1 UNIT

Computer System

Part-1 ..... **(1–2B to 1–9B)**

- *Introduction : Operating System and Function*
- *Classification of Operating System : Batch*
- *Interactive*
- *Time-Sharing*
- *Real Time System*
- *Multi-processor Systems*
- *Multi-user Systems*
- *Multi-process Systems*
- *Multi-threaded Systems*

A. Concept Outline : Part-1 ..... 1–2B
B. Long and Medium Answer Type Questions 1–2B

Part-2 ..... **(1–9B to 1–20B)**

- *Operating System Structure*
- *Layered Structure*
- *System Components*
- *Operating System Services*
- *Re-entrant Kernels*
- *Monolithic and Micro-kernel System*

A. Concept Outline : Part-2 ..... 1–9B
B. Long and Medium Answer Type Questions 1–10B

PART-1

Introduction : Operating System and Function, Classification of Operating System : Batch, Interactive, Time-Sharing, Real Time System, Multi-processor Systems, Multi-user Systems, Multi-process Systems, Multi-threaded System.

CONCEPT OUTLINE : PART-1

- An operating system is a program that acts an intermediary between a user of a computer and the computer hardware.
- **Function of operating system :**
  - i. Memory management
  - ii. Processor management
  - iii. Device management
  - iv. File management
- The feature of a batch system is the lack of interaction between the user and the job while that job is executing.
- A multi-processor is a computer system with two or more central processing units, with each one sharing the common main memory as well as peripherals.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.1. **What is operating system ? Write the major functions of an operating system.**

Answer

1. An operating system acts as an intermediary between the user of a computer and computer hardware.
2. An operating system is software that manages the computer hardware.
3. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

Functions of an operating system :

The operating systems perform the following functions :

1. **Booting**
   i. Uses diagnostic routines to test system for equipment failure.
   ii. Copies BIOS (Basic Input System) programs from ROM chips to main memory.
   iii. Loads operating system into computer's main memory.
2. **Formatting :**
   Formats (initializes) diskettes so they can store data and programs.
3. **Managing computer resources :**
   i. Keeps track of locations in main memory where programs and data are stored (memory management).
   ii. Moves data and programs back and forth between main memory and secondary storage (swapping) via partitioning, using foreground and background areas, and using buffers and queues.
4. **Managing files :**
   i. Copies files/programs from one disk to another.
   ii. Backup files/programs.
   iii. Erases (deletes) files/programs.
   iv. Renames files.
5. **Managing tasks :**
   May be able to perform multi-tasking, multi-programming, time-sharing or multi-processing.

Que 1.2. **Describe the classification of operating system.**

OR

Write down the different types of operating system.

AKTU 2016-17, Marks 10

Answer

Various types of operating system are as follows :

1. **Batch system :**
   a. This type of operating system was used in the earlier age.
   b. To speedup processing, jobs with similar needs were batched together and were run through the computer as a group.
   c. The definitive feature of a batch system is the lack of interaction between the user and the job while that job is executing.
   d. In this execution environment, the CPU is often idle.
2. **Multi-programming system :**
   a. In this type of operating system, more than one program will reside into main memory.
   b. The operating system picks and begins to execute one of the jobs in the memory.

   c. Eventually, the job may have to wait for some task, the operating system simply switches to another job and executes it.
   d. When the first job finishes, waiting job gets the CPU back.
   e. As long as there is always some job to execute, the CPU will never be idle.
3. **Time-sharing system :**
   a. A time-shared operating system allows the many users to share the computer simultaneously.
   b. A time-shared operating system uses CPU scheduling and multi-programming to provide each user with a small portion of a time-shared computer.
4. **Real time operating system :**
   a. Real time operating system is a special purpose operating system, used when there are rigid time requirements on the operation of a processor or the flow of data.

Que 1.3. **What is real time operating system ? What is the difference between hard real time and soft real time operating system ?**

AKTU 2013-14, Marks 05

Answer

1. Real time operating system is a special purpose operating system, used when there are rigid time requirements on the operation of a processer or the flow of data.
2. Real time operating system has well-defined, fixed time constraints otherwise system will fail.
3. For example, scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, home-appliances controllers, air traffic control system etc.

Difference between hard real time and soft real time operating system :

S. No.	Characteristic	Hard real time	Soft real time
1.	Response time	Hard required	Soft desired
2.	Peak-load performance	Predictable	Degraded
3.	Control of pace	Environment	Computer
4.	Safety	Often critical	Non-critical
5.	Size of data files	Small/medium	Large
6.	Redundancy type	Active	Checkpoint-recovery
7.	Data integrity	Short-term	Long-term
8.	Error detection	Autonomous	User assisted

Que 1.4. **Discuss essential properties of time-sharing operating system, real time operating system and distributed operating system.**

AKTU 2012-13, Marks 05

Answer

Properties of time-sharing operating system :

1. Uses CPU scheduling and multi-programming to provide economical interactive use of a system.
2. The CPU switches rapidly from one user to another *i.e.*, the CPU is shared between a number of interactive users.
3. Instead of having a job defined by spooled card images, each program reads its next control instructions from the terminal and output is normally printed immediately on the screen.

Properties of real time operating system :

1. Real time systems are usually dedicated, embedded systems.
2. They typically read from and react to sensor data.
3. The system must guarantee response to events within fixed periods of time to ensure correct performance.

Properties of distributed operating system :

1. Distributes computation among several physical processors.
2. The processors do not share memory or a clock, instead, each processor has its own local memory.
3. They communicate with each other through various communication lines.

Que 1.5. **Explain the following terms :**

i. Multi-programming system

ii. Multi-processor system

Answer

i. **Multi-programming system :** Refer Q. 1.2, Page 1–3B, Unit-1.

ii. **Multi-processor system :**

1. Multi-processor systems have more than one processor in close communication. They share the computer bus, system and input-output devices and sometimes memory.
2. In multi-processing system, it is possible for two processes to run in parallel.
3. Multi-processor systems are of two types : symmetric multi-processing and asymmetric multi-processing.

a. In symmetric multi-processing, each processor runs identical copy of the operating system and they communicate with one another as needed. All the CPU shared the common memory.

b. In asymmetric multi-processing, each processor is assigned a specific task. It uses master-slave relationship.

Que 1.6. **Explain the following terms clearly :**

i. Multi-programming

ii. Multi-threading

AKTU 2011-12, Marks 05

Answer

i. **Multi-programming :** Refer Q. 1.2, Page 1–3B, Unit-1.

ii. **Multi-threading :**

1. Multi-threading extends the idea of multi-tasking into applications, so we can sub-divide specific operations within a single application into individual threads.
2. Each of the threads can run in parallel.
3. The OS divides processing time not only among different applications, but also among each thread within an application.
4. In a multi-threaded program, an example application might be divided into four threads : a user interface thread, a data acquisition thread, network communication, and a logging thread.
5. We can prioritize each of these, so that they operate independently.
6. Thus, in multi-threaded applications, multiple tasks can progress in parallel with other applications that are running on the system.

Que 1.7. **Write down the difference between multi-processing and multi-programming operating system.**

AKTU 2013-14, Marks 05

Answer

S. No.	Multi-processing	Multi-programming
1.	Multi-processing refers to processing of multiple processes at same time by multiple CPUs.	Multi-programming keeps several programs in main memory at the same time and execute them concurrently utilizing single CPU.
2.	It utilizes multiple CPUs.	It utilizes single CPU.
3.	It permits parallel processing.	Context switching takes place.

4.	Less time taken to process the jobs.	More time taken to process the jobs.
5.	It facilitates much efficient utilization of devices of the computer system.	Less efficient than multi-processing.
6.	Usually more expensive.	Such systems are less expensive.

Que 1.8. **Describe the differences between symmetric and asymmetric multi-processing.** **AKTU 2014-15, Marks 05**

Answer

S.No.	Basis	Symmetric multi-processing	Asymmetric multi-processing
1.	Architecture	All processor in symmetric multi-processing has the same architecture.	All processor in asymmetric multi-processing may have same or different architecture.
2.	Communication	All processors communicate with another processor by a shared memory.	Processors need not to communicate as they are controlled by the master processor.
3.	Failure	If a processor fails, the computing capacity of the system reduces.	If a master processor fails, a slave is turned to the master processor to continue the execution. If a slave processor fails, its task is switched to other processors.
4.	Ease	Symmetric multi-processor is complex.	Asymmetric multi-processor is simple.

Que 1.9. **What is spooling ?**

OR

What is spooling ? What are the advantage of spooling over buffering ?

Answer

1. SPOOLING is acronym for Simultaneous Peripheral Operations Online.
2. Spooling refers to putting jobs in a buffer, a special area in memory or on a disk where a device can access them when it is ready.

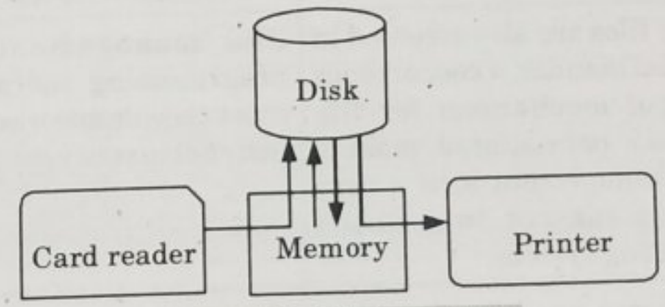


Fig. 1.9.1. Print spooling.

3. Spooling is useful because device access data at different rates.
4. The buffer provides a waiting station where data can rest while the slower device catches up.
5. The most common spooling application is print spooling.
6. In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate.
7. Spooling is also used for processing data at remote sites. The CPU sends the data via communications path to a remote printer.

Advantages of spooling over buffering :

1. The spooling operation uses a disk as a very large buffer.
2. Spooling is capable of overlapping I/O operation for one job with processor operations for another job.

Que 1.10. **Differentiate between batch processing system and multi-programming system with example.**

Answer

S. No.	Batch processing system	Multi-programming system
1.	In batch systems, the instructions, data and some control information are submitted to the computer operator in the form of a job.	Multi-programming system allow concurrent execution of multiple programs, and hence multi-programming operating system require more sophisticated scheduling algorithms.

2.	Only one program is in execution at a time, any time-critical device management is not required, which simplifies the I/O management.	Multi-programming operating system allow sharing of I/O devices among multiple users, more sophisticated I/O management is required.
3.	Since files are also accessed in a serial manner, a concurrency control mechanism for file access is not required, making file management also a very simple matter in a batch operating system.	File management in multi-programming operating system must provide advanced protection, and concurrency control methods.
4.	The users are not allowed to interact with the computer system.	The programs should be scheduled in such a way that the CPU remains busy for maximum amount of time.
5.	**Example :** Execution of batch files such as autoexec.bat by the computer.	**Example :** Interleaved execution of two or more different and independent programs by the same computer.

PART-2

Operating System Structure, Layered Structure, System Components, Operating System, Services Re-entrant Kernels, Monolithic and Micro-kernel System.

CONCEPT OUTLINE : PART-2

- **Structure of OS :**
  - i. Simple structure
  - ii. Layered approach
  - iii. Micro-kernels
  - iv. Modules
- **System components :**
  - i. Memory management
  - ii. Processor management
  - iii. Device management
  - iv. File management

- **Operating system service :**
  - i. Program execution
  - ii. I/O operation
  - iii. File system manipulation
  - iv. Error detection
  - v. Protection and security

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 1.11. **What is an operating system ? Discuss the operating system structure.**

OR

Explain layered structure of an operating system. Also explain advantages and disadvantages of the layered approach to system design.

AKTU 2014-15, Marks 05

Answer

Operating system : Refer Q. 1.1, Page 1–2B, Unit-1.

Structure of OS :

1. **Simple structure :** MS-DOS written to provide the most functionality in the least space :
   - a. Not divided into modules.
   - b. Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated.

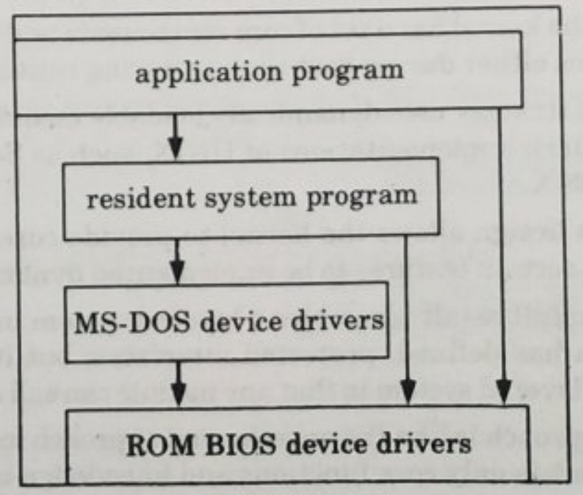


Fig. 1.11.1.

2. **Layered approach :**

   a. The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer *N*) is the user interface.

   b. With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.

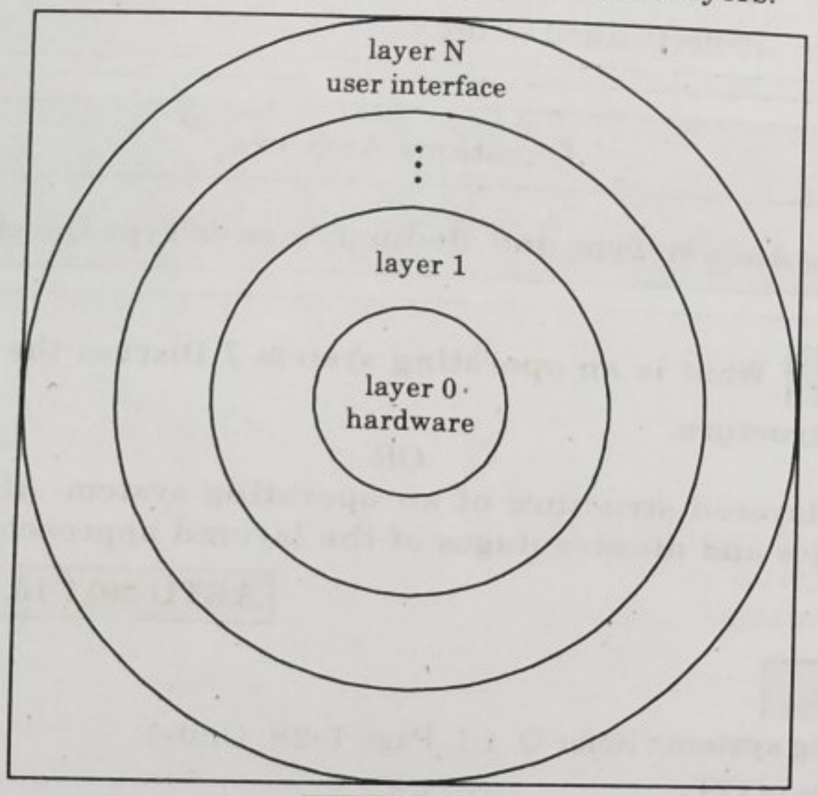


Fig. 1.11.2.

3. **Micro-kernels :**

   a. Moves as much from the kernel into "user" space communication takes place between user modules using message passing.

4. **Modules :**

   a. Here the kernel has a set of core components and links in additional services either during boot time or during runtime.

   b. Such a strategy uses dynamically loadable modules and is common in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X.

   c. Such a design allows the kernel to provide core services yet also allows certain features to be implemented dynamically.

   d. The overall result resembles a layered system in that each kernel section has defined, protected interfaces; but it is more flexible than a layered system in that any module can call any other module.

   e. The approach is like the microkernel approach in that the primary module has only core functions and knowledge of how to load and

communicate with other modules; but it is more efficient, because modules do not need to invoke message passing in order to communicate.

Advantages :

1. The main advantage of the layered approach is simplicity of construction and debugging.
2. The layers are selected so that each uses functions (operations) and services of only lower-level layers.
3. This approach simplifies debugging and system verification.
4. The first layer can be debugged without any concern for the rest of the system, because, it uses only the basic hardware to implement its functions.
5. Once the first layer is debugged, its correct functioning can be assumed while the second layer is debugged, and so on.
6. If an error is found during the debugging of a particular layer, the error must be on that layer, because the layers below it are already debugged.
7. Thus, the design and implementation of the system is simplified.

Disadvantages :

1. The major difficulty with the layered approach involves appropriately defining the various layers. Because a layer can use only lower level layers, careful planning is necessary.
2. A final problem with layered implementations is that they tend to be less efficient than other types. At each layer, the parameters may be modified, data may need to be passed, and so on. Each layer adds overhead to the system call; the net result is a system call that takes longer than one on a non-layered system.
3. These limitations have caused a small backlash against layering. Fewer layers with more functionality are being designed, providing most of the advantages of modularized code while avoiding the difficult problems of layer definition and interaction.

Que 1.12. **What is an operating system ? Define the components of an operating system.** **AKTU 2013-14, Marks 05**

OR

Discuss various operating system components. **AKTU 2014-15, Marks 05**

Answer

Operating system : Refer Q. 1.1, Page 1–2B, Unit-1.

Components of an operating system are :

1. **Memory management :**

   An operating system does the following activities for memory management :

   a. Keeps tracks of primary memory, *i.e.*, what part of it are in use by whom, what part are not in use.

   b. In multi-programming, the OS decides which process will get memory when and how much.

   c. Allocates the memory when a process requests it to do so.

   d. De-allocates the memory when a process no longer needs it or has been terminated.

2. **Processor management :**

   An operating system does the following activities for processor management :

   a. Keeps tracks of processor and status of process. The program responsible for this task is known as traffic controller.

   b. Allocates the processor (CPU) to a process.

   c. De-allocates processor when a process is no longer required.

3. **Device management :**

   An operating system manages device communication via their respective drivers. It does the following activities for device management :

   a. Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.

   b. Decides which process gets the device when and for how much time.

   c. Allocates the device in the efficient way.

   d. De-allocates devices.

4. **File management :**

   An operating system does the following activities for file management :

   a. Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

   b. Decides who gets the resources.

   c. Allocates the resources.

   d. De-allocates the resources.

Que 1.13. **Define the services provided by the operating system.**

AKTU 2013-14, Marks 05

Explain the services provided by operating system.

AKTU 2016-17, Marks 10

Answer

Some of the services provided by operating system are as follows :

1. **Program execution :**

   a. The purpose of computer system is to allow the users to execute programs in an efficient manner.

   b. The operating system provides an environment where the user can conveniently run these programs.

   c. To run a program, the program is required to be loaded into the RAM first and then to assign CPU time for its execution.

2. **I/O operations :**

   a. Each program requires an input and after processing the input submitted by user it produces output.

   b. This involves the use of I/O devices.

   c. The I/O service cannot be provided by user-level programs and it must be provided by the operating system.

3. **File system manipulation :**

   a. While working on the computer, generally a user is required to manipulate various types of files like opening a file, saving a file and deleting a file from the storage disk.

   b. This is an important task that is also performed by the operating system.

4. **Communication :**

   a. Operating system performs the communication among variou types of processes in the form of shared memory.

   b. Communications may be implemented via shared memory, in whic two or more processes read and write to a shared section of memor or message passing, in which packets of information in predefine formats are moved between processes by the operating system.

5. **Error detection :**

   a. The main function of operating system is to detect the errors lil bad sectors on hard disk, memory overflow and errors related I/O devices.

   b. After detecting the errors, operating system takes an appropria action for consistent computing.

6. **Resource allocation :**
   a. In the multitasking environment, when multiple jobs are running at a time, it is the responsibility of an operating system to allocate the required resources (like CPU, main memory, tape drive or secondary storage etc.) to each process for its better utilization.

8. **Protection and security :**
   a. Protection involves ensuring that all access to system resources is controlled.
   b. Such security starts with requiring each user to authenticate him or her to the system, usually by means of a password, to gain access to system resources.

Que 1.14. **What is kernel ? Describe various operations performed by kernel ?**

AKTU 2016-17, Marks 10

Answer

1. Kernel is the main component of most computer operating systems.
2. It provides an interface between applications and actual data processing at the hardware level.
3. Kernel is considered as the heart of an operating system.
4. Kernel provides the lowest-level abstraction layer for the resources (especially processors and I/O devices) that application software must control to perform its function.

The various types of operations performed by the kernel are :

1. It controls the state of the process that is it checks whether the process is running or process is waiting for the request of the user.
2. It provides the memory for the processes those are running on the system that is kernel runs the allocation and de-allocation of process.
3. The kernel also maintains a time table for all the processes those are running that is the kernel also prepare the schedule time which will provide the time to various process of the CPU.
4. The kernel puts the waiting and suspended jobs into the different memory area.
5. When a kernel determines that the logical memory does not fit to store the programs, then it uses the concept of the physical memory which will store the programs into temporary manner. Therefore, physical memory of the system can be used as temporary memory.

Que 1.15. **Describe re-entrant kernels.**

Answer

1. A re-entrant kernel is one where many processes or threads can execute in the same kernel program concurrently without affecting one another.
2. In non-reentrant kernels, a process does not modify kernel programs, but modify global kernel data.
3. If a process is executing operating system programs, no other processes may be allowed to execute the programs, nor may system start another kernel path execution when the kernel access the global data.
4. An interrupts from I/O devices may not be handled immediately by the operating system.
5. Normally the operating system is composed of re-entrant and non-reentrant functions. The re-entrant functions may modify local data, but they do not modify any global data.
6. Concurrent executions of re-entrant functions do not affect the behaviour of one another. The operating system makes sure that non-reentrant functions are executed mutually exclusively by the kernel paths so that the function executions do not modify global data at the same time.

Que 1.16. **Differentiate between the following :**

i. Shell and kernel

ii. Monolithic kernel and micro-kernel

AKTU 2011-12, Marks 05

OR

What are the differences between shell and kernel ?

AKTU 2014-15, Marks 05

Answer

i. Shell and kernel :

S. No.	Shell	Kernel
1.	When a user logs in, the login program checks the username and password, and then starts another program called the shell.	The kernel is the hub of the operating system, it allocates time and memory to programs and handles the file storage and communications in response to system calls.
2.	Shell gives interface between user and kernel.	Kernel gives the hardware interaction with user.
3.	The shell is the GUI or interface to the kernel.	The kernel is the part of the operating system that interacts with the hardware.

ii. Monolithic kernel and micro-kernel :

S. No.	Monolithic kernel	Micro-kernel
1.	Kernel size is large.	Kernel size is small.
2.	OS is complex to design.	OS is easy to design, implement and install.
3.	All the operating system services are included in the kernel.	Kernel provides only IPC and low level device management services.
4.	Request may be serviced faster.	Request may be serviced slower than monolithic kernel.
5.	No message passing and no context switching are required while the kernel is performing the job.	Micro-kernel requires message passing and context switching.

Que 1.17. **What do you understand by system call ? How is a system call made ? How is a system call handled by the system ? Choose suitable examples for explanation.** **AKTU 2011-12, Marks 05**

Answer

System calls :

1. System calls provide an interface to the services made available by an operating system.
2. These calls are generally available as routines written in C and C++, although certain low-level tasks (for example, task where hardware must be accessed directly), may need to be written using assembly language instructions.

System call is made :

1. An operating system makes system calls available, let's first use an example to illustrate how system calls are used : writing a simple program to read data from one file and copy them to another file.
2. The first input that the program will need is the names of the two files : the input file and the output file.
3. One approach is for the program to ask the user for the names of the two files.
4. In an interactive system, this approach will require a sequence of system calls, first to write a prompting message on the screen and then to read from the keyboard the characters that define the two files.
5. Once the two file names are obtained, the program must open the input file and create the output file. Each of these operations requires another system call.
6. When the program tries to open the input file, it may find that there is no file of that name or that the file is protected against access.

7. In these cases, the program should print a message on the console and then terminate abnormally.
8. If the input file exists, then we must create a new output file.
9. We may find that there is already an output file with the same name. This situation may cause the program to abort, or we may delete the existing file and create a new one.
10. On input, the program may find that the end of the file has been reached or that there was a hardware failure in the read (such as a parity error.)
11. Finally, after the entire file is copied, the program may close both files, write a message to the console or window, and finally terminate normally.
12. Typically, application developers design programs according to an application programming interface (API).
13. The API specifies a set of functions that are available to an application programmer, including the parameter that are passed to each function and the return values the programmers can expect.
14. The functions that make up an API typically invoke the actual system calls on behalf of the application programmer.

System call handled by system :

1. The runtime support system for most programming languages provides a system call interface that serves as the link to system calls made available by the operating system.
2. The system call interface intercepts function calls in the API and invokes the necessary system calls within the operating system.
3. Typically, a number is associated with each system call, and the system call interface maintains a table indexed according to these numbers.
4. The system call interface then invokes the intended system calls in the operating system kernel and returns the status of the system call and any return values.
5. The caller need know nothing about how the system calls is implemented or what it does during execution.
6. Rather, it need only obey the API and understand what the operating system will do as a result of the execution of that system call.
7. Thus, most of the details of the operating system interface are hidden from the programmer by the API and are managed by the runtime support library.

Que 1.18. **What do you understand by system call ? Enumerate five system calls used in process management.**

AKTU 2013-14, Marks 05

OR

What do you understand by system call ? List and explain three system calls used for process management.

AKTU 2012-13, Marks 05

Answer

System call : Refer Q. 1.17, Page 1–17B, Unit-1.

System calls used for process management :

fork() : To create a new process
exec() : To execute a new program in a process
wait() : To wait until a created process completes its execution
exit() : To exit from a process execution
getpid() : To get a process identifier of the current process
getppid() : To get parent process identifier
brk() : To increase/decrease the data segment size of a process

Que 1.19. **What are the advantages of the layered approach to the design of operating system ?** **AKTU 2012-13, Marks 05**

1. The main advantage of the layered approach is simplicity of construction and debugging.
2. The layers are selected so that each uses functions (operations) and services of only lower-level layers.
3. This approach simplifies debugging and system verification.
4. The first layer can be debugged without any concern for the rest of the system, because, it uses only the basic hardware to implement its functions.
5. Once the first layer is debugged, its correct functioning can be assumed while the second layer is debugged, and so on.
6. If an error is found during the debugging of a particular layer, the error must be on that layer, because the layers below it are already debugged.
7. Thus, the design and implementation of the system is simplified.

Que 1.20. **What is the reason behind dual mode operation of processors ?** **AKTU 2011-12, Marks 05**

OR

Whether it is possible to construct a secure operating system without having dual mode of operation in the system ? Give arguments in favour of your answer. **AKTU 2012-13, Marks 05**

Answer

1. In dual mode operation, two separate modes are used for working of operating system. These modes are user mode and monitor mode.
2. For indicating mode of the system, mode bit is used in the computer hardware. The mode bit is 0 for monitor and 1 for user.
3. With the mode bit, we are able to distinguish between a task that is executed in user mode or monitor mode.
4. This feature helps to the operating system in many ways :
   a. At the booting time, the hardware starts in the monitor mode, then operating system is loaded. The hardware switches from user mode to monitor mode when interrupts occur. When the operating system gains control of the system, it is in monitor mode.
   b. The dual mode operation provides the protection to the operating system from unauthorized users.

   c. The privileged instructions are executed only in the monitor mode. The computer hardware is not allowed for executing the privilege instructions in other mode, *i.e.* user mode.
   d. These provide greater protection for the operating system. Fig. 1.20.1 shows the dual mode protecting.

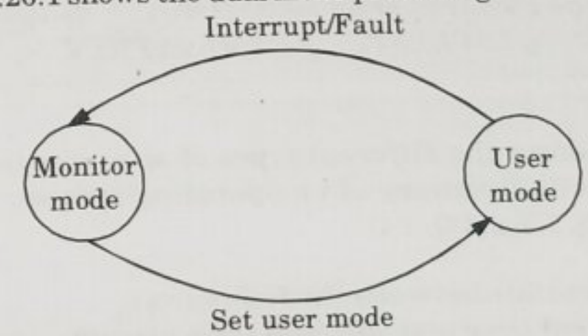


Fig. 1.20.1.

Que 1.21. **Differentiate between process switch and mode switch.** **AKTU 2012-13, Marks 2.5**

Answer

S. No.	Process switch	Mode switch
1.	Process switching occurs when the processor switches from one process to another.	Mode switching is the switching of a process between kernel mode and user mode. A process can execute in either of these two modes.
2.	When process switching occurs, the kernel saves the context of old process in its PCB and loads the saved context of new process scheduled to run.	Mode switching can be done using system call. This is a special instruction that sets the system's state to kernel mode.
3.	Example : If an interrupt occurs, the system needs to save the current context of running process on CPU, so that it can restore the saved context when processing of interrupt is over, *i.e.*, process switching is useful for suspending and resuming of a process.	Example : If a user process needs to access things controlled by kernel, it is necessary to perform mode switching.

```
//wait for available barbers
        get_haircut();
}
                else { //        do nothing (leave) when all chairs
                                                        are used.
        signal(mutex);
                        }
}
cut_hair( ){
        waiting(cutting);
}
get_haircut( ){
get hair cut for some time;
        signal(cutting);
}
```

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Discuss process model and principle of concurrency.

Ans. Refer Q. 2.1 and Q. 2.3.

Q. 2. Write short note on producer consumer problem along with its solution using semaphore.

Ans. Refer Q. 2.4 and Q. 2.16.

Q. 3. What is mutual exclusion ? What are the four conditions of mutual exclusion ?

Ans. Refer Q. 2.5.

Q. 4. Write and explain Peterson solution to the critical section problem.

Ans. Refer Q. 2.11.

Q. 5. Give a solution to reader/writer problem using semaphores.

Ans. Refer Q. 2.19.

Q. 6. Discuss classical problem of concurrency.

Ans. Refer Q. 2.22.

Q. 7. Write short note on interprocess communication.

Ans. Refer Q. 2.23.

☺☺☺

3 UNIT

CPU Scheduling

Part-1 ..... **(3–2B to 3–12B)**

- *CPU Scheduling : Scheduling Concepts*
- *Performance Criteria*
- *Process States*
- *Process Transition Diagram*
- *Schedulers*
- *Process Control Block (PCB)*
- *Process Address Space*
- *Process Identification Information*

A. Concept Outline : Part-1 ..... 3–2B
B. Long and Medium Answer Type Questions 3–2B

Part-2 ..... **(3–12B to 3–44B)**

- *Threads and their Management*
- *Scheduling Algorithms*
- *Multiprocessor Scheduling*
- *Deadlock : System Model*
- *Deadlock Characterization*
- *Prevention*
- *Avoidance and Detection*
- *Recovery from Deadlock*

A. Concept Outline : Part-2 ..... 3–12B
B. Long and Medium Answer Type Questions 3–13B

PART-1

CPU Scheduling : Scheduling Concepts, Performance Criteria, Process States, Process Transition Diagram, Schedulers, Process Control Block (PCB), Process Address Space, Process Identification Information.

CONCEPT OUTLINE : PART-1

- CPU scheduling is the management of CPU resources.
- **Performance scheduling criteria :**
  - i. CPU utilization
  - ii. Throughput
  - iii. Waiting time
  - iv. Turnaround time
  - v. Response time
- **Various states of process :**
  - i. New
  - ii. Running
  - iii. Waiting
  - iv. Ready
  - v. Terminated
- **Types of scheduler :**
  - i. Long term scheduler
  - ii. Short term scheduler
  - iii. Mid term scheduler

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.1. **What is CPU scheduling ? Give the criteria for scheduling.**

OR

Discuss the performance criteria for CPU scheduling.

AKTU 2015-16, Marks 10

Answer

1. CPU scheduling is the management of CPU resources.
2. The scheduling mechanism is the part of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
3. CPU scheduling is the basis of multi-programmed operating systems.
4. The scheduler is responsible for multiplexing processes on the CPU.
5. CPU scheduling is used to increase CPU utilization.

Scheduling criteria : The scheduling policy determines the importance of each of the criteria. Some commonly used criteria are :

1. **CPU utilization :**
   - a. CPU utilization is the average function of time, during which the processor is busy.
   - b. The load on the system affects the level of utilization that can be achieved.
   - c. CPU utilization may range from 0 % to 100 %. On large and expensive system *i.e.*, time shared system, CPU utilization may be the primary consideration.
2. **Throughput :**
   - a. Throughput refers to the amount of work completed in a unit of time.
   - b. The number of processes the system can execute in a period of time.
   - c. The higher the number, the more work is done by the system.
3. **Waiting time :**
   - a. The average period of time a process spends waiting.
   - b. Waiting time may be expressed as turnaround time less the actual execution time.
4. **Turnaround time :**
   - a. The interval from the time of submission of a process to the time of completion is the turnaround time.
   - b. Turnaround time is the sum of the periods spent waiting to get into memory, waiting into the ready queue, executing on the CPU and doing I/O.
5. **Response time :**
   - a. Response time is the time from the submission of a request until the first response is produced.
6. **Priority :**
   - a. Give preferential treatment to processes with higher priorities.

7. **Balanced utilization :**
   a. Utilization of memory, I/O devices and other system resources are also considered.
   b. Not only CPU utilization considered for performance.
8. **Fairness :**
   a. Avoid the process from the starvation.
   b. All the processes must be given equal opportunity to execute.

Que 3.2. **Discuss the objectives of CPU scheduling.**

Answer

Objectives of CPU scheduling :

1. **Efficiency :** Keep the CPU busy all the time.
2. **Maximize throughput :** Service the largest possible number of jobs in a given amount of time; minimize the amount of time users must wait for their results.
3. **Minimize response time :** Interactive users should see good performance.
4. **Minimize overhead :** Do not waste too many resources. Keep scheduling time and context switch time at a minimum.
5. **Maximize resource use :** Favour processes that will use under-utilized resources.
6. **Avoid indefinite postponement :** Every process should get a chance to run eventually.
7. **Enforce priorities :** If the scheduler allows a process to be assigned a priority, it should be meaningful and enforced.

Que 3.3. **What is the difference between pre-emptive and non pre-emptive scheduling ? State, why strict non pre-emptive scheduling is unlikely to be used in computer centre ? Explain the operation of multi-level scheduling.**

Answer

Difference :

S. No.	Pre-emptive scheduling	Non pre-emptive scheduling
1.	Processor can be pre-empted to execute a different process in the middle of execution of any current process.	Once processor starts to execute a process it must finish it before executing the other. It cannot be paused in middle.
2.	CPU utilization is more compared to non pre-emptive scheduling.	CPU utilization is less compared to pre-emptive scheduling.
3.	Waiting time and response time is less.	Waiting time and response time is more.
4.	If a high priority process frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then another process with less CPU burst time may starve.
5.	Pre-emptive scheduling is flexible.	Non pre-emptive scheduling is rigid.
6.	For example, SRTF, Priority, Round-Robin, etc.	For example, FCFS, SJF, Priority, etc.

Non pre-emptive scheduling is unlikely to be used in computer centre :

1. 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.
2. In a general purpose computer system, users share the CPU and care about system responsiveness.
3. If the system uses non pre-emptive scheduling, some users may sit before the monitor for several hours without doing anything other than waiting for the set of processes in front of them in the system queue to finish.
4. So, strictly non pre-emptive scheduling is unlikely to be used in a general purpose computer system.

Operation of multilevel scheduling :

1. A multilevel queue scheduling algorithm partitions the ready queue into several separate queues (Fig. 3.3.1).
2. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.
3. Each queue has its own scheduling algorithm.
4. For example, separate queues might be used for foreground and background processes.
5. The foreground queue might be scheduled by an RR algorithm, while the background queue is scheduled by an FCFS algorithm.

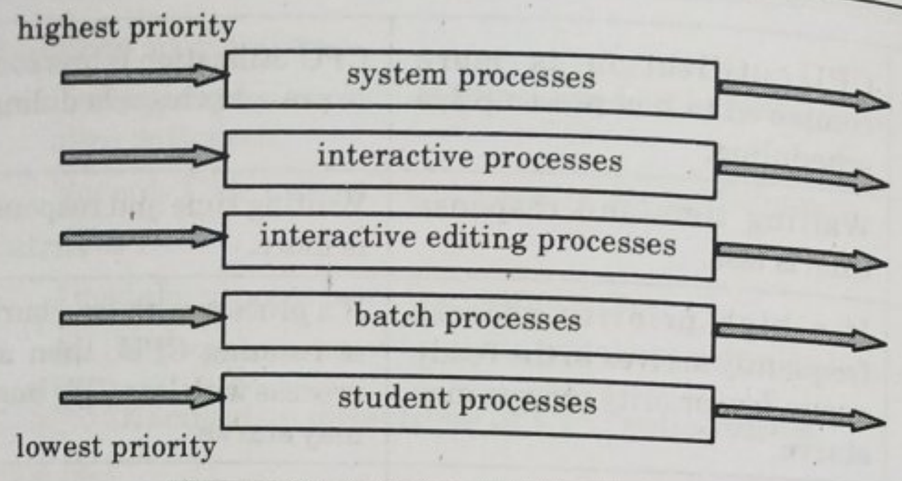


Fig. 3.3.1. Multilevel queue scheduling.

6. There must be scheduling among the queues, which is commonly implemented as fixed-priority pre-emptive scheduling.
7. For example, the foreground queue may have absolute priority over the background queue.

Que 3.4. **Explain the objectives and implementation of short term scheduling and long term scheduling.**

Answer

Objective and implementation of short term scheduling :

1. The main objective is increasing system performance in accordance with the chosen set of criteria.
2. It is the change of ready state to running state of the process.
3. It is also called CPU scheduler.
4. CPU scheduler selects from among the processes that are ready to execute and allocates the CPU to one of them.

Objective and implementation of long term scheduling :

1. The primary objective is to provide a balanced mix of jobs, such as I/O bound and processor bound.
2. It also controls the degree of multi-programming.
3. If the degree of multi-programming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.
4. Long term scheduler determines which programs are admitted to the system for processing.
5. It is also called job scheduler.
6. Job scheduler selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduler.

Que 3.5. **List out the various states of process.**

OR

Draw the process state diagram and describe the various process states.

OR

What are the various process states ? Depict process state diagram. What do you understand by context switching and various processes involved in it.

OR

Draw and explain the process state transition diagram.

AKTU 2011-12, Marks 05

Answer

Various process states are :

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.

Process state diagram :

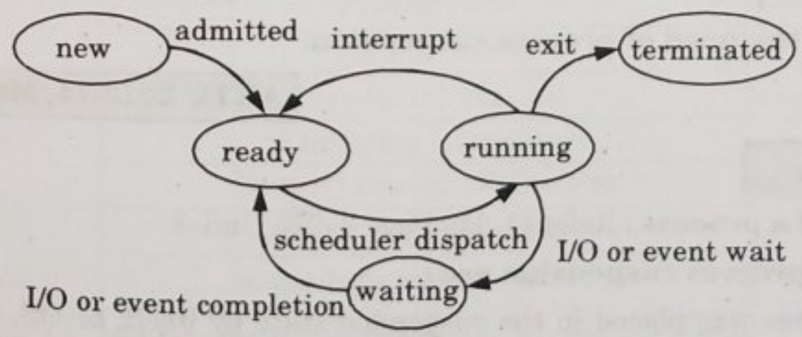


Fig. 3.5.1.

Context switching :

1. A context switching is the mechanism to store and restore the state or context of a CPU in process control block so that a process execution can be resumed from the same point at a later time.
2. Any operating system that allows for multitasking relies heavily on the use of context switching to allow different processes to run at the same time.
3. A context switch occurs when a computer's CPU switches from one process or thread to a different process or thread.

4. Context switching allows for one CPU to handle numerous processes or threads without the need for additional processors.

Process involved in context switching are :

The steps in a full process switch are :

1. Save the context of the processor, including program counter and other registers.
2. Update the process control block of the process that is currently in the running state. This includes changing the state of the process to one of the other states (Ready; Blocked; Ready/Suspend or Exit). Other relevant fields must also be updated, including the reason for leaving the running state and accounting information.
3. Move the process control block of this process to the appropriate queue (Ready; Blocked on Event *i*; Ready/Suspend).
4. Select another process for execution.
5. Update the process control block of the process selected. This includes changing the state of this process to running.
6. Update memory management data structures. This may be required, depending on how address translation is managed.
7. Restore the context of the processor to that which existed at the time the selected process was last switched out of the running state, by loading in the previous values of the program counter and other registers.

Que 3.6. **Define the different states of a process with diagram. Explain the need of process suspension.**

AKTU 2013-14, Marks 05

Answer

States of a process : Refer Q. 3.5, Page 3–7B, Unit-3.

Need of process suspension are :

The process was placed in the suspended state by itself, or OS, a parent process for the purpose of preventing its execution. There are following needs of process suspension :

1. **Parent process request :** A process may be suspended in order of parent process request in order to modify the process.
2. **Swapping :** If the process is ready and there is no place in the main memory, the process get suspended.
3. **Other OS reason :** When one process in main memory which was blocked and there is another process ready for execution, but waiting in secondary memory, then the process in main memory is suspended.

Que 3.7. **What is process control block ? Also list out the information it contains.**

OR

What is the process control block ? Explain all its components.

Answer

Process control block :

1. Process control block is a data structure used to store the information about the processes.
2. The information of the process is used by CPU at the runtime.

The following are the various information that is contained by process control block :

1. Naming the process
2. State of the process
3. Resources allocated to the process
4. Memory allocated to the process
5. Scheduling information
6. Input / output devices associated with process

Components of process control block :

Process Id
Process state
Program counter
Register information
Scheduling information
Memory related information
Accounting information
Status information related to I/O

Fig. 3.7.1. Structure of process control block.

The following are the various components that are associated with the process control block (PCB) :

1. **Process ID :**
   a. In computer system, there are various processes running simultaneously and each process has its unique Id.
   b. This Id helps system in scheduling the processes.
   c. This Id is provided by the process control block.
   d. In other words, it is an identification number that uniquely identifies the processes of computer system.

2. **Process state :**
   a. As we know that the process state of any process can be new, running, waiting, executing, blocked, suspended, terminated.
   b. Process control block is used to define the process state of any process.
   c. In other words, process control block refers the states of the processes.
3. **Program counter :**
   a. Program counter is used to point to the address of the next instruction to be executed in any process.
   b. This is also managed by the process control block.
4. **Register information :**
   a. It comprises of the various registers, such as index and stack that are associated with the process.
   b. This information is also managed by the process control block.
5. **Scheduling information :**
   a. Scheduling information is used to set the priority of different processes.
   b. This is very useful information which is set by the process control block.
   c. The priority of primary feature of RAM is higher than other secondary features.
   d. Scheduling information is very useful in managing any computer system.
6. **Memory related information :**
   a. This section of the process control block comprises of page and segment tables.
   b. It also stores the data contained in base and limit registers.
7. **Accounting information :**
   a. This section of process control block stores the details relate to central processing unit (CPU) utilization and execution time of a process.
8. **Status information related to input / output :**
   a. This section of process control block stores the details pertaining to resource utilization and file opened during the process execution.

Que 3.8. **Differentiate between long term, short term and mid term schedulers.**

OR

With a diagram, explain the different states of a process. Differentiate between long term and short term schedulers.

Answer

Different states of process : Refer Q. 2.2, Page 2–3B, Unit-2.

Difference between various schedulers :

S. No.	Long term	Short term	Mid term
1.	It is job scheduler.	It is CPU scheduler.	It is process swapping scheduler.
2.	Speed is less than short term scheduler.	Speed is very fast.	Speed is in between both.
3.	It controls degree of multi-programming.	Less control over degree of multi-programming.	Reduced the degree of multi-programming.
4.	Absent or minimal in time sharing system.	Minimal in time sharing system.	Time sharing system use mid term scheduler.
5.	It selects process from pool and loads them into memory for execution.	It select among the processes that are ready to execute.	Process can be reintroduced into memory and its execution can be continued.

Que 3.9. **Describe process address space.**

Answer

1. Process address space means a space that is allocated in memory for a process.
2. Address space is a space in computer memory.
3. Every process has an address space.
4. Address space can be of two types :
   a. **Physical address space :** Physical address space is created in RAM.
   b. **Virtual address space :** Virtual address space is an address space that is created outside the main memory inside the virtual memory, and it is created in hard disk.

Que 3.10. **Write a short note on process identification information.**

Answer

1. Process registration involves recording all information necessary to identify a registered process and to differentiate it from other processes

in the system. This information is called process identification information.

2. This information is recorded right after the process is created or born.
3. Usually, every process is also given a unique identifier, called a process ID, in order to make future references easier and unambiguous.
4. Process identification information is usually static, which means it does not change as time goes by and as the process moves from one state of life to another, for that purpose, the system will record process state information.
5. Process identification information use following numeric identifiers :
   a. **Unique process identifier (PID) :** PID provide index id to the process.
   b. **User identifier (UID) :** UID identifies the user who is responsible for the job.

PART-2

Threads and their Management, Scheduling Algorithms, Multiprocessor Scheduling, Deadlock : System Model, Deadlock Characterization, Prevention, Avoidance and Detection, Recovery from Deadlock.

CONCEPT OUTLINE : PART-2

- A thread is a flow of execution through the process code, with its own program counter, system registers and stack.
- **Types of thread :**
  i. User level thread ii. Kernel level thread
- **Scheduling algorithm :**
  i. FCFS ii. SJF
  iii. Priority scheduling
  iv. Round Robin scheduling
  v. Multilevel queue scheduling
  vi. Multilevel feedback queue scheduling
- **Necessary conditions for deadlock :**
  i. Mutual exclusion ii. Hold and wait
  iii. Circular wait iv. No pre-emption

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 3.11. **What is thread ? Discuss its advantages.**

Answer

1. A thread is a flow of execution through the process code, with its own program counter, system registers and stack.
2. Threads represent a software approach to improving performance of operating system by reducing the overhead of process switching.
3. Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control.
4. They also provide a suitable foundation for parallel execution of applications on shared memory multi-processors.

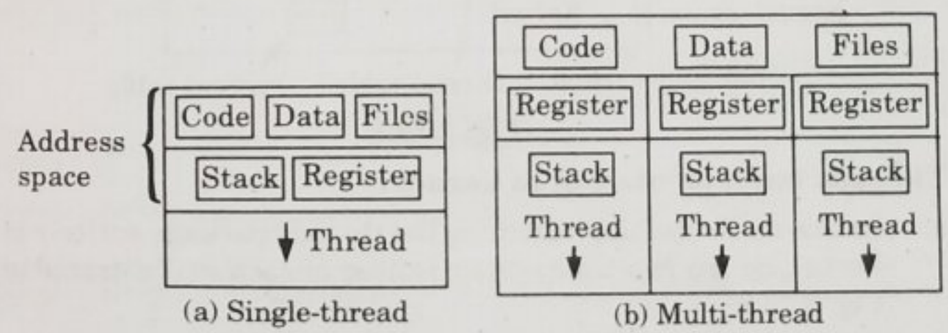


Fig. 3.11.1. Single and multi-thread processes.

Advantages of thread :

1. Thread minimize context switching time.
2. Use of thread provides concurrency within a process.
3. Efficient communication.
4. It is more economical to create and context switch threads.
5. The benefits of multi-threading can be greatly increased in a multi-processor architecture.

Que 3.12. **How threads are implemented ?**

Answer

To implement a threads package, there are the following three ways :

1. **Threads implementation in user space :**
   a. Threads could be implemented at the user level.

b. A user-level thread is known only to the user or actually the process that has created it.

c. These tasks include but are not restricted to : CPU assignment, device assignment, state transition, etc.

d. To be clearer, the operating system will not know that a process has created any threads and hence cannot help managing them.

e. What the operating system does, in respect to user-level threads, is to provide a set of basic routines for creation, manipulation, and destruction of threads.

f. These can be used by the user whenever needed, without any managerial responsibility as the part of the operating system.

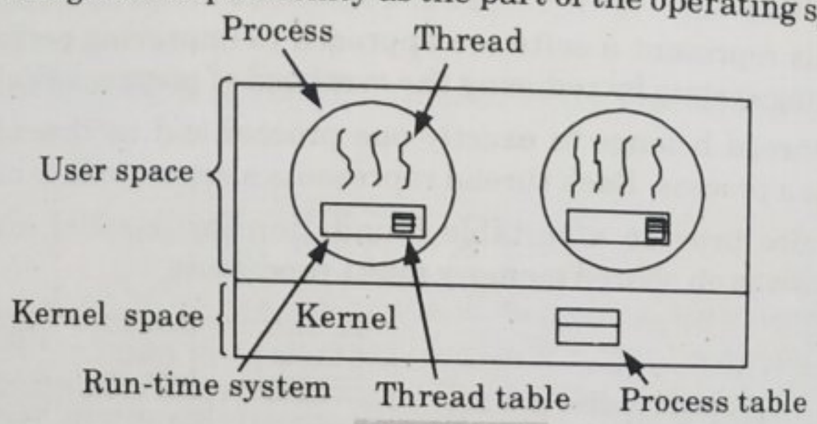


Fig. 3.12.1.

2. **Threads implementation in kernel :**

a. In this method of implementing the threads package entirely in the kernel, no any run-time system is need in each as illustrated in the Fig. 3.12.2.

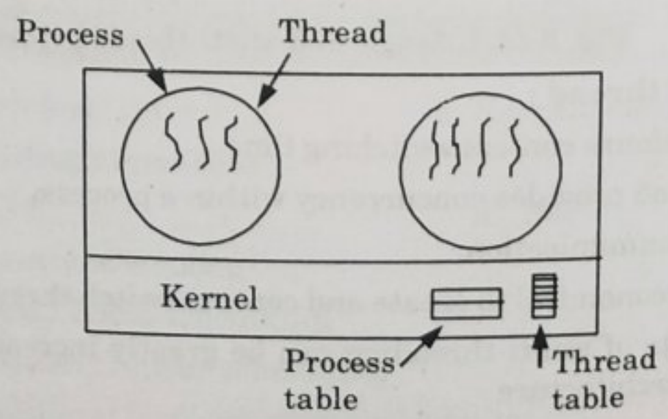


Fig. 3.12.2.

b. In this, there is no any thread table in each process. But to keep track of all the threads in the system, the kernel has the thread table.

c. Whenever a thread wants to create a new thread or destroy an existing thread, then it makes a kernel call, which does the creation or destruction just by updating the kernel thread table.

d. The thread table of the kernel holds each registers, state, and some other useful information of the thread.

e. Here the information is the same as with the user level threads.

f. This information is a subset of the information that traditional kernels maintain about each of their single-threaded processes, that is, the process state.

g. In addition to these, to keep track of processes, the kernel also maintains the traditional process table.

3. **Hybrid implementation :**

a. The third type of thread implementation is hybrid implementation that is the implementation of a combination of user level and kernel level threads.

b. The managerial responsibility of this kind of thread is performed partially by the user who has created the thread and partially by the operating system.

Que 3.13. **Differentiate between user level and kernel level thread.**

AKTU 2012-13, Marks 2.5

Answer

Difference between user level and kernel level thread :

S. No.	User level threads	Kernel level threads
1.	User level threads are faster to create and manage.	Kernel level threads are slower to create and manage.
2.	Implemented by a thread library at the user level.	Operating system support directly to kernel threads.
3.	User level thread can run on any operating system.	Kernel level threads are specific to the operating system.
4.	Support provided at the user level called user level thread.	Support may be provided by kernel is called kernel level thread.
5.	Multi-thread application cannot take advantage of multi-processing.	Kernel routines themselves can be multi-threaded.

Que 3.14. **What are the types of thread ? Give advantages and disadvantages.**

Answer

Types of thread are :

1. **User level thread :**
   i. In a user level thread, all of the work of thread management is done by the application.
   ii. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts.
   iii. The application begins with a single thread and begins running in that thread.

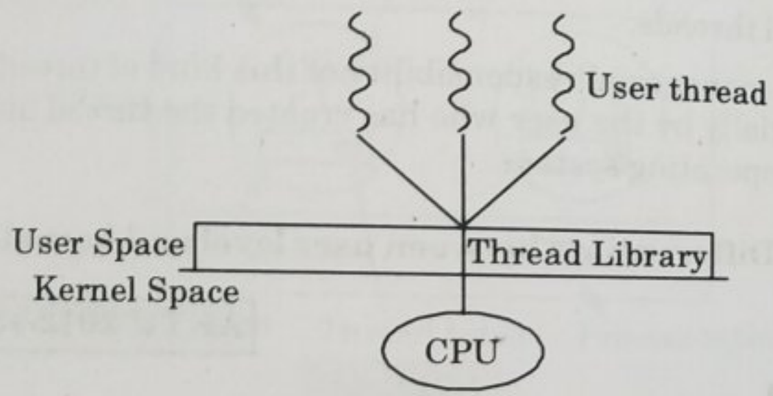


Fig. 3.14.1. User level thread.

Advantages of user level thread :

i. Thread switching does not require kernel mode privileges.
ii. User level thread can run on any operating system.
iii. Scheduling can be application specific.
iv. User level threads are fast to create and manage.

Disadvantages of user level thread :

i. In a typical operating system, most system calls are blocking.
ii. Multi-threaded application cannot take advantage of multi-processing.

2. **Kernel level thread :**
   i. In kernel level thread, thread management is done by the kernel. There is no thread management code in the application area.
   ii. Kernel threads are supported directly by the operating system.
   iii. All of the threads within an application are supported within a single process.
   iv. Scheduling by the kernel is done on a thread basis.
   v. The kernel performs thread creation, scheduling and management

Advantages of kernel level thread :

i. Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
ii. If one thread in a process is blocked, the kernel can schedule another thread of the same process.
iii. Kernel routines themselves can multithread.

Disadvantages of kernel level thread :

i. Kernel threads are generally slower to create and manage than the user threads.
ii. Transfer of control from one thread to another within same process requires a mode switch to the kernel.

Que 3.15. **What is a thread ? How thread is different from a process ? What resources are used when a thread is created ?**

AKTU 2014-15, Marks 10

Answer

Thread : Refer Q. 3.11, Page 3–13B, Unit-3.

Difference between thread and process :

S. No.	Thread	Process
1.	Thread is called light weight process.	Process is called heavy weight process.
2.	Thread switching does not need to call an operating system and cause an interrupt to the kernel.	Process switching needs interface with operating system.
3.	All threads can share same set of open files, child processes.	In multiple process implementation each process executes the same code but has its own memory and file resources.
4.	While one server thread is blocked and waiting, second thread in the same task could run.	If one server process is blocked no other server process can execute until the first process unblocked.
5.	One thread can read, write or even completely wipe out another threads stack.	In multiple process each process operates independently of the other.

Resources used when thread is created : When a thread is created, the thread does not require any new resource to execute, the thread shares the resources like memory of the process to which they belong to.

Que 3.16. **Differentiate between user thread and kernel thread. What is thread cancellation ?** **AKTU 2013-14, Marks 05**

Answer

User thread vs Kernel thread : Refer Q. 3.13, Page 3–15B, Unit-3.

Thread cancellation :

1. Thread cancellation is the task of terminating a thread before it has completed.
2. For example, if multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be canceled.
3. Another situation might occur when a user presses a button on a web browser that stop a web page from loading any further.
4. Often, a web page is loaded using several threads and each image is loaded in a separate thread.
5. When a user presