Computer Science &
Information Technology

Operating System

Comprehensive Theory

with Solved Examples and Practice Questions

MADE EASY
Publications

MADE EASY Publications Pvt. Ltd.

Corporate Office: 44-A/4, Kalu Sarai (Near Hauz Khas Metro Station), New Delhi-110016
E-mail: infomep@madeeasy.in
Contact: 011-45124660, 8860378007
Visit us at: www.madeeasypublications.org

Operating System

First Edition: 2015
Second Edition : 2016
Third Edition : 2017
Fourth Edition : 2018
Fifth Edition : 2019
Sixth Edition : 2020
Seventh Edition : 2021
Eighth Edition : 2022

Contents

Operating System

■■■■

Operating System

Goal of the Subject

The goal of this book is to provide all important concepts of operating systems such as Processes and Threads, Mutual Exclusion, CPU Scheduling, Deadlock, Memory Management, Virtual Memory and File Systems.

- Understand the purpose of the operating system
- Distinguish between a resource, a program, and a process
- Solutions of semaphores
- Describe various memory page replacement algorithms
- Describe how files are stored in secondary storage

Operating System

INTRODUCTION

An operating system is a program that acts as an intermediary between a user of a computer and the computer hardware. Two primary aims of an operating systems are to manage resources (e.g. CPU time, memory) and to control users and software. The book consists of topic wise *Examples* and *Student Assignment* questions which test the important concepts from the lesson and provide practice problems. This subject includes the following chapters.

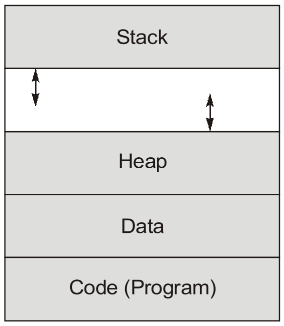
1. **Basic Concepts of Operating System:** In this chapter we discuss Basic Concepts of Operating System, Types of Operating System, Dual Mode Operations and System Call.
2. **Processes and Threads:** In this chapter we discuss Process, Operations on a Process, Scheduling, Thread and Co-operating Processes (Inter Process Communication).
3. **CPU Scheduling:** In this chapter we discuss Goals of CPU Scheduling and Scheduling Algo.
4. **Process Synchronization:** In this chapter we discuss Synchronization, Critical-Section Problem, Synchronization Techniques, Semaphores and Classical Problems of Synchronization with Semaphore Solution.
5. **Deadlock and Concurrency:** In this chapter we discuss Concurrency, Deadlock, Conditions for a deadlock, Methods of Handling Deadlocks, Prevention, Avoidance and Detection & Recovery.
6. **Memory Management:** In this chapter we discuss Logical (Virtual) Vs Physical Address Space, Memory Allocation Techniques, Internal Fragmentation and External Fragmentation, Paging, Segmentation, Segmented Paging and Buddy System.
7. **Virtual Memory:** In this chapter we discuss Page Fault, Page Replacement, Dynamic Paging Algorithms and Frame Allocation.
8. **File System:** In this chapter we discuss Directories, File Management System, File Allocation Methods and Free Space Management.
9. **Input/Output System:** In this chapter we discuss Input/Output System Structure, Magnetic Storage Devices, Disk Scheduling and Disk Scheduling Algorithms.

■■■■

CHAPTER 02

Processes and Threads

2.1 Process



Abstraction view of a process

A process is an activity of executing a program. It is a program under execution. Every process needs certain resources to complete its task.

Under execution means:

(*i*) It should reside in main memory.

(*ii*) It occupies the CPU to execute the instruction.

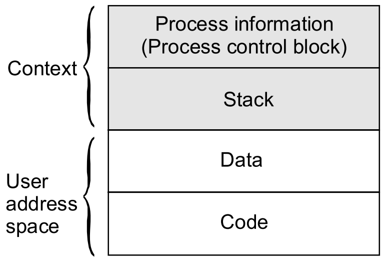
(*iii*) It is active and dynamic.

There are two types of processes:

1. **User process:** User processes are executed in user mode and user processes can be preempted while executing.
2. **System process:** System processes are executed in privileged mode. System process executed automatically without preemption.

