I.C.E.™

Inspiring Creativity & Endeavour

IES | GATE | PSU

OPERATING SYSTEM

For

COMPUTER SCIENCE

I.C.E.
Inspiring Creativity & Endeavour
IES | GATE | PSU

OPERATING SYSTEM

SYLLABUS

Processes, threads, interprocess communication, concurrency and synchronization. Deadlock. CPU scheduling. Memory management and virtual memory. File systems.

ANALYSIS OF GATE PAPERS

Exam Year	1 Mark Ques.	2 Mark Ques.	Total
2003	2	5	12
2004	3	4	11
2005	-	2	4
2006	1	8	17
2007	2	6	14
2008	2	5	12
2009	2	5	12
2010	3	2	7
2011	4	2	8
2012	1	4	9
2013	2	4	10
2014 Set-1	2	3	8
2014 Set-2	1	3	7
2014 Set-3	1	3	7
2015 Set-1	2	4	10
2015 Set-2	2	3	8
2015 Set-3	2	2	6
2016 Set-1	1	4	9
2016 Set-2	1	3	7
2017 Set-1	2	2	6
2017 Set-2	2	2	6

CONTENTS

1 PROCESS AND DEADLOCK

PARTS OF PROCESS MANAGEMENT

Process Management can be divided into following parts:

1. PROCESS CONCEPT
2. CPU SCHEDULING
3. PROCESS SYNCHRONIZATION
4. DEADLOCKS

1.1 PROCESS CONCEPT

A process is more than the program code (same as the text section). It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers. A process generally also includes the process stack, containing temporary data (such as subroutine parameters, return address, and temporary variables), and a data section containing global variables.

1.1.1 Process State

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

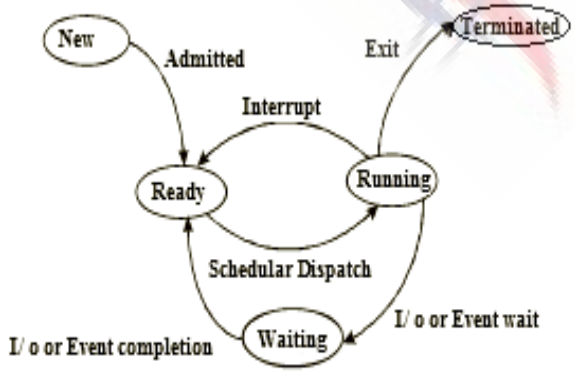


Diagram of process state

1) **New:** The process is being created.
2) **Running:** Instructions are being executed.
3) **Waiting:** The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
4) **Ready:** The process is waiting to be assigned to a processor.
5) **Terminated:** The process has finished execution.

1.1.2 STATE DIAGRAM OF A PROCESS

A process goes through various states with respect to CPU. It is depicted in following figure:

a) **Initial State:** A process/program/job starts in a state in which it has no resources at all. It is called initial state.
b) **Hold State:** A process is said to be hold state if it lies in the secondary storage.
c) **Ready State:** A job in ready state is in the main memory. It is ready to use CPU but CPU is not available.
d) **Run State:** A process is said to be in Run state if it is using the CPU. Hence a process wanting to reach RUN state goes through the ready state first.
e) **Wait State:** A process is said to be in Wait state if it is waiting for something to happen and it is not in a position to use the CPU (e.g. there may be page fault).
f) **Complete State:** In this state, the process completes its erection. It does not need any resources.

1.1.3 PROCESS CONTROL BLOCK

Each process is represented in the operating system by a process control block(PCB)-also called a task control block. A PCB is shown in the figure below:

Pointer	Process state
Process number	
Program counter	
Register	
Memory limits	
List of open files	
:	

1.1.4 PROCESS SCHEDULING

The objective of multiprogramming is to have some process running to all times, to maximize, CPU utilization.

