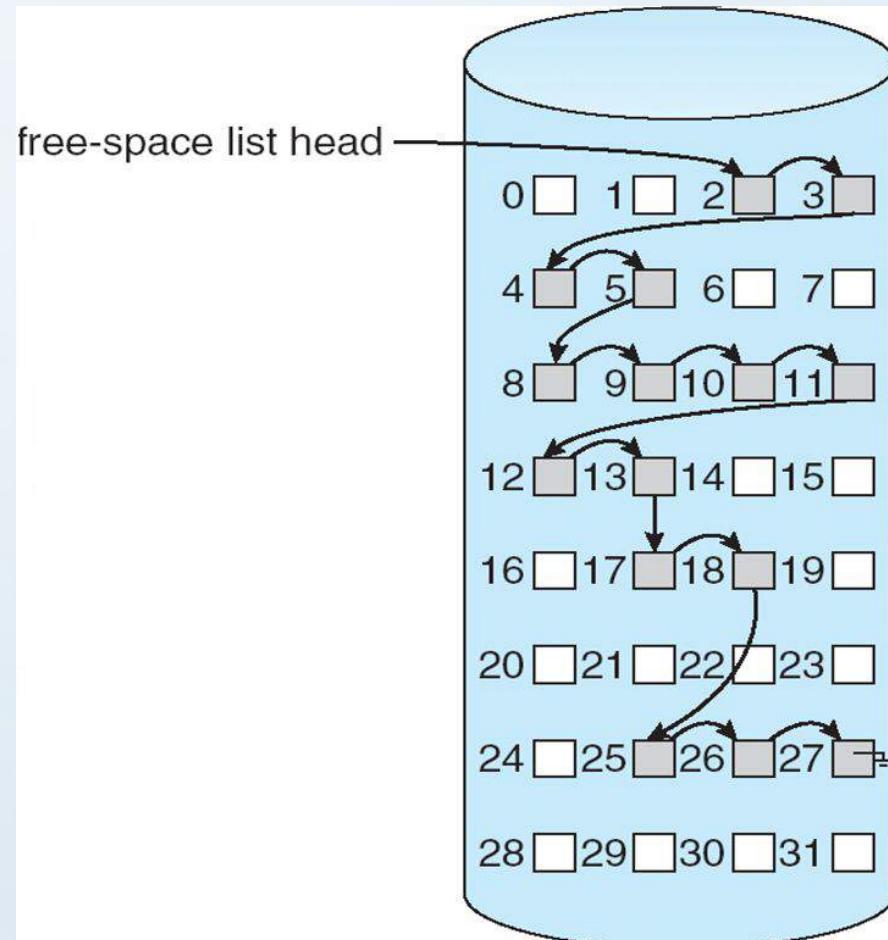
Free-Space Management (Cont.)

- Need to protect:
 - Pointer to free list
 - Bit map
 - Must be kept on disk
 - Copy in memory and disk may differ
 - Cannot allow for $\text{block}[i]$ to have a situation where $\text{bit}[i] = 1$ in memory and $\text{bit}[i] = 0$ on disk
 - Solution:
 - Set $\text{bit}[i] = 1$ in disk
 - Allocate $\text{block}[i]$
 - Set $\text{bit}[i] = 1$ in memory

Directory Implementation

- Linear list of file names with pointer to the data blocks
 - simple to program
 - time-consuming to execute
- Hash Table – linear list with hash data structure
 - decreases directory search time
 - **collisions** – situations where two file names hash to the same location
 - fixed size

Linked Free Space List on Disk



Efficiency and Performance

- Efficiency dependent on:
 - disk allocation and directory algorithms
 - types of data kept in file's directory entry
- Performance
 - disk cache – separate section of main memory for frequently used blocks
 - free-behind and read-ahead – techniques to optimize sequential access
 - improve PC performance by dedicating section of memory as virtual disk, or RAM disk

Free Space Management

Objectives :

- Operating system maintains a list of free disk spaces to keep track of all disk blocks which are not being used by any file.
- Whenever a file has to be created, the list of free disk space is searched for and then allocated to the new file.
- A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk.
- Hence there is need for understanding the methods available for managing free space in the disk

Understanding the methods available for maintaining the free spaces in the disk

The system keeps tracks of the free disk blocks for allocating space to files when they are created.

Also, to reuse the space released from deleting the files, free space management becomes crucial.

The system maintains a free space list which keeps track of the disk blocks that are not allocated to some file or directory.

- The free space list can be implemented mainly as:

1. Bitmap or Bit vector

- A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block.
- The bit can take two values: 0 and 1: *0 indicates that the block is allocated and 1 indicates a free block.*
- The given instance of disk blocks on the disk in *Figure 1* (where green blocks are allocated) can be represented by a bitmap of 16 bits as: **0000111000000110**.