- Code (text section): It contains executable instructions it contains the value of program counter and the contents of the processor's registers.
- Heap: This is dynamically allocated memory to a process during its run time.
- Data: This section contains the global and static variables.
- Stack: The process stack contains the temporary data such as method (function) parameters, return address and local variables.

2.1.1 Process Description



Code and data section of a program are called as "user address space". Context section of a process is managed by operating system which contains stack and process control information.

2.1.2 Process Control Block (PCB)

A process control block is a data structure maintained by the operating system for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process as listed below:

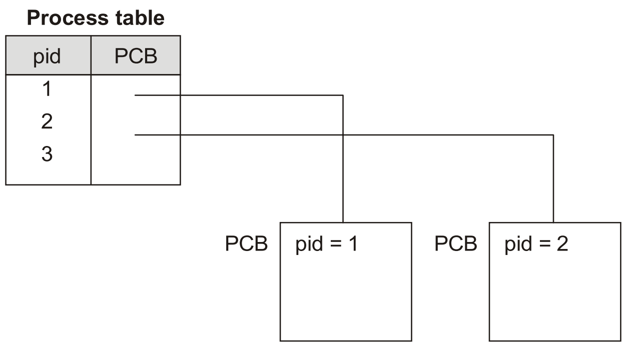
Process ID
Process state
Pointer
Priority
Program counter
CPU registers
I/O information
Accounting information

- Process ID: Unique identification for each process in OS.
- Process state: Current state of the process, i.e. whether running, ready or waiting.
- Pointer: Pointer to parent process.
- Priority: Priority of the process.
- Program counter: It is a pointer to the address of the next instruction to be executed for this process.
- CPU registers: Registers where process need to be stored for execution for running state.
- I/O information: It includes a list of I/O devices allocated to the process.
- Accounting information: It includes the amount of CPU used for process execution, time limits etc.

Doubly linked list data structure is generally used to implement process control block. If the attributes are less arrays can also be used. But generally PCBs are large data instruction and so stored in doubly linked list which can be accommodated for any process to any level.

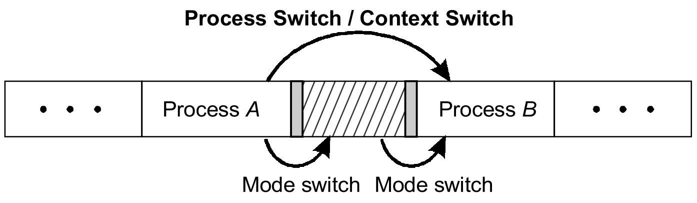
The Process Control Block (PCB) is maintained for a process throughput its lifetime, and is deleted once the process terminates.

When the process make transitions from one state to another, the OS updates its information in the process's PCB in a process table so that it can access the PCB quickly.



2.1.3 Process Switch (Context Switch)

1. **Process switch:** Process switch also called as context switch which involves saving the current CPU information, updating the control information and restore the CPU information.
   Process switch includes the following steps:
   (*i*) Save CPU context [Mode switch from user mode to Kernel mode using mode bit].
   (*ii*) Update PCB of current process.
   (*iii*) Move PCB of current process to appropriate queue.
   (*iv*) Select another process for execution [By CPU scheduler].
   (*v*) Update PCB of selected process.
   (*vi*) Update memory management structures.
   (*vii*) Restore CPU context of new PCB [Mode switch from Kernel mode to user mode].



2. Context switch time is dependent on hardware and it is overhead because during context switch the system does do not useful work.

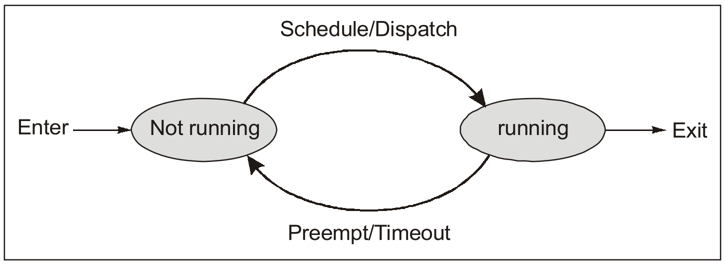
2.2 Process State Models

Process state defines the current activity of the process. Process states are: new, running, ready, blocked, suspended (suspended ready, suspended blocked), etc.

