[](https://www.studocu.com/vn?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=os-mcqs-os-mcqs)

[OS-MCQs - OS-MCQs](https://www.studocu.com/vn/document/truong-dai-hoc-bach-khoa-ha-noi/operating-systems/os-mcqs-os-mcqs/48967550?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=os-mcqs-os-mcqs)

[Operating Systems (Trường Đại học Bách khoa Hà Nội)](https://www.studocu.com/vn/course/truong-dai-hoc-bach-khoa-ha-noi/operating-systems/3260070?utm_campaign=shared-document&utm_source=studocu-document&utm_medium=social_sharing&utm_content=os-mcqs-os-mcqs)



Scan to open on Studocu

Studocu is not sponsored or endorsed by any college or university

# Processes Process

* 1. What is an operating system?
     1. collection of programs that manages hardware resources
     2. system service provider to the application programs
     3. interface between the hardware and application programs
     4. all of the mentioned
  2. To access the services of operating system, the interface is provided by the
     1. System calls
     2. API
     3. Library
     4. Assembly instructions
  3. Which one of the following is not true?
     1. kernel is the program that constitutes the central core of the operating system
     2. kernel is the first part of operating system to load into memory during booting
     3. kernel is made of various modules which can not be

loaded in running operating system

* + 1. kernel remains in the memory during the entire computer session

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

* 1. Which one of the following error will be handle by the operating system?
     1. power failure
     2. lack of paper in printer
     3. connection failure in the network
     4. all of the mentioned
  2. What is the main function of the command interpreter?
     1. to get and execute the next user-specified command
     2. to provide the interface between the API and application program
     3. to handle the files in operating system
     4. none of the mentioned

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

* 1. In Operating Systems, which of the following is/are CPU scheduling algorithms?
     1. Round Robin
     2. Shortest Job First
     3. Priority
     4. All of the mentioned
  2. If a process fails, most operating system write the error information to a
     1. log file
     2. another running process
     3. new file
     4. none of the mentioned

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

* 1. Which facility dynamically adds probes to a running system, both in user processes and in the kernel?
     1. DTrace
     2. DLocate
     3. DMap
     4. DAdd

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

* 1. Which one of the following is not a real time operating system?
     1. VxWorks
     2. QNX
     3. RTLinux
     4. Palm OS

*Explanation:* *Palm* *OS* *is* *a* *mobile* *operating* *system.* *Palm* *OS* *is* *developed* *for* *Personal* *Digital* *Assistants* *(PDAs).*

* 1. The OS X has
     1. monolithic kernel
     2. hybrid kernel
     3. microkernel
     4. monolithic kernel with modules

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

1. The systems which allow only one process execution at a time, are called
   1. uniprogramming systems
   2. uniprocessing systems
   3. unitasking systems
   4. none of the mentioned
2. In operating system, each process has its own
   1. address space and global variables
   2. open files
   3. pending alarms, signals and signal handlers
   4. all of the mentioned

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

1. In Unix, Which system call creates the new process?
   1. fork
   2. create
   3. new
   4. none of the mentioned
2. A process can be terminated due to
   1. normal exit
   2. fatal error
   3. killed by another process
   4. all of the mentioned
3. What is the ready state of a process?
   1. when process is scheduled to run after some execution
   2. when process is unable to run until some task has been completed
   3. when process is using the CPU
   4. none of the mentioned
4. What is interprocess communication?
   1. communication within the process
   2. communication between two process
   3. communication between two threads of same process
   4. none of the mentioned

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

1. A set of processes is deadlock if
   1. each process is blocked and will remain so forever
   2. each process is terminated
   3. all processes are trying to kill each other
   4. none of the mentioned

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

1. A process stack does not contain
   1. Function parameters
   2. Local variables
   3. Return addresses
   4. PID of child process
2. Which system call can be used by a parent process to determine the termination of child process?
   1. wait
   2. exit
   3. fork
   4. get
3. The address of the next instruction to be executed by the current process is provided by the
   1. CPU registers
   2. Program counter
   3. Process stack
   4. Pipe

# Process Control Block

1. A Process Control Block(PCB) does not contain which of the following?
   1. Code
   2. Stack
   3. Bootstrap program
   4. Data
2. The number of processes completed per unit time is known as
   1. Output
   2. Throughput
   3. Efficiency
   4. Capacity
3. The state of a process is defined by
   1. the final activity of the process
   2. the activity just executed by the process
   3. the activity to next be executed by the process
   4. the current activity of the process
4. Which of the following is not the state of a process?
   1. New
   2. Old
   3. Waiting
   4. Running
5. What is a Process Control Block?
   1. Process type variable
   2. Data Structure
   3. A secondary storage section
   4. A Block in memory

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

1. The entry of all the PCBs of the current processes is in
   1. Process Register
   2. Program Counter
   3. Process Table
   4. Process Unit
2. What is the degree of multiprogramming?
   1. the number of processes executed per unit time
   2. the number of processes in the ready queue
   3. the number of processes in the I/O queue
   4. the number of processes in memory
3. A single thread of control allows the process to perform
   1. only one task at a time
   2. multiple tasks at a time
   3. only two tasks at a time
   4. all of the mentioned
4. What is the objective of multiprogramming?
   1. Have a process running at all time
   2. Have multiple programs waiting in a queue ready to run
   3. To increase CPU utilization
   4. None of the mentioned

# Process Scheduling Queues

1. Which of the following do not belong to queues for processes?
   1. Job Queue
   2. PCB queue
   3. Device Queue
   4. Ready Queue
2. When the process issues an I/O request
   1. It is placed in an I/O queue
   2. It is placed in a waiting queue
   3. It is placed in the ready queue
   4. It is placed in the Job queue
3. What will happen when a process terminates?
   1. It is removed from all queues
   2. It is removed from all, but the job queue
   3. Its process control block is de-allocated
   4. Its process control block is never de-allocated
4. What is a long-term scheduler?
   1. It selects processes which have to be brought into the ready queue
   2. It selects processes which have to be executed next and allocates CPU
   3. It selects processes which heave to remove from memory by swapping
   4. None of the mentioned

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

1. If all processes I/O bound, the ready queue will almost always be and the Short-term Scheduler will have a

to do.