Bitmap or Bit vector

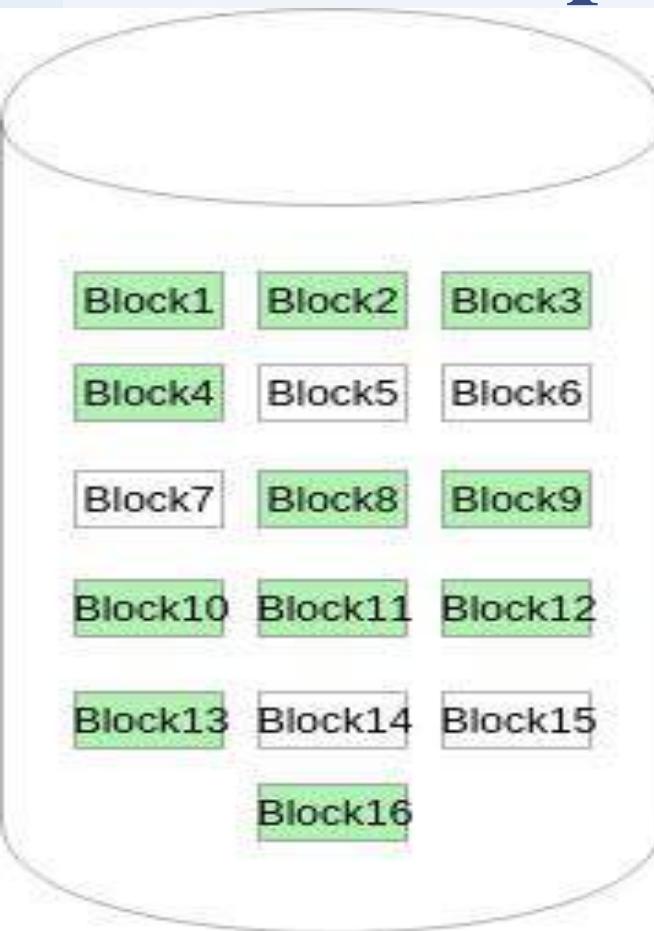


Figure - 1

- The block number can be calculated as:

$$(number\ of\ bits\ per\ word) * (number\ of\ 0-values\ words) + offset\ of\ bit\ first\ bit\ 1\ in\ the\ non-zero\ word.$$
- For the *Figure-1*, we scan the bitmap sequentially for the first non-zero word.
- The first group of 8 bits (00001110) constitute a non-zero word since all bits are not 0.
-
- After the non-0 word is found, we look for the first 1 bit. This is the 5th bit of the non-zero word. So, offset = 5.

Bitmap or Bit vector

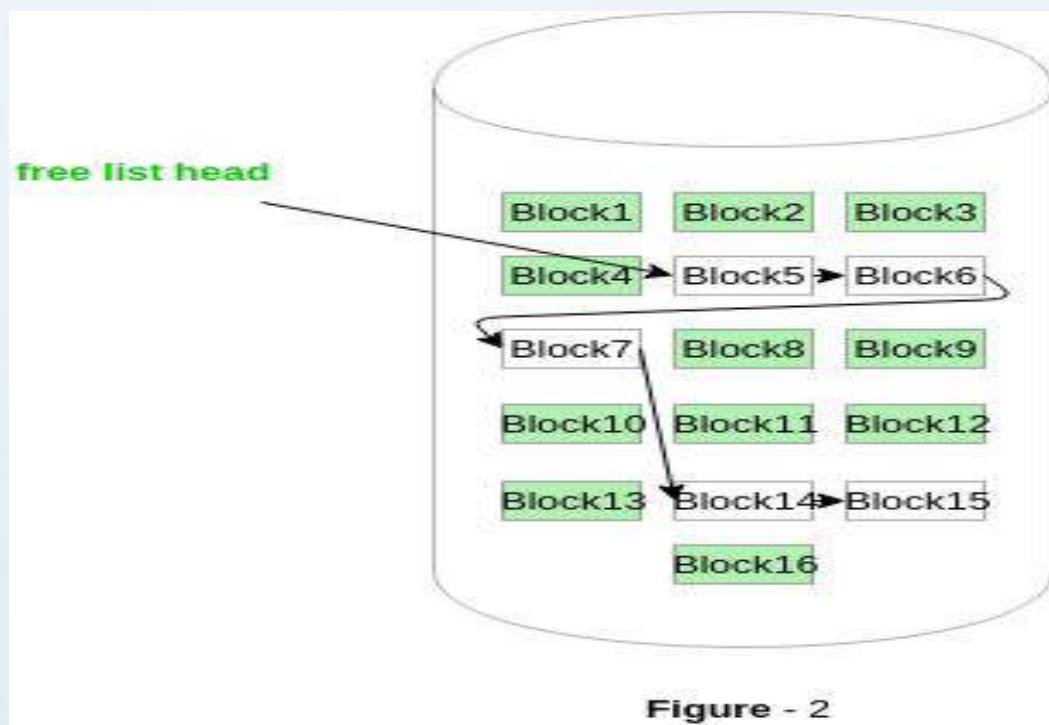
Advantages

- Simple to understand.
- Finding the first free block is efficient. It requires scanning the words (a group of 8 bits) in a bitmap for a non-zero word. (A 0-valued word has all bits 0). The first free block is then found by scanning for the first 1 bit in the non-zero word.

2.Linked List

- In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block.
- The block number of the very first disk block is stored at a separate location on disk and is also cached in memory.

2.Linked List



- In *Figure-2*, the free space list head points to Block 5 which points to Block 6, the next free block and so on.
- The last free block would contain a null pointer indicating the end of free list.

3. Grouping

- This approach stores the address of the free blocks in the first free block.
- The first free block stores the address of some, say n free blocks.
- Out of these n blocks, the first $n-1$ blocks are actually free and the last block contains the address of next n blocks.
- An **advantage** of this approach is that the addresses of a group of free disk blocks can be found easily.

4. Counting

- This approach stores the address of the first free disk block and a number n of free contiguous disk blocks that follow the first block.
Every entry in the list would contain:
- Address of first free disk block
- A number n
- For example, *in Figure-1*, the first entry of the free space list would be: ([Address of Block 5], 2), because 2 contiguous free blocks follow block 5.

Swap-Space Management

- **Swapping** is a memory management technique used in multi-programming to increase the number of process sharing the CPU.
- It is a technique of removing a process from main memory and storing it into secondary memory, and then bringing it back into main memory for continued execution.
- This action of moving a process out from main memory to secondary memory is called **Swap Out** and the action of moving a process out from secondary memory to main memory is called **Swap In**.