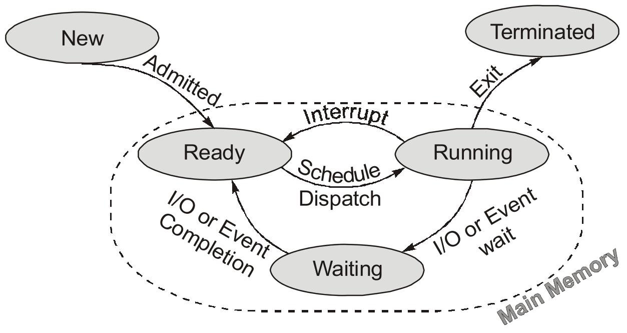
2.2.1 Types of Process State Models

Number of process states are decided by the type of process state model.

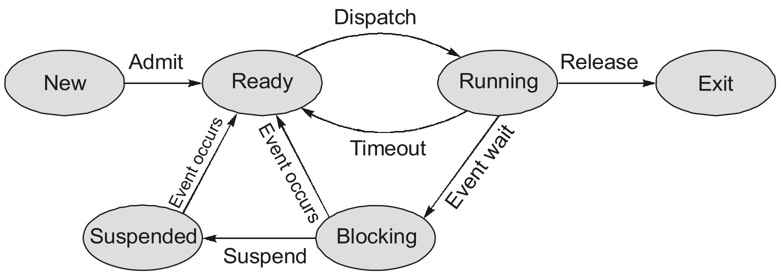
1. **Two state process model:**



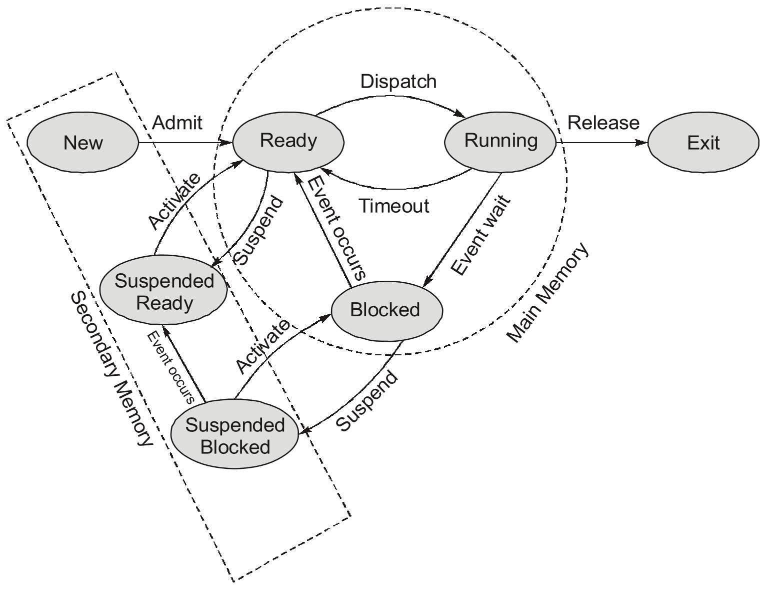
2. **Five state process model:**



3. **Six state process model:**



4. **Seven state process model:**



2.2.2 Queues / States Description

- New: In this state, the process is about to be created but not yet created; it is the program which is present in secondary memory that will be picked up by OS to create the process.
- Ready: When a process is ready, the program and data are loaded, and PCB of that process kept in "Ready" queue. The process is available for execution and available in main memory.
- Blocked: Whenever the process requests access to I/O or needs input from the user or needs access to a critical region, it enters the blocked or wait state. Once the process completes I/O operation, it enters ready state.
- Running: The process currently executing is in running state atmost one process is in running state at any time.
- Suspended blocked: The process is in secondary memory and awaiting an event will be in suspended blocked. The process in "Blocked" will move to "Suspended blocked" due to many reasons such as blocked process might be consuming more memory.
- Suspended ready: The process is in the secondary memory but is available for execution whenever it is loaded into memory. The process moved from "Ready" queue to "Suspended Ready" queue due to more priority for other processes and several reasons exist.
- Exit: A process which completed its execution will be terminated by swapping out and its PCB is deleted.

2.2.3 Active and Inactive Jobs

The following processes (active) are in main memory.

- Ready
- Running
- Blocked

The following processes (inactive) are in secondary memory.

- New
- Suspended blocked
- Suspended ready
- Exit

2.2.4 Actions Performed by OS

Timeout/Preemption: The process receives a timer interrupt and relinquishes control back to the OS dispatcher. The OS puts the process in "Ready" queue and dispatches another process to the CPU.

Dispatch: A process in "Ready" queue has been chosen to be the next running process.

Event Wait (Block/I/O Wait): A process invokes an I/O system call for an event that blocks the currently executing process. OS puts the process in "Blocked" queue and dispatches another process to the CPU from "Ready" queue.

Event Occurs (Unblock/ I/O Completion): An I/O system sends an interrupt to CPU to inform the completion of its task. OS may then decide to unblock the process which is waiting in blocked "queue" and puts in "Ready" queue.

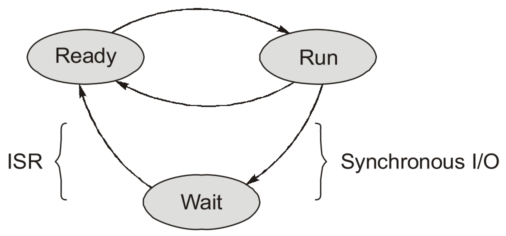
2.2.5 Types of Process

Processes may be categorized as:

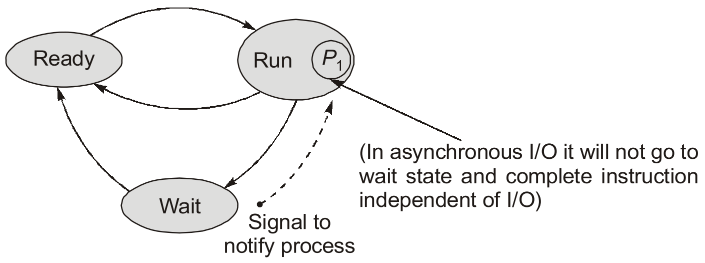
- CPU-bound: Process does not need much I/O service, almost always want the CPU.
- I/O-bound: Short CPU burst times, needs lots of I/O service.
- Interactive: Short CPU burst times, lots of time waiting for user input (keyboard, mouse).

2.2.6 Types of I/O (Synchronous and Asynchronous I/O)

1. **Synchronous I/O:** The process performing I/O operation will be placed in the block state till I/O operation is completed. Once I/O operations is completed ISR (Interrupt Service Routine) will be initiated which places the process from block to ready state.



2. **Asynchronous I/O:** In asynchronous I/O, while initiating I/O request, a handler function will be registered. The process is not placed in block state it continues to execute remaining code after initiating I/O request. Once the I/O request is completed the signal mechanism is used to notify the process that data is available and registered handler function is asynchronously invoked. All information about process will be stored in handler function like type, what it does etc.


Ready
Run
P_1
Wait
Signal to notify process
(In asynchronous I/O it will not go to wait state and complete instruction independent of I/O)

2.2.7 Process Memory Image

Memory image of a process has following 5 sections which are used while running a process.

- **Text Segment/Test Region:** It stores the machine instructions of stored program and size is fixed.
- **Data Segment:** It stores initialized static and global data of stored program and size is fixed.
- **BSS (Block Stated by Symbol) Segment:** It stores uninitialized static data that is defined in the program (e.g., global uninitialized strings, numbers, structures). Size of BSS is fixed.
- **Heap Segment:** It is dynamically allocated memory. Memory can be allocated and deallocated dynamically at run time.
- **Stack Segment:** It used to maintain the call stack, which holds return addresses, local variables, temporary data, and saved registers. Stack is dynamic.

2.2.8 Important Commands (System Calls)

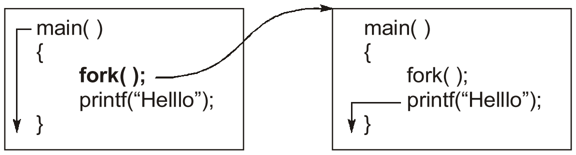
- **Fork:** Fork creates a new process. The new process is a copy of the parent. It's running the same program and has the same open files. It is, however, a copy and not a reference to the parent. Any memory modifications or changes to open file status will be invisible to the parent and vice versa. Fork system call: (i) Returns – ve value if process creation is unsuccessful, (ii) Returns +ve value if process creation is successful, (iii) Returns 0 to newly created process.
  The parent and child process have same virtual address but physical address will be different.

NOTE: For N fork statement $2^N - 1$ child process will be created.

Example - 2.1 Find the number of child processes created for the following code and also find how many times "Hello" is printed.

```
main( )
{
    fork( );
    printf("Hello");
}
```

Solution:

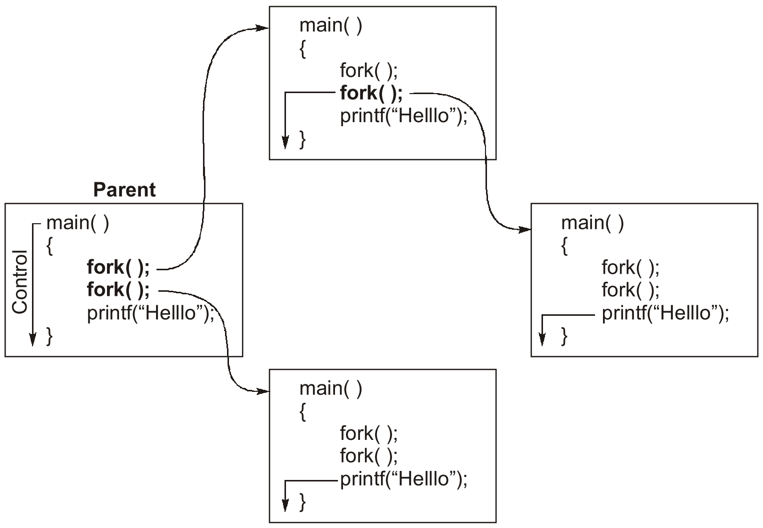


Only one child process is created. "Hello" is printed twice for the given code.

Example - 2.2 Find the number of child processes created for the following code and also find how many times "Hello" is printed.

```
main( )
{
    fork( );
    fork( );
    printf("Hello");
}
```

Solution:

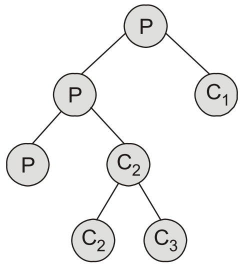


Three child processes are created. "Hello" is printed four times.

Example-2.3 Find the output of following code:

```
int main( )
{
1.  if (fork( ))
    {
2.      if (! fork( ))
        {
3.          fork( );
            printf("1");
        }
4.      else
        {
            printf("2");
        }
    }
5.      else
    {
        printf("3");
    }
        printf("4");
        return 0;
    }
```

Solution:



Output: 2 4 1 4 1 3 4

It creates two process: Parent with process ID of child process and child C_1 with process ID = 0.

When condition is true parent P executes if statement (statement 1) and child executes else (statement 5) and prints 3.

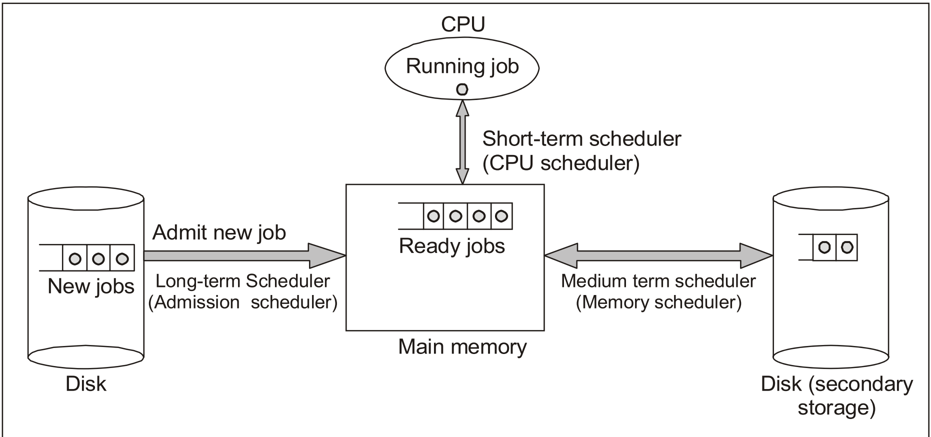
Parent P again checks if statement (statement 2) and creates 2 process (one parent and one child C_2). In if statement, not operator (!) is used, it executes for child process C_2 and parent P executes else part (statement 4) and prints 2.

Child C_2 further creates two processes (one parent C_2 and other is child C_3).

2.3 Scheduling

Scheduling depends on three scheduler in the system

1. Long-term scheduler
2. Medium-term scheduler
3. Short-term scheduler


CPU
Running job
Short-term scheduler (CPU scheduler)
Admit new job
New jobs
Long-term Scheduler (Admission scheduler)
Ready jobs
Medium term scheduler (Memory scheduler)
Main memory
Disk
Disk (secondary storage)

Three-level Scheduling

2.3.1 Long-term Scheduler

- It is also called as job scheduler.
- It determines which process are to be admitted to the system for processing. It selects processes from the queue (from disk) and leads them into memory for execution.
- Its primary objective is to provide a balanced mix of I/O bound and CPU bound processes.
- It controls the degree of multiprogramming.

Student's Assignment

Q.1 Consider the following components in a computer system:
C1: Computer Hardware
C2: Application Programs
C3: Utilities
Find the place of operating system using the above components.
(a) In-between C1 and C2
(b) In-between C2 and C3
(c) In-between C1 and C3
(d) None of the above

Q.2 Which of the following instruction is allowed only in Kernel mode?
(a) Switch user mode to Kernel mode
(b) Read the time
(c) Disable all interrupts
(d) None of these

Q.3 Which of the following is advantage of micro Kernel approach to the system design?
(a) Adding a new service does not require modifying the Kernel.
(b) It is more secure because more operations are done in user mode than in Kernel mode.
(c) Both (a) and (b)
(d) Neither (a) nor (b)

Q.4 Which of the following operations need not to be privileged?
(a) Read the clock (b) Access I/O device
(c) Clear the memory (d) None of these

Q.5 Identify one of the following which need not be part of operating system?
(a) CPU scheduling
(b) Page replacement
(c) Demand paging and virtual memory
(d) Compiler

Q.6 Which of the following is hardware generated signal?
(a) Interrupt (b) Trap
(c) Both (a) and (b) (d) Neither (a) nor (b)

Q.7 Identify the scheduler which involves only in the decision for selection of partially serviced jobs?
(a) Short-term scheduler
(b) Long-term scheduler
(c) Medium-term scheduler
(d) None of these

Q.8 In multi thread process, which of the following is shared by the threads of a process?
(a) CPU state (b) Stack
(c) Address space (d) All of these

Q.9 Match the following groups:
Group-I (Scheduler)
A. Long-term scheduler
B. Medium-term scheduler
C. Short-term scheduler
Group-II (Transition of process)
1. New to ready state
2. Ready to running state
3. Suspended to blocked
Codes:

	A	B	C
(a)	1	2	3
(b)	1	3	2
(c)	3	1	2
(d)	2	3	1

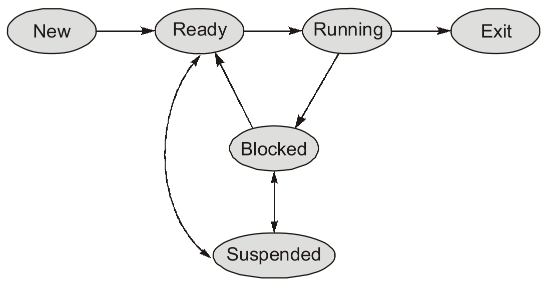
Q.10 Identify the correct statement from the following
(a) Job may transit from “Ready” to “Suspend ready”
(b) Job may transit from “Suspend blocked” to “Suspend ready”
(c) Job may transit from “Suspend ready” to “Ready”
(d) All of these

Q.11 Which of the following scheduling can be done by thread library?
(a) Process scheduling
(b) Kernel thread scheduling
(c) User thread scheduling
(d) None of these

Q.12 If a system uses 5-state model for processes execution (New, Ready, Running, Blocked, Exit), then which of the following state processes are in the main memory?

(a) New, Ready
(b) Ready, Running
(c) Ready, Running, blocked
(d) New, Ready, Running, blocked

Q.13 Consider the following six-state model.



Identify the states in which processes are in the secondary memory (disk).
(a) New
(b) New, Suspended
(c) New, Suspended, blocked
(d) New, Suspended, blocked, Ready

Q.14 A system can support a process with user level threads or Kernel level threads. Identify the correct statement from the following.
(a) User thread is known to Kernel, blocking one thread causes entire thread is blocked.
(b) Kernel level threads: Contain separate TCB for each thread within a process, blocking one thread is independent to other thread
(c) Both (a) and (b)
(d) Neither (a) nor (b)

Q.15 Which of the following is not TCB information.
(a) Thread identifier
(b) CPU state information: control and status registers, stack pointers
(c) Scheduling: state, priority awaited event
(d) Used memory and I/O, program code (Address space)

Q.16 Choose the incorrect statement from the following.
(a) Thread can't suspended (Swapped out) from blocked state to suspended state.
(b) Processes can be suspended from blocked to suspended state
(c) Threads are lightweight processes
(d) None of these

Q.17 Monitor mode is called as
(a) Privileged mode (b) Kernel mode
(c) System mode (d) All of these

Q.18 Which of the following statement is false.
(a) Interrupt is hardware generated interrupt
(b) Trap is software generated interrupt
(c) Trap can be used to call OS routines or to catch arithmetic errors
(d) None of these

Q.19 Which of the following statements are correct when user level threads are compared to Kernel level threads.
(a) User level threads require memory management where Kernel threads do not.
(b) User level thread scheduling is faster than Kernel thread scheduling.
(c) Both (a) and (b)
(d) Neither (a) nor (b)

Q.20 Which of the following can run in parallel on different processors in a multiprocessor?
(a) Processes
(b) Kernel threads of a process
(c) Both (a) and (b)
(d) User threads of a process

Q.21 Which of the following are equal things when there is only one CPU in a system?
(a) Multiprogramming and Multitasking
(b) Multiprocessing and Multiprogramming
(c) Multitasking and Multiprocessing
(d) None of these

Q.22 Which of the following item is not a part of Process Control Block (PCB)?
(a) CPU state information
(b) Scheduling information of a process
(c) Stack section
(d) None of these

Q.23 In multi-threading, which of the following statements is correct?
(a) All threads of a process share same address space and resources
(b) Creation of a new thread only involves allocating a new stack and a new thread Control Block

(c) Both (a) and (b)
(d) Neither (a) nor (b)

Q.24 Match **List-I** and **List-II** and select correct answer using codes given below:

List-I
A. Fork
B. Context switch
C. Degree of multiprogramming
D. Message passing

List-II
1. Inter process communication
2. Process creation
3. Dispatcher
4. Long term scheduler

	A	B	C	D
(a)	2	1	4	3
(b)	1	4	3	2
(c)	1	4	2	3
(d)	2	3	4	1

Q.25 Consider the following statements:

S_1: Multiprogramming is used to increase CPU utilization, while time-sharing is used to increase CPU responsiveness in interacting with user.

S_2: If a user-level thread is blocked for I/O operation, then kernel of operating system will perform context switching to run another user-thread which is not blocked.

S_3: Many-to-One is the most efficient model of multi-threading because it allows several user-level threads to be assigned to different processors in a multi-processor computer system.

Which of the following is true?
(a) Only S_1 and S_2 (b) Only S_2 and S_3
(c) Only S_1 and S_3 (d) None of these

Q.26 Consider following statements with respect to interprocess communication:

S_1: Message passing is useful for exchanging smaller amount of data.

S_2: Message passing require kernel support

S_3: Shared memory is faster than message passing mechanisms.

S_4: Shared memory are implemented with the help of system calls and continuous requires system call support for exchanging data.

