1

. 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 :

* Blocked state
* Ready state
* Suspended state
* Terminated state

B. Ready state

2

. In a multi-programming environment :

* the processor executes more than one process at a time
* the programs are developed by more than one person
* more than one process resides in the memory
* a single user can execute many programs at the same time

C. more than one process resides in the memory

3

. Suppose that a process is in “Blocked” state waiting for some I/O service. When the service is completed, it goes to the :

* Running state
* Ready state
* Suspended state
* Terminated state

B. Ready state

4

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

* A device
* Timer
* Scheduler process
* Power failure

C. Scheduler process

5

. 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) \_\_\_\_.

* Shared Memory Segments
* Entry Section
* Race condition
* Process Synchronization

C. Race condition

6

. Which of the following state transitions is not possible ?

* blocked to running
* ready to running
* blocked to ready
* running to blocked

A. blocked to running

7

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

* cooperating process
* child process
* parent process
* init process

A. cooperating process

8

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

* mutual exclusion
* critical exclusion
* synchronous exclusion
* asynchronous exclusion

A. mutual exclusion

9

. Which one of the following is a synchronization tool?

* thread
* pipe
* semaphore
* socket

C. semaphore

10

. A semaphore is a shared integer variable

* that can not drop below zero
* that can not be more than zero
* that can not drop below one
* that can not be more than one

A. that can not drop below zero

1

. What is interprocess communication?

* communication within the process
* communication between two process
* communication between two threads of same process
* none of the mentioned

B. communication between two process

2

. A set of processes is deadlock if

* each process is blocked and will remain so forever
* each process is terminated
* all processes are trying to kill each other
* none of the mentioned

A. each process is blocked and will remain so forever

3

. A process stack does not contain

* function parameters
* local variables
* return addresses
* PID of child process

D. PID of child process

4

. Which system call returns the process identifier of a terminated child?

* wait
* exit
* fork
* get

A. wait

5

. The address of the next instruction to be executed by the current process is provided by the

* CPU registers
* program counter
* process stack
* pipe

6

. When the process issues an I/O request :

* It is placed in an I/O queue
* It is placed in a waiting queue
* It is placed in the ready queue
* It is placed in the Job queue

A. It is placed in an I/O queue

7

. What is a long-term scheduler ?

* It selects which process has to be brought into the ready queue
* It selects which process has to be executed next and allocates CPU
* It selects which process to remove from memory by swapping
* None of these

A. It selects which process has to be brought into the ready queue

8

. What is a medium-term scheduler ?

* It selects which process has to be brought into the ready queue
* It selects which process has to be executed next and allocates CPU
* It selects which process to remove from memory by swapping
* None of these

C. It selects which process to remove from memory by swapping

9

. What is a short-term scheduler ?

* It selects which process has to be brought into the ready queue
* It selects which process has to be executed next and allocates CPU
* It selects which process to remove from memory by swapping
* None of these

B. It selects which process has to be executed next and allocates CPU

10

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

* The length of their queues
* The type of processes they schedule
* The frequency of their execution
* None of these

C. The frequency of their execution

1

. What is operating system?

* collection of programs that manages hardware resources
* system service provider to the application programs
* link to interface the hardware and application programs
* all of the mentioned

D. all of the mentioned

2

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

* system calls
* API
* library
* assembly instructions

D. assembly instructions

3

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

* power failure
* lack of paper in printer
* connection failure in the network
* all of the mentioned

D. all of the mentioned

4

. The main function of the command interpreter is

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

A. to get and execute the next user-specified command

5

. By operating system, the resource management can be done via

* time division multiplexing
* space division multiplexing
* both (a) and (b)
* none of the mentioned

C. both (a) and (b)

6

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

* log file
* another running process
* new file
* none of the mentioned

A. log file

7

. The systems which allows only one process execution at a time, are called

* uniprogramming systems
* uniprocessing systems
* unitasking systems
* none of the mentioned

A. uniprogramming systems

8

. In operating system, each process has its own

* address space and global variables
* open files
* pending alarms, signals and signal handlers
* all of the mentioned

D. all of the mentioned

9

. A process can be terminated due to

* normal exit
* fatal error
* killed by another process
* all of the mentioned

D. all of the mentioned

10

. What is the ready state of a process?

* when process is scheduled to run after some execution
* when process is unable to run until some task has been completed
* when process is using the CPU
* none of the mentioned

A. when process is scheduled to run after some execution

1

. Process synchronization can be done on

* hardware level
* software level
* both (a) and (b)
* none of the mentioned

C. both (a) and (b)

2

. A monitor is a module that encapsulates

* shared data structures
* procedures that operate on shared data structure
* synchronization between concurrent procedure invocation
* all of the mentioned

D. all of the mentioned

3

. What is the reusable resource?

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

A. that can be used by one process at a time and is not depleted by that use

4

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

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

D. all of the mentioned

5

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

* banker’s algorithm
* round-robin algorithm
* elevator algorithm
* karn’s algorithm

A. banker’s algorithm

6

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

* every time a resource request is made
* at fixed time intervals
* both (a) and (b)
* none of the mentioned

C. both (a) and (b)

7

. To avoid deadlock

* there must be a fixed number of resources to allocate
* resource allocation must be done only once
* all deadlocked processes must be aborted
* inversion technique can be used

A. there must be a fixed number of resources to allocate

8

. An un-interruptible unit is known as :

* single
* atomic
* static
* None of these

B. atomic

9

. Semaphore is a/an \_\_\_\_\_\_\_ to solve the critical section problem.

* hardware for a system
* special program for a system
* integer variable
* None of these

C. integer variable

10

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

* program counter
* status register
* instruction register
* program status word

A. program counter

1

. Operating System maintains the page table for

* each process
* each thread
* each instruction
* each address

A. each process

2

. Because of virtual memory, the memory can be shared among

* processes
* threads
* instructions
* none of the mentioned

A. processes

3

. \_\_\_\_\_ is the concept in which a process is copied into main memory from the secondary memory according to the requirement.

* Paging
* Demand paging
* Segmentation
* Swapping

B. Demand paging

4

. The pager concerns with the

* individual page of a process
* entire process
* entire thread
* first page of a process

A. individual page of a process

5

. Swap space exists in

* primary memory
* secondary memory
* CPU
* none of the mentioned

B. secondary memory

6

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

* stack pointer
* cache
* accumulator
* disk buffer

B. cache

7

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

* physical address
* absolute address
* logical address
* none of the mentioned

C. logical address

8

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

* memory management unit
* CPU
* PCI
* none of the mentioned

A. memory management unit

9

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

* fragmentation
* paging
* mapping
* none of the mentioned

B. paging

10

. Program always deals with

* logical address
* absolute address
* physical address
* relative address

A. logical address

1

. In FIFO page replacement algorithm, when a page must be replaced

* **oldest page is chosen**
* newest page is chosen
* random page is chosen
* none of the mentioned

2

. \_\_\_\_\_\_ is a unique tag, usually a number, identifies the file within the file system.

* File identifier
* File name
* File type
* none of the mentioned

3

. To create a file

* allocate the space in file system
* make an entry for new file in directory
* both (a) and (b)
* none of the mentioned

4

. By using the specific system call, we can

* open the file
* read the file
* write into the file
* **all of the mentioned**

5

. File type can be represented by

* file name
* file extension
* file identifier
* none of the mentioned

6

. Mapping of file is managed by

* file metadata
* page table
* virtual memory
* file system

7

. file system fragmentation occurs when

* unused space or single file are not contiguous
* used space is not contiguous
* unused space is non-contiguous
* multiple files are non-contiguous

8

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

* CPU
* Monitor
* wirefull
* bus

9

. A \_\_\_\_\_\_\_\_ is a collection of electronics that can operate a port, a bus, or a device.

* controller
* driver
* host
* bus

10

. An I/O port typically consists of four registers status, control, \_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_ registers.

* system in, system out
* data in, data out
* flow in, flow out
* input, output

. The \_\_\_\_\_\_ register is read by the host to get input.

* flow in
* flow out
* data in
* data out

C. data in

2

. The \_\_\_\_\_\_ register is written by the host to send output.

* status
* control
* data in
* data out

D. data out

3

. The hardware mechanism that allows a device to notify the CPU is called \_\_\_\_\_\_\_.

* polling
* interrupt
* driver
* controlling

B. interrupt

1

. What does Belady’s Anomaly related to?

* Page Replacement Algorithm
* Memory Management Algorithm
* Deadlock Prevention Algorithm
* Disk Scheduling Algorithm

A. Page Replacement Algorithm

2

. What are the two types of Semaphore?

* Digital Semaphores and Binary Semaphores
* Analog Semaphores and Octal Semaphores
* Counting Semaphores and Binary Semaphores
* Critical Semaphores and System Semaphores

C. Counting Semaphores and Binary Semaphores

3

. What are the requirements for the solution to critical section problem?

* Mutual Exclusion
* Progress
* Bounded Waiting
* All of Above

D. All of Above

4

. Which of the following memory unit that processor can access more rapidly

* Main Memory
* Virtual Memory
* Cache memory
* Read Only Memory

C. Cache memory

5

. Which of the following is the system tool?

* Backup
* Disk defragmenter
* Both of above
* File

C. Both of above

6

. When a computer is first turned on or restarted, a special type of absolute loader called \_\_\_\_ is executed

* Compile and Go loader
* Boot loader
* Bootstrap loader
* Relating loader

C.  Bootstrap loader

7

. In which type of the following OS, the response time is very crucial.

* Network Operating System
* Real Time Operating System
* Batch Operating System
* Unix Operating System

B. Real Time Operating System

8

. Virtual Memory is

* Extremely Large Main memory
* Extremely Large Secondary memory
* An illusion of extremely large main memory
* An illusion of extremely large secondary memory

C. An illusion of extremely large main memory

9

. Swapping

* Works best with many small partitions
* Allows many programs to use memory simultaneously
* Allows each program in turn to use the memory
* Does not work with overlaying

C. Allows each program in turn to use the memory

10

. A program in execution is called

* A Paging
* A Process
* A virtual memory
* **A Demand Page**

B. A Process

1

. What is thrashing?

* A high paging activity is called thrashing.
* A high executing activity is called thrashing
* A extremely long process is called thrashing
* A extremely long virtual memory is called thrashing

A. A high paging activity is called thrashing.

2

. Symptoms of swap-file problems include extremely slow system speed and a disk that is constantly being accessed, which is referred to as \_\_\_\_\_\_\_\_\_\_\_\_\_\_.

* Clocking
* Thrashing
* Booting
* Filtering

B. Thrashing

3

. What hole will allocates in “Worst-Fit” algorithm of memory management?

* It allocates the smaller hole than required memory hole
* It allocates the smallest hole from the available memory holes
* It allocates the largest hole from the available memory holes
* It allocates the exact same size memory hole

C. It allocates the largest hole from the available memory holes

4

. Which of the following is the allocation method of a disk space?

* Contiguous allocation
* Linked allocation
* Indexed allocation
* All of the Above

D. All of the Above

5

. Which of the following concept is best to preventing page faults?

* Paging
* The working set
* Hit ratios
* Address location resolution

B. The working set

6

. Poor response time are usually caused by

* Process busy
* High I/O rates
* High paging rate
* Any of the above

D. Any of the above

7

. Virtual memory typically located on

* RAM
* CPU
* Flash card
* Hard drive

D. Hard drive

8

. What is contained in the page table?

* Base address of each frame and corresponding page number
* Memory address and corresponding page number
* File name and corresponding page number
* None of Above

A. Base address of each frame and corresponding page number

9

. Tree structure displays the

* File only
* Directory only
* File and directory name
* None of above

C. File and directory name

10

. Multiprogramming systems:

* Are easier to develop than single programming systems
* Execute each job faster
* Execute more jobs in the same time period
* Are used only one large mainframe computers.

C. Execute more jobs in the same time period

. Bringing a page into memory only when it is needed,this mechanism is called

* Deadlock
* Page Fault
* Dormant Paging
* Demand Paging

D. Demand Paging

2

. What do you mean by Memory Compaction?

* Combine multiple equal memory holes into one big hole
* Combine multiple small memory holes into one big hole
* Divide big memory hole into small holes
* Divide memory hole by 2

B. Combine multiple small memory holes into one big hole

3

. Copying a process from memory to disk to allow space for other processes is Called

* Swapping
* Deadlock
* Demand Paging
* Page Fault

A. Swapping

4

. Page stealing

* Is a sign of an efficient system
* Is taking page frame from other working sets
* Should be the turning goal
* Is taking layer disk space for page in page out

B. Is taking page frame from other working sets

5

. A spooler is a

* Location in memory that maintains the contents of documents until it prints out
* Queue of print job that are waiting to print
* Program that coordinates the print job that are waiting to process
* Message sent from the printer to the operating system when a print job is completed

C. Program that coordinates the print job that are waiting to process

6

. What is the method of handling deadlocks?

* Use a protocol to ensure that the system will never enter a deadlock state.
* Allow the system to enter the deadlock state and then recover.
* Pretend that deadlocks never occur in the system
* All of the Above

D. All of the Above

7

. The Banker’s algorithm is used

* to rectify deadlock
* to detect deadlock
* to prevent deadlock
* to slove deadlock

C. to prevent deadlock

8

. All of the following are TRUE regarding virtual memory EXCEPT

* Any amount of RAM can be allocated to virtual memory
* The setting for the amount of hard disk drive space to allocate virtual memory can be manually change
* This temporary storage is called the swap file or page file
* Virtual memory is the physical space of the hard drive

B. The setting for the amount of hard disk drive space to allocate virtual memory can be manually change

9

. What is dispatch latency?

* The time taken by the dispatcher to stop one process and start another
* The time taken by the processor to write a file into disk
* The whole time taken by all processor
* None of Above

A. The time taken by the dispatcher to stop one process and start another

10

. A page fault occurs when

* the Deadlock happens
* the Segmentation starts
* the page is found in the memory
* the page is not found in the memory

D. the page is not found in the memory

1

. When a program is loaded into the memory and it becomes a process, it can be divided into which of the  sections?

1. stack
2. heap
3. text
4. data

* 1
* 1,2
* 1,2,3
* 1,2,3,4

D. 1,2,3,4

Explanation :

1. stack
2. heap
3. text
4. data

2

. Which of the following includes the current activity represented by the value of Program Counter and the contents of the processor's registers.

* stack
* Text
* Data
* Heap

B. Text

3

. A process is more than the program code, which is sometimes known as the \_\_\_\_\_\_\_\_\_\_

* text section
* contents of the processor’s registers
* stack
* data section

A. text section

4

. \_\_\_\_ section is dynamically allocated memory to a process during its run time

* stack
* Text
* Data
* Heap

D. Heap

5

. A process's Current Activity is represented by

1. program counter
2. contents of the processor’s registers
3. stack
4. data section

* 1
* 1,2
* 1,2,3
* 1,2,3,4

B. 1,2

Explanation :

1. program counter
2. contents of the processor’s registers

6

. A process generally also includes the process \_\_\_\_\_\_, which contains temporary data (such as function parameters, return addresses, and local variables)

* text section
* program counter
* stack
* data section

C. stack

7

. A process generally also includes the process \_\_\_\_\_, which contains global variables.

* heap
* program counter
* stack
* data section

D. data section

8

. A process may also include a \_\_\_\_\_, which is memory that is dynamically allocated during process run time

* heap
* program counter
* stack
* data section

A. heap

9

. As a process executes, it changes state. The state of a process is defined in part by the current activity of that process. A process may be in one of the following states

1. New
2. Running
3. Waiting
4. Ready
5. Terminated

* 1,2
* 1.2.3
* 1,2,3,4,5
* 1,2,3,4

C. 1,2,3,4,5

Explanation :

1. New
2. Running
3. Waiting
4. Ready
5. Terminated

10

. Which state of a process defined "The process is being created"

* New
* Running
* Ready
* Waiting

A. New

11

. Which state of a process defined "Instructions are being executed"

* New
* Running
* Waiting
* Ready

B. Running

12

. .Which state of a process defined "The process is waiting for some event to occur (such as an I/O completion or reception of a signal)"

* Ready
* Running
* Waiting
* Terminated

C. Waiting

13

. Which state of a process defined "The process is waiting to be assigned to a processor."