Swap-Space Management

- **Swap-space** — virtual memory uses **disk space** as an extension of main memory.
- Main goal for the design and implementation of swap space is to provide the *best throughput* for VM system
- Swap-space use
 - Swapping – use swap space to hold entire process image
 - Paging – store pages that have been pushed out of memory
- Some OS may support multiple swap-space
 - Put on separate disks to balance the load
- Better to **overestimate** than underestimate
 - If out of swap-space, some processes must be aborted or system crashed

Swap-Space Location

- Swap-space can be carved out of the *normal file system*, or in a *separate disk partition*.
- *A large file within the file system*: simple but inefficient
 - Navigating the *directory structure* and the disk-allocation *data structure* takes time and potentially extra disk accesses
 - *External fragmentation* can greatly increase swapping times by forcing multiple seeks during reading or writing of a process image
 - Improvement
 - Caching block location information in main memory
 - Contiguous allocation for the swap file
 - But, the cost of traversing FS data structure still remains

Swap-Space Location

- *In a separate partition: raw partition*
 - Create a swap space during disk partitioning
 - A separate swap-space storage manager is used to *allocate and de-allocate blocks*
 - Use algorithms optimized for speed, rather than storage efficiency
 - *Internal fragment* may increase
- Linux supports both approaches

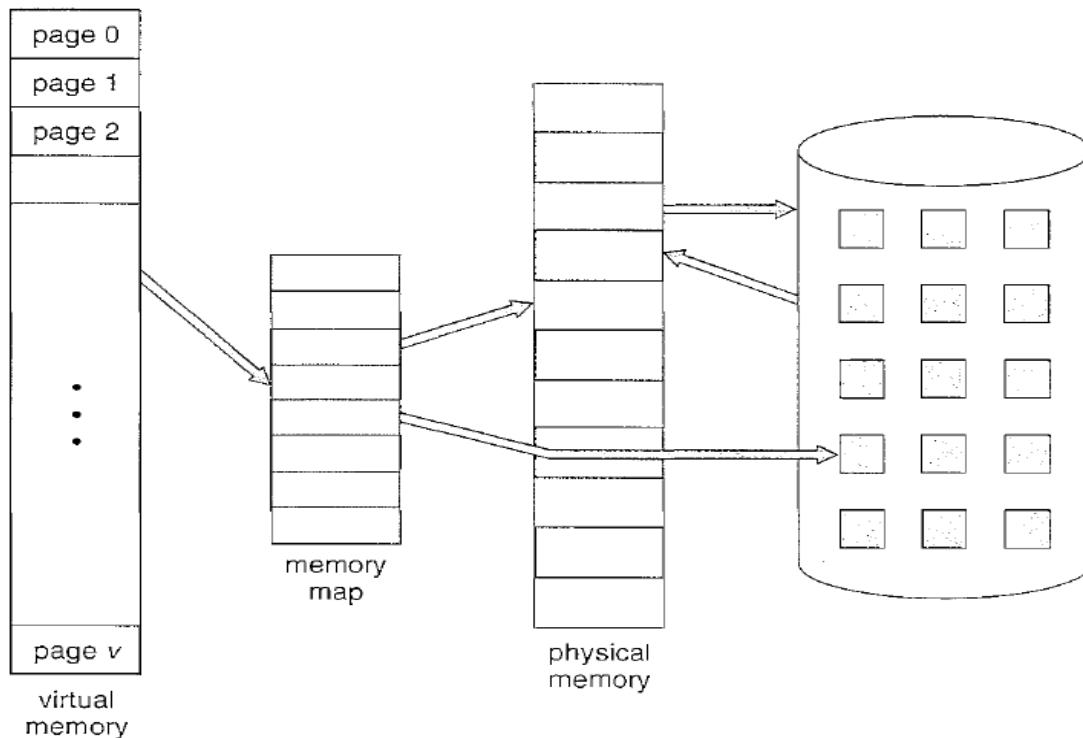
Swap-space Management: Example

- Solaris 1
 - Text-segment pages are brought in from the file system and are thrown away if selected for paged out
 - More efficient to re-read from FS than write it to the swap space
 - Swap space: only used as a backing store for pages of **anonymous memory**
 - Stack, heap, and uninitialized data
- Solaris 2
 - Allocates swap space only when a page is forced out of physical memory
 - Not when the virtual memory page is first created.

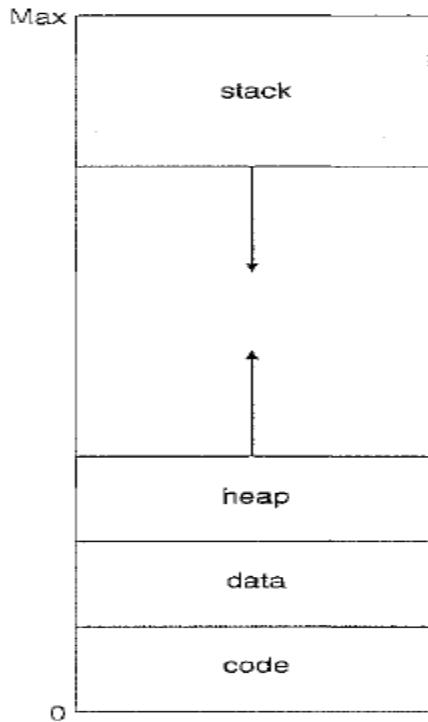
Thank you

VIRTUAL MEMORY

- Virtual memory is a technique that allows the execution of processes that are not completely in memory. One major advantage of this scheme is that programs can be larger than physical memory.
- Further, virtual memory abstracts main memory into an extremely large, uniform array of storage, separating logical memory as viewed by the user from physical memory.
- Virtual Memory involves the separation of logical memory as perceived by users from physical memory. This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available (Shown in Figure).
- Virtual memory makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available; she can concentrate instead on the problem to be programmed.