Which of the statements are NOT TRUE?
(a) S_2 and S_4 (b) S_3 only
(c) S_4 only (d) S_1 and S_4 only

Answer Key:

1. (a)	**2.** (c)	**3.** (c)	**4.** (a)	**5.** (d)
6. (a)	**7.** (c)	**8.** (c)	**9.** (b)	**10.** (d)
11. (c)	**12.** (c)	**13.** (b)	**14.** (b)	**15.** (d)
16. (d)	**17.** (d)	**18.** (d)	**19.** (b)	**20.** (c)
21. (a)	**22.** (c)	**23.** (c)	**24.** (d)	**25.** (a)
26. (c)				

Student's Assignments Explanations

1. (a)
Operating system is a software that acts as an interface between user (applications) and hardware.

2. (c)
Disabling the instruction is privileged instruction, so it is only allowed in Kernel mode.

3. (c)
- As all new services are added to user space and consequently do not require modification of the Kernel.
- The micro Kernel also provides more security and reliability, since most services are running as user processes rather than Kernel processor.

4. (a)
Read the clock is non-privileged instruction.

5. (d)
Compiler is not a part of OS. It is a software program that transforms a program written in high level language to low level language.

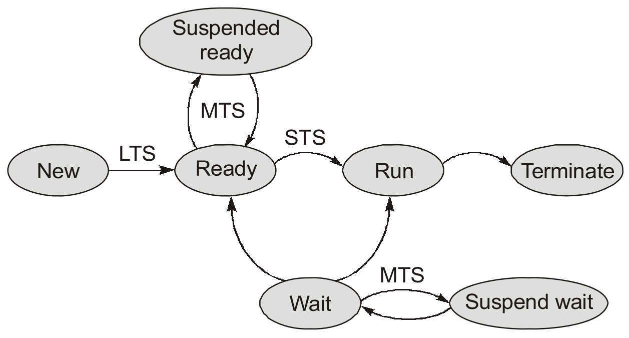
6. (a)
Interrupt are hardware generated whereas as trap are software generated signal.

7. (c)
Medium term scheduler is responsible for selection of partially services jobs, i.e., it swaps processes in and out of the memory. It swaps processes between:
- Ready and suspend ready
- Blocked and suspend blocked

Here, partially serviced jobs means the jobs that are waiting for IO or resources to be completed.

8. (c)
Memory address space is shared by the threads of a process whereas each thread maintains its state and stack. Each thread requires its own stack to store the local variables.

9. (b)



Here, LTS: long term scheduler
MTS: Medium term scheduler
STS: Short term scheduler

10. (d)
- A job moves from ready to suspend ready if space is not available in main memory.
- A job moves from suspend blocked to suspend ready if the event for which it has been waiting occurs.
- A job moves from suspend ready to ready if enough memory space is available.

11. (c)
Thread libraries provide programmers with API for creation and management of threads. Kernel level threads are supported and managed directly by OS.
User level threads are managed without Kernel support. They are entirely managed by user level library.

12. (c)
Only ready, running and wait state are in main memory.

13. (b)
Suspended and new state are in secondary memory (or disk). Jobs are brought from disk to main memory using scheduler.

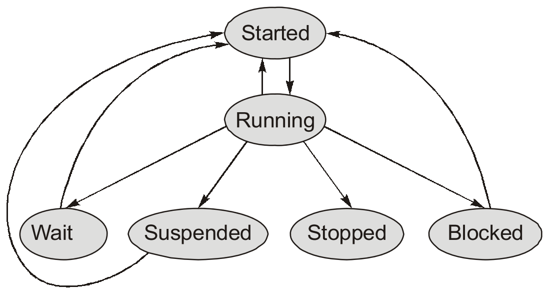
14. (b)
Option (a) is partially incorrect as user level threads are not known to Kernel. Blocking one user level thread causes entire threads to be blocked.
Option (b) is true.

15. (d)
Thread control block (TCB) contains the following information:

Thread Id
Thread state
CPU information: Program counter Register contents
Thread priority
Pointer to process that created this thread
Pointer(s) to other thread(s) that were created by this thread

TCB

16. (d)
(a) The thread life cycle (or thread states) are as follows:


Started
Running
Wait
Suspended
Stopped
Blocked