* New
* Running
* Waiting
* Ready

D. Ready

14

. Which state of a process defined "The process has finished execution"

* Running
* Running
* Ready
* Terminated

D. Terminated

1

. Each process is represented in the operating system by a\_\_\_\_

* process control block
* printed circuit board
* program control block
* problem control block

A. process control block

2

. Process control block (PCB) contains which of the following

* list of open files
* list of open files
* process state
* process number
* All of the above

E. All of the above

3

. the address of the next instruction to be executed for the current process is stored in

* CPU registers
* program counter
* process state
* process number

B. program counter

1

. The objective of multi programming is

* I. to have some process running at all times.
* II. Single process should run at a time.
* III. to maximize CPU utilization
* IV. to minimize CPU utilization
* I
* II,III
* I,III
* I,IV

C. I,III

Explanation :

1. to have some process running at all times.
2. to maximize CPU utilization

2

. For a single-processor system

* I. there will never be more than one running process
* II. if more processes, the rest will have to wait until the CPU is free and can be rescheduled.
* III. it maximize the CPU utilization
* IV. there will be more than one running process
* I, II
* II,III
* I,III
* I,IV

A. I, II

Explanation :

1. there will never be more than one running process
2. if more processes, the rest will have to wait until the CPU is free and can be rescheduled.

3

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

* device queue
* ready queue
* job queue
* None of the above

B. ready queue

4

. The ready queue is generally stored as a\_\_\_\_\_\_

* Array
* Stack
* Linked List
* None of the above

C. Linked List

5

. Which of the following is True regarding A ready-queue header

* IT contains pointers to the first and final PCBs in the list
* IT contains pointers field that points to the next PCB in the ready queue
* **Only A is True**
* **Both A and B**

C. Only A is True

Explanation : A ready-queue header contains pointers to the first and final PCBs in the list. Each PCB includes a pointer field that points to the next PCB in the ready queue.

6

. The list of processes waiting for a particular I/O device is called a\_\_\_\_\_\_\_\_

* device queue
* ready queue.
* job queue
* None of the above

A. device queue

7

. Once the process is allocated the CPU and is executing, which of several events could occur:

* i. The process could issue an I/O request and then be placed in an I/O queue
* ii. The process could create a new child process and wait for the child’s termination.
* iii. The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.
* Only 1
* 1 and 2
* 1, 2,3
* 2 and 3

C. 1, 2,3

Explanation : i. The process could issue an I/O request and then be placed in an I/O queue ii. The process could create a new child process and wait for the child’s termination. iii. The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.

8

. At which time it is removed from all queues and has its PCB and resources deallocated.

* The process could issue an I/O request and then be placed in an I/O queue
* The process could create a new child process and wait for the child’s termination.
* The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.
* None of the Above

D. None of the Above

9

. For short-term scheduler Which of the following stands True

* i. selects from among the processes that are ready to execute and allocates the CPU to one of them.
* ii. selects processes from mass-storage device (typically a disk) and loads them into memory for execution.
* iii. The short-term scheduler must select a new process for the CPU frequently.
* iv. The short-term scheduler executes much less frequently.
* 1 only
* 2 and 3 only
* 1 ,2 and 4 only
* 1,3 only

D. 1,3 only

Explanation :

1. selects from among the processes that are ready to execute and allocates the CPU to one of them
2. The short-term scheduler must select a new process for the CPU frequently.

10

. For Long-term scheduler which of the following stands true

* i. The long-term scheduler executes much less frequently.
* ii. 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
* iii. Because of the smaller interval between executions, the long-term scheduler can afford to take Less time to decide which process should be selected for execution
* iv. The long-term scheduler executes More frequently.
* 1,2 only
* 1 only
* 1 and 3 only
* 1 ,2 and 3 only

A. 1,2 only

Explanation :

1. The long-term scheduler executes much less frequently.
2. 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

11

. Which of the following is TRUE for An I/O-bound process.

* i. is one that spends more of its time doing I/O
* ii. is one that spends more of its time doing computations.
* iii. If all processes are I/O bound, the ready queue will almost always be empty
* iv. If all processes are I/O bound, the ready queue will almost always be Full
* 1 only
* 1 and 3 only
* 2 and 4 only
* 2 and 3 Only

B. 1 and 3 only

Explanation :

1. is one that spends more of its time doing I/O
2. If all processes are I/O bound, the ready queue will almost always be empty

12

. Which of the following is TRUE for CPU bound process?

* i. is one that spends more of its time doing I/O
* ii. is one that spends more of its time doing omputations.
* iii. If all processes are CPU bound, the I/O waiting queue will almost always be empty
* iv. If all processes are CPU bound, the I/O waiting queue will almost always be Full
* 1 only
* 1 and 3 only
* 2 and 3 only
* 2 and 4

C. 2 and 3 only

Explanation : is one that spends more of its time doing omputations. If all processes are CPU bound, the I/O waiting queue will almost always be empty

13

. Copying a process from memory to disk to allow space for other processes is called

* Swapping
* Deadlock
* Demand Paging
* Page Fault

A. Swapping

14

. The process is swapped out of memory, and is later swapped into memory, by the\_\_\_\_\_\_\_\_

* Long-term Scheduler
* Short-term Scheduler
* medium-term scheduler
* None of the Above

C. medium-term scheduler

15

. Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as a \_\_\_\_\_\_\_\_\_\_\_\_.

* Swapping
* context switch
* Demand Paging
* Page Fault

B. context switch

. Most operating systems (including UNIX, Linux, and Windows) identify processes according to a unique \_\_\_\_\_\_\_\_\_\_

* program counter
* process state
* process number
* Process Identifier

D. Process Identifier

2

. The init process which always has a pid of

* **0**
* 1
* 2
* 3

B. 1

3

. When a process creates a new process, two possibilities for execution exist:

1. The parent continues to execute concurrently with its children.
2. The parent Stop to execute concurrently with its children.
3. The parent waits until some or all of its children have terminated.
4. The parent do not wait until some or all of its children have terminated.

* 1 and 2
* 2 and 3
* 2 and 4
* 1 and 3

D. 1 and 3

4

. When a process creates a new process,There are also two address-space possibilities for the new process:

1. The child process is a duplicate of the parent process (it has the same program and data as the parent).
2. The child process is a Not duplicate of the parent process (it has not have the same program and data as the parent).
3. The child process has a new program loaded into it.
4. The child process has a same program loaded into it.

* 2 and 4
* 2 and 3
* 1 and 3
* 1 and 4

C. 1 and 3

5

. The new process consists of a copy of the \_\_\_\_ of the original process. This mechanism allows the parent process to communicate easily with its child process

* address space
* process state
* process number
* Process Identifier

A. address space

6

. 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 \_\_\_\_\_\_ for the new (child) process, whereas the \_\_\_\_\_ process identifier of the child is returned to the parent

* Negative integer, Zero
* Zero, Negative integer
* nonzero integer, Zero
* Zero, nonzero integer

D. Zero, nonzero integer

7

. Creating a separate process using the UNIX \_\_\_\_\_\_\_system call.

* fork
* init
* exec
* **wait**

A. fork

8

. After a fork() system call, one of the two processes typically uses the \_\_\_\_\_ system call to replace the process’s memory space with a new program.

* exit
* init
* exec
* wait

C. exec

9

. The parent can create more children; or, if it has nothing else to do while the child runs, it can issue a \_\_\_\_system call to move itself off the ready queue until the termination of the child

* exit
* exit
* exec
* wait

D. wait

10

. Processes are created in the Windows API using the \_\_\_\_\_\_ function, which is similar to fork() in that a parent creates a new child process.

* CreateProcess()
* InitiateProcess()
* StartProcess()
* None of the above

A. CreateProcess()

11

. .A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the \_\_\_ system call.

* wait()
* exit()
* forck()
* exec()

B. exit()

12

. A parent may terminate the execution of one of its children for a variety of reasons, Which of the following is true:

1. The child has exceeded its usage of some of the resources that it has been allocated.
2. The task assigned to the child is no longer required.
3. The parent is exiting, and the operating system does not allow a child to continue if its parent terminates.

* 1 only
* 1 and 2 only
* 2 and 3 only
* all of the above

D. all of the above

13

. A process that has terminated, but whose parent has not yet called wait(), is known as a \_\_\_ process.

* zombie
* orphans
* terminated
* None of the above

A. zombie

14

. if a parent did not invoke wait() and instead terminated, thereby leaving its child processes as it is ,the child process is known as \_\_\_

* zombie
* orphans
* terminated
* None of the above

B. orphans

15

. When a process creates a new process using the fork() operation, which of the following states is shared between the parent process and the child process?

* Stack
* Heap
* Shared memory segments
* None of the above

C. Shared memory segments

. Which are two fundamental models of interprocess communication:

1. shared memory
2. message passing.
3. independent
4. cooperating

* 1,2
* 2,3
* 3,4
* **1,4**

A. 1,2

2

. Which of the following is True for Shared Memory and message passing interprocess communicatoin.  
I. In the shared-memory model, a region of memory that is shared by cooperating processes is established.  
II. In the message-passing model, communication takes place by means of messages exchanged between the cooperating processes  
III. Message passing is useful for exchanging Biger amounts of data, because no conflicts need to be avoided.  
IV. Message passing is also Difficult to implement in a distributed system than shared memory.  
V. Shared memory can be faster than message passing, since message-passing systems are typically implemented using system calls

* 1 2 3 only
* 1 2 5 only
* 2 3 4 only
* 1 3 4 only

B. 1 2 5 only

3

. In producer–consumer problem using shared memory ,Select appropriate statement from the below.

* we must have available a buffer of items that can be filled by the producer and emptied by the consumer
* A producer can consume one item while the consumer is producing another item
* No synchronization required between The producer and consumer , so that the consumer does not try to consume an item that has not yet been produced.
* the buffer will reside in a region of memory that need not be to shared by the producer and consumer processes.

A. we must have available a buffer of items that can be filled by the producer and emptied by the consumer

Explanation : One solution to the producer–consumer problem uses shared memory. To allow producer and consumer processes to run concurrently, we must have available a buffer of items that can be filled by the producer and emptied by the consumer. This buffer will reside in a region of memory that is shared by the producer and consumer processes. A producer can produce one item while the consumer is consuming another item. The producer and consumer must be synchronized, so that the consumer does not try to consume an item that has not yet been produced

4

. Using producer–consumer problem using Shared Memory Interprocess Communication using Unbounded Buffer:

* There is Limit on the size of the buffer.
* Assumes a fixed buffer size.
* the consumer must wait if the buffer is empty, and the producer must wait if the buffer is full
* There is no practical limit on the size of the buffer. The consumer may have to wait for new items, but the producer can always produce new items

D. There is no practical limit on the size of the buffer. The consumer may have to wait for new items, but the producer can always produce new items

Explanation : The unbounded buffer places no practical limit on the size of the buffer. The consumer may have to wait for new items, but the producer can always produce new items.

5

. Using producer–consumer problem using Shared Memory Interprocess Communication using Bounded Buffer:

* There is Limit on the size of the buffer.
* There is a fixed buffer size
* The consumer must wait if the buffer is empty, and the producer must wait if the buffer is full
* There is no practical limit on the size of the buffer. The consumer may have to wait for new items, but the producer can always produce new items.
* B and C

E. B and C

Explanation : The bounded buffer assumes a fixed buffer size. In this case, the consumer must wait if the buffer is empty, and the producer must wait if the buffer is full.

6

. Message passing provides a mechanism to allow processes to communicate and to synchronize their actions

* By sharing the same address space
* without sharing the same address space
* by sharing the same process number and Process Identifier
* None of the above

B. without sharing the same address space

7

. In Message-Passing Systems of Interprocess Communication

* Messages sent by a process can be either fixed or variable in size.
* Messages sent by a process can be fixed in size
* Messages sent by a process can be variable in size
* None of the above

A. Messages sent by a process can be either fixed or variable in size.

8

. In Message-Passing Systems ,A message-passing facility provides at least two operations:

* send(message) and delete(message)
* delete(message) and receive (message)
* send(message) and receive(message)
* write(message) and delete(message)

C. send(message) and receive(message)

9

. Under direct communication, each process P and Q that wants to communicate must explicitly name the recipient or sender of the communication. In this scheme, which of the following primitives are required:

1. send(P, message)-Send a message to process P.
2. receive(Q, message)-Receive a message from process Q..
3. receive(id, message)-Receive a message from any process.
4. send(id, message)-Send a message to any process

* 1 and 3
* 2 and 4
* 3 and 4
* 1 and 2

D. 1 and 2

10

. Under direct communication, when asymmetry in addressing scheme is employed. the send() and receive() primitives are defined as follows:

1. send(P, message)-Send a message to process P.
2. receive(Q, message)-Receive a message from process Q..
3. receive(id, message) - Receive a message from any process.
4. send(id, message) - Send a message to any process

* 1 and 3
* 2 and 4
* 3 and 4
* 1 and 2

A. 1 and 3

. Thread shares with other threads belonging to the same process its

* thread ID
* program counter
* register set and a stack
* code section and data section

D. code section and data section

2

. A process can be

* single threaded
* multithreaded
* both single threaded and multithreaded
* none of the above

C. both single threaded and multithreaded

3

. User threads

* are supported above the kernel and are managed without kernel support
* are supported below the kernel and are managed without kernel support
* are supported above the kernel and are managed with kernel support
* are supported below the kernel and are managed with kernel support

A. are supported above the kernel and are managed without kernel support

4

. kernel threads

* Cannot be supported and managed directly by the operating system
* can be supported and managed directly by the operating system
* are supported below the kernel and are managed without kernel support
* None of the above

A. Cannot be supported and managed directly by the operating system

5

. Which of the following Multithreading model maps many user-level threads to one kernel thread.

* Many to One model
* One to Many model
* Many to Many model
* One to One model

A. Many to One model

6

. which of the following Multithreading model ,The entire process will block if a thread makes a blocking system call.

* Many to One model
* One to Many model
* Many to Many model
* One to One model

A. Many to One model

7

. Which of the following Multithreading model,only one thread can access the kernel at a time, multiple threads are unable to run in parallel on multicore systems

* Many to One model
* One to Many model
* Many to Many model
* One to One model

A. Many to One model

8

. Which of the following Multithreading model maps each user thread to a kernel thread?

* Many to One model
* One to Many model
* Many to Many model
* One to One model

D. One to One model

9

. Which of the following Multithreading model provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call.

* Many to One model
* One to Many model
* Many to Many model
* One to One model

D. One to One model

10

. Which of the following Multithreading model also allows multiple threads to run in parallel on multiprocessors

* Many to One model
* One to Many model
* Many to Many model
* One to One model

D. One to One model

11

. Which of the following Multithreading model has drawback "that creating a user thread requires creating the corresponding kernel thread".

* Many to One model
* One to Many model
* Many to Many model
* One to One model

D. One to One model

12

. Which of the following Multithreading model multiplexes many user-level threads to a smaller or equal number of kernel threads?

* Many to One model
* One to Many model
* Many to Many model
* One to One model

C. Many to Many model

13

. Which of the following components of program state are shared across threads in a multithreaded process?

* Register values
* Heap memory
* Global variables
* Stack memory
* B and C

E. B and C

Explanation : The threads of a multithreaded process share heap memory and global variables. Each thread has its separate set of register values and a separate stack

. A Race condition refers to

* A situation where several processes access and manipulate the same data concurrently
* A situation where Single process access and manipulate the same data concurrently .
* A situation where No process access and manipulate the same data concurrently.
* None of the above

A. A situation where several processes access and manipulate the same data concurrently

2

. Each process has a segment of code, called a \_\_\_\_\_\_\_\_\_\_\_\_,in which the process may be changing common variables, updating a table, writing a file, and so on

* None-critical section
* Critical section
* Exit Section
* Entry Section

B. Critical section

3

. In Critical section

* several processes access and manipulate the same data concurrently
* No process access and manipulate the same data concurrently
* when one process is executing in its critical section, no other process is allowed to execute in its critical section
* None of the above

C. when one process is executing in its critical section, no other process is allowed to execute in its critical section

4

. A solution to the critical-section problem must satisfy which of the following three requirements:  
I. Mutual exclusion  
II. Progress  
III. Un-Bounded waiting  
IV. Bounded waiting

* 1,2 ,3
* 2,3,4
* 1,3,4
* 1,2,4

D. 1,2,4

5

. Which of the following condition stands true for Mutual Exclusion?

* No process access and manipulate the same data concurrently
* If process Pi is executing in its critical section, then no other processes can be executing in their critical sections
* several processes access and manipulate the same data concurrently
* None of above

B. If process Pi is executing in its critical section, then no other processes can be executing in their critical sections

6

. Which of the following condition stands true for Progress?

* When a thread is executing in its critical section, no other threads can be executing in their critical sections.
* If no thread is executing in its critical section, and if there are some threads that wish to enter their critical Sections, then one of these threads will get into the critical section.
* several processes access and manipulate the same data in the critical section
* All of the above

B. If no thread is executing in its critical section, and if there are some threads that wish to enter their critical Sections, then one of these threads will get into the critical section.

7

. Which of the following condition stands true for Bounded Waiting?