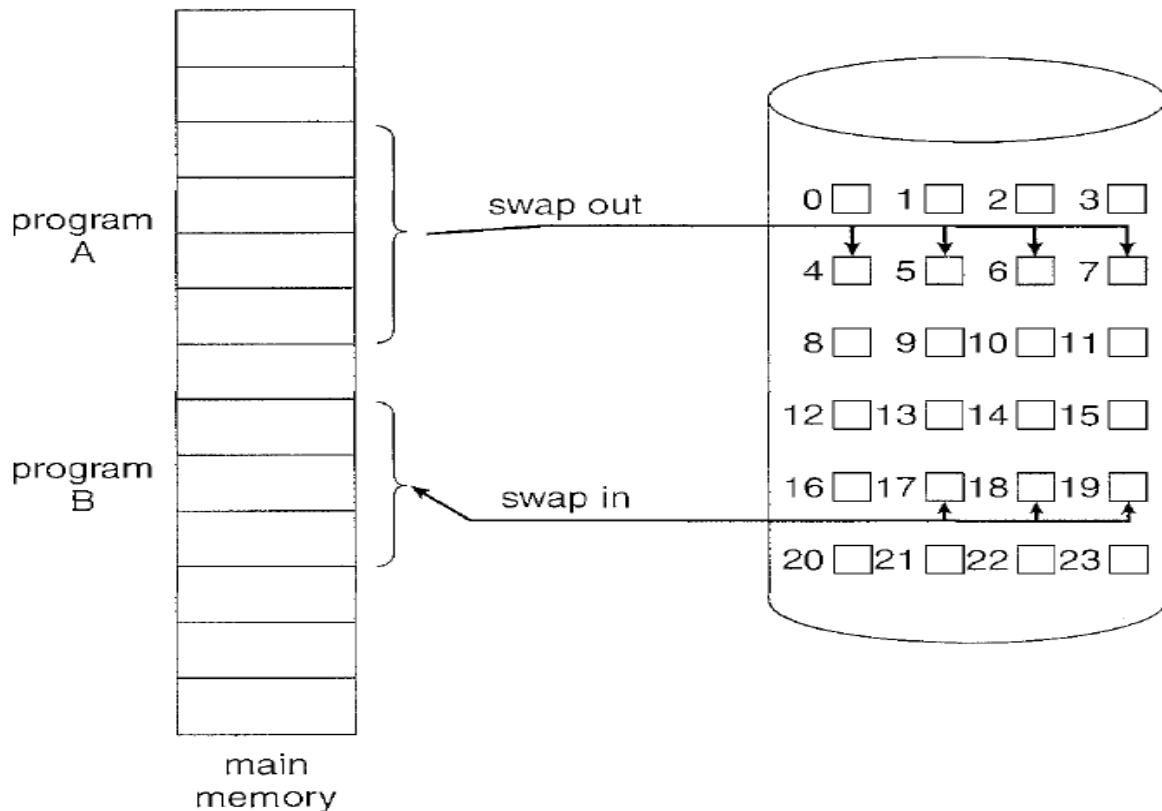


- The virtual address space of a process refers to the logical (or virtual) view of how a process is stored in memory. Typically, this view is that a process begins at a certain logical address-say, address 0-and exists in contiguous memory, as shown in Figure.