* 1. full, little
  2. full, lot
  3. empty, little
  4. empty, lot

*Explanation:* *I/O* *bound* *processes* *spend* *more* *time* *doing* *I/O* *than* *computation.*

1. What is a medium-term scheduler?
   1. It selects which process has to be brought into the ready queue
   2. It selects which process has to be executed next and allocates CPU
   3. It selects which process to remove from memory by

swapping

* 1. None of the mentioned

*Explanation:* *The* *medium-term* *scheduler* *swapped* *out* *the* *process* *and* *later* *swapped* *in.* *Swapping* *helps* *to* *free* *up* *memory.*

1. What is a short-term scheduler?
   1. It selects which process has to be brought into the ready queue
   2. It selects which process has to be executed next and

allocates CPU

* 1. It selects which process to remove from memory by swapping
  2. None of the mentioned

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

1. The primary distinction between the short term scheduler and the long term scheduler is
   1. The length of their queues
   2. The type of processes they schedule
   3. The frequency of their execution
   4. None of the mentioned
2. The only state transition that is initiated by the user process itself is
   1. block
   2. wakeup
   3. dispatch
   4. none of the mentioned
3. 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
   1. Blocked state
   2. Ready state
   3. Suspended state
   4. Terminated state

*Explanation:* *In* *a* *time-sharing* *operating* *system,* *when* *the* *time* *slot* *given* *to* *a* *process* *is* *completed,* *the* *process* *goes* *from* *the* *running* *state* *to* *the* *Ready* *State.* *In* *a* *time-sharing* *operating* *system* *unit* *time* *is* *defined* *for* *sharing* *CPU,* *it* *is* *called* *a* *time* *quantum* *or* *time* *slice.* *If* *a* *process* *takes* *less* *than* *1* *time* *quantum,* *then* *the* *process* *itself* *releases* *the* *CPU.*

1. In a multiprogramming environment
   1. the processor executes more than one process at a time
   2. the programs are developed by more than one person
   3. more than one process resides in the memory
   4. a single user can execute many programs at the same time
2. Suppose that a process is in “Blocked” state waiting for some I/O service. When the service is completed, it goes to the
   1. Running state
   2. Ready state
   3. Suspended state
   4. Terminated state
3. The context of a process in the PCB of a process does not contain
   1. the value of the CPU registers
   2. the process state
   3. memory-management information
   4. context switch time
4. Which of the following need not necessarily be saved on a context switch between processes?
   1. General purpose registers
   2. Translation lookaside buffer
   3. Program counter
   4. All of the mentioned

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

1. Which of the following does not interrupt a running process?
   1. A device
   2. Timer
   3. Scheduler process
   4. Power failure

*Explanation:* *Scheduler* *process* *selects* *an* *available* *process* *from* *a* *pool* *of* *available* *processes* *and* *allocates* *CPU* *to* *it.*

# Process Synchronization

1. Which process can be affected by other processes executing in the system?
   1. cooperating process
   2. child process
   3. parent process
   4. init process

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

1. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place is called
   1. dynamic condition
   2. race condition
   3. essential condition
   4. critical condition
2. If a process is executing in its critical section, then no other processes can be executing in their critical section. What is this condition called?
   1. mutual exclusion
   2. critical exclusion
   3. synchronous exclusion
   4. asynchronous exclusion
3. Which one of the following is a synchronization tool?
   1. thread
   2. pipe
   3. semaphore
   4. socket
4. A semaphore is a shared integer variable
   1. that can not drop below zero
   2. that can not be more than zero
   3. that can not drop below one
   4. that can not be more than one

*Explanation:* *In* *binary* *semaphore,* *if* *the* *value* *of* *the* *semaphore* *variable* *is* *zero* *that* *means* *there* *is* *a* *process* *that* *uses* *a* *critical* *resource* *and* *no* *other* *process* *can* *access* *the* *same* *critical* *resource* *until* *it* *is* *released.* *In* *Counting* *semaphore,* *if* *the* *value* *of* *the* *semaphore* *variable* *is* *zero* *that* *means* *there* *is* *no* *resource* *available.*

1. Mutual exclusion can be provided by the
   1. mutex locks
   2. binary semaphores
   3. both mutex locks and binary semaphores
   4. none of the mentioned

*Explanation:* *Mutex* *is* *a* *short* *form* *of* ***Mut****ual* ***Ex****clusion.* *Binary* *semaphore* *also* *provides* *a* *mechanism* *for* *mutual* *exclusion.* *Binary* *semaphore* *behaves* *similar* *to* *mutex* *locks.*

1. When high priority task is indirectly preempted by medium priority task effectively inverting the relative priority of the two tasks, the scenario is called
   1. priority inversion
   2. priority removal
   3. priority exchange
   4. priority modification
2. Process synchronization can be done on
   1. hardware level
   2. software level
   3. both hardware and software level
   4. none of the mentioned

*Explanation:* *Critical* *section* *problems* *can* *be* *resolved* *using* *hardware* *synchronisation.* *But* *this* *method* *is* *not* *simple* *for* *implementation* *so* *software* *synchronization* *is* *mostly* *used.*

1. A monitor is a module that encapsulates
   1. shared data structures
   2. procedures that operate on shared data structure
   3. synchronization between concurrent procedure invocation
   4. all of the mentioned
2. To enable a process to wait within the monitor
   1. a condition variable must be declared as condition
   2. condition variables must be used as boolean objects
   3. semaphore must be used
   4. all of the mentioned

Process Creation

1. Restricting the child process to a subset of the parent’s resources prevents any process from
   1. overloading the system by using a lot of secondary storage
   2. under-loading the system by very less CPU utilization
   3. overloading the system by creating a lot of sub- processes
   4. crashing the system by utilizing multiple resources
2. A parent process calling system call will be suspended until children processes terminate.
   1. wait
   2. fork
   3. exit
   4. exec