* When a thread is executing in its critical section, no other threads can be executing in their critical sections
* If no thread is executing in its critical section, and if there are some threads that wish to enter their critical sections, then one of these threads will get into the critical section.
* several processes access and manipulate the same data concurrently
* After a thread makes a request to enter its critical section, there is a bound on the number of times that other threads are allowed to enter their critical sections, before the request is granted

D. After a thread makes a request to enter its critical section, there is a bound on the number of times that other threads are allowed to enter their critical sections, before the request is granted

8

. A preemptive kernel allows

* a process to be preempted while it is not running in kernel mode
* a process to be preempted while it is running in kernel mode
* a process to be preempted while it is running in User mode
* does not allow a process running in kernel mode to be preempted

B. a process to be preempted while it is running in kernel mode

9

. nonpreemptive kernels

* a process to be preempted while it is not running in kernel mode
* a process to be preempted while it is running in kernel mode
* a process to be preempted while it is running in User mode
* does not allow a process running in kernel mode to be preempted

D. does not allow a process running in kernel mode to be preempted

Explanation : a kernel-mode process will run until it exits kernel mode, blocks,or voluntarily yields control of the CPU.

10

. a nonpreemptive kernel is essentially free from race conditions

* True
* False

A. True

Explanation : a nonpreemptive kernel is essentially free from race conditions on kernel data structures, as only one process is active in the kernel at a time

11

. a preemptive kernel is essentially free from race conditions

* True
* False

B. False

Explanation : they must be carefully designed to ensure that shared kernel data are free from race conditions. Preemptive kernels are especially difficult to design for SMP architectures, since in these environments it is possible for two kernel-mode processes to run simultaneously on different processors.

. Peterson’s solution is restricted to \_\_\_ processes that alternate execution between their critical sections and remainder sections

* One
* Three
* Two
* All of the above

C. Two

2

. We use the mutex lock to protect critical regions and thus prevent race conditions.the term mutex is short for

* mutual exception
* mutual exclusion
* mutually explained
* mutual excluded

B. mutual exclusion

3

. Select the correct statements regarding mutex lock to prevent race condition.  
I. a process must acquire the lock before entering a critical section;  
II. a process need not acquire the lock before entering a critical section;  
III. it releases the lock when it exits the critical section  
IV. a process must acquire the lock when it exits the critical section.

* 1 and 3
* 2 and 4
* 3 and 4
* 1 and 4

A. 1 and 3

4

. The main disadvantage of the Mutex Lock is

* No waiting
* No waiting
* mutex locks can not be used to solve classical synchronization problems
* none of the above

B. No waiting

5

. In Spinlocks :

* no context switch is required when a process must wait on a lock
* locks are expected to be held for short times
* employed on multiprocessor systems
* All of the mentioned

D. All of the mentioned

Explanation : Spinlocks do have an advantage, however, in that no context switch is required when a process must wait on a lock, and a context switch may take considerable time. Thus, when locks are expected to be held for short times, spinlocks are useful. They are often employed on multiprocessor systems where one thread can “spin” on one processor while another thread performs its critical section on another processor

6

. A semaphore S is an integer variable that, apart from initialization, is accessed only through two standard atomic operations:

* exec() and exit()
* exec() and signal()
* wait() and exit()
* wait() and signal()

D. wait() and signal()

7

. where two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes is called.

* MutexLocks
* deadlocked
* spinlocks
* None of the above

B. deadlocked

Explanation : 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.

8

. Indefinite blocking may occur if we remove processes from the list associated with a semaphore in \_\_\_\_\_\_\_\_ order.

* LIFO (last-in, first-out)
* FIFO (Fisrst-in, first-out)
* LIFO and FIFO
* FILO (First in Last Out)

A. LIFO (last-in, first-out)

9

. indefinite blocking or starvation is problem related to \_\_\_

* MutexLocks
* deadlocks
* spinlocks
* None of the above

B. deadlocks

Explanation : indefinite blocking or starvation is a situation in which processes wait indefinitely within the semaphore. Indefinite blocking may occur if we remove processes from the list associated with a semaphore in LIFO (last-in, first-out) order.

10

. In Priority Inversion Which of following Stands TRUE:

* A scheduling challenge arises when a higher-priority process needs to read or modify kernel data that are currently being accessed by a lower-priority process
* Since kernel data are typically protected with a lock, the higher-priority process will have to wait for a lower-priority one to finish with the resource
* The situation becomes more complicated if the lower-priority process is preempted in favor of another process with a higher priority
* All of the above

D. All of the above

Explanation : Priority inversion can be more than a scheduling inconvenience. On systems with tight time constraints—such as real-time systems—priority inversion can cause a process to take longer than it should to accomplish a task. When that happens, other failures can cascade, resulting in system failure

11

. a classic \_\_\_\_\_\_solution to the critical-section problem known as Peterson’s solution

* Hardware based
* software-based
* software and Hardware based
* None of the above

B. software-based

. The bounded buffer problem is also known as :

* Producer – Consumer problem
* Readers – Writers problem
* Dining – Philosophers problem
* Dining – Readers problem

A. Producer – Consumer problem

2

. In bounded buffer problem Which of the following Assumption Stands TRUE:

1. I. We assume that the pool consists of n buffers, each capable of holding one item.
2. II. The mutex semaphore provides mutual exclusion for accesses to the buffer pool and is initialized to the value 1.
3. III. The mutex semaphore provides mutual exclusion for accesses to the buffer pool and is initialized to the value 0.
4. IV. The empty and full semaphores count the number of empty and full buffers.The semaphore empty is initialized to the value n; the semaphore full is initialized to the value 0.
5. V. The empty and full semaphores count the number of empty and full buffers.The semaphore empty is initialized to the value 0; the semaphore full is initialized to the value n.

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

B. 1,2,4

Explanation : In problem, the producer and consumer processes share the following data structures:   int n; semaphore mutex = 1; semaphore empty = n; semaphore full = 0   We assume that the pool consists of n buffers, each capable of holding one item. The mutex semaphore provides mutual exclusion for accesses to the buffer pool and is initialized to the value 1. The empty and full semaphores count the number of empty and full buffers. The semaphore empty is initialized to the value n; the semaphore full is initialized to the value 0.

3

. The readers–writers problem and its solutions have been generalized to provide reader–writer locks on some systems. When a process wishes only to read shared data, it requests the reader–writer lock in\_\_\_\_

* Read MODE
* Read MODE
* Read and Write MODE
* All of the above

A. Read MODE

4

. The readers–writers problem and its solutions have been generalized to provide reader–writer locks on some systems. A process wishing to modify the shared data must request the lock in\_\_\_\_

* Read MODE
* Write MODE
* Read and Write MODE
* All of the above

B. Write MODE

5

. Multiple processes are permitted to concurrently acquire a reader–writer lock in read mode, but only one process may acquire the lock for \_\_\_, as exclusive access is required for \_\_\_\_

* reading ,readers
* reading,writers
* Writing ,Writers
* None of the above

C. Writing ,Writers

6

. The Dining-Philosophers Problem Solution is:

* Deadlock free solution
* Starvation free solution
* page fault free solution
* All of the above

A. Deadlock free solution

7

. All processes share a semaphore variable mutex, which is initialized to 1. Each process must execute wait(mutex) before entering the critical section and signal(mutex) afterward. Suppose that a process interchanges the order in which the wait() and signal() operations on the semaphore mutex are executed, resulting in the following execution:   signal(mutex); ... critical section ... wait(mutex);   In this situation

* a deadlock will occur
* processes will starve to enter critical section
* several processes maybe executing in their critical section
* all of the mentioned

C. several processes maybe executing in their critical section

8

. All processes share a semaphore variable mutex, which is initialized to 1. Each process must execute wait(mutex) before entering the critical section and signal(mutex) afterward.   Suppose that a process replaces signal(mutex) with wait(mutex). That is, it executes   wait(mutex); ... critical section ... wait(mutex);   In this Case

* a deadlock will occur
* processes will starve to enter critical section
* several processes maybe executing in their critical section
* a deadlock will not occur

A. a deadlock will occur

9

. in the following execution:   signal(mutex); ... critical section ... wait(mutex);   Suppose that a process omits the wait(mutex), or the signal(mutex), or both. In this case:

* processes will starve to enter critical section
* several processes maybe executing in their critical section
* either mutual exclusion is violated or a deadlock will occur.
* processes will not starve to enter critical section

C. either mutual exclusion is violated or a deadlock will occur.

. CPU-scheduling decisions may take place under which of the following  circumstances:

1. When a process switches from the running state to the waiting state
2. When a process switches from the running state to the ready state
3. When a process switches from the waiting state to the ready state
4. When a process terminates

* 1 and 2
* 1,2,3
* 2,3,4
* 1,2,3,4

D. 1,2,3,4

2

. Under nonpreemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until .

1. it releases the CPU by terminating
2. it releases the CPU by switching to the waiting state
3. it releases the CPU By switching to the ready state
4. it releases the CPU By switching to the Running state

* 1 and 2
* 1,2,3
* 2,3,4
* 1,2,3,4

A. 1 and 2

3

. An I/O-bound program typically has

* a few long CPU bursts
* a few short CPU bursts
* many short CPU bursts
* None of the above

C. many short CPU bursts

4

. A CPU-bound program might have

* a few long CPU bursts
* a few short CPU bursts
* many short CPU bursts
* None of the above

A. a few long CPU bursts

5

. Whenever the CPU becomes idle, the operating system must select one of the processes in the \_\_\_ to be executed.

* execution queue
* process queue
* waiting queue
* ready queue

D. ready queue

Explanation : As we shall see when we consider the various scheduling algorithms, a ready queue can be implemented as a FIFO queue, a priority queue, a tree, or simply an unordered linked list. 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

6

. The \_\_\_\_ is the module that gives control of the CPU to the process selected by the short-term scheduler.

* interrupt
* scheduler
* dispatcher
* All of the above

C. dispatcher

Explanation : Another component involved in the CPU-scheduling function is the dispatcher. This function involves the following:

* Switching context
* Switching to user mode
* Jumping to the proper location in the user program to restart that program

7

. The dispatcher should be as fast as possible, since it is invoked during every process switch. The time it takes for the dispatcher to stop one process and start another running is known as the \_\_\_\_.

* hardware latency
* dispatch latency
* memory latency
* RAM Latency

B. dispatch latency

1

. Many criteria have been suggested for comparing CPU-scheduling algorithms.The criteria include which of the following:

1. CPU utilization
2. Throughput
3. Turnaround time
4. Waiting time
5. Response time

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

D. 1,2,3,4,5

2

. We want to keep the CPU as busy as possible,This criteria refers to as

* Throughput
* CPU utilization
* Response time

B. CPU utilization

3

. If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called \_\_\_\_.

* Throughput
* CPU utilization
* Turnaround time
* Response time

A. Throughput

4

. From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the \_\_\_\_

* Throughput
* Waiting time
* Turnaround time
* Response time

C. Turnaround time

Explanation : Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

5

. \_\_\_\_\_\_\_\_\_is the sum of the periods spent waiting in the ready queue.

* Throughput
* Waiting time
* Turnaround time
* Response time

B. Waiting time

Explanation : The CPU-scheduling algorithm does not affect the amount of time during which a process executes or does I/O. It affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue

6

. another measure is the time from the submission of a request until the first response is produced. This measure is called\_\_\_\_\_

* Throughput
* Waiting time
* Turnaround time
* Response time

D. Response time

Explanation : : it is the time it takes to start responding, not the time it takes to output the response.

7

. It is desirable to \_\_\_\_\_\_.

* minimize CPU utilization and throughput and to maximize turnaround time, waiting time, and response time
* maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time
* maximize CPU utilization and turnaround time and to minimize throughput, waiting time, and response time
* minimize waiting time and throughput and to maximize turnaround time, CPU utilization, and response time

B. maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time

1

. In Which of the following algorithm the process that requests the CPU first is allocated the CPU first

* Shortest-Job-First Scheduling
* First-Come, First-Served Scheduling
* Priority Scheduling
* Round-Robin Scheduling

B. First-Come, First-Served Scheduling

2

. Which of the following  scheduling algorithm is nonpreemptive

* Shortest-Job-First Scheduling
* First-Come, First-Served Scheduling
* Priority Scheduling
* Round-Robin Scheduling

B. First-Come, First-Served Scheduling

3

. Which of the following  scheduling algorithm the average waiting time for the process to start execution  is often quite long

* Shortest-Job-First Scheduling
* Shortest-Job-First Scheduling
* First-Come, First-Served Scheduling
* Priority Scheduling

C. First-Come, First-Served Scheduling

4

. Which of the following  scheduling algorithm associates with each process the length of the process’s next CPU burst

* Shortest-Job-First Scheduling
* First-Come, First-Served Scheduling
* Priority Scheduling
* Round-Robin Scheduling

A. Shortest-Job-First Scheduling

5

. In Shortest-Job-First Scheduling If the next CPU bursts of two processes are the same, \_\_\_\_ scheduling is used to break the tie.

* Shortest-Job-First
* First-Come, First-Served
* Priority
* Round-Robin

B. First-Come, First-Served

6

. more appropriate term for Which of the following scheduling method would be the shortest-next CPU-burst algorithm

* Round-Robin Scheduling
* First-Come, First-Served Scheduling
* Priority Scheduling
* Shortest-Job-First Scheduling

D. Shortest-Job-First Scheduling

7

. The SJF algorithm can be

* preemptive Only
* nonpreemptive Only
* either preemptive or nonpreemptive
* None of the above

C. either preemptive or nonpreemptive

8

. Which of the following  scheduling algorithm is used frequently in long-term scheduling

* Shortest-Job-First Scheduling
* First-Come, First-Served Scheduling
* Priority Scheduling
* Round-Robin Scheduling

A. Shortest-Job-First Scheduling

9

. Which of the following  scheduling algorithm is provably optimal, in that it gives the minimum average waiting time for a given set of processes.

* Round-Robin Scheduling
* First-Come, First-Served Scheduling
* Priority Scheduling
* Shortest-Job-First Scheduling

D. Shortest-Job-First Scheduling

10

. \_\_\_\_ is sometimes called shortest-remaining-time-first scheduling

* Round-Robin Scheduling
* Preemptive SJF scheduling
* Priority Scheduling
* First-Come, First-Served Scheduling

B. Preemptive SJF scheduling

11

. Which of the following Statement is TRUE for Shortest-Job-First Scheduling

1. The SJF algorithm is a special case of the general priority-scheduling algorithm
2. it can be implemented at the level of short-term CPU scheduling
3. A nonpreemptive SJF algorithm will allow the currently running process to finish its CPU burst
4. A preemptive SJF algorithm will preempt the currently executing process

* 1 ,2,4
* 1,2,3
* 1,3,4
* 1,2,3,4

C. 1,3,4

Explanation : The Disadvantages of SJF algorithm is it cannot be implemented at the level of short-term CPU scheduling

12

. Which of the following Statement is TRUE for First-Come, First-Served Scheduling

1. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue
2. FCFS scheduling algorithm is preemptive
3. The implementation of the FCFS policy is easily managed with a FIFO queue
4. the average waiting time under the FCFS policy is often quite small

* 1 ,2,4
* 1,3
* 1,3,4
* 1,2,3,4

B. 1,3

Explanation : Note that the FCFS scheduling algorithm is nonpreemptive ,the major Disadvantages of FCFS policy is the average waiting time under the FCFS policy is often quite long.

13

. The next CPU burst of the newly arrived process may be shorter than what is left of the currently executing process. A \_\_\_\_\_ will preempt the currently executing process

* preemptive priority scheduling algorithm
* nonpreemptive SJF algorithm
* preemptive SJF algorithm
* First-Come, First-Served Scheduling

C. preemptive SJF algorithm

14

. The \_\_\_\_ is a special case of the general priority-scheduling algorithm

* FCFS scheduling
* RR scheduling
* FCLS scheduling
* SJF algorithm

D. SJF algorithm

15

. In which of the following Scheduling Algorithm A priority is associated with each process, and the CPU is allocated to the process with the highest priority.