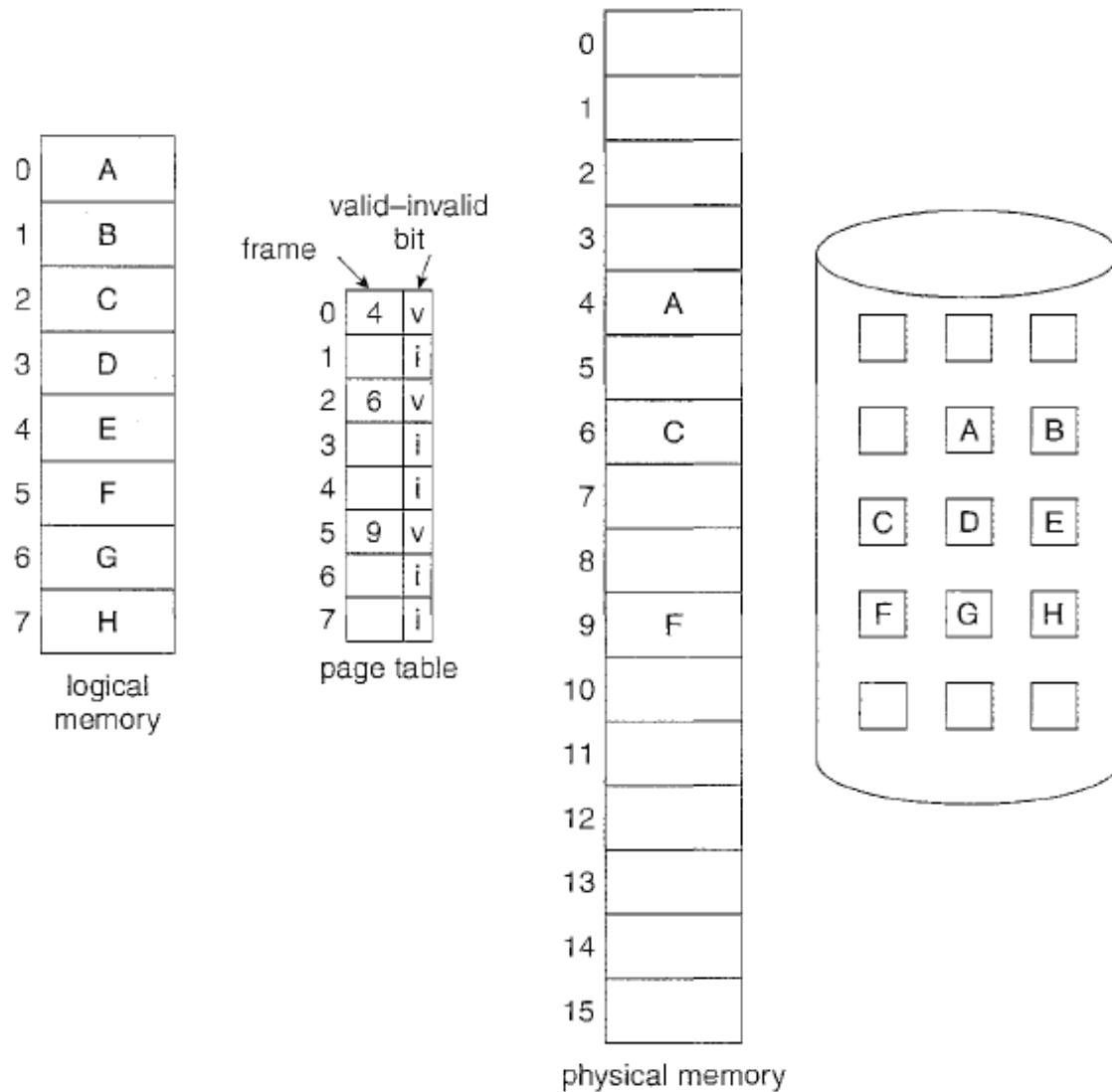
DEMAND PAGING

- Consider how an executable program might be loaded from disk into memory. One option is to load the entire program in physical memory at program execution time. However, a problem with this approach is that we may not initially *need* the entire program in memory.
- Suppose a program starts with a list of available options from which the user is to select. Loading the entire program into memory results in loading the executable code for *all* options, regardless of whether an option is ultimately selected by the user or not. An alternative strategy is to load pages only as they are needed. This technique is known as **Demand paging** and is commonly used in virtual memory systems.
- With demand-paged virtual memory, pages are only loaded when they are demanded during program execution; pages that are never accessed are thus never loaded into physical memory.
- A demand-paging system is similar to a paging system with swapping (shown in figure below) where processes reside in secondary memory (usually a disk). When we want to execute a process, we swap it into memory.
- Since we are now viewing a process as a sequence of pages, rather than as one large contiguous address space. We thus use *pager*, rather than *swapper* (*which is used in case of process*), in connection with demand paging.



Basic Concepts

- When a process is to be swapped in, the pager guesses which pages will be used before the process is swapped out again. Instead of swapping in a whole process, the pager brings only those pages into memory. Thus, it avoids reading into memory pages that will not be used anyway, decreasing the swap time and the amount of physical memory needed.
- With this scheme, we need some form of hardware support to distinguish between the pages that are in memory and the pages that are on the disk. **The valid -invalid bit scheme** can be used for this purpose. When the bit is set to "valid/" the associated page is both legal and in memory. If the bit is set to "invalid/" the page either is not valid (that is, not in the logical address space of the process) or is valid but is currently on the disk.
- The page-table entry for a page that is brought into memory is set as usual but the page-table entry for a page that is not currently in memory is either simply marked invalid or contains the address of the page on disk.



But what happens if the process tries to access a page that was not brought into memory?

- Access to a page marked invalid causes a **Page Fault**. The paging hardware, in translating the address through the page table, will notice that the invalid bit is set, causing a trap to the operating system. This trap is the result of the operating system's failure to bring the desired page into memory.
- The procedure for handling this page fault is:
 - We check an internal table (usually kept with the process control block) for this process to determine whether the reference was a valid or an invalid memory access.
 - If the reference was invalid, we terminate the process. If it was valid, but we have not yet brought in that page, we now page it in.
 - We find a free frame (by taking one from the free-frame list, for example).
 - We schedule a disk operation to read the desired page into the newly allocated frame.
 - When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
 - We restart the instruction that was interrupted by the trap. The process can now access the page as though it had always been in memory.

Performance of Demand Paging

Demand paging can significantly affect the performance of a computer system. To see why, let's compute the effective access time for a demand-paged memory.

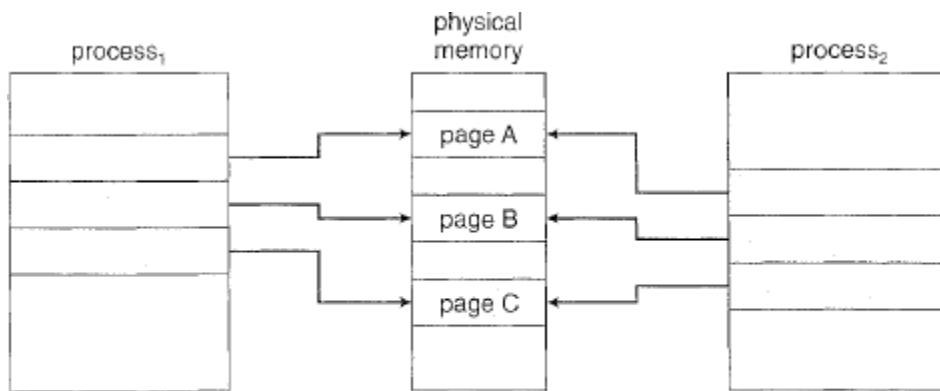
For most computer systems, the memory-access time, denoted ma , ranges from 10 to 200 nanoseconds. As long as we have no page faults, the effective access time is equal to the memory access time. If, a page fault occurs, we must first read the relevant page from disk and then access the desired word.