1.1.5 Scheduling Queues

As processes enter the system, they are put into a job queue. The queue consists of all processes in the system. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the ready queue.
A common representation for a discussion of process scheduling is a queuing diagram, such as shown in Figure

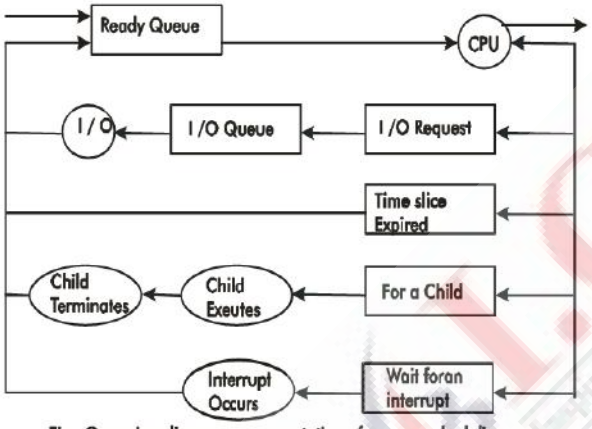


Fig. Queueing-diagram respresentation of process scheduling.

1.2 SCHEDULERS

A process migrates between the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler.
The long-term scheduler (or job scheduler) The short-term scheduler (or CPU scheduler)The primary distinction between these two schedulers is the frequency of their execution. The short-term scheduler must select a new process for the CPU quite frequently. A process may execute for only a few milliseconds before waiting for an I/O request. Often, the short-term scheduler executes at least once every 100 milliseconds. The long -term scheduler, on the other hand, executes much less frequently. There may be minutes between the creations of new processes in the system. The long-term scheduler controls the degree of multiprogramming (the number of processes in memory).

1.2.1 CONTEXT SWITCH

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is known as a context switch.

1.2.2 PERATION ON PROCESSES

The processes in the system can execute concurrently, and must be created and deleted dynamically. Thus, the operating system must provide a mechanism for process creation and termination.
A thread, sometimes called a lightweight process (LWP), is a basic unit of CPU utilization, and consists of a program counter, a register set, and a stack space. It shares with peer threads, its code section, data section, and operating-system resources such as open files and signals, collectively known as a task. A traditional or heavyweight process is equal to a task with one thread. A task does nothing if no threads are in it, and a thread must be in exactly one task. The extensive sharing makes CPU switching among peer threads and the creation of threads inexpensive, compared with context switches among heavyweight processes. Although a thread context switch still requires a register set switch, no memory-management- related work need be done.
Threads can be in one of several states: Ready, blocked, running, or terminated.
Like processes, threads share the CPU, and only one thread at a time is active (running). A thread within a process executes sequentially, and each thread has its own stack and program counter. Threads

can create child threads, and can block waiting for system calls to complete; if one thread is blocked, another thread can run.

1.2.3 CPU SCHEDULING

CPU scheduling is the basis of multi-programmed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

The idea of multiprogramming is relatively simple. A process is executed until it must wait, typically for the completion of some I/O request. In a simple computer system, the CPU would then just sit idle. All this waiting time is wasted; no useful work is accomplished. With multiprogramming, we try to use this time productively. Several processes are kept in memory at one time. When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process. This pattern continues.

1.2.4 PREEMPTIVE AND NON-PREEMPTIVE SCHEDULING

CPU scheduling decisions may take place under the following four circumstances:

1) When a process switches from the running state to the waiting state, I/O request, or Invocation of wait for the termination of one of the child processes)
2) When a process switches from the running state to the ready state, when an interrupt Occurs)
3) When a process switches from the waiting state to the ready state e.g. completion of I/O)
4) When a process terminates Under these circumstances, there is no choice in terms of scheduling. A new process (If one exists in the ready queue) must be selected for execution. There is a choice, however, for circumstances (ii) and (iii).

When scheduling takes place only under circumstances (i) to (iv), the scheduling scheme is non pre-emptive; otherwise, the scheduling scheme is preemptive. Under non pre-emptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.