* Round-Robin Scheduling
* Preemptive SJF scheduling
* Priority Scheduling
* First-Come, First-Served Scheduling

C. Priority Scheduling

16

. In Priority Scheduling Equal-priority processes are scheduled in \_\_\_ order

* LIFO -Last In Fist Out
* FCFS - First Come First Served
* SJF - Shortest job First
* LILO -Last in Last Out

B. FCFS - First Come First Served

17

. In Priority Scheduling Algorithm Which of the following stands True

* the priority (p) is equal to the next CPU burst. The smaller the CPU burst, the higher the priority
* the priority (p) is the inverse of the (predicted) next CPU burst. The smaller the CPU burst, the higher the priority
* the priority (p) is the inverse of the (predicted) next CPU burst. The larger the CPU burst, the lower the priority
* the priority (p) is equal to the next CPU burst. The larger the CPU burst, the lower The priority

C. the priority (p) is the inverse of the (predicted) next CPU burst. The larger the CPU burst, the lower the priority

18

. The Priority Scheduling algorithm can be

* preemptive Only
* nonpreemptive Only
* either preemptive or nonpreemptive
* None of the above

C. either preemptive or nonpreemptive

19

. A preemptive priority scheduling algorithm will preempt the CPU if

* The priority of the newly arrived process is higher than the priority of the currently running process
* The priority of the newly arrived process is Lower than the priority of the currently running process
* The priority of the newly arrived process is equal to the priority of the currently running process
* all of the above

A. The priority of the newly arrived process is higher than the priority of the currently running process

20

. A nonpreemptive priority scheduling algorithm will simply put the new process at \_\_\_

* The tail of the ready queue.
* The head of the ready queue.
* head and tail of the ready queue
* None of the above

B. The head of the ready queue.

21

. The indefinite blocking, or starvation is A major problem related to which of the following scheduling algorithm

* Shortest-Job-First Scheduling
* First-Come, First-Served Scheduling
* Priority Scheduling
* Round-Robin Scheduling

C. Priority Scheduling

22

. Which of the following scheduling algorithm can leave some low priority processes waiting indefinitely?

* Shortest-Job-First Scheduling
* Priority Scheduling
* First-Come, First-Served Scheduling
* Round-Robin Scheduling

B. Priority Scheduling

23

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

* Switching
* Swapping
* Paging
* aging

D. aging

24

. Which of the following statement is True for Aging

* Aging involves gradually increasing the priority of processes that wait in the system for a long time
* **Aging involves gradually decreasing the priority of processes that wait in the system for a long time**
* Aging involves gradually increasing the priority of processes that wait in the system for a small time
* Aging involves gradually decreasing the priority of processes that wait in the system for a small time

A. Aging involves gradually increasing the priority of processes that wait in the system for a long time

25

. Which of the following Scheduling Algorithm is designed especially for timesharing systems?

* Shortest-Job-First Scheduling
* Priority Scheduling
* First-Come, First-Served Scheduling
* Round-Robin Scheduling

D. Round-Robin Scheduling

26

. The RR scheduling algorithm is

* preemptive
* nonpreemptive
* either preemptive or nonpreemptive
* All of the above

A. preemptive

27

. A Round-Robin Scheduling algorithm will simply put the new process at \_\_\_\_\_

* the tail of the ready queue
* the head of the ready queue.
* head and tail of the ready queue.
* None of the above

A. the tail of the ready queue

28

. In which of the following scheduling algorithm The ready queue is treated as a circular queue. and we again treat the ready queue as a FIFO queue of processes

* Round-Robin Scheduling
* Preemptive SJF scheduling
* Priority Scheduling
* First-Come, First-Served Scheduling

A. Round-Robin Scheduling

29

. In which of the following scheduling algorithm new processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.

* Shortest-Job-First Scheduling
* Priority Scheduling
* First-Come, First-Served Scheduling
* Round-Robin Scheduling

D. Round-Robin Scheduling

Explanation : A small unit of time, called a time quantum or time slice, is defined. A time quantum is generally from 10 to 100 milliseconds in length

30

. In Round-Robin Scheduling algorithm, The process may have a CPU burst of less than 1 time quantum. In this case:

* 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 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 head of the ready queue, The CPU scheduler will then select the next process in the ready queue
* the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue
* the process itself will not release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue

C. the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue

31

. In Round-Robin scheduling algorithm, If the CPU burst of the currently running process is longer than 1 time quantum. In this case:

* 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 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 head of the ready queue, The CPU scheduler will then select the next process in the ready queue
* the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue
* the process itself will not release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue

A. 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.

32

. In which of the following scheduling algorithm the performance of the algorithm depends heavily on the size of the time quantum

* Shortest-Job-First Scheduling
* Priority Scheduling
* First-Come, First-Served Scheduling
* Round-Robin Scheduling

D. Round-Robin Scheduling

1

. A deadlock situation can arise if which of the following conditions hold simultaneously in a system:

* Mutual exclusion
* Hold and wait
* No preemption
* All of the above

D. All of the above

2

. At least one resource must be held in a nonsharable mode; that is, only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.This condition for deadlock is refered to as :

* Mutual exclusion
* Hold and wait.
* No preemption
* Circular wait

A. Mutual exclusion

3

. A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.This condition for deadlock is refered to as :

* Mutual exclusion
* Hold and wait.
* No preemption
* Circular wait

B. Hold and wait.

4

. Resources cannot be preempted; that is, a resource can be released only voluntarily by the process holding it, after that process has completed its task.This condition for deadlock is refered to as :

* Mutual exclusion
* Hold and wait.
* No preemption
* Circular wait.

C. No preemption

5

. A set {P0, P1, ..., Pn} of waiting processes must exist such that P0 is waiting for a resource held by P1, P1 is waiting for a resource held by P2, ..., Pn−1 is waiting for a resource held by Pn, and Pn is waiting for a resource held by P0.This condition for deadlock is refered to as :

* Mutual exclusion
* Mutual exclusion
* No preemption
* Circular wait.

D. Circular wait.

6

. Deadlocks can be described more precisely in terms of a directed graph called a

* Bar graph
* system resource-allocation graph
* Pie charts
* Line graphs

B. system resource-allocation graph

7

. Which of the following statements is TRUE for dealing with the deadlock problem:

1. We can use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlocked state.
2. We can allow the system to enter a deadlocked state, detect it, and recover.
3. We can ignore the problem altogether and pretend that deadlocks never occur in the system.

* 1
* 1,2
* 1,2,3
* None of the above

C. 1,2,3

8

. To ensure that deadlocks never occur, the system can use

1. deadlock prevention scheme
2. deadlock-avoidance scheme
3. deadlock-detection scheme
4. deadlock-recovery scheme

* 1,2
* 1,2,3
* 3,4
* 1,2,3,4

A. 1,2

. Deadlock Prevention Scheme can applied if:

* each of the four necessary conditions must hold
* By ensuring that at least one of these conditions can hold
* By ensuring that at least one of these conditions cannot hold
* All of the above

C. By ensuring that at least one of these conditions cannot hold

2

. For Deadlock Prevention which of the following condition Stands TRUE:

1. The mutual exclusion condition must hold
2. hold-and-wait condition never occurs in the system
3. No Preemption condition does not hold
4. Circular Wait condition never holds

* 1
* 1,2
* 1,2,3
* 1,2,3,4

D. 1,2,3,4

3

. In Deadlock Prevention Scheme ,The mutual exclusion condition must hold if:

* at least one resource must be nonsharable
* at least one resource must be sharable
* whenever a process requests a resource, it does not hold any other resources
* All of the above

A. at least one resource must be nonsharable

4

. In Deadlock Prevention Scheme ,To ensure that the hold-and-wait condition never occurs in the system if:

* at least one resource must be nonsharable
* at least one resource must be sharable
* whenever a process requests a resource, it does not hold any other resources
* All of the above

C. whenever a process requests a resource, it does not hold any other resources

5

. To ensure that the hold-and-wait condition never occurs in the system Which of the following protocol should be used

* each process to request and be allocated all its resources before it begins execution. We can implement this provision by requiring that system calls requesting resources for a process precede all other system calls.
* protocol allows a process to request resources only when it has none. A process may request some resources and use them. Before it can request any additional resources, it must release all the resources that it is currently allocated.
* If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources the process is currently holding are preempted
* A and B Both

C. If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources the process is currently holding are preempted

6

. To ensure that the hold-and-wait condition never occurs in the system In Deadlock Prevention Scheme,which of the following is the major disadvantages of the protocol used.

1. Swapping
2. resource utilization may be low,
3. starvation
4. page fault

* 1,2
* 1,2,3
* 2,3
* 2,3,4

C. 2,3

Explanation : First, resource utilization may be low, since resources may be allocated but unused for a long period. In the example given, for instance, we can release the DVD drive and disk file, and then request the disk file and printer, only if we can be sure that our data will remain on the disk file. Otherwise, we must request all resources at the beginning for both protocols. Second, starvation is possible. A process that needs several popular resources may have to wait indefinitely, because at least one of the resources that it needs is always allocated to some other process.

7

. The third necessary condition for deadlocks is that there be no preemption of resources that have already been allocated. To ensure that this condition does not hold, Which of the following protocol should be used.

* **each process to request and be allocated all its resources before it begins execution. We can implement this provision by requiring that system calls requesting resources for a process precede all other system calls.**
* protocol allows a process to request resources only when it has none. A process may request some resources and use them. Before it can request any additional resources, it must release all the resources that it is currently allocated.
* If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources the process is currently holding are preempted
* All of the above

C. If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources the process is currently holding are preempted

8

. If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources the process is currently holding are preempted.This protocol is often applied to  resources like

1. CPU register
2. memory space
3. mutex locks
4. semaphores

* 1 and 2
* 3 and 4
* 1,2,3
* 2,3,4

A. 1 and 2

9

. The fourth and final condition for deadlocks is the circular-wait condition. One way to ensure that this condition never holds .Which of the following protocol should be used.

1. a process requesting an instance of resource type Rj must have released any resources Ri such that F(Ri) ≥ F(Rj).
2. protocol allows a process to request resources only when it has none. A process may request some resources and use them. Before it can request any additional resources, it must release all the resources that it is currently allocated.
3. If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources the process is currently holding are preempted
4. Each process can request resources only in an increasing order of enumeration. That is, a process can initially request any number of instances of a resource type —say, Ri. After that, the process can request instances of resource type Rj if and only if F(Rj) > F(Ri).

* 1,2
* 1,3
* 1,4
* 3,4

C. 1,4

10

. Prevent deadlocks by limiting how requests can be made. The limits ensure that at least one of the necessary conditions for deadlock cannot occur. Possible side effects of preventing deadlocks by this method are:

1. Low device utilization
2. Reduced system throughput.
3. Starvation
4. Page fault

* 1,2
* 1, 2, 3
* 1, 2, 4
* All of the above

B. 1, 2, 3

1

. An alternative method for avoiding deadlocks is

* by limiting how requests can be made.
* to require additional information about how resources are to be requested
* a and b
* None of the above

B. to require additional information about how resources are to be requested

2

. avoiding deadlocks is to require additional information about how resources are to be requested .In order to  avoid a possible future deadlock which of the following information is required?

* Each request requires that in making this decision the system consider the resources currently available
* the resources currently allocated to each process
* the future requests and releases of each process.
* All of the above

D. All of the above

3

. avoiding deadlocks ,The simplest and most useful model requires that each process declare the \_\_\_\_ of resources of each type that it may need

* Minimum Number
* maximum number
* Average Number
* None of the above

B. maximum number

4

. A deadlock-avoidance algorithm dynamically examines the \_\_\_\_ to ensure that a circular-wait condition can never exist.

* resource-allocation state
* resource-available state
* resource-Utilization state
* All of the above

A. resource-allocation state

5

. In A deadlock-avoidance algorithm The resource allocation state is defined by

* the number of available resources
* the allocated resources
* the maximum demands of the processes
* all of the above

D. all of the above

6

. In A deadlock-avoidance algorithm A state is safe if

* there exists a safe sequence
* the system can allocate resources to each process (up to its maximum) in some order and still avoid a deadlock
* A and B both
* None of the above

C. A and B both

7

. consider a system with twelve magnetic tape drives and three processes: P0, P1, and P2. Process P0 requires ten tape drives, process P1 may need as many as four tape drives, and process P2 may need up to nine tape drives. Suppose that, at time t0, process P0 is holding five tape drives, process P1 is holding two tape drives, and process P2 is holding two tape drives. (Thus, there are three free tape drives.)      
Maximum Needs Current Needs  
P0   10                               5  
P1   4                                2  
P2   9                                2  
At time t0, Which of the following sequence is a safe sequence ?

* P0, P1, P2
* P1, P0, P2
* P2, P0, P1
* P1, P2, P0

B. P1, P0, P2

Explanation : Process P1 can immediately be allocated all its tape drives and then return them (the system will then have five available tape drives); then process P0 can get all its tape drives and return them (the system will then have ten available tape drives); and finally process P2 can get all its tape drives and return them (the system will then have all twelve tape drives available).

8

. Which of the following stands true for Resource-allocation graph for deadlock avoidance.

* An algorithm for detecting a cycle in this graph requires an order of n^2 operations, where n is the number of processes in the system
* If no cycle exists, then the allocation of the resource will leave the system in a safe state.
* If a cycle is found, then the allocation will put the system in an unsafe state
* All of the above

D. All of the above

9

. The resource-allocation-graph algorithm is not applicable to\_\_\_

* a resource allocation system with single instances of each resource type
* a resource allocation system with multiple instances of each resource type
* a resource allocation system with Single and multiple instances of each resource type
* **All of the above**

B. a resource allocation system with multiple instances of each resource type

10

. which of the following statement is True for Banker algorithm for deadlock-avoidance.

* When a new process enters the system, it must declare the maximum number of instances of each resource type that it may need. This number may not exceed the total number of resources in the system
* It allows a process to request resources only when it has none. A process may request some resources and use them. Before it can request any additional resources, it must release all the resources that it is currently allocated.
* If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources the process is currently holding are preempted
* All of the above

A. When a new process enters the system, it must declare the maximum number of instances of each resource type that it may need. This number may not exceed the total number of resources in the system

11

. Which of the following data structures  must be maintained to implement the banker’s algorithm , where n is the number of processes in the system and m is the number of resource types:

1. A vector of length m indicates the number of available resources of each type. If Available[j] equals k, then k instances of resource type Rj are available.
2. An n × m matrix defines the maximum demand of each process. If Max[i][j] equals k, then process Pi may request at most k instances of resource type Rj
3. An n × m matrix defines the number of resources of each type currently allocated to each process. If Allocation[i][j] equals k, then process Pi is currently allocated k instances of resource type Rj.
4. An n × m matrix indicates the remaining resource need of each process. If Need[i][j] equals k, then process Pi may need k more instances of resource type Rj to complete its task. Note that Need[i][j] equals Max[i][j] − Allocation[i][j].

* 1
* 1,2
* 1,2,3
* 1,2,3,4

D. 1,2,3,4

1

. In Deadlock Detection Algorithm The wait-for graph scheme is  applicable to

* a resource-allocation system with multiple instances of each resource type
* a resource-allocation system with Single instances of each resource type
* A and B
* None of the above

B. a resource-allocation system with Single instances of each resource type

2

. wait-for graph can be obtained from  the resource-allocation graph by

* removing the resource nodes and collapsing the appropriate edges.
* collapsing the resource nodes and removing the appropriate edges.
* A and B
* None of the above

A. removing the resource nodes and collapsing the appropriate edges.

3

. an edge from Pi to Pj in a wait-for graph implies that

* process Pj is waiting for process Pi to release a resource that Pi needs
* process Pj is waiting for process Pi to release a resource that Pi needs
* process Pi is waiting for process Pj to release a resource that Pj needs
* process Pi is waiting for process Pj to release a resource that Pi needs

D. process Pi is waiting for process Pj to release a resource that Pi needs

4

. An edge Pi → Pj exists in a wait-for graph if and only if

* the corresponding resource allocation graph contains two edges Pj → Rq and Rq → Pi for some resource Rq
* the corresponding resource allocation graph contains two edges Rq → pi and Pi → Rq for some resource Rq
* the corresponding resource allocation graph contains two edges Pi → Rq and Rq → Pj for some resource Rq
* All of the above

C. the corresponding resource allocation graph contains two edges Pi → Rq and Rq → Pj for some resource Rq

5

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

* Cycle
* No Cycle
* Square
* All of the above

A. Cycle

6

. To detect deadlocks, the system needs to maintain the wait for graph and periodically invoke an algorithm that searches for a cycle in the graph. An algorithm to detect a cycle in a graph requires an

* order of n2 operations, where n is the number of vertices in the graph.
* order of n^2 operations, where n is the number of vertices in the graph.
* order of n2 operations, where n is the number of edges in the graph.
* order of n^2 operations, where n is the number of edges in the graph

B. order of n^2 operations, where n is the number of vertices in the graph.

7

. When should we invoke the detection algorithm?

* How often is a deadlock likely to occur?
* How many processes will be affected by deadlock when it happens?
* All of the above
* None of the above

C. All of the above

Explanation : If deadlocks occur frequently, then the detection algorithm should be invoked frequently. Resources allocated to deadlocked processes will be idle until the deadlock can be broken. In addition, the number of processes involved in the deadlock cycle may grow. Deadlocks occur only when some process makes a request that cannot be granted immediately. This request may be the final request that completes a chain of waiting processes. In the extreme, then, we can invoke the deadlockdetection algorithm every time a request for allocation cannot be granted immediately.