Let p be the probability of a page fault ($0 \leq p \leq 1$). We would expect p to be close to zero—that is, we would expect to have only a few page faults. The effective access time is then

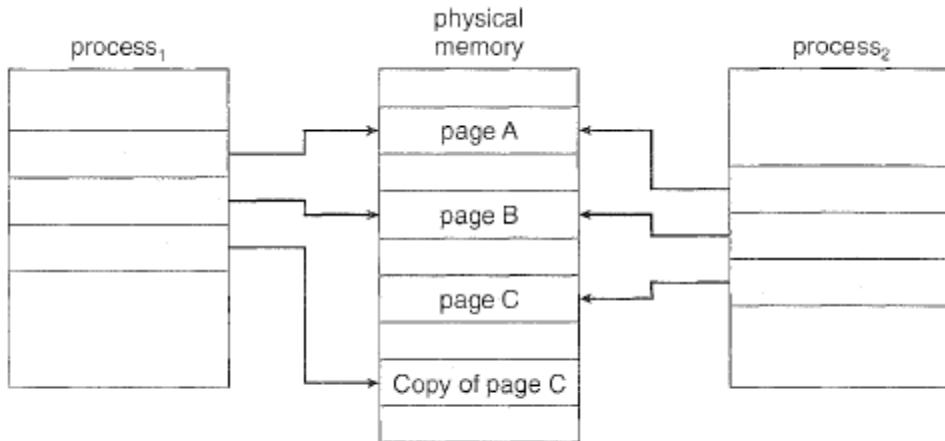
$$\text{effective access time} = (1 - p) \times ma + p \times \text{page fault time}$$

Copy-On-Write

- As a process can start quickly by merely demand paging in the page containing the first instruction. However, process creation using the `fork()` system call may initially bypass the need for demand paging by using a technique similar to page sharing.
- This technique provides for rapid process creation and minimizes the number of new pages that must be allocated to the newly created process.
- The `fork()` system call creates a child process that is a duplicate of its parent. Traditionally, `fork()` worked by creating a copy of the parent's address space for the child, duplicating the pages belonging to the parent.
- We can use a technique known as **Copy-On-Write** which works by allowing the parent and child processes initially to share the same pages.
- These shared pages are marked as copy-on-write pages, meaning that if either process writes to a shared page, a copy of the shared page is created.
- Copy-on-write is illustrated in Figures below, which show the contents of the physical memory before and after process 1 modifies page C.



Before process 1 modifies page C.



After process 1 modifies page C.

- For example, assume that the child process attempts to modify a page containing portions of the stack, with the pages set to be copy-on-write. The operating system will create a copy of this page, mapping it to the address space of the child process. The child process will then modify its copied page and not the page belonging to the parent process. Obviously, when the copy-on-write technique is used, only the pages that are modified by either process are copied; all unmodified pages can be shared by the parent and child processes.
- When it is determined that a page is going to be duplicated using copy-on- write, it is important to note the location from which the free page will be allocated. Many operating systems provide a **pool** of free pages for such requests.

PAGE REPLACEMENT ALGORITHM

Page replacement takes the following approach. If no frame is free, we find one that is not currently being used and free it. We can free a frame by writing its contents to swap space and changing the page table (and all other tables) to indicate that the page is no longer in memory. We can now use the freed frame to hold the page for which the process faulted. We modify the page-fault service routine to include page replacement:

1. Find the location of the desired page on the disk.
2. Find a free frame:
 - a. If there is a free frame, use it.
 - b. If there is no free frame, use a page-replacement algorithm to select a **Victim Frame**.
 - c. Write the victim frame to the disk; change the page and frame tables accordingly.
3. Read the desired page into the newly freed frame; change the page and frame tables.
4. Restart the user process.

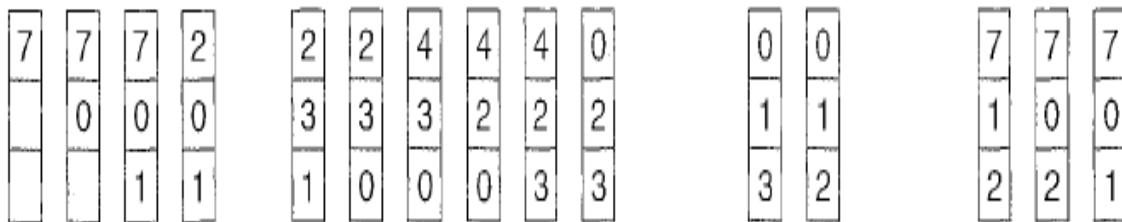
Notice that, if no frames are free, *two* page transfers (one out and one in) are required. This situation effectively doubles the page-fault service time and increases the effective access time accordingly.

FIFO PAGE REPLACEMENT ALGORITHM

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

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



page frames

- To illustrate the problems that are possible with a FIFO page-replacement algorithm, we consider the following reference string:
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- Given Figure shows the curve of page faults for this reference string versus the number of available frames. Notice that the number of faults for four frames (ten) is greater than the number of faults for three frames (nine)! This most unexpected result is known as **Belady's Anomaly** for some page-replacement algorithms, the page-fault rate may increase as the number of allocated frames increases.

Optimal Page Replacement

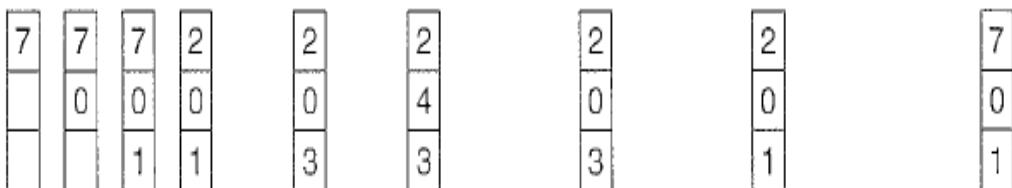
One of the discovery due to Belady's Anomaly is Optimal Page Replacement Algorithm- which has the lowest page-fault rate of all algorithms and will never suffer from Belady's anomaly. Such an algorithm does exist and has been called OPT or MIN. It is simply this:

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

Use of this page-replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames.

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



page frames

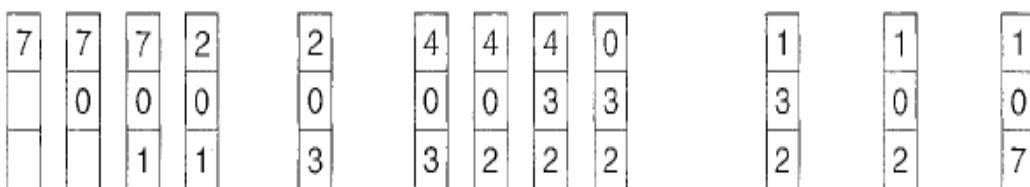
LRU Page Replacement Algorithm

In LRU Algorithm the main concern is to use the recent past as an approximation of the near future, then we can replace page that *has not been used* for the longest period of time.

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

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



page frames

Counting-Based Page Replacement

There are many other algorithms that can be used for page replacement. For example, we can keep a counter of the number of references that have been made to each page and develop the following two schemes.

- **The least frequently used (LFU)** page-replacement algorithm requires that the page with the smallest count be replaced. The reason for this selection is that an actively used page should have a large reference count. A problem arises, however, when a page is used heavily during the initial phase of a process but then is never used again. Since it was used heavily, it has a large count and remains in memory even though it is no longer needed. One solution is to shift the counts right by 1 bit at regular intervals, forming an exponentially decaying average usage count.
- **The most frequently used (MFU)** page-replacement algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used. As you might expect, neither MFU nor LFU replacement is common. The implementation of these algorithms is expensive, and they do not approximate OPT replacement well.

Page-Buffering Algorithms

- Other procedures are often used in addition to a specific page-replacement algorithm. For example, systems commonly keep a pool of free frames. When a page fault occurs, a victim frame is chosen as before. However, the desired page is read into a free frame from the pool before the victim is written out. This procedure allows the process to restart as soon as possible, without waiting for the victim page to be written out. When the victim is later written out, its frame is added to the free-frame pool.
- An expansion of this idea is to maintain a list of modified pages. Whenever the paging device is idle, a modified page is selected and is written to the disk. Its modify bit is then reset. This scheme increases the probability that a page will be clean when it is selected for replacement and will not need to be written out.
- Another modification is to keep a pool of free frames but to remember which page was in each frame. Since the frame contents are not modified when a frame is written to the disk, the old page can be reused directly from the free-frame pool if it is needed before that frame is reused. No I/O is needed in this case. When a page fault occurs, we first check whether the desired page is in the free-frame pool. If it is not, we must select a free frame and read into it.

Allocation of Frames

How do we allocate the fixed amount of free memory among the various processes? If we have 93 free frames and two processes, how many frames does each process get?

- The simplest case is the single-user system. Consider a single-user system with 128 KB of memory composed of pages 1 KB in size. This system has 128 frames. The operating system may take 35 KB, leaving 93 frames for the user process. Under pure demand paging, all 93 frames would initially be put on the free-frame list. When a user process started execution, it would generate a sequence of page faults. The first 93 page faults would all get free frames from the free-frame list. When the free-frame list was exhausted, a page-replacement algorithm would be used to select one of the 93 in-memory pages to be replaced with the 94th, and so on. When the process terminated, the 93 frames would once again be placed on the free-frame list.

Allocation Algorithms

- The easiest way to split m frames among n processes is to give everyone an equal share, m/n frames. For instance, if there are 93 frames and five processes, each process will get 18 frames. The three leftover frames can be used as a free-frame buffer pool. This scheme is called **Equal Allocation**.
- An alternative is to recognize that various processes will need differing amounts of memory. Consider a system with a 1-KB frame size. If a small student process of 10 KB and an interactive database of 127 KB are the only two processes running in a system with 62 free frames, it does not make much sense to give each process 31 frames. The student process does not need more than 10 frames, so the other 21 are, strictly speaking, wasted.
- To solve this problem, we can use **Proportional Allocation** in which we allocate available memory to each process according to its size. Let the size of the virtual memory for process p_i be s_i , and define

$$S = \sum s_i.$$

- Then, if the total number of available frames is m , we allocate a_i frames to process p_b where a_i is approximately

$$a_i = s_i / S \times m.$$

We must adjust each a_i to be an integer that is greater than the minimum number of frames required by the instruction set, with a sum not exceeding m .

With proportional allocation, we would split 62 frames between two processes, one of 10 pages and one of 127 pages, by allocating 4 frames and 57 frames, respectively, since

$$10/137 \times 62 \sim 4, \text{ and}$$

$$127/137 \times 62 \sim 57.$$

In this way, both processes share the available frames according to their "needs," rather than equally.

Global versus Local Allocation

- Another important factor in the way frames are allocated to the various processes is page replacement. With multiple processes competing for frames, we can classify page-replacement algorithms into two broad categories: **Global Replacement** and **Local Replacement**.
- Global replacement allows a process to a replacement frame from the set of all frames, even if that frame is currently allocated to some other process; that is, one process can take a frame from another.
- Local replacement requires that each process select from only its own set of allocated frames.
- For example, consider an allocation scheme wherein we allow high-priority processes to select frames from low-priority processes for replacement. A process can select a replacement from among its own frames or the frames of any lower-priority process. This approach allows a high-priority process to increase its frame allocation at the expense of a low-priority process.
- With a local replacement strategy, the number of frames allocated to a process does not change. With global replacement, a process may happen to select only frames allocated to other processes, thus increasing the number of frames allocated to it

Thrashing

- If the number of frames allocated to a low-priority process falls below the minimum number required by the computer architecture, we must suspend that process's execution. We should then page out its remaining pages, freeing all its allocated frames. This provision introduces a swap-in, swap-out level of intermediate CPU scheduling.
- In fact, look at any process that does not have "enough" frames. If the process does not have the number of frames it needs to support pages in active use, it will quickly page-fault. At this point, it must replace some page.

- However, since all its pages are in active use, it must replace a page that will be needed again right away. Consequently, it quickly faults again, and again, and again, replacing pages that it must back in immediately.
- This high paging activity is called **Thrashing**. A process is thrashing if it is spending more time paging than executing.

Causes of Thrashing

Consider the following scenario, which is based on the actual behavior of early paging systems.

The operating system monitors CPU utilization. If CPU utilization is too low, we increase the degree of multiprogramming by introducing a new process to the system.

A global page-replacement algorithm is used; it replaces pages without regard to the process to which they belong.

Now suppose that a process enters a new phase in its execution and needs more frames. It starts faulting and taking frames away from other processes. These processes need those pages, however, and so they also fault, taking frames from other processes.

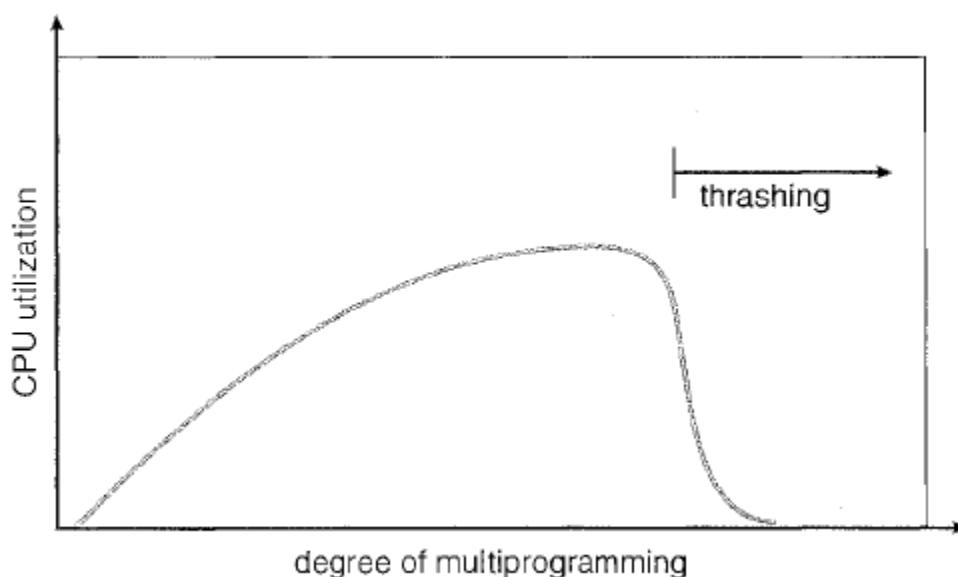
These faulting processes must use the paging device to swap pages in and out. As they queue up for the paging device, the ready queue empties. As processes wait for the paging device, CPU utilization decreases.

The CPU scheduler sees the decreasing CPU utilization and *increases* the degree of multiprogramming as a result.

The new process tries to get started by taking frames from running processes, causing more page faults and a longer queue for the paging device. As a result, CPU utilization drops even further, and the CPU scheduler tries to increase the degree of multiprogramming even more.

Thrashing has occurred, and system throughput plunges. The page fault rate increases tremendously. As a result, the effective memory-access time increases. No work is getting done, because the processes are spending all their time paging.

This phenomenon is illustrated in given Figure in which CPU utilization is plotted against the degree of multiprogramming.



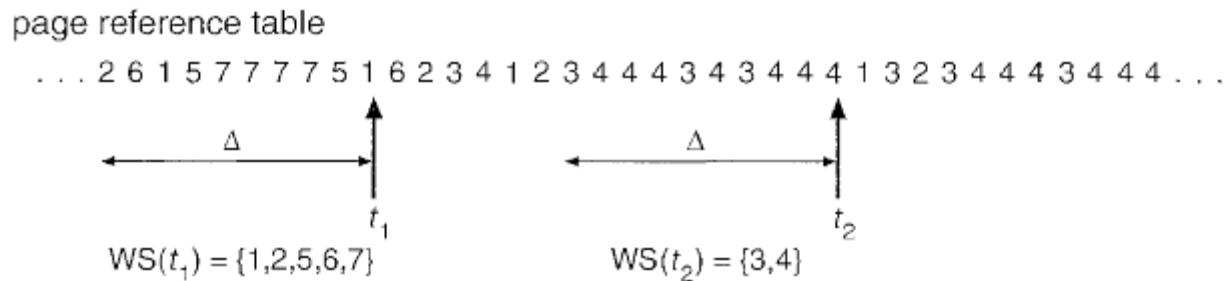
As the degree of multiprogramming increases, CPU utilization also increases, although more slowly, until a maximum is reached. If the degree of multiprogramming is increased even further, thrashing sets in, and CPU

utilization drops sharply. At this point, to increase CPU utilization and stop thrashing, we must *decrease* the degree of multiprogramming.

- To prevent thrashing, we must provide a process with as many frames as it needs. But how do we know how many frames it "needs"? There are several techniques. The **working-set strategy** starts by looking at how many frames a process is actually using. This approach defines the locality model of process execution.

Working-Set Model

The Working-Set Model is based on the assumption of locality. This model uses a parameter Δ , to define the working set window. The idea is to examine the most recent Δ pages references. The set of pages in the most recent Δ page references is the **working set**.



If a page is in active use, it will be in the working set. If it is no longer being used, it will drop from the working set Δ time units after its last reference. Thus, the working set is an approximation of the program's locality.

For example, given the sequence of memory references shown in Figure above, if $\Delta=10$ memory references, then the working set at time t_1 is $\{1, 2, 5, 6, 7\}$. By time t_2 , the working set has changed to $\{3, 4\}$.

The accuracy of the working set depends on the selection of Δ . If Δ is too small, it will not encompass the entire locality; if Δ is too large, it may overlap several localities.

The most important property of the working set, then, is its size. If we compute the working-set size, WSS_i for each process in the system, we can then consider that

$$D = \sum WSS_i,$$

where D is the total demand for frames.

Each process is actively using the pages in its working set. Thus, process i needs WSS_i frames. If the total demand is greater than the total number of available frames ($D > m$), thrashing will occur, because some processes will not have enough frames.