Unfortunately, preemptive scheduling incurs a cost. Consider the case of two processes sharing data. One may be in midst of updating the data when it is preempted and the Second process is run. The second process may try to read the data, which are currently in an inconsistent

1.3 DISPATCHER

Another component involved in the CPU scheduling function is the dispatcher. The dispatches is the module that gives control of the CPU to the process selected by the short- term scheduler.

This function involves:

1) Switching context
2) Switching to user mode
3) Jump it to the proper location in the user program to restart that program.

The dispatcher should be as fast as possible, given that 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 dispatcher latency.

1.3.1 SCHEDULING CRITERIA

The scheduling criteria that are used includes

1) CPU utilization

The CPU should be kept as busy as possible. CPU utilization may range from 0 to 100 per cent. In a real system, it should range from 40 per cent (for a lightly loaded system) to 90 per cent (for a heavily used system).

2) **Throughput**
   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. For long processes, this rate may be one.
1) **Turnaround time**
   The interval from the time of submission of a process to the time of completion is the turnaround time.
2) **Waiting time**
   The amount of time a process spends waiting in the ready queue.
3) **Response time**
   Response time, is the amount of time it takes to start responding, but not the time that it takes to output that response. The turnaround time is generally limited by the speed of the output device.

1.3.2 SCHEDULING ALGORITHMS

First-Come, First-Served Scheduling (FCFS)

The average waiting time under the FCFS policy, however, is often quite long.
Consider the following set of processes that arrive at time 0, with the length of the CPU-burst time given in milliseconds:

PROCESS	BURST TIME
P_1	24
P_2	3
P_3	3

If the processes arrive in the order P_1, P_2, P_3, and are served in FCFS order, we get the result shown in the following Gantt chart:

P_1	P_2	P_3	
0	24	27	30

Now, the waiting time is 0 milliseconds for process P_1, 24 milliseconds for process P_2, and 27 milliseconds for process P_3. Thus, the average waiting time = $\frac{0+24+27}{3} = 17$ milliseconds.

If the processes arrive in the order P_2, P_3, P_1, however, the results will be as shown in the following Gantt chart:

P_2	P_3	P_1	
0	3	6	30

Now, the average waiting time = $\frac{6+0+3}{3} = 3$ milliseconds.
This reduction is substantial. Thus, the average waiting time under a FCFS policy is generally not minimal.
The FCFS scheduling algorithm is non-preemptive. Once, the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating or by requesting I/O.

1.3.4 Shortest –Job-First Scheduling (SJF)

Consider the following set of process, with the length of the CPU burst time given in milliseconds:

PROCESS	BURST TIME
P_1	6
P_2	8
P_3	7
P_4	3

Using SJF scheduling, we would schedule these processes according the following Gantt Chart:

P_4	P_1	P_3	P_2	
0	3	9	16	24

The waiting time is 3 milliseconds for process P_1, 16 milliseconds for process P_2, 9 milliseconds for process P_3, and 0 milliseconds for process P_4.
Thus, the average waiting time = $\frac{3+16+9+0}{4}$ = 7 milliseconds.
If we were using the FCFS scheduling scheme, then the average waiting time would be 10.25 milliseconds.
The SJF scheduling algorithm is provably optimal, in that it gives the minimum average waiting time for a given set of processes. If two processes have the same length next CPU burst, FCFS scheduling is used to break the tie.

GATE QUESTIONS

Topics **Page No**

1 PROCESS MANAGEMENT-I

Q.1 A processor needs software interrupt to
a) Test the interrupt system of the processor
b) Implement co- routines
c) Obtain system series which need execution of privileged instructions
d) Return from subroutine

[GATE -2001]

Q.2 A CPU has two modes- privileged and non- privileged. In order to change the mode from privileged to non- privileged
a) a hardware interrupt is needed
b) a software interrupt is needed
c) a privileged instruction (which does not generate an interrupt) is needed.
d) a non-privileged instruction (which does not generate an interrupt) is needed.