8

. invoking the deadlock-detection algorithm for every resource request will incur considerable overhead in

* seek time
* invoke time
* computation time
* All of the above

C. computation time

Explanation : A less expensive alternative is simply to invoke the algorithm at defined intervals—for example, Once per hour or whenever CPU utilization drops below 40 percent. (A deadlock eventually cripples system throughput and causes CPU utilization to drop.) If the detection algorithm is invoked at arbitrary points in time, the resource Graph may contain many cycles. In this case, we generally cannot tell which of the many deadlocked processes “caused” the deadlock.

1

. Which of the following is  options for breaking a deadlock

* simply abort one Process
* simply abort one or more processes to break the circular wait
* preempt some resources from one or more of the deadlocked processes
* All of the above

D. All of the above

2

. which of the following method can be used to eliminate deadlock under process termination

* Abort all deadlocked processes
* Abort one process at a time until the deadlock cycle is eliminated
* It is not possible to abort all deadlocked process
* A and B

D. A and B

3

. In which of the following method after each process is aborted, a deadlock-detection algorithm must be invoked to determine whether any processes are still deadlocked.

* Abort all deadlocked processes
* Abort one process at a time until the deadlock cycle is eliminated
* preempt some resources from one or more of the deadlocked processes
* None of the above

B. Abort one process at a time until the deadlock cycle is eliminated

4

. we should abort those processes whose termination will incur

* the average cost
* the Maximum cost
* the minimum cost
* None of the above

C. the minimum cost

5

. Many factors may affect which process is chosen for termination, including:

* What the priority of the process is
* How long the process has computed and how much longer the process will compute before completing its designated task
* How many and what types of resources the process has used (for example,whether the resources are simple to preempt)
* All of the above

D. All of the above

6

. If preemption is required to deal with deadlocks, which of the following issues need to be addressed:

* Selecting a victim
* Rollback
* Starvation
* All of the above

D. All of the above

7

. Which resources and which processes are to be preempted? As in process termination, we must determine the order of preemption to minimize cost. Cost factors may include which of the below parameters

* as the number of resources a deadlocked process is holding
* the amount of time the process has thus far consumed.
* A and B
* None of the above

C. A and B

8

. If we preempt a resource from a process, Which of the following stands true?

1. it cannot continue with its normal execution; it is missing some needed resource.
2. We must roll back the process to some safe state and restart it from that state
3. it can continue with its normal execution; it is missing some needed resource.
4. We should not roll back the process to some safe state and restart it from that state

* 1
* 1,2
* 2,3
* 3,4

B. 1,2

9

. Which of  the following method requires the system to keep more information about the state of all running processes to break the deadlock in Resource Preemption

* Selecting a victim
* Rollback
* Starvation
* deadlock-detection

B. Rollback

Explanation : Since, in general, it is difficult to determine what a safe state is, the simplest solution is a total rollback: abort the process and then restart it. Although it is more effective to roll back the process only as far as necessary to break the deadlock, this method requires the system to keep more information about the state of all running processes.

10

. In Resource preemtion which of the following causes  starvation to occur

* resources are always preempted from the same process
* resources will not always be preempted from the same process
* Abort all deadlocked processes
* Abort one process at a time until the deadlock cycle is eliminated

A. resources are always preempted from the same process

11

. Which following is the solution to starvation problem in resource preemtion

* the same process is always picked as a victim
* we must ensure that a process can be picked as a victim only a (small) finite number of times
* A and B both
* None of the above

B. we must ensure that a process can be picked as a victim only a (small) finite number of times

. Memory hierarchy comprises of Disk ,RAM Memory,Caches,select the increasing order of their access time by CPU for an instruction or data

* Disk ,RAM , Caches
* RAM,Disk,Caches
* Caches,RAM,Disk
* RAM,Caches,Disk

C. Caches,RAM,Disk

2

. for which of the below allocation is managed by Kernel and use of the allocated memory is managed by run time library?

* Caches
* RAM
* Disk
* All of the above

B. RAM

3

. for which of the below allocation and use is managed entirely in the hardware?

* Caches
* RAM
* Disk
* All of the above

A. Caches

4

. for which of the below allocation and use is managed by kernel?

* Caches
* RAM
* Disk
* All of the above

C. Disk

5

. for which of the below memory Accommodating more process in memory and  ensuring high hit ratio is the main performance issue

* Caches
* RAM
* Disk
* All of the above

B. RAM

6

. for which of the below memory ensuring high hit ratio is the main performance issue

* Caches
* RAM
* Disk
* All of the above

A. Caches

7

. for which of the below memory Quick loading and storing of parts of process address spaces is the main perfomanace issue

* Caches
* RAM
* Disk
* All of the above

C. Disk

8

. The effective memory access time depends on a

* Miss Ratio
* Hit Ratio
* Bit Ratio
* Byte Ratio

B. Hit Ratio

Explanation : performance of process depends on the hit ratio in various levels of the memory hierarchy,where the hit ratio in a level indicates what fraction of instruction or data bytes that were looked for in that level were actually present in it.

9

. During operation, a process creates data structures within the memory already allocated to it by the kernel,This function is actually performed by the\_\_\_\_

* Run time library
* dynamic library
* static library
* load time library

A. Run time library

10

. The kernel may decide keeping part of each process's address spaces in memory.it is achieved by part of memory hierarchy called\_\_\_\_

* Caches
* Disk
* RAM
* Virtual memory

D. Virtual memory

11

. Virtual memory in memory hierarchy consists of

* Caches and RAM
* Caches and Disk
* RAM and Disk
* All of the above

C. RAM and Disk

1

. Memory allocation and deallocation can be managed by\_\_\_\_\_\_\_\_

* CPU
* MMU
* GPR
* None of the above

B. MMU

Explanation : MMU-Memory management unit

2

. Which of the below factor considered the most in management of memory

* Efficient use of memory
* speedy allocation of memory
* speedy deallocation of memory
* All of the above

D. All of the above

3

. Which of the following statements stands true for Static memory allocation.

1. Static memory allocated can be performed by a compiler,linker or loader.
2. Static memory allocation to a process is possible only if size of its data structures are known before its execution begins.
3. static memory allocation is performed in a lazy manner during execution of program.
4. static memory is allocated to a function or a variable just before its is used for the first time.

* 1,2
* 2,3
* 3,4
* 1,4

A. 1,2

Explanation : Static memory allocated can be performed by a compiler,linker or loader while a program is being readied for execution. Static memory allocation to a process is possible only if size of its data structures are known before its execution begins.if sizes are not known,they have to be guessed :wrong estimation can lead to wastage of memory and lack of flexibility

4

. Which of the following statements stands true for Dynamic memory allocation.

1. Dynamic memory allocated can be performed by a compiler,linker or loader.
2. Dynamic memory allocation to a process is possible only if size of its data structures are known before its execution begins.
3. Dynamic memory allocation is performed in a lazy manner during execution of program.
4. Dynamic memory is allocated to a function or a variable just before its is used for the first time.

* 1,2
* 2,3
* 3,4
* 1,4

C. 3,4

Explanation : Dynamic memory allocation is performed in a lazy manner during execution of program. Dynamic memory is allocated to a function or a variable just before its is used for the first time.

5

. In which of the following memory allocation memory is wasted if we overestimate the its size or process may not operate correctly if we underestimate its size

* Dynamic memory allocation
* Static memory allocation
* contiguous memory allocation
* None-Contiguous memory allocation

B. Static memory allocation

6

. \_\_\_\_\_\_memory allocation avoid problems of wastage of memory and  the problems of underestimating the memory size

* Dynamic memory allocation
* Static memory allocation
* contiguous memory allocation
* None-Contiguous memory allocation

A. Dynamic memory allocation

Explanation : It can avoid  the problems by allocating a memory area whose size matches the actual size of the array, which would be known by the time the allocation is performed.it can even permit the array size to vary during operation of the process

7

. \_\_\_\_\_\_\_\_\_\_memory allocation incurs the overhead of memory allocation actions performed during operation of a process.

* Dynamic
* Static
* contiguous
* None-Contiguous

A. Dynamic

8

. \_\_\_ is generally called a translator

* loader
* Linker
* Compiler or Assembler
* re-locator

C. Compiler or Assembler

Explanation : it translates program P into an equivalent program in the object module form.this program contains instruction in the machine language of the computer.

9

. source program is converted to machine language of the computers by\_\_\_\_\_\_\_\_\_

* loader
* Linker
* relocator
* Compiler or Assembler

D. Compiler or Assembler

10

. Program P may call other programs during its execution,eg functions from mathematical libraries ,these functions should be included in the program,and their start address should be used in the function call instruction in P. This Procedure is called \_\_\_\_\_\_\_\_\_\_

* loading
* Linking
* Compiling or Assembling
* relocation

B. Linking

11

. some object module merged with program P may have conflicting translated time address.This conflict is resolved by changing the memory binding of the object modules;this action is called\_\_\_\_\_\_

* linking of object module
* loading of object module
* Relocation of object module
* compiling of object module

C. Relocation of object module

Explanation : It involves changing addresses of operands used in their instruction.

12

. The relocation and linking functions are performed by program called\_\_\_\_\_\_

* compiler
* linker
* loader
* relocator

B. linker

Explanation : The addresses assigned by it are called linked addresses.The user may specify the linked origin for program ;otherwise ,the linker assumes the linked origin to be same as the translated origin;In accordance with the linked origin and relocation necessary to avoid address conflicts,the linker binds instruction and data of the program to a set of linked address.

13

. A Binary program has to be loaded in memory for execution.This function is performed by the \_\_\_\_\_\_\_\_\_\_

* compiler
* linker
* loader
* relocator

C. loader

14

. Which of the following statements stands TRUE for Static Relocation of programs?

1. static relocation is performed before execution of the programs begins.
2. static relocation is performed during execution of the programs .
3. static relocation can be performed by suspending a program's execution,carrying out the relocation procedures and then resuming its execution.
4. it would require information concerning the translated origin and address sensitive instructions to be available during the program execution.

* 1
* 1,2
* 1,2,3
* 1,2,3,4

A. 1

Explanation : static relocation is performed before execution of the programs begins.

15

. Which of the following statements stands TRUE for Dynamic Relocation of programs

1. Dynamic relocation is performed before execution of the programs begins.
2. Dynamic relocation is performed during execution of the programs .
3. Dynamic relocation can be performed by suspending a program's execution,carrying out the relocation procedures and then resuming its execution.
4. it would require information concerning the translated origin and address sensitive instructions to be available during the program execution.

* 1
* 1,2
* 2,3
* 2,3,4

D. 2,3,4

Explanation : Dynamic relocation is performed during execution of the programs . Dynamic relocation can be performed by suspending a program's execution,carrying out the relocation procedures and then resuming its execution. it would require information concerning the translated origin and address sensitive instructions to be available during the program execution.

16

. some computer architecture provide a \_\_\_\_\_\_\_\_\_\_ to simplify dynamic relocation.

* Base Register
* special register
* Relocation register
* PSW-Program Status word

C. Relocation register

Explanation : The relocation register is special register in the CPU whose contents are added to every memory address used during execution of a program. whose result is another memory address,which is actually used to make a memory reference. **Effective memory address=memory address used in the current instruction+content of the relocation registe**r

17

. Which of the following statements stands true for Static linking?

1. the linker links all modules of a program before its execution begins.
2. The linker is invoked when an unresolved external reference is encountered during its execution.the linker resolves the external reference and resumes execution of the program.
3. if several programs use the same module from a library,each program will get a private copy of the module;several copies of the module might present in a memory at the same time if program using the same module.
4. All of the above.

* 1 only
* 1,2 only
* 1,3 only
* **4**

C. 1,3 only

Explanation : the linker links all modules of a program before its execution begins. it produce binary program that does not contain any unresolved external references. if several programs use the same module from a library,each program will get a private copy of the module;several copies of the module might present in a memory at the same time if program using the same module.

18

. Which of the following statements stands true for Dynamic linking?

1. linking is performed during execution of a binary program.
2. if the module referenced by the program has already been linked to another program that is execution,the same copy of the module could be linked to this program as well,thus saving the memory
3. it produce binary program that does not contain any unresolved external references.
4. All of the above

* 1 only
* 1,2 only
* 1,3 only
* 4

B. 1,2 only

Explanation : linking is performed during execution of a binary program. if the module referenced by the program has already been linked to another program that is execution,the same copy of the module could be linked to this program as well,thus saving the memory. The linker is invoked when an unresolved external reference is encountered during its execution.the linker resolves the external reference and resumes execution of the program.

19

. \_\_\_\_\_contains instructions and data of a program and information required for its relocation and linking.

* reentrant program
* object module
* Binary Program
* self-relocating program

B. object module

20

. \_\_\_\_\_\_ is ready to execute form of a program

* reentrant program
* object module
* Binary Program
* self-relocating program

C. Binary Program

21

. In \_\_\_\_\_\_\_\_ linking is performed in a lazy manners,ie an object module defining a symbol is linked to a program only when that symbol is referenced during the program's execution.

* Dynamically linked program
* Object module
* Binary Program
* self-relocating program

A. Dynamically linked program

22

. \_\_\_\_\_\_\_ can relocate itself to execute in any area of memory

* Dynamically linked program
* Reentrant program
* Binary Program
* self-relocating program

D. self-relocating program

23

. \_\_\_\_\_\_ can be executed on several sets of data concurrently.

* Dynamically linked program
* Reentrant program
* Binary Program
* self-relocating program

B. Reentrant program

1

. Variables whose scope is associated with functions, procedures, or blocks, in a program and parameters of function or procedure calls. This kind of data is allocated on the\_\_\_\_\_\_\_\_\_\_\_

* Heap
* stack
* CPU register
* Program Counter

B. stack

Explanation : This data is allocated when a function, procedure or blocks is entered and is deallocated when it is exited.

2

. In Stack, Allocation and deallocations are performed in a \_\_\_\_\_\_\_\_\_Manner in response to push and pop operations.

* FIFO
* LIFO
* FILO
* LILO

B. LIFO

Explanation : We assume each entry in the stack to be of some standard size, say l bytes. Only the last entry in the stack is accessible at any time. Contiguous area of memory is reserved for the stack.

3

. A pointer called the \_\_\_\_\_\_ points to the first entry of the stack, while a pointer called the \_\_\_\_\_ points to the last entry allocated in the stack.

* top of Stack (TOS),Frame Base (FB)
* top of Stack (TOS),Stack Base (SB)
* Stack Base (SB),top of Stack (TOS)
* Frame Base (FB),top of Stack (TOS)

C. Stack Base (SB),top of Stack (TOS)

4

. During execution of a program, a stack is used to support function calls. the group of stack entries that pertain to one function call is called

* stack
* Frame Base
* base Frame
* Stack frame

D. Stack frame

Explanation : A stack frame is pushed on the stack when function is called.

5

. Stack frame contains

* Address of the function's parameters
* values of the functions parameters
* the return address
* All of the above

D. All of the above

Explanation : During execution of the function, the run time support of the programming language in which the program is coded creates local data of the function within the stack frame. At the end of the function's execution, the entire stack frame us popped off the stack and the return address contained in it is used to pass control back to the calling program.

6

. The first entry in the stack frame contains

* The next value of the frame base.
* The previous value of the frame base.
* the return address of the function
* All of the above.

B. The previous value of the frame base.

7

. The second entry in the stack frame contains

* The next value of the frame base
* The previous value of the frame base.
* the return address of the function
* All of the above.

C. the return address of the function

8

. The frame base points to the

* Last entry in the stack frame
* the return address of the function
* the last local data in the stack frame
* First entry in the stack frame

D. First entry in the stack frame

9

. The TOS points to the

* the last local data in the stack frame
* Last entry in the stack frame
* the return address of the function
* First entry in the stack frame

A. the last local data in the stack frame

10

. The size of a program can be obtained from its

* Directory entry
* size of the stack
* PCD-program controlled dynamic data vary during execution of the program
* all of the above

D. all of the above

11

. Which of the following statements stands TRUE for Memory Allocation Model?

1. The code and static data components in the program are allocated memory area that exactly matches their size.
2. The PCD data and stack share a single large area of memory but grow in opposite directions when memory is allocated to new data.
3. The PCD data is allocated by starting at the low end of this area while the stack is allocated by starting at the high end of the area. the memory between these two components is free.
4. In this model the stack and PCD data components do not have individual size restrictions.

* 1,4
* 1,2
* 1,3
* 1,2,3,4

D. 1,2,3,4

Explanation : The code and static data components in the program are allocated memory area that exactly matches their size. The PCD data and stack share a single large area of memory but grow in opposite directions when memory is allocated to new data. The PCD data is allocated by starting at the low end of this area while the stack is allocated by starting at the high end of the area. The memory between these two components is free. In this model the stack and PCD data components do not have individual size restrictions.

12

. PCD data can be allocated deallocated by

* MMU
* run time libraries of the programming language ie.library routines
* Kernel
* All of the above

B. run time libraries of the programming language ie.library routines

Explanation : A Program creates or destroys PCD data by calling appropriate routines of the run time library of the programming language in which it is coded.

. A Heap permits allocation and deallocation of memory in a \_\_\_\_\_\_\_\_ Manner.

* Random and Sequential order
* Sequential order
* Random Order
* None of the above

C. Random Order

Explanation : An Allocation request by a process returns with a pointer to the allocated memory area in the heap,and the process accesses the allocated memory area through this pointer. a deallocation request must present a pointer to the memory are through this pointer.

2

. \_\_\_\_\_\_\_\_\_\_allocators are used by run-time support of programming language to manage PCD data,and by the kernel to manage its own memory requirements.

* Stack
* heap
* stack and heap
* None of the above

B. heap

Explanation : Stack based allocators are not used for data that are allocated and released in an unordered manner.

3

. The speed of memory and efficient used of memory are two fundamental concerns in the design of memory allocator.\_\_\_\_\_\_\_allocation addresses both these concerns effectively.

* stack-based
* heap-based
* stack and heap based
* None of the above

A. stack-based

Explanation : Since memory allocation and deallocation is very fast-the allocators modifies only the SB,FB and TOS pointers to manage the free and allocated memory.

4

. In \_\_\_\_\_ allocator released memory is not reused automatically when fresh allocation are made.

* stack-based
* heap-based
* stack and heap based
* None of the above

B. heap-based

Explanation : In heap ,reuse of memory is not automatic;the heap allocator must try to reuse a free memory are while making fresh allocations.however ,the size of a memory request rarely matches the size of previously used memory area.

5

. Some memory are is left over when a fresh allocation is made in \_\_\_ allocator.

* stack-based
* heap-based
* stack and heap based
* None of the above

B. heap-based

Explanation : In heap ,reuse of memory is not automatic;the heap allocator must try to reuse a free memory are while making fresh allocations.however ,the size of a memory request rarely matches the size of previously used memory area.

6

. Which of the below kernel functions can be used for reuse of memory in heap management?

* Maintaining a free list.
* select a memory area for allocation.
* merge free memory area.
* All of the above

D. All of the above

Explanation : The kernel maintains a free list to keep information about free memory area in the system.A memory request is satisfied by using the free memory area that is considered the most suitable for the request,and the memory left over from this memory area is entered in the free list.The kernel tries to merge free areas of memory into larger free area so that larger memory request can be granted.

7

. which of the following control information the kernel needs to maintains for each memory area in the free list;

* The size of the memory areas
* the pointers used for forming the lists.
* The size of the memory areas and the pointers used for forming the lists.
* Stack pointer

C. The size of the memory areas and the pointers used for forming the lists.

8

. Which of the following free lists is used to facilitate faster insertion and deletion operations on memory area while managing the free lists.

* Singly linked free list.
* Doubly linked free list
* Stack
* singly linked free list and stack

B. Doubly linked free list

Explanation : Singly linked list requires lot of work when a memory area is to be inserted into the list or deleted from it.as deletion of a memory area requires a change in the pointers stored in the previous memory area in the list.same this is for insertion operation.in Doubly linked free list each entry contains two pointers-one points to the next memory area in the list.while other points to the previous memory area.if a memory area with a specific address is to be deleted from the list,the kernel can simply take the pointer to the previous and following memory area in the list,and manipulate pointer in these areas to perform the deletion.

9

. Which of the following techniques can be used  to perform memory allocation by using a free list;

* First Fit technique
* Best-Fit technique
* Next-Fit technique
* All of the above.

D. All of the above.

10

. To service a request for n bytes of memory,the \_\_\_\_\_ technique uses the first free memory area it can find whose size is ≥ n bytes.

* First Fit technique
* Best-Fit technique
* Next-Fit technique
* Good-Fit technique.

A. First Fit technique

11

. The \_\_\_\_ technique uses the smallest free memory area with size ≥ n bytes.

* First Fit technique
* Best-Fit technique
* Next-Fit technique
* Good-Fit technique.

B. Best-Fit technique

12

. The \_\_\_\_\_ technique remembers which entry in the free list was used to make the last allocation.to make new allocation,it searches the free list starting from the next entry and perform allocation using first free memory area whose size  ≥ n bytes that it can find.

* First Fit technique
* Best-Fit technique
* Next-Fit technique
* Good-Fit technique.

C. Next-Fit technique

13

. \_\_\_\_\_\_\_\_ technique may split memory area at the start of the free list repeatedly,so free memory area become smaller with time.

* First Fit
* Best-Fit
* Next-Fit
* worst-Fit

A. First Fit

Explanation : the allocator may not have any large free memory area left to satisfy large memory request.Also several free memory areas may become unusably small.

14

. \_\_\_\_\_\_\_\_ technique avoids needless splitting of large memory area.

* First Fit
* Best-Fit
* Next-Fit
* worst-Fit

B. Best-Fit

15

. \_\_\_\_\_\_\_\_ technique incurs higher allocation overhead

* First Fit
* Best-Fit
* Next-Fit
* worst-Fit

B. Best-Fit

Explanation : because it either has to process the entire free list at every allocation or maintain the free list in ascending order by size of free memory areas.

16

. \_\_\_\_\_\_\_\_ technique avoids splitting the same free area repeatedly also avoids allocation overhead.

* First Fit
* Best-Fit
* Next-Fit
* worst-Fit

C. Next-Fit

. Memory fragmentation can be defined as

* The existence of usable area in the memory of computer system
* The existence of unusable area in the memory of computer system
* The existence of unreachable area in the memory of computer system
* None of the above

B. The existence of unusable area in the memory of computer system

2

. External fragmentation occurs when a

* memory area remain unused because it is too large to be allocated
* memory area remain unused because it is too small to be allocated
* More memory is allocated than requested by the process
* less memory is allocated than requested by the process

B. memory area remain unused because it is too small to be allocated

3

. Internal fragmentation occurs when a

* memory area remain unused because it is too large to be allocated
* memory area remain unused because it is too small to be allocated
* More memory is allocated than requested by the process
* less memory is allocated than requested by the process

C. More memory is allocated than requested by the process

Explanation : some of the memory allocated to process remain unused,which happens if a process is allocated more memory than it needed

4

. memory fragmentation results in\_\_\_\_\_\_\_

* stack overflow
* page faults
* Better Utilization of memory
* poor utilization of memory

D. poor utilization of memory

5

. In External fragmentation,merging of free memory areas using boundary tags.which of the following statement stands TRUE.

1. Boundary tags ,is a status descriptor for a memory area.
2. It consists of an ordered pair giving allocation status of the area;whether it is free or allocated.
3. Boundary tags are identical tags stored at the start and end of memory area.
4. when an area of memory becomes free ,the kernel checks the boundary tags of its neighboring areas.

* 1,2 only
* 1,3 only
* 1,4 only
* 1,2,3,4

D. 1,2,3,4

Explanation : Boundary tags ,is a status descriptor for a memory area. It consists of an ordered pair giving allocation status of the area;whether it is free or allocated. Boundary tags are identical tags stored at the start and end of memory area. when an area of memory becomes free ,the kernel checks the boundary tags of its neighboring areas.

6

. A relation called the 50-percent rule holds

* When an area of memory is freed,the total number of free area in the system decreases by 1
* When an area of memory is freed,the total number of free area in the system increases by 1
* When an area of memory is freed,the total number of free area in the system remain the same depending on whether the area being freed has zero,two,or one free area as neighbors.
* All of the above

D. All of the above

7

. Which of following statements stands true of a relation called 50-percent rule.

1. the number of free area is half the number of allocated area ie. m=n/2
2. it helps in estimating the size of the free list
3. it also gives us method of estimating the free area in the memory at any time
4. if sf is the average size of free area of memory,the total free memory is sf×n/2

* 1 only
* 1,2 only
* 1,2,3 only
* 1,2,3,4

D. 1,2,3,4

Explanation : the number of free area is half the number of allocated area ie. m=n/2 it helps in estimating the size of the free list it also gives us method of estimating the free area in the memory at any time if sf is the average size of free area of memory,the total free memory is sf×n/2

8

. In\_\_\_\_\_\_\_\_memory binding are changed in such manner that all free memory area can be merged to form a single free memory area.

* Memory Paging
* Memory Swapping
* Memory Compaction
* Memory segmentation

C. Memory Compaction

Explanation : It is achieved by packing all allocated areas towards once end of the memory.

9

. Which of the following statements stands true for memory compaction method.

1. it involves movement of code and data in the memory.
2. it is feasible only if computer system provides relocation register;the relocation can be achieved by simply changing the address in the relocation register
3. it does not involves movement of code and data in the memory
4. it does not involves use of relocation register

* 1,2 only
* 3,4 only
* 2,3 only
* 1,4 only

A. 1,2 only

Explanation : Memory compaction is not simple as suggested.it involves movement of code and data in memoryin free list if process memory has free memory on either side of it.it needs to be relocated to execute correctly from the new memory area allocated to it.Relocation involves modification of all addresses used by a process,including address of heap-allocated data and address contained in the general purppose registers.it is feasible only if the computer system provides a relocation register;relocation can be achieved by simply changing the address in the relocation register.

10

. Buddy system and power of 2 allocators leads to \_\_\_\_\_\_\_\_\_\_\_

* External fragmentation
* Internal Fragmentation
* No fragmentation
* None of the above

B. Internal Fragmentation

Explanation : it performs allocation of memory in blocks of a few standard sizes.this features leads to internal fragmentation because some memory in each allocated memory block may be wasted.

11

. In the Binary buddy system the sizes of memory blocks are \_\_\_\_\_\_\_\_\_\_\_ and separate free lists are maintained for blocks of \_\_\_\_\_\_\_\_\_\_\_sizes.

* power of 2,same
* square of 2,same
* power of 2,different
* square of 2,different

C. power of 2,different

12

. Which of the following statement is true for buddy system allocators?

1. Buddy System splits and recombines memory blocks in a predetermined manner during allocation and deallocation.
2. No splitting of blocks takes place, also no effort is made to coalesce adjoining blocks to form larger blocks; when released, a block is simply returned to its free list.
3. When a request is made for m bytes, the allocator first check the free list containing blocks whose size is 2i for the smallest value of i such that 2i ≥ m.if the free list is empty ,it checks the list containing blocks that are higher the next higher power of 2 in size on so on. an entire block is allocated to a request.
4. When a request is made for m bytes. the system finds the smallest power of 2 that is ≥ m. Let this be 2i.if the list is empty, it checks the lists for block of size 2i+1.it takes one block off this list and splits it into two halves of size 2i.it put one of these blocks into the free list of size 2i,and uses the other block to satisfy the request.

* 1 only
* 1,2 only
* 2,3 only
* 1,4 only

D. 1,4 only

Explanation : Buddy System splits and recombines memory blocks in a predetermined manner during allocation and deallocation. When a request is made for m bytes. the system finds the smallest power of 2 that is ≥ m. Let this be 2i.if the list is empty, it checks the lists for block of size 2i+1.it takes one block off this list and splits it into two halves of size 2i.it put one of these blocks into the free list of size 2i,and uses the other block to satisfy the request.

13

. Which of the following statement is true for Power-of-2 allocators?

1. Buddy System splits and recombines memory blocks in a predetermined manner during allocation and deallocation.
2. No splitting of blocks takes place, also no effort is made to coalesce adjoining blocks to form larger blocks; when released, a block is simply returned to its free list.
3. When a request is made for m bytes, the allocator first check the free list containing blocks whose size is 2i for the smallest value of i such that 2i ≥ m.if the free list is empty ,it checks the list containing blocks that are higher the next higher power of 2 in size on so on. an entire block is allocated to a request.
4. When a request is made for m bytes. the system finds the smallest power of 2 that is ≥ m. Let this be 2i.if the list is empty, it checks the lists for block of size 2i+1.it takes one block off this list and splits it into two halves of size 2i.it put one of these blocks into the free list of size 2i,and uses the other block to satisfy the request.

* 1 only
* 1,2 only
* 2,3 only
* 1,4 only

C. 2,3 only

Explanation : No splitting of blocks takes place, also no effort is made to coalesce adjoining blocks to form larger blocks; when released, a block is simply returned to its free list. When a request is made for m bytes ,the allocator first check the free list containing blocks whose size is 2i for the smallest value of i such that 2i ≥ m. If the free list is empty, it checks the list containing blocks that are higher the next higher power of 2 in size on so on. an entire block is allocated to a request.

14

. The buddy and power of 2 allocators are faster than the first-fit ,best-fit, next-fit allocators

* TRUE
* FALSE

A. TRUE

Explanation : because they avoid searches in free lists

15

. The power of 2 allocators is faster than the Buddy System allocators.

* TRUE
* FALSE

A. TRUE

Explanation : because it does not need to performs splitting and merging.

1

. Contiguous memory allocation is the classical memory allocation model in which

* Same process is allocated in a different area in the memory
* all the process is allocated a single contiguous area in the memory
* Each process is allocated a single contiguous area in the memory
* All of the above

C. Each process is allocated a single contiguous area in the memory

2

. Contiguous memory allocation faces the problem of

* Memory fragmentation
* Page Faults
* less throughput
* Less hit ratio

A. Memory fragmentation

3

. In Contiguous memory allocation \_\_\_\_\_\_\_\_\_ has no cure

* Internal fragmentation
* External fragmentation
* inline fragmentation
* outline fragmentation

A. Internal fragmentation

Explanation : Because Kernel has no means of estimating the memory requirement of a process accurately

4

. The technique of memory compaction and reuse of memory can be applied to overcome the problem of

* Internal fragmentation
* External fragmentation
* Page Faults
* Swapping

B. External fragmentation

5

. Memory compaction involves \_\_\_\_\_\_\_\_\_

* removing of memory
* No Relocation
* Static relocation
* Dynamic relocation

D. Dynamic relocation

6

. Dynamic Relocation is not feasible without\_\_\_\_\_

* Program Counter
* Special Purpose Register
* Relocation Register
* Program status Word

C. Relocation Register

7

. NonContiguous memory allocation is a model in which

* Portion of its address space are distributed among many areas of memory
* all the process is allocated a single continuous area in the memory
* Each process is allocated a single continuous area in the memory
* All of the above

A. Portion of its address space are distributed among many areas of memory

8

. In NonContiguous memory allocation \_\_\_\_\_\_\_

* it increases the external fragmentation
* it reduce the external fragmentation
* it increases the internal fragmentation
* it reduce the internal fragmentation

B. it reduce the external fragmentation

Explanation : This model of memory allocation permits the kernel to reuse free memory areas that are smaller than the size of a process, so it can reduce external fragmentation.

9

. logical address is the address\_\_\_\_\_\_

* constituents the physical address space
* of an instruction or data byte as used in a process.
* in a memory where an instruction or data byte exists
* All of the above

B. of an instruction or data byte as used in a process.

Explanation : it may be obtained using index,base,or segment registers.The logical address in a process constitutes the logical address space of the process

10

. physical address is the address\_\_\_\_\_\_

* constituents the logical address space
* of an instruction or data byte as used in a process.
* in a memory where an instruction or data byte exists
* All of the above

C. in a memory where an instruction or data byte exists

Explanation : The set of physical addresses in the system constitutes the physical address space of the system.

11

. The CPU sends the \_\_\_\_\_\_\_\_\_\_ of each data or instruction used in the process to the MMU.

* Physical address
* logical address
* effective address
* None of the above

B. logical address

12

. The MMU uses the memory allocation information stored in the table to compute the corresponding \_\_\_\_\_\_\_\_\_\_

* Physical address
* logical address
* effective address
* None of the above

A. Physical address

13

. The procedure of computing the effective memory address from a logical address is called\_\_\_\_\_\_

* physical address translation
* logical address translation
* address translation
* All of the above

C. address translation

14

. in Non-contiguous memory allocation the kernel allocates a single memory area to a process.

* TRUE
* FALSE

B. FALSE

Explanation : Because Kernel allocates single memory area to a process in contiguous memory allocation.In NonContiguous allocation,the kernel allocates several memory area to a process-each memory area holds one component of the process.

15

. In Contiguous allocation address translation is not required

* TRUE
* FALSE

A. TRUE

Explanation : In NonContiguous allocation address translation is performed by the MMU during program execution.

16

. In Contiguous allocation external fragmentation arises if first-fit,best-fit,or next-fit allocation is used.internal fragmentation arises if memory allocation is performed in blocks of a few standard sizes.

* TRUE
* FALSE

A. TRUE

17

. In NonContiguous allocation unless the computer system provides the relocation register,swapped-in process must be places in its originally allocated area.

* TRUE
* FALSE

B. FALSE

Explanation : Because in NonContiguous allocation component of a swapped-in process can be placed anywhere in memory and and it Contiguous allocation unless the computer system provides the relocation register,swapped-in process must be places in its originally allocated area.

18

. Which of the following is the fundamental approaches to implement Non-contiguous memory allocation?

1. Paging
2. Segmentation
3. Memory compaction
4. power of 2 allocator

* 1,2
* 2,3
* 3,4
* 1,4

A. 1,2

1

. In approaches to noncontiguous memory allocation, the memory can accommodate an integral number of pages. It is partitioned into memory areas that have same size as page. This approach is known as\_\_\_\_\_\_

* Segmentation
* Paging
* Thrashing
* none of the above

B. Paging

2

. Each process consists of fixed size component called \_\_\_\_\_\_\_\_\_

* Segment
* Page table
* Pages
* All of the above

C. Pages

3

. In which of the memory allocation techniques internal fragmentation can arise

* Segmentation
* Paging
* Segmentation with paging
* **All of the above**

B. Paging

Explanation : In paging internal fragmentation can arise because the last page of a process is allocated a page-size memory area even if it is smaller than a page in size.

4

. Which of the following non contiguous memory allocation approach have component of different sizes, so that kernel has to use memory reuse techniques such as first-fit or best-fit allocation?

* Segmentation
* Paging
* Segmentation with paging
* All of the above

A. Segmentation

5

. In \_\_\_\_\_\_\_\_\_\_\_, a programmer identifies component in a process Called\_\_\_\_\_\_\_\_\_ in a process.

* Paging, Pages
* Segmentation, Pages
* Segmentation, Segments
* Segment table, Pages

C. Segmentation, Segments

6

. Which of the following facilitates sharing of codes, data and program modules between processes?

* Segments
* Page table
* Pages
* Segment table

A. Segments

7

. In which of the following memory allocation techniques external fragmentation can arise?

* Segmentation
* Paging
* Segmentation with paging
* All of the above.

A. Segmentation

8

. In which of the following memory allocation techniques facilitates sharing of codes, data and program modules between processes without incurring external fragmentation?

* Segmentation
* Paging
* Segmentation with paging
* All of the above

C. Segmentation with paging

9

. A process is considered to consists of pages numbered from \_\_\_\_\_\_ onward

* -1
* 1
* 2
* 0

D. 0

10

. A process is considered to consist of pages. Each pages is of size *s*bytes, where *s* is power of \_\_\_\_

* -1
* 1
* 2
* 0

C. 2

. A memory hierarchy ,consisting of a computer system’s memory and a disk, that enables a process to operate with only some portion of its address space in memory known as \_\_\_\_\_\_\_

* RAM
* ROM
* Virtual memory
* Disk

C. Virtual memory

2

. A software component of Virtual memory is known as \_\_\_\_\_\_\_\_\_\_

* Memory management unit
* Virtual memory manager
* Memory manager
* All of the above.

B. Virtual memory manager

3

. Which of the following translate the logical address of the code in the instructions of a process into the address in memory where the operand or instruction actually resides?

* Memory management Unit
* Virtual memory manager
* Software Management Unit
* A and B both

D. A and B both

4

. A memory  that is larger than the real memory of the computer system is \_\_\_\_\_\_

* RAM
* ROM
* Disk
* Virtual Memory

D. Virtual Memory

5

. The \_\_\_\_\_\_\_\_\_\_\_loads only one component of a process address space in memory to begin with-the component that contain starting address of the process.

* Memory management Unit
* Virtual memory manager
* Memory manager
* All of the above.

B. Virtual memory manager

6

. The virtual memory manager loads other components of the process only when they are needed. This techniques is called\_\_\_\_\_\_\_\_\_\_\_\_

* Thrashing
* Swapping
* Demand paging
* Segmentation

C. Demand paging

Explanation : to keep the memory commitment to a process low, the virtual memory manager removes component of the process from memory time to time. These components would be loaded back in memory when needed again.

7

. The performance of a process in virtual memory depends on the rate at which its components have to be loaded into memory. to achieve a low rate of loading of process components Virtual memory manager uses the \_\_\_\_\_\_\_\_

* Law of locality of objects
* Law of locality of pointers
* Thrashing
* Law of locality of reference

D. Law of locality of reference

8

. Classify the below statement as true or false. as far as functions of the virtual memory manager is concerned it manage logical address space

* True
* False

A. True

Explanation : Set up the swap space of a process, organize its logical address space in memory through page-in and page-out operations, and maintain its page table.

9

. Classify the below statement as true or false. as far as functions of the virtual memory manager is concerned it manages memory.

* True
* False

A. True

Explanation : VMM keep track of occupied and free page frames in memory.

10

. Classify the below statement as true or false. as far as functions of the virtual memory manager is concerned it implement memory protection.

* True
* False

A. True

Explanation : VMM maintain the information needed for memory protection

11

. Classify the below statement as true or false. as far as functions of the virtual memory manager is concerned it collects page reference information

* True
* **False**

A. True

Explanation : Paging hardware provides information concerning page references. This information is maintained in appropriate data structures for use by the page replacement algorithm.

12

. Classify the below statement as true or false. as far as functions of the virtual memory manager is concerned it Does not perform page replacement.

* True
* False

B. False

Explanation : VMM perform replacement of a page when a page fault arises and all page frames in memory, or all page frames allocated to a process are occupied.

13

. Allocate physical memory is not a function of virtual memory manager

* True
* False

B. False

Explanation : VMM decide how much memory should be allocated to a process and revise this decision from time to time to suit the needs of the process and the OS

14

. Virtual memory manager implement page sharing functionality

* True
* False

A. True

Explanation : it arranges sharing of pages to process.

. The memory of the computer system is considered to consist of *page frames*, where a page frame is memory area that has the same size as a \_\_\_\_

* Page Table
* Page
* Segment
* Page frame table

B. Page

2

. Page frames are numbered from \_\_\_\_\_\_ Where *#frames*is the number of page frame of memory

* 1 to #frames
* 1 to #frames-1
* 0 to #frames -1
* Any of the above

C. 0 to #frames -1

3

. Each logical address used in a process is considered to be a pair (*pi,bi*), where *pi* is a page number and *bi* is the byte number in *pi,*Each page is of size *s* And 0 ≤ *bi*< s . The effective memory address of a logical address (*pi,bi*) is computed as follows:

* effective memory address of a logical address (*pi,bi*) = Start address of the Segment containing page *pi* + *bi*
* effective memory address of a logical address (*pi,bi*) = Start address of the Page Frame containing page *pi* + *bi*
* effective memory address of a logical address (*pi,bi*) = Start address of the Page Frame containing page *pi* - *bi*
* None of the above.

B. effective memory address of a logical address (*pi,bi*) = Start address of the Page Frame containing page *pi* + *bi*

4

. Memory allocation information for a process is stored in a \_\_\_\_\_\_\_

* Page table
* Frame table
* Pages
* Frame list

A. Page table

Explanation : Each entry in the page table contains memory allocation information for one page of a process. It contains the page frame number where a page resides.

5

. The page table for a process facilitates implementation of address translation, demand loading, and page replacement operations. Which of the followings are the fields in a page table entry

* Valid bit
* Page frame#
* Modified
* Prot info
* All of the above.

E. All of the above.

6

. Which of the following field in a page table entry indicates whether the page described by the entry currently exists in memory?

* Valid bit
* Page frame#
* Modified
* Prot info

A. Valid bit

7

. Which of the following field in a page table entry indicates which page frame of memory is occupied by the page?

* Valid bit
* Page frame#
* Modified
* Prot info

B. Page frame#

8

. Which of the following field in a page table entry indicates how the process may use contents of the page - whether read, write, or execute?

* Ref info
* Modified
* Prot info
* Other info

C. Prot info

9

. Which of the following field in a page table entry indicates information concerning references made to the page white it is in memory?

* Ref info
* Modified
* Other info
* Prot info

A. Ref info

10

. Which of the following field in a page table entry indicates whether the page has been modified while in memory?

* Ref info
* Modified
* Other info
* Prot info

B. Modified

11

. Which of the following field in a page table entry indicates other useful information concerning the page e.g., its position in the swap space?

* Ref info
* Modified
* Other info
* Prot info

C. Other info

12

. Which of the following field in the page table entry is also known as *presence bit*?

* Valid bit
* Page frame#
* Modified
* Prot info

A. Valid bit

13

. Which of the following field in the page table entry is also known as *dirty bit,* and this field is single bit?

* Valid bit
* Page frame#
* Modified
* Prot info

C. Modified

14

. Which of the following field is used to decide whether a page out operation is needed while replacing the page?

* Valid bit
* Page frame#
* Modified
* Prot info

C. Modified

Explanation : the modified bit indicates whether the page has been modified .i.e. whether it is dirty.

15

. Which of the following is the step performed in address translation by the MMU?

* Look up page table
* Obtain page number and byte number in a page
* Form effective memory address.
* All of the above.

D. All of the above.

16

. When a page is not present in the memory ,MMU raises an interrupt called a \_\_\_\_\_\_\_\_\_\_

* Page hit
* Page miss
* Page Fault
* All of the above.

C. Page Fault

Explanation : while performing address translation for a logical address pair (*pi,bi*) , the MMU check the valid bit of the page table entry of *pi*.if the bit indicates the *pi*is not present in memory, the MMU raises an interrupt called a ***missing page interrupt*** or a ***page fault***.

17

. Which of the following operations are performed by the Virtual memory manager when page fault occurs?

* Page in
* Page out
* Page replacement operations
* All of the above.

D. All of the above.

18

. The virtual memory manager uses a page replacement algorithm to select one of the pages currently in memory for replacement, accesses the page table entry of the selected page to mark it as “not present” in memory and initiates a \_\_\_\_\_\_\_\_operation for it if the modified bit of its page table entry indicates that it is dirty page.

* Page hit
* Page out
* Page Miss
* Page in

B. Page out

19

. The virtual memory manager initiates a \_\_\_\_\_ operation to load the required page into the page frame that was occupied by the selected page and it updates the page table entry of the page to record frame number of the page frame, marks the page as “present” and makes provision to resume operation of the process.

* Page hit
* Page out
* Page Miss
* Page in

D. Page in

20

. Page-in and page-out operations are required to implement demand paging constitutes \_\_\_\_\_\_\_

* Page I/O
* Process I/O
* Program I/O
* Disk I/O.

A. Page I/O

21

. Which of the following stands true for Effective memory access time for a process in demand paging?

* Only Time consumed by the MMU in performing address translation
* Only the average time consumed by the virtual memory manager in handling a page fault
* Time consumed by the MMU in performing address translation and the average time consumed by the virtual memory manager in handling a page fault
* None of the above.

C. Time consumed by the MMU in performing address translation and the average time consumed by the virtual memory manager in handling a page fault

Explanation : The effective memory access time can be improved by reducing the number of page faults. One way to achieving it is to load pages before they are needed by a process.

22

. Which of the following stands true for Effective memory access time for a process in demand paging? Where ***pr1***= probability that a page exists in memory, ***tmem*** = memory access time , ***tpfh*** = time overhead of page fault handling

* Effective memory access time = pr1 × 2 × tmem + (1 - pr1 ) × (tmem + tpfh + 2 × tmem)
* Effective memory access time = pr1 × 1 × tmem+ (1 - pr1 ) × (tmem + tpfh + 2 × tmem
* Effective memory access time = pr1 × 1 × tmem1 ) × (tmem + tpfh + 2 × tmem)
* Effective memory access time = pr1 × 2 × tmem - (1 + pr1 ) × (tmem + tpfh + 2 × tmem )

A. Effective memory access time = pr1 × 2 × tmem + (1 - pr1 ) × (tmem + tpfh + 2 × tmem)

Explanation : The effective memory access time can be improved by reducing the number of page faults. One way to achieving it is to load pages before they are needed by a process

. Page replacement becomes necessary when

1. Page faults occur and there are no free page frames in the memory.
2. Page faults occur and there are free page frames in the memory.
3. Page faults would arise if the replaced page is referenced again.
4. It is important to replace a page that is not likely to be referenced again in the immediate future.

* 1 only
* 1 and 3 only
* 1 , 2 and 4 only
* 1,3 and 4 only

D. 1,3 and 4 only

2

. Which of the following statements stands true for locality of reference in page replacement and handling page faults?

* It states that the physical addresses used by a process in any short interval of time during its operation tend to be bunched together in certain portion of its logical address space.
* It states that the logical addresses used by a process in any short interval of time during its operation tend to be bunched together in certain portion of its logical address space.
* It states that the physical addresses used by a process in any short interval of time during its operation tend to be bunched together in certain portion of its physical address space.
* It states that the logical addresses used by a process in any Long interval of time during its operation tend to be bunched differently in certain portion of its logical address space.

B. It states that the logical addresses used by a process in any short interval of time during its operation tend to be bunched together in certain portion of its logical address space.

Explanation : Process exhibits this behavior for two reasons; Execution of instructions in the process is mostly sequential in nature, because only 10-20 percent of instructions executed by a process are branch instructions.

3

. The computer exploits the law of locality to ensure\_\_\_\_\_\_\_\_\_\_

* More page faults and high hit ratio in cache
* Fewer page faults and high hit ratio in the disk
* High hit ratio in the cache and fewer page faults
* None of the above.

C. High hit ratio in the cache and fewer page faults

4

. How much memory should the virtual memory manager allocates to a process depends on which of the below factors?

* An overcommitment of memory to a process implies a low page fault rate for the process;hence it ensures good process performance.
* An undercommitment of memory to a process causes a high page fault rate,which would lead to poor performance of the process
* In An overcommitment of memory to a process however if a smaller number of processes would fit in memory which would causes CPU idling and poor system performance.
* All of the above.

D. All of the above.

5

. When a process is operating in low memory allocation and has high page fault rate, this process spends a lot its time in the \_\_\_\_\_\_\_\_\_\_\_\_

* Active state
* Pending State
* Blocked State
* Ready State

C. Blocked State

6

. A situation where All the processes in the system operate in the region of high page fault rates, the CPU would be engaged in performing page traffic and process switching most of the time and throughput would be poor Called\_\_\_\_\_\_\_\_

* Swapping
* Switching
* Paging
* Thrashing

D. Thrashing

7

. A situation where too few processes exists in memory or all process in memory perform I/O operations frequently is called \_\_\_\_\_\_\_

* Swapping
* Context Switching
* Paging
* Thrashing

D. Thrashing

Explanation : this situation is different in that all processes make poor progress because of high page fault rates.

8

. Which of the following Statement stands true for the optimal page size of process? It determines\_\_\_\_\_\_

1. the number of bits required to represent the byte number in a page
2. Memory wastage due to internal fragmentation
3. Size of page table for a process
4. Page Fault rates when a fixed amount of memory is allocated to a process

* 1 Only
* 1 and 2 only
* 1,2 and 3 only
* All of the above.

D. All of the above.

9

. Consider a process ***Pi*** of size ***z*** A page size of ***s*** bytes implies that the process has an ***n*** pages, where \_\_\_\_\_\_

* n = [z/s]
* n = [s/z]
* n = [zs]
* n = [z - s]

A. n = [z/s]

10

. Consider a process ***Pi*** of size ***z*** bytes. A page size of ***s*** bytes implies that the process has an ***n*** pages, where *n = [z/s]*.An Average internal fragmentation is \_\_\_\_\_\_\_\_\_\_

* s bytes
* s/2 bytes
* 1/s bytes
* 2s bytes

B. s/2 bytes

11

. The MMU contains special register called\_\_\_\_\_\_\_\_\_\_ to point to the start of a page table

* Page address resistor (PAR)
* Page Frame Address resistor (PFAR)
* Page table Address resistor (PTAR)
* **Page table size Resistor (PTSR)**

C. Page table Address resistor (PTAR)

Explanation : for a logical address (*pi, bi*) , the MMU computes <PTAR> +  *pi*× *lPT\_entry* to obtain the address of the page table entry of page *pi,*where *lPT\_entry* is the length of the page table entry and <PTAR> denotes the content of the PTAR. The PTAR has to be loaded with the correct address when a process is scheduled.

12

. Kernel can store the address of the page table of a process in its \_\_\_\_\_\_\_\_\_

* Relocation Register
* Process Control block
* Stack
* Heap

B. Process Control block

13

. The MMU provides a special register called the \_\_\_\_\_\_ if a process tries to access a nonexistent page, or exceeds its access privileges while accessing a page. and a memory protection interrupt raised.

* relocation register
* page table address register
* page table size register
* Process control block

C. page table size register

14

. Which of the following is the function of paging hardware

* Memory protection
* Efficient address translation
* Page replacement support
* All of the above.

D. All of the above.

15

. The \_\_\_\_\_\_\_\_ is a small and fast associative memory that is used to eliminate the reference to the page table.

* PCB
* VM Manager
* Free frame list
* TLB

D. TLB

Explanation : TLB translation look-aside buffer is a small and fast associative memory that speeds up address translation

16

. The TLB contains entries of the form \_\_\_\_\_\_\_\_ for a few recently accessed pages of a program that are in memory.

* Valid bit, page frame #, Prot info
* Page # , Page frame # , Prot info
* Valid bit, page frame #, Page #
* Page # , Page frame # , Valid bit

B. Page # , Page frame # , Prot info

Explanation : During address translation of a logical address (*pi,bi*) , the TLB hardware searches for an entry of page pi, if an entry is found, the page frame # from the entry is used to complete address translation for the logical address (*pi,bi*).

17

. A probability that a page exists in memory is known as \_\_\_\_\_\_\_

* TLB hit ratio
* TLB miss ratio
* Memory hit ratio
* Memory miss ratio

C. Memory hit ratio

18

. A probability that a page exists in TLB is known as \_\_\_\_\_\_\_

* TLB hit ratio
* TLB miss ratio
* Memory hit ratio
* Memory miss ratio

A. TLB hit ratio

19

. When TLB is used for a address translation, effective memory access time is \_\_\_\_\_\_ Where *Pr1* = Probability that page exists in memory *Pr2* = Probability that page exists in TLB *tmem* = memory access time *tTLB* = access time of TLB *tpfh* = time overhead of page fault handling

* Pr2 × (tTLB + tmem ) + (Pr1 - Pr2) × (tTLB + 2 × tmem) - (1 - Pr1) × (tTLB + tmem + tpfh + tTLB + 2 × tmem)
* Pr2 × (tTLB + tmem ) - (Pr1 - Pr2) × (tTLB + 2 × tmem) - (1 - Pr1) × (tTLB + tmem + tpfh + tTLB + 2 × tmem)
* Pr2 × (tTLB + tmem ) + (Pr1 - Pr2) × (tTLB + 2 × tmem) + (1 - Pr1) × (tTLB + tmem + tpfh + tTLB + 2 × tmem)
* Pr2 × (tTLB - tmem ) + (Pr1 + Pr2) × (tTLB - 2 × tmem) + (1 - Pr1) × (tTLB - tmem + tpfh + tTLB + 2 × tmem)

C. Pr2 × (tTLB + tmem ) + (Pr1 - Pr2) × (tTLB + 2 × tmem) + (1 - Pr1) × (tTLB + tmem + tpfh + tTLB + 2 × tmem)

20

. Which of the following is used to reduce the size of memory committed to page tables?

* Inverted page table
* Single level page table
* Multilevel page table
* Both a and c.

D. Both a and c.

21

. In which of the following the page table of a process is itself is paged; the entire page table therefore does not need to exist in memory at any time?

* Inverted page table
* Single level page table
* Multilevel page table
* Multiprogramming page table

C. Multilevel page table

Explanation : A higher level page table is used to access pages of the page table. if the higher level page table is large, it could itself be paged and so on. In this organization, the page table entry of a page has to be accessed through relevant entries of the higher level page tables.

22

. In which of the following the size of page table is governed by the size of memory, so it is independent of the number and size of processes. However information about a page cannot be accessed directly as in page table?

* Inverted page table
* Single level page table
* Multilevel page table
* Multiprogramming page table

A. Inverted page table

Explanation : the Inverted page table (IPT) has one entry for each page frame in memory that indicates which page, if any occupies the page frame; the table got this name because the information in it is the ‘inverse’ of the information in a page table.

1

. Page replacement policy should replace a page that is \_\_\_\_\_

* Likely to be referenced in the immediate future
* Not Likely to be referenced in the immediate future
* Currently in use by the process
* None of the above.

B. Not Likely to be referenced in the immediate future

2

. In which of the following Page replacement Algorithm making pages replacement decisions in such a manner that the total number of page faults during operation of a process is the minimum possible?

* LRU Page replacement Algorithm
* FIFO page replacement Algorithm
* Optimal page replacement algorithm
* NRU Page replacement algorithm

C. Optimal page replacement algorithm

3

. In which of the following Page replacement Policy, at every page fault the page replacement policy replaces the page that was loaded into memory earlier than any other page of the process?

* LRU Page replacement Algorithm
* FIFO page replacement Algorithm
* Optimal page replacement algorithm
* NRU Page replacement algorithm

B. FIFO page replacement Algorithm

4

. To facilitate FIFO Page replacement, the virtual memory manager records the time of loading of a page in the \_\_\_\_\_\_\_\_\_ field of its page table entry.

* Prot info
* Valid bit
* Ref info
* None of the above.

C. Ref info

5

. To facilitate Optimal Page replacement, the virtual memory manager records the time of loading of a page in the \_\_\_\_\_\_\_\_\_ field of its page table entry.

* Prot info
* Valid bit
* Ref info
* None of the above.

C. Ref info

6

. To facilitate LRU Page replacement, the virtual memory manager records the time of loading of a page in the \_\_\_\_\_\_\_\_\_ field of its page table entry.

* Prot info
* Valid bit
* Ref info
* None of the above.

C. Ref info

7

. In which of the following Page replacement Policy uses the law of locality of reference as the basis for its replacement decisions?

* LRU Page replacement Algorithm
* FIFO page replacement Algorithm
* Optimal page replacement algorithm
* NRU Page replacement algorithm

A. LRU Page replacement Algorithm

Explanation : At every page fault the least recently used (LRU) page is replaced by the required page. The page table entry of a page records the time when the page was last referenced. This information is initialized when page is loaded, and it is updated every time the page is referenced. When page fault occurs, this information is used to locate the page pLRUwhose last reference is earlier than that of every other page. This page is replaced with the page whose reference led to the page fault.

8

. To achieve the desirable page fault characteristic , a page replacement policy must possess the \_\_\_\_\_\_\_\_\_

* Heap property
* Array Property
* Stack Property
* All of the above.

C. Stack Property

9

. Which of the following page replacement algorithm exhibit the stack property ?

* LRU Page replacement Algorithm
* FIFO page replacement Algorithm
* Optimal page replacement algorithm
* NRU Page replacement algorithm

A. LRU Page replacement Algorithm

10

. Belady’s Anomaly occurs in which of the following page replacement algorithm?

* LRU Page replacement Algorithm
* FIFO page replacement Algorithm
* Optimal page replacement algorithm
* NRU Page replacement algorithm

B. FIFO page replacement Algorithm

11

. Which of the following statement stands true for Belady’s anomaly?

* The number of page fault decreases when memory allocation for the process is increased
* The number of page fault increases when memory allocation for the process is decreased
* The number of page fault increases when memory allocation for the process is increased
* None of the above.

C. The number of page fault increases when memory allocation for the process is increased

12

. In which of the following algorithm it is possible to combat *thrashing* by increasing the value of *alloc*for each process?

* LRU Page replacement Algorithm
* FIFO page replacement Algorithm
* Optimal page replacement algorithm
* NRU Page replacement algorithm

A. LRU Page replacement Algorithm

Explanation : The virtual memory manager cannot use FIFO page replacement policy because increasing the allocation to a process may increase the page fault frequency of the process. This feature would make it difficult to combat thrashing in the system. However, when LRU page replacement is used, the number of page fault is nonincreasing function of *alloc*. Hence it is possible to combat thrashing by increasing the value of *alloc* for each process.

13

. In which of the following Algorithm where the reference bit is used to determine whether a page has been recently referenced, and some page that has not been recently referenced is replaced?

* LRU Page replacement Algorithm
* FIFO page replacement Algorithm
* Optimal page replacement algorithm
* NRU Page replacement algorithm

D. NRU Page replacement algorithm

Explanation : The reference bit of a page is initialized to 0 when the page is loaded, and it is set to 1 when page is referenced. it replaces a page whose reference bit is 0.When virtual memory manager finds that the reference bits of all pages have become 1, it reset the bits of all pages to 0 and arbitrarily selects one of the pages for replacement.

. Which of the following is a named collection of related information that is recorded on secondary storage and is the smallest allotment of logical secondary storage?

* Directory
* File
* Disk
* All of the above

B. File

Explanation : Data cannot be written to secondary storage unless they are within a file.

2

. In Which of the following is a sequence of characters organized into lines (and possibly pages) ?

* Text file
* Source file
* Executable file
* None of the above.

A. Text file

3

. In which of the following a sequence of functions, each of which is further organized as declarations followed by executable statements is stored?

* Text file
* Source file
* Executable file
* All of the above.

B. Source file

4

. In which of the following file is a series of code sections that the loader can bring into memory and execute?

* Text file
* Source file
* Executable file
* Distributed file

C. Executable file

5

. A file is named, for the convenience of its human users, and is referred to by its \_\_\_\_\_\_

* Name
* Type
* Identifier
* size

A. Name

6

. Which of the following file attribute is the unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file?

* Name
* Identifier
* Size
* Location

B. Identifier

7

. Which of the following file attribute information is needed for systems that support different types of files?

* Name
* Type
* Identifier
* size

B. Type

8

. Which of the following file attribute a information is a pointer to a device and to the location of the file on that device?

* Name
* Size
* Time, date, and user identification.
* Location

D. Location

9

. Which of the following file attribute, the current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included?

* Protection
* Size
* Identifier
* Type

B. Size

10

. Which of the following file attribute Access-control information determines who can do reading, writing, executing, and so on?

* Protection
* identifier
* Type
* Time, date, and user identification

A. Protection

11

. The information about all files is kept in the \_\_\_\_\_\_\_\_ which also resides on secondary storage.

* Disk structure
* Directory structure
* Array structure
* All of the above

B. Directory structure

Explanation : The information about all files is kept in the directory structure, which also resides on secondary storage. Typically, a directory entry consists of the file’s name and its unique identifier. The identifier in turn locates the other file attributes.

12

. Which of following basic operations that can be performed on files by the operating System’s system calls?

* Read, Write, Delete
* Write, Paint, reposition
* Delete, truncate files, Sorting
* All of the above.

A. Read, Write, Delete

Explanation : The operating system can provide system calls to create, write, read, reposition, delete, and truncate files.

13

. Which of the following steps are necessary to create a file?

* space in the file system must be found for the file
* an entry for the new file must be made in the directory
* space in the file system must be found for the file and an entry for the new file must be made in the directory
* None of the above.

C. space in the file system must be found for the file and an entry for the new file must be made in the directory

14

. Which of the following file operation is also known as file seek?

* Truncating a file
* Deleting a file
* Repositioning within a file
* Reading a file

C. Repositioning within a file

Explanation : The directory is searched for the appropriate entry, and the current-file-position pointer is re-positioned to a given value. Repositioning within a file need not involve any actual I/O. This file operation is also known as a file **seek**

15

. Which of the following Statements stands true for **creating a file**?

* We search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.
* The directory is searched for the appropriate entry, and the current-file-position pointer is re-positioned to a given value.
* Space in the file system must be found for the file and an entry for the new file must be made in the directory.
* We use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place.

C. Space in the file system must be found for the file and an entry for the new file must be made in the directory.

16

. Which of the following Statements stands true for **writing a file**?

* Given the name of the file, the system searches the directory to find the file’s location. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.
* The directory is searched for the appropriate entry, and the current-file-position pointer is re positioned to a given value.
* space in the file system must be found for the file and an entry for the new file must be made in the directory.
* We use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place.

A. Given the name of the file, the system searches the directory to find the file’s location. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

17

. Which of the following Statements stands true for **Reading a file**?

* Given the name of the file, the system searches the directory to find the file’s location. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.
* We search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.
* space in the file system must be found for the file and an entry for the new file must be made in the directory.
* We use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place.

D. We use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place.

18

. Which of the following Statements stands true for **deleting a file?**

* Given the name of the file, the system searches the directory to find the file’s location. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.
* We search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.
* Space in the file system must be found for the file and an entry for the new file must be made in the directory.
* We use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a delete pointer to the location in the file where the next read is to take place.

B. We search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.

19

. Which of the following Statements stands true for **truncating a file**?

* We search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.
* The directory is searched for the appropriate entry, and the current-file-position pointer is re-positioned to a given value.
* Space in the file system must be found for the file and an entry for the new file must be made in the (directory).
* The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged—except for file length—but lets the file be reset to length zero and its file space released.

D. The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged—except for file length—but lets the file be reset to length zero and its file space released.

20

. The operating system keeps a table, called the **\_\_\_\_\_\_\_**, containing information about all open files.

* per-process table
* system-wide table
* Open-file table
* A and C

D. A and C

Explanation : When a file operation is requested, the file is specified via an index into this table, so no searching is required. When the file is no longer being actively used, it is closed by the process, and the operating system removes its entry from the open-file table. Typically, the operating system uses two levels of internal tables: a per-process table and a system-wide table. The per-process table tracks all files that a process has open. Stored in this table is information regarding the process’s use of the file.

21

. The \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_needed to locate the file on disk is kept in memory so that the system does not have to read it from disk for each operation.

* File pointer
* File-open count
* Disk location of the file.
* Access rights

C. Disk location of the file.

22

. This \_\_\_\_\_\_\_\_ is stored on the per-process table so the operating system can allow or deny subsequent I/O requests.

* File pointer
* File-open count
* Disk location of the file.
* Access rights

D. Access rights

23

. \_\_\_\_\_\_ tracks the number of opens and closes and reaches zero on the last close.The system can then remove the entry.

* File pointer
* File-open count
* Disk location of the file.
* Access rights

B. File-open count

Explanation : As files are closed, the operating system must reuse its open-file table entries, or it could run out of space in the table. Multiple processes may have opened a file, and the system must wait for the last file to close before removing the open-file table entry. The file-open count tracks the number of opens and closes and reaches zero on the last close. The system can then remove the entry.

24

. \_\_\_\_\_\_ is unique to each process operating on the file and therefore must be kept separate from the on-disk file attributes.

* File pointer
* File-open count
* Disk location of the file.
* Access rights

A. File pointer

Explanation : On systems that do not include a file offset as part of the read() and write() system calls, the system must track the last read– write location as a current-file-position pointer. This pointer is unique to each process operating on the file and therefore must be kept separate from the on-disk file attributes

25

. A **\_\_\_\_\_**is akin to a reader lock in that several processes can acquire the lock concurrently.

* Exclusive file lock
* Shared file lock
* Inclusive lock
* Mandatory lock

B. Shared file lock

26

. An **\_\_\_\_\_\_\_\_\_\_**behaves like a writer lock;only one process at a time can acquire such a lock.

* Exclusive file lock
* Shared file lock
* Inclusive file lock
* Mandatory file lock

A. Exclusive file lock

27

. If a lock is \_\_\_\_\_\_, then once a process acquires an exclusive lock, the operating system will prevent any other process from accessing the locked file

* Exclusive file lock
* Shared file lock
* Inclusive file lock
* Mandatory file lock

D. Mandatory file lock

28

. The system uses the \_\_\_\_\_\_\_ to indicate the type of the file and the type of operations that can be done on that file.

* Size
* Location
* Date time
* Extension

D. Extension

29

. A file with extension\_\_\_\_\_\_ is the binary executable file.

* .txt
* .docx
* .sh
* .exe

D. .exe

30

. Which of the following file with extension \_\_\_\_\_\_is the binary executable file?

* .txt
* .docx
* .sh
* .com

D. .com

31

. \_\_\_\_\_\_\_\_\_ is a **shell script**containing, in ASCII format, commands to the operating system.

* .txt
* .docx
* .sh
* .com

C. .sh

32

. binary file containing audio or A/V information

* exe, com, bin or none
* c, cc, java, perl, asm
* xml, html, tex
* mpeg, mov, mp3, mp4, avi

D. mpeg, mov, mp3, mp4, avi

33

. related files grouped into one file, sometimes compressed, for archiving or storage

* exe, com, bin or none
* rar, zip, tar
* xml, html, tex
* mpeg, mov, mp3, mp4, avi

B. rar, zip, tar

34

. libraries of routines for programmers

* exe, com, bin or none
* rar, zip, tar
* xml, html, tex
* lib, a, so, dll

D. lib, a, so, dll

35

. which of the following are various word-processor formats?

* exe, com, bin or none
* rar, zip, tar
* xml, html, tex
* xml, rtf, docx

D. xml, rtf, docx

36

. Which of the following has the file type of markup?

* exe, com, bin or none
* rar, zip, tar
* xml, html, tex
* xml, rtf, docx

C. xml, html, tex

37

. Which of the following is known as batch file?

* exe, com, bin or none
* obj, o
* lib, a, so, dll
* bat, sh

D. bat, sh

38

. Which of the following file extension is for the file containing source code in various languages?

* exe, com, bin or none
* obj, o
* lib, a, so, dll
* c, cc, java, perl

D. c, cc, java, perl

39

. \_\_\_\_\_\_\_\_file extension has the function of compiled, machine language, not linked.

* exe, com, bin or none
* obj, o
* lib, a, so, dll
* c, cc, java, perl

B. obj, o

40

. \_\_\_\_\_\_\_\_file extension contain ready-to-run machine language program.

* exe, com, bin or none
* obj, o
* lib, a, so, dll
* c, cc, java, perl

A. exe, com, bin or none