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

1. Cascading termination refers to termination of all child processes if the parent process terminates
   1. Normally
   2. Abnormally
   3. Normally or abnormally
   4. None of the mentioned
2. With only one process can execute at a time; meanwhile all other process are waiting for the processor. With more than one process can be running simultaneously each on a different processor.
   1. Multiprocessing, Multiprogramming
   2. Multiprogramming, Uniprocessing
   3. Multiprogramming, Multiprocessing
   4. Uniprogramming, Multiprocessing
3. In UNIX, each process is identified by its
   1. Process Control Block
   2. Device Queue
   3. Process Identifier
   4. None of the mentioned
4. In UNIX, the return value for the fork system call is for the child process and for the parent process.
   1. A Negative integer, Zero
   2. Zero, A Negative integer
   3. Zero, A nonzero integer
   4. A nonzero integer, Zero
5. The child process can
   1. be a duplicate of the parent process
   2. never be a duplicate of the parent process
   3. cannot have another program loaded into it
   4. never have another program loaded into it
6. The child process completes execution, but the parent keeps executing, then the child process is known as
   1. Orphan
   2. Zombie
   3. Body
   4. Dead

# Inter Process Communication

1. What is Interprocess communication?
   1. allows processes to communicate and synchronize their actions when using the same address space
   2. allows processes to communicate and synchronize

their actions

* 1. allows the processes to only synchronize their actions without communication
  2. none of the mentioned

1. Message passing system allows processes to
   1. communicate with each other without sharing the same address space
   2. communicate with one another by resorting to shared data
   3. share data
   4. name the recipient or sender of the message
2. Which of the following two operations are provided by the IPC facility?
   1. write & delete message
   2. delete & receive message
   3. send & delete message
   4. receive & send message
3. Messages sent by a process
   1. have to be of a fixed size
   2. have to be a variable size
   3. can be fixed or variable sized
   4. none of the mentioned

*Explanation:* *If* *the* *message* *size* *of* *the* *process* *is* *fixed* *then* *system* *level* *implementation* *is* *straightforward* *but* *it* *makes* *the* *task* *of* *programming* *more* *difficult.* *If* *the* *message* *size* *of* *the* *process* *is* *variable* *then* *system* *level* *implementation* *is* *more* *complex* *but* *it* *makes* *the* *task* *of* *programming* *simpler.*

1. The link between two processes P and Q to send and receive messages is called
   1. communication link
   2. message-passing link
   3. synchronization link
   4. all of the mentioned
2. Which of the following are TRUE for direct communication?
   1. A communication link can be associated with N number of process (N = max. number of processes supported by system)
   2. A communication link is associated with exactly two

processes

* 1. Exactly N/2 links exist between each pair of processes(N = max. number of processes supported by system)
  2. Exactly two link exists between each pair of processes

1. In indirect communication between processes P and Q
   1. there is another process R to handle and pass on the messages between P and Q
   2. there is another machine between the two processes to help communication
   3. there is a mailbox to help communication between P

and Q

* 1. none of the mentioned

1. In the non blocking send
   1. the sending process keeps sending until the message is received
   2. the sending process sends the message and resumes

operation

* 1. the sending process keeps sending until it receives a message
  2. none of the mentioned

1. In the Zero capacity queue
   1. the queue can store at least one message
   2. the sender blocks until the receiver receives the message
   3. the sender keeps sending and the messages don’t wait in the queue
   4. none of the mentioned
2. The Zero Capacity queue
   1. is referred to as a message system with buffering
   2. is referred to as a message system with no buffering
   3. is referred to as a link
   4. none of the mentioned
3. Bounded capacity and Unbounded capacity queues are referred to as
   1. Programmed buffering
   2. Automatic buffering
   3. User defined buffering
   4. No buffering

# CPU Scheduling Concepts of CPU Scheduling

* 1. Which module gives control of the CPU to the process selected by the short-term scheduler?
     1. dispatcher
     2. interrupt
     3. scheduler
     4. none of the mentioned
  2. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called
     1. job queue
     2. ready queue
     3. execution queue
     4. process queue
  3. The interval from the time of submission of a process to the time of completion is termed as
     1. waiting time
     2. turnaround time
     3. response time
     4. throughput
  4. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?
     1. first-come, first-served scheduling
     2. shortest job scheduling
     3. priority scheduling
     4. none of the mentioned
  5. In priority scheduling algorithm
     1. CPU is allocated to the process with highest priority
     2. CPU is allocated to the process with lowest priority
     3. Equal priority processes can not be scheduled
     4. None of the mentioned
  6. In priority scheduling algorithm, when a process arrives at the ready queue, its priority is compared with the priority of
     1. all process
     2. currently running process
     3. parent process
     4. init process
  7. Which algorithm is defined in Time quantum?
     1. shortest job scheduling algorithm
     2. round robin scheduling algorithm
     3. priority scheduling algorithm
     4. multilevel queue scheduling algorithm
  8. Process are classified into different groups in
     1. shortest job scheduling algorithm
     2. round robin scheduling algorithm
     3. priority scheduling algorithm
     4. multilevel queue scheduling algorithm
  9. In multilevel feedback scheduling algorithm
     1. a process can move to a different classified ready queue
     2. classification of ready queue is permanent
     3. processes are not classified into groups
     4. none of the mentioned
  10. Which one of the following can not be scheduled by the kernel?
      1. kernel level thread
      2. user level thread
      3. process
      4. none of the mentioned

# CPU Scheduling Benefits

1. CPU scheduling is the basis of
   1. multiprocessor systems
   2. multiprogramming operating systems
   3. larger memory sized systems
   4. none of the mentioned
2. With multiprogramming is used productively.
   1. time
   2. space
   3. money
   4. all of the mentioned
3. What are the two steps of a process execution?
   1. I/O & OS Burst
   2. CPU & I/O Burst
   3. Memory & I/O Burst
   4. OS & Memory Burst
4. An I/O bound program will typically have
   1. a few very short CPU bursts
   2. many very short I/O bursts
   3. many very short CPU bursts
   4. a few very short I/O bursts
5. A process is selected from the queue by the

scheduler, to be executed.

* 1. blocked, short term
  2. wait, long term
  3. ready, short term
  4. ready, long term

1. In the following cases non – preemptive scheduling occurs?
   1. When a process switches from the running state to the ready state
   2. When a process goes from the running state to the

waiting state

* 1. When a process switches from the waiting state to the ready state
  2. All of the mentioned

1. The switching of the CPU from one process or thread to another is called
   1. process switch
   2. task switch
   3. context switch
   4. all of the mentioned
2. What is Dispatch latency?
   1. the speed of dispatching a process from running to the ready state
   2. the time of dispatching a process from running to ready state and keeping the CPU idle
   3. the time to stop one process and start running

another one

* 1. none of the mentioned

1. Scheduling is done so as to
   1. increase CPU utilization
   2. decrease CPU utilization
   3. keep the CPU more idle
   4. none of the mentioned
2. Scheduling is done so as to
   1. increase the throughput
   2. decrease the throughput
   3. increase the duration of a specific amount of work
   4. none of the mentioned
3. What is Turnaround time?
   1. the total waiting time for a process to finish execution
   2. the total time spent in the ready queue
   3. the total time spent in the running queue
   4. the total time from the completion till the submission of a process
4. Scheduling is done so as to
   1. increase the turnaround time
   2. decrease the turnaround time
   3. keep the turnaround time same
   4. there is no relation between scheduling and turnaround time
5. What is Waiting time?
   1. the total time in the blocked and waiting queues
   2. the total time spent in the ready queue
   3. the total time spent in the running queue
   4. the total time from the completion till the submission of a process
6. Scheduling is done so as to
   1. increase the waiting time
   2. keep the waiting time the same
   3. decrease the waiting time
   4. none of the mentioned
7. What is Response time?
   1. the total time taken from the submission time till the completion time
   2. the total time taken from the submission time till the first response is produced
   3. the total time taken from submission time till the response is output
   4. none of the mentioned

# CPU Scheduling Algorithm

1. Round robin scheduling falls under the category of
   1. Non-preemptive scheduling
   2. Preemptive scheduling
   3. All of the mentioned
   4. None of the mentioned
2. With round robin scheduling algorithm in a time shared system
   1. using very large time slices converts it into First come

First served scheduling algorithm

* 1. using very small time slices converts it into First come First served scheduling algorithm
  2. using extremely small time slices increases performance
  3. using very small time slices converts it into Shortest Job First algorithm

1. The portion of the process scheduler in an operating system that dispatches processes is concerned with
   1. assigning ready processes to CPU
   2. assigning ready processes to waiting queue
   3. assigning running processes to blocked queue
   4. all of the mentioned
2. Complex scheduling algorithms
   1. are very appropriate for very large computers
   2. use minimal resources
   3. use many resources
   4. all of the mentioned
3. What is FIFO algorithm?
   1. first executes the job that came in last in the queue
   2. first executes the job that came in first in the queue
   3. first executes the job that needs minimal processor
   4. first executes the job that has maximum processor needs
4. The strategy of making processes that are logically runnable to be temporarily suspended is called
   1. Non preemptive scheduling
   2. Preemptive scheduling
   3. Shortest job first
   4. First come First served
5. What is Scheduling?
   1. allowing a job to use the processor
   2. making proper use of processor
   3. all of the mentioned
   4. none of the mentioned
6. There are 10 different processes running on a workstation. Idle processes are waiting for an input event in the input queue. Busy processes are scheduled with the Round-Robin time sharing method. Which out of the following quantum times is the best value for small response times, if the processes have a short runtime, e.g. less than 10ms?
   1. tQ = 15ms
   2. tQ = 40ms
   3. tQ = 45ms
   4. tQ = 50ms
7. Orders are processed in the sequence they arrive if

rule sequences the jobs.

* 1. earliest due date
  2. slack time remaining
  3. first come, first served
  4. critical ratio

1. Which of the following algorithms tends to minimize the process flow time?
   1. First come First served
   2. Shortest Job First
   3. Earliest Deadline First
   4. Longest Job First
2. Under multiprogramming, turnaround time for short jobs is usually and that for long jobs is slightly
   1. Lengthened; Shortened
   2. Shortened; Lengthened
   3. Shortened; Shortened
   4. Shortened; Unchanged
3. Which of the following statements are true? (GATE 2010)
4. Shortest remaining time first scheduling may cause starvation
5. Preemptive scheduling may cause starvation
6. Round robin is better than FCFS in terms of response time
   1. I only
   2. I and III only
   3. II and III only
   4. I, II and III

*Explanation:*

* + 1. *Shortest* *remaining* *time* *first* *scheduling* *is* *a* *preemptive* *version* *of* *shortest* *job* *scheduling.* *It* *may* *cause* *starvation* *as* *shorter* *processes* *may* *keep* *coming* *and* *a* *long* *CPU* *burst* *process* *never* *gets* *CPU.*
    2. *Preemption* *may* *cause* *starvation.* *If* *priority* *based* *scheduling* *with* *preemption* *is* *used,* *then* *a* *low* *priority* *process* *may* *never* *get* *CPU.*
    3. *Round* *Robin* *Scheduling* *improves* *response* *time* *as* *all* *processes* *get* *CPU* *after* *a* *specified* *time.*
       1. Which is the most optimal scheduling algorithm?
          1. FCFS – First come First served
          2. SJF – Shortest Job First
          3. RR – Round Robin
          4. None of the mentioned
       2. The real difficulty with SJF in short term scheduling is
          1. it is too good an algorithm
          2. knowing the length of the next CPU request
          3. it is too complex to understand
          4. none of the mentioned
       3. The FCFS algorithm is particularly troublesome for
          1. time sharing systems
          2. multiprogramming systems
          3. multiprocessor systems
          4. operating systems

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

* + - 1. Consider the following set of processes, the length of the CPU burst time given in milliseconds.

|  |  |
| --- | --- |
| Process | Burst Time |
| P1 | 6 |
| P2 | 7 |
| P3 | 8 |
| P4 | 3 |

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

* + - * 1. The waiting time for process P1 is 3ms
        2. The waiting time for process P1 is 0ms
        3. The waiting time for process P1 is 16ms
        4. The waiting time for process P1 is 9ms
      1. Preemptive Shortest Job First scheduling is sometimes called
         1. Fast SJF scheduling
         2. EDF scheduling – Earliest Deadline First
         3. HRRN scheduling – Highest Response Ratio Next
         4. SRTN scheduling – Shortest Remaining Time Next
      2. An SJF algorithm is simply a priority algorithm where the priority is
         1. the predicted next CPU burst
         2. the inverse of the predicted next CPU burst
         3. the current CPU burst
         4. anything the user wants
      3. Choose one of the disadvantages of the priority scheduling algorithm?
         1. it schedules in a very complex manner
         2. its scheduling takes up a lot of time
         3. it can lead to some low priority process waiting indefinitely for the CPU
         4. none of the mentioned
      4. What is ‘Aging’?
         1. keeping track of cache contents
         2. keeping track of what pages are currently residing in memory
         3. keeping track of how many times a given page is referenced
         4. increasing the priority of jobs to ensure termination

in a finite time

* + - 1. A solution to the problem of indefinite blockage of low – priority processes is
         1. Starvation
         2. Wait queue
         3. Ready queue
         4. Aging
      2. Which of the following statements are true?

1. Shortest remaining time first scheduling may cause starvation
2. Preemptive scheduling may cause starvation
3. Round robin is better than FCFS in terms of response time
   * + - 1. i only
         2. i and iii only
         3. ii and iii only
         4. i, ii and iii
       1. Which of the following scheduling algorithms gives minimum average waiting time?
          1. FCFS
          2. SJF
          3. Round – robin
          4. Priority

# Process Synchronization Critical Section

* 1. Concurrent access to shared data may result in
     1. data consistency
     2. data insecurity
     3. data inconsistency
     4. none of the mentioned
  2. A situation where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which access takes place is called
     1. data consistency
     2. race condition
     3. aging
     4. starvation
  3. The segment of code in which the process may change common variables, update tables, write into files is known as
     1. program
     2. critical section
     3. non – critical section
     4. synchronizing
  4. Which of the following conditions must be satisfied to solve the critical section problem?
     1. Mutual Exclusion
     2. Progress
     3. Bounded Waiting
     4. All of the mentioned
  5. Mutual exclusion implies that
     1. if a process is executing in its critical section, then no other process must be executing in their critical sections
     2. if a process is executing in its critical section, then other processes must be executing in their critical sections
     3. if a process is executing in its critical section, then all the resources of the system must be blocked until it finishes execution
     4. none of the mentioned
  6. Bounded waiting implies that there exists a bound on the number of times a process is allowed to enter its critical section
     1. after a process has made a request to enter its

critical section and before the request is granted

* + 1. when another process is in its critical section
    2. before a process has made a request to enter its critical section
    3. none of the mentioned
  1. A minimum of variable(s) is/are required to be shared between processes to solve the critical section problem.
     1. one
     2. two
     3. three
     4. four
  2. In the bakery algorithm to solve the critical section problem
     1. each process is put into a queue and picked up in an ordered manner
     2. each process receives a number (may or may not be

unique) and the one with the lowest number is served next

* + 1. each process gets a unique number and the one with the highest number is served next
    2. each process gets a unique number and the one with the lowest number is served next

# Semaphore

1. An un-interruptible unit is known as
   1. single
   2. atomic
   3. static
   4. none of the mentioned
2. TestAndSet instruction is executed
   1. after a particular process
   2. periodically
   3. atomically
   4. none of the mentioned
3. Semaphore is a/an to solve the critical section problem.
   1. hardware for a system
   2. special program for a system
   3. integer variable
   4. none of the mentioned
4. What are the two atomic operations permissible on semaphores?
   1. wait
   2. stop
   3. hold
   4. none of the mentioned
5. What are Spinlocks?
   1. CPU cycles wasting locks over critical sections of programs
   2. Locks that avoid time wastage in context switches
   3. Locks that work better on multiprocessor systems
   4. All of the mentioned
6. What is the main disadvantage of spinlocks?
   1. they are not sufficient for many process
   2. they require busy waiting
   3. they are unreliable sometimes
   4. they are too complex for programmers
7. The wait operation of the semaphore basically works on the basic system call.
   1. stop()
   2. block()
   3. hold()
   4. wait()
8. The signal operation of the semaphore basically works on the basic system call.
   1. continue()
   2. wakeup()
   3. getup()
   4. start()
9. If the semaphore value is negative
   1. its magnitude is the number of processes waiting on that semaphore
   2. it is invalid
   3. no operation can be further performed on it until the signal operation is performed on it
   4. none of the mentioned
10. The code that changes the value of the semaphore is
    1. remainder section code
    2. non – critical section code
    3. critical section code
    4. none of the mentioned
11. The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialized as S0 = 1, S1 = 0, S2 = 0.
12. Two processes, P1 and P2, need to access a critical section of code. Consider the following synchronization construct used by the processes.

How many times will P0 print ‘0’?

Process P0 while(true)

{

wait(S0);

print '0'; release(S1); release(S2);

}

Process P1 wait(S1); release(S0);

Process P2 wait(S2);

release(S0);

* 1. At least twice
  2. Exactly twice
  3. Exactly thrice
  4. Exactly once

1. Each process Pi, i = 0,1,2,3,……,9 is coded as follows.

repeat P(mutex)

{Critical Section} V(mutex)

forever

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

* 1. 1
  2. 2
  3. 3
  4. None of the mentioned

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

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

1. It does not ensure mutual exclusion

Process P1 : while(true)

{

w1 = true; while(w2 == true); Critical section w1 = false;

}

Remainder Section

Process P2 : while(true)

{

w2 = true; while(w1 == true); Critical section w2 = false;

}

Remainder Section

1. It does not ensure bounded waiting
2. It requires that processes enter the critical section in strict alternation
3. It does not prevent deadlocks but ensures mutual

exclusion

1. What will happen if a non-recursive mutex is locked more than once?
   1. Starvation
   2. Deadlock
   3. Aging
   4. Signaling
2. What is a semaphore?
   1. is a binary mutex
   2. must be accessed from only one process
   3. can be accessed from multiple processes
   4. none of the mentioned
3. What are the two kinds of semaphores?
   1. mutex & counting
   2. binary & counting
   3. counting & decimal
   4. decimal & binary
4. What is a mutex?
   1. is a binary mutex
   2. must be accessed from only one process
   3. can be accessed from multiple processes
   4. none of the mentioned
5. At a particular time of computation the value of a counting semaphore is 7.Then 20 P operations and 15 V operations were completed on this semaphore. The resulting value of the semaphore is? (GATE 1987)
   1. 42
   2. 2
   3. 7
   4. 12
6. A binary semaphore is a semaphore with integer values
   1. 1
   2. -1
   3. 0.8
   4. 0.5
7. The following pair of processes share a common variable X.

Process A int Y;

A1: Y = X\*2; A2: X = Y;

Process B int Z;

B1: Z = X+1;

B2: X = Z;

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

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

1. two
2. three
3. four
4. eight

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

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

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

Process A int Y;

A1: Y = X\*2; A2: X = Y;

signal(T);

Process B int Z;

B1: wait(T);

B2: Z = X+1; X = Z;

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

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

* 1. one
  2. two
  3. three
  4. four

1. Semaphores are mostly used to implement
   1. System calls
   2. IPC mechanisms
   3. System protection
   4. None of the mentioned
2. Spinlocks are intended to provide only.
   1. Mutual Exclusion
   2. Bounded Waiting
   3. Aging
   4. Progress

# Classic Synchronization Problems

1. The bounded buffer problem is also known as
   1. Readers – Writers problem
   2. Dining – Philosophers problem
   3. Producer – Consumer problem
   4. None of the mentioned
2. In the bounded buffer problem, there are the empty and full semaphores that
   1. count the number of empty and full buffers
   2. count the number of empty and full memory spaces
   3. count the number of empty and full queues
   4. none of the mentioned
3. In the bounded buffer problem
   1. there is only one buffer
   2. there are n buffers ( n being greater than one but finite)
   3. there are infinite buffers
   4. the buffer size is bounded
4. To ensure difficulties do not arise in the readers – writers problem are given exclusive access to the shared object.
   1. readers
   2. writers
   3. readers and writers
   4. none of the mentioned
5. The dining – philosophers problem will occur in case of
   1. 5 philosophers and 5 chopsticks
   2. 4 philosophers and 5 chopsticks
   3. 3 philosophers and 5 chopsticks
   4. 6 philosophers and 5 chopsticks
6. A deadlock free solution to the dining philosophers problem
   1. necessarily eliminates the possibility of starvation
   2. does not necessarily eliminate the possibility of starvation
   3. eliminates any possibility of any kind of problem further
   4. none of the mentioned
7. All processes share a semaphore variable mutex, initialized to 1. Each process must execute wait(mutex) before entering the critical section and signal(mutex) afterward.

Suppose a process executes in the following manner.

signal(mutex);

.....

critical section

.....

wait(mutex);

In this situation :

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

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

Suppose a process executes in the following manner.

wait(mutex);

.....

critical section

.....

wait(mutex);

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

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

Method used by P1 :

while(S1==S2); Critical section

Which of the following statements describes properties achieved?

* 1. Mutual exclusion but not progress

S1 = S2;

Method used by P2 : while(S1!=S2); Critical section

S2 = not(S1);

* 1. Progress but not mutual exclusion
  2. Neither mutual exclusion nor progress
  3. Both mutual exclusion and progress

# Monitors

1. A monitor is a type of
   1. semaphore
   2. low level synchronization construct
   3. high level synchronization construct
   4. none of the mentioned
2. A monitor is characterized by
   1. a set of programmer defined operators
   2. an identifier
   3. the number of variables in it
   4. all of the mentioned
3. A procedure defined within a can access only those variables declared locally within the and its formal parameters.
   1. process, semaphore
   2. process, monitor
   3. semaphore, semaphore
   4. monitor, monitor
4. The monitor construct ensures that
   1. only one process can be active at a time within the monitor
   2. n number of processes can be active at a time within the monitor (n being greater than 1)
   3. the queue has only one process in it at a time
   4. all of the mentioned
5. What are the operations that can be invoked on a condition variable?
   1. wait & signal
   2. hold & wait
   3. signal & hold
   4. continue & signal
6. Which is the process of invoking the wait operation?
   1. suspended until another process invokes the signal operation
   2. waiting for another process to complete before it can itself call the signal operation
   3. stopped until the next process in the queue finishes

execution

* 1. none of the mentioned

1. If no process is suspended, the signal operation
   1. puts the system into a deadlock state
   2. suspends some default process execution
   3. nothing happens
   4. the output is unpredictable

# Atomic Transactions

1. A collection of instructions that performs a single logical function is called
   1. transaction
   2. operation
   3. function
   4. all of the mentioned
2. A terminated transaction that has completed its execution successfully is otherwise it is
   1. committed, destroyed
   2. aborted, destroyed
   3. committed, aborted
   4. none of the mentioned
3. The state of the data accessed by an aborted transaction must be restored to what it was just before the transaction started executing. This restoration is known as of transaction.
   1. safety
   2. protection
   3. roll – back
   4. revert – back
4. Write ahead logging is a way
   1. to ensure atomicity
   2. to keep data consistent
   3. that records data on stable storage
   4. all of the mentioned
5. In the write ahead logging a is maintained.
   1. a memory
   2. a system
   3. a disk
   4. a log record
6. An actual update is not allowed to a data item
   1. before the corresponding log record is written out to stable storage
   2. after the corresponding log record is written out to stable storage
   3. until the whole log record has been checked for inconsistencies
   4. all of the mentioned
7. The undo and redo operations must be to guarantee correct behaviour, even if a failure occurs during recovery process.
   1. idempotent
   2. easy
   3. protected
   4. all of the mentioned
8. The system periodically performs checkpoints that consists of the following operation(s)
   1. Putting all the log records currently in main memory onto stable storage
   2. putting all modified data residing in main memory onto stable storage
   3. putting a log record onto stable storage
   4. all of the mentioned
9. Consider a transaction T1 that committed prior to checkpoint. The <T1 commits> record appears in the log before the <checkpoint> record. Any modifications made by T1 must have been written to the stable storage either with the checkpoint or prior to it. Thus at recovery time
   1. There is a need to perform an undo operation on T1
   2. There is a need to perform a redo operation on T1
   3. There is no need to perform an undo and redo operation on T1
   4. All of the mentioned
10. Serializable schedules are ones where
    1. concurrent execution of transactions is equivalent to the transactions executed serially
    2. the transactions can be carried out one after the other
    3. a valid result occurs after execution transactions
    4. none of the mentioned
11. A locking protocol is one that
    1. governs how locks are acquired
    2. governs how locks are released
    3. governs how locks are acquired and released
    4. none of the mentioned
12. The two phase locking protocol consists of
    1. growing & shrinking phase
    2. shrinking & creation phase
    3. creation & growing phase
    4. destruction & creation phase
13. The growing phase is a phase in which?
    1. A transaction may obtain locks, but does not release any
    2. A transaction may obtain locks, and releases a few or all of them
    3. A transaction may release locks, but does not obtain any new locks
    4. A transaction may release locks, and does obtain new locks
14. The shrinking phase is a phase in which?
    1. A transaction may obtain locks, but does not release any
    2. A transaction may obtain locks, and releases a few or all of them
    3. A transaction may release locks, but does not obtain

any new locks

* 1. A transaction may release locks, and does obtain new locks

1. Which of the following concurrency control protocols ensure both conflict serializability and freedom from deadlock?
2. 2-phase locking
3. Timestamp ordering
   1. I only
   2. II only
   3. Both I and II
   4. Neither I nor II

# Deadlocks Concepts of Deadlock

* 1. What is a reusable resource?
     1. that can be used by one process at a time and is not depleted by that use
     2. that can be used by more than one process at a time
     3. that can be shared between various threads
     4. none of the mentioned
  2. Which of the following condition is required for a deadlock to be possible?
     1. mutual exclusion
     2. a process may hold allocated resources while awaiting assignment of other resources
     3. no resource can be forcibly removed from a process holding it
     4. all of the mentioned
  3. A system is in the safe state if
     1. the system can allocate resources to each process in some order and still avoid a deadlock
     2. there exist a safe sequence
     3. all of the mentioned
     4. none of the mentioned
  4. The circular wait condition can be prevented by
     1. defining a linear ordering of resource types
     2. using thread
     3. using pipes
     4. all of the mentioned
  5. Which one of the following is the deadlock avoidance algorithm?
     1. banker’s algorithm
     2. round-robin algorithm
     3. elevator algorithm
     4. karn’s algorithm
  6. What is the drawback of banker’s algorithm?
     1. in advance processes rarely know how much resource they will need
     2. the number of processes changes as time progresses
     3. resource once available can disappear
     4. all of the mentioned
  7. For an effective operating system, when to check for deadlock?
     1. every time a resource request is made
     2. at fixed time intervals
     3. every time a resource request is made at fixed time intervals
     4. none of the mentioned
  8. A problem encountered in multitasking when a process is perpetually denied necessary resources is called
     1. deadlock
     2. starvation
     3. inversion
     4. aging
  9. Which one of the following is a visual ( mathematical ) way to determine the deadlock occurrence?
     1. resource allocation graph
     2. starvation graph
     3. inversion graph
     4. none of the mentioned
  10. To avoid deadlock
      1. there must be a fixed number of resources to allocate
      2. resource allocation must be done only once
      3. all deadlocked processes must be aborted
      4. inversion technique can be used

# Deadlock Prevention

1. The number of resources requested by a process
   1. must always be less than the total number of resources available in the system
   2. must always be equal to the total number of resources available in the system
   3. must not exceed the total number of resources

available in the system

* 1. must exceed the total number of resources available in the system

1. The request and release of resources are
   1. command line statements
   2. interrupts
   3. system calls
   4. special programs
2. What are Multithreaded programs?
   1. lesser prone to deadlocks
   2. more prone to deadlocks
   3. not at all prone to deadlocks
   4. none of the mentioned
3. For a deadlock to arise, which of the following conditions must hold simultaneously?
   1. Mutual exclusion
   2. No preemption
   3. Hold and wait
   4. All of the mentioned
4. For Mutual exclusion to prevail in the system
   1. at least one resource must be held in a non sharable mode
   2. the processor must be a uniprocessor rather than a multiprocessor
   3. there must be at least one resource in a sharable mode
   4. all of the mentioned
5. For a Hold and wait condition to prevail
   1. A process must be not be holding a resource, but waiting for one to be freed, and then request to acquire it
   2. A process must be holding at least one resource and

waiting to acquire additional resources that are being held by other processes

* 1. A process must hold at least one resource and not be waiting to acquire additional resources
  2. None of the mentioned

1. Deadlock prevention is a set of methods
   1. to ensure that at least one of the necessary conditions cannot hold
   2. to ensure that all of the necessary conditions do not hold
   3. to decide if the requested resources for a process have to be given or not
   4. to recover from a deadlock
2. For non sharable resources like a printer, mutual exclusion
   1. must exist
   2. must not exist
   3. may exist
   4. none of the mentioned
3. For sharable resources, mutual exclusion
   1. is required
   2. is not required
   3. may be or may not be required
   4. none of the mentioned
4. To ensure that the hold and wait condition never occurs in the system, it must be ensured that
   1. whenever a resource is requested by a process, it is not holding any other resources
   2. each process must request and be allocated all its resources before it begins its execution
   3. a process can request resources only when it has none
   4. all of the mentioned
5. The disadvantage of a process being allocated all its resources before beginning its execution is
   1. Low CPU utilization
   2. Low resource utilization
   3. Very high resource utilization
   4. None of the mentioned
6. To ensure no preemption, if a process is holding some resources and requests another resource that cannot be immediately allocated to it
   1. then the process waits for the resources be allocated to it
   2. the process keeps sending requests until the resource is allocated to it
   3. the process resumes execution without the resource being allocated to it
   4. then all resources currently being held are

preempted

1. One way to ensure that the circular wait condition never holds is to
   1. impose a total ordering of all resource types and to

determine whether one precedes another in the ordering

* 1. to never let a process acquire resources that are held by other processes
  2. to let a process wait for only one resource at a time
  3. all of the mentioned

# Deadlock Avoidance

1. Each request requires that the system consider the

to decide whether the current request can be satisfied or must wait to avoid a future possible deadlock.

* 1. resources currently available
  2. processes that have previously been in the system
  3. resources currently allocated to each process
  4. future requests and releases of each process

1. Given a priori information about the number of resources of each type that maybe requested for each process, it is possible to construct an algorithm that ensures that the system will never enter a deadlock state.
   1. minimum
   2. average
   3. maximum
   4. approximate
2. A deadlock avoidance algorithm dynamically examines the to ensure that a circular wait condition can never exist.
   1. resource allocation state
   2. system storage state
   3. operating system
   4. resources
3. A state is safe, if
   1. the system does not crash due to deadlock occurrence
   2. the system can allocate resources to each process in some order and still avoid a deadlock
   3. the state keeps the system protected and safe
   4. all of the mentioned
4. A system is in a safe state only if there exists a
   1. safe allocation
   2. safe resource
   3. safe sequence
   4. all of the mentioned
5. All unsafe states are
   1. deadlocks
   2. not deadlocks
   3. fatal
   4. none of the mentioned
6. A system has 12 magnetic tape drives and 3 processes : P0, P1, and P2. Process P0 requires 10 tape drives, P1 requires 4 and P2 requires 9 tape drives.
7. The Banker’s algorithm is than the resource allocation graph algorithm.
   1. less efficient
   2. more efficient
   3. equal
   4. none of the mentioned
8. The data structures available in the Banker’s algorithm are
   1. Available
   2. Need
   3. Allocation
   4. All of the mentioned
9. The content of the matrix Need is
   1. Allocation – Available
   2. Max – Available
   3. Max – Allocation
   4. Allocation – Max
10. A system with 5 processes P0 through P4 and three resource types A, B, C have A with 10 instances, B with 5 instances, and C with 7 instances. At time t0, the following snapshot has been taken:

Which of the following sequence is a safe sequence?

a) P0, P1, P2

Process P0

P1 P2 P3 P4

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

0 1 0

2 0 0

3 0 2

2 1 1

0 0 2

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

7 5 3

3 2 2

9 0 2

2 2 2

4 3 3

Available A B C

3 3 2

Process P0

P1 P2

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

10

4

9

Currently allocated (process-wise) 5

2

2

1. P1, P2, P0
2. P2, P0, P1
3. P1, P0, P2
4. If no cycle exists in the resource allocation graph
   1. then the system will not be in a safe state
   2. then the system will be in a safe state
   3. all of the mentioned
   4. none of the mentioned
5. The resource allocation graph is not applicable to a resource allocation system
   1. with multiple instances of each resource type
   2. with a single instance of each resource type
   3. single & multiple instances of each resource type
   4. none of the mentioned

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

1. an unsafe state
2. a safe state
3. a protected state
4. a deadlock

# Deadlock Detection

1. The wait-for graph is a deadlock detection algorithm that is applicable when
   1. all resources have a single instance
   2. all resources have multiple instances
   3. all resources have a single 7 multiple instances
   4. all of the mentioned
2. An edge from process Pi to Pj in a wait for graph indicates that
   1. Pi is waiting for Pj to release a resource that Pi needs
   2. Pj is waiting for Pi to release a resource that Pj needs
   3. Pi is waiting for Pj to leave the system
   4. Pj is waiting for Pi to leave the system
3. If the wait for graph contains a cycle
   1. then a deadlock does not exist
   2. then a deadlock exists
   3. then the system is in a safe state
   4. either deadlock exists or system is in a safe state
4. If deadlocks occur frequently, the detection algorithm must be invoked
   1. rarely
   2. frequently
   3. rarely & frequently
   4. none of the mentioned
5. What is the disadvantage of invoking the detection algorithm for every request?
   1. overhead of the detection algorithm due to consumption of memory
   2. excessive time consumed in the request to be allocated memory
   3. considerable overhead in computation time
   4. all of the mentioned
6. A deadlock eventually cripples system throughput and will cause the CPU utilization to
   1. increase
   2. drop
   3. stay still
   4. none of the mentioned
7. Every time a request for allocation cannot be granted immediately, the detection algorithm is invoked. This will help identify
   1. the set of processes that have been deadlocked
   2. the set of processes in the deadlock queue
   3. the specific process that caused the deadlock
   4. all of the mentioned
8. A computer system has 6 tape drives, with ‘n’ processes competing for them. Each process may need 3 tape drives. The maximum value of ‘n’ for which the system is guaranteed to be deadlock free is?

a) 2

b) 3

c) 4

d) 1

1. A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units then, deadlock
   1. can never occur
   2. may occur
   3. has to occur
   4. none of the mentioned
2. ‘m’ processes share ‘n’ resources of the same type. The maximum need of each process doesn’t exceed ‘n’ and the sum of all their maximum needs is always less than m+n. In this setup, deadlock
   1. can never occur
   2. may occur
   3. has to occur
   4. none of the mentioned

# Deadlock Recovery

1. A deadlock can be broken by
   1. abort one or more processes to break the circular wait
   2. abort all the process in the system
   3. preempt all resources from all processes
   4. none of the mentioned
2. The two ways of aborting processes and eliminating deadlocks are
   1. Abort all deadlocked processes
   2. Abort all processes
   3. Abort one process at a time until the deadlock cycle is eliminated
   4. All of the mentioned
3. Those processes should be aborted on occurrence of a deadlock, the termination of which?
   1. is more time consuming
   2. incurs minimum cost
   3. safety is not hampered
   4. all of the mentioned
4. The process to be aborted is chosen on the basis of the following factors?
   1. priority of the process
   2. process is interactive or batch
   3. how long the process has computed
   4. all of the mentioned
5. Cost factors for process termination include
   1. Number of resources the deadlock process is not holding
   2. CPU utilization at the time of deadlock
   3. Amount of time a deadlocked process has thus far consumed during its execution
   4. All of the mentioned
6. If we preempt a resource from a process, the process cannot continue with its normal execution and it must be
   1. aborted
   2. rolled back
   3. terminated
   4. queued
7. To to a safe state, the system needs to keep more information about the states of processes.
   1. abort the process
   2. roll back the process
   3. queue the process
   4. none of the mentioned
8. If the resources are always preempted from the same process can occur.
   1. deadlock
   2. system crash
   3. aging
   4. starvation
9. What is the solution to starvation?
   1. the number of rollbacks must be included in the cost factor
   2. the number of resources must be included in resource preemption
   3. resource preemption be done instead
   4. all of the mentioned