[GATE -2001]

Q.3 Consider a set of n tasks with known runtimes $r_1, r_2, \ldots\ldots, r_n$ to be run on a uni-processor machine. Which of the following processor scheduling algorithms will result in the maximum throughput?
a) Round Robin
b) Shortest Job First
c) Highest Response Ratio Next
d) First Come First Served

[GATE-2001]

Q.4 Which of the following scheduling algorithms is non-pre-emptive?
a) Round Robin
b) First-in-First out
c) Multilevel queue Scheduling
d) Multilevel Queue Scheduling with Feedback

[GATE-2001]

Q.5 Which of the following does not interrupt a running process?
a) A device
b) Timer
c) Scheduler process
d) Power failure

[GATE -2001]

Q.6 Which combination of the following features will suffice to characterize an operating system as a multi-programmed operating system?
A) More than one program may be loaded into main memory at the same time for execution.
B) If a program waits for certain events such as IN/OUT, another program is immediately scheduled for execution.
C) If the execution of a program terminates, another program is immediately scheduled for execution.
a) (A) only b) (A) and (B)
c) (A) and (C) d) (A)(B)and(C)

[GATE-2002]

Q.7 Draw the process state transition diagram of an OS in which (i) each process is in one of the five states: created, ready, running, blocked (i.e. sleep or wait). Or terminated, and (ii) only non- preemptive scheduling is used by the OS Label the transitions appropriately.

[GATE -2002]

Q.8 A uni-processor computer system only has two processes, both of which alternate 10 ms CPU bursts with 90 ms IN/OUT bursts. Both the processes were created at nearly the same time. The I N/OUT of both processes can proceed in parallel. Which of the following scheduling

EXPLANATIONS

Q.1 **(c)**
To execute privileged instructions, system services can be obtained using software interrupt.

Q.2 **(d)**
Because we want to change the mode from privileged to non-privileged, to the next instruction to be executed should be non-privileged instruction.

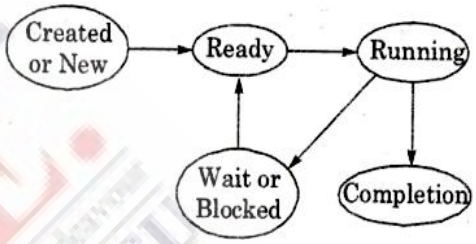
Q.3 **(b)**
Throughput is defined as the measure of the number of processes completed per unit time and if the CPU is busy in executing some process than the work is being done. If the number of processors is increased then amount of work done is increased and time is decreased. In case of uni-processor first come first serve gives the maximum throughput but when the case of known run-time is considered shortest job first is used which is also used in case of turn around. :

Q.4 **(b)**
Non-pre-emption means that once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating the process or by switching to waiting state. Thus, FIFO, i.e., first in first out algorithm is non-pre-emptive as it does not pre-empty the CPU while it is been executed.

Q.5 **(a)**
Device can not interrupt a running process but timer, scheduler and power failure can interrupt a running process.

Q.6 **(b)**
Both A and B conditions are necessary to characterize an operating system as a multi programmed operating system.
Option C) is characteristic of both multi programmed and single programmed OS.

Q.7 Just draw the diagram by using all criteria as below:



Q.8 **(a)**
FCFS produces least CPU utilization when two: processes do both IN/OUT and computing whereas round robin produces maximum CPU utilization.

Q.9 **(b)**
Statement I False: The context switch is slower with kernel supported threads.
Statement II True: For user level threads, a system call can block the entire system. The statement is true as it is the drawback of user level threads blocking system call can block the entire process.
Statement III True: Kernel supported threads can be scheduled independently as kernel supported threads have their own areas thus, are scheduled independently.
Statement IV False: User level threads can never be transparent to the kernel.

Q.10 **(a)**

ASSIGNMENT QUESTIONS

Q.1 Virtual memory is
a) An externally large main memory
b) An externally large secondary memory
c) An illusion of an extremely large memory
d) A type of memory used in super computers

Q.2 Spatial localities refers to the problem that once a location is referenced
a) It will not be referenced again
b) It will be referenced again
c) A nearby location will be referenced soon
d) None of the above

Q.3 Which of the following is an example of a SPOOLED device?
a) The terminal used to enter the input data for a program being executed.
b) The secondary memory device in a virtual memory system
c) A line printer used to print the output of a number of jobs.
d) None of the above.

Q.4 Page fault occurs when
a) The page is corrupted by application software
b) The page is main memory
c) The page is not in main memory
d) One tries to divide a number by 0

Q.5 Overlay is
a) A part of the operating system
b) A specific memory location
c) A single contiguous memory that used in the olden days for Running large programs by swapping.
d) Overloading the system with many user files

Q.6 Determine the number of page faults when references to pages occur in the order -1, 2, 4, 5, 2, 1, 2, 4. Assume that the main memory can accommodate 3 pages and the main memory already has the pages 1 and 2, with page 1 having been brought earlier than page 2. (Assume LRU algorithm is used)
a) 3 b) 5
c) 4 d) None of the above

Q.7 Concurrent processes are processes that
a) Do not overlap in time
b) Overlap in time
c) Are executed by a processor at the same time
d) None of the above

Q.8 The page replacement policy that sometimes leads to more page faults when the is size of the memory is increased is
a) FIFO
b) LRU
c) No such policy exists
d) None of the above

Q.9 The only state transition that is initiated by the user process itself is
a) Block b) dispatch
c) Wakeup d) None of the above

Q.10 Working set (t, k) at an instant of time, t, is the set of
a) K future reference that the operating system will make
b) Future reference that the operating system will make in the next 'k' time units
c) K reference with high frequency
d) Pages that have been referenced in the last k time units

EXPLANATIONS

Q.15 (b)
Deadlock occurs when each other of the 3 user processes hold one resource and make simultaneous demand for another. If there are 4 resources one of the 3 user processes will get the fourth instance of the resource and relinquish one or both of the resource(s) it is currently holding after

Q.25 (c)
Each p operation will decrease the Semaphore value by 1 and V operations increase it by 1. If x is 18, then 7P operations will make semaphore value 0. If this is followed by 7V operations the value comes back to 7. So after 18 P and 18 V operations, the value of the semaphore will be 7. The remaining 2P operations result in the semaphore value 5.

Q.29 (d)
Even in a non-multiprogramming system, memory protection may be used, when for example, spooling is being used.

Q.32 (a)
2 processes can never lead to deadlock as the peak time demand of (3+3) tape drives can be satisfied. But 3 processes can lead to deadlock if each holds 2 drives and then demand one more.

Q.41 (c)
0.35×10+(1– 0.35)×100= 68.5 ns

Q.43 (d)
Size of Virtual Memory depends on Size of Hard Disk. It doesn't depend on Address Bus, since we have many addressing modes.

Q.44 (d)
The arrival pattern is a Poisson distribution.
P (K requests) $= e^{-\mu T}(\mu T)^k / k!$
Here k = 0, μ= 20, T = 3/4.

Q.52 (a)
It is 0+10+(15–2)+(18–5)+(38–10) divided by 5, i.e., 12.8 ms.

Q.53 (b)
It is 8+0+3+15+8 divided by 5, i.e., 6.8 ms.

Q.54 (c)
It is 10+3+0+15+0 divided by 5, i.e., 5.6 ms.

Q.55 (c)
It is 30 + 0 + 3 + 3 + 18 divided by 5, i.e., 10.8 ms.

Q.56 (b)
It is 30 + 3 + 0 + 5 + 0 divided by 5, i.e., 7.6 ms.

Q.60 (a)
Having 11 resources ensure that at least one process will have no pending request. This process after using will release the resources and so Deadlock can never occur.

Q.61 (a)
At least one process will be holding 2 resources in case of a Simultaneous demand from all the processes. That process will release the 2 resources, thereby avoiding any possible deadlock.

Q.62 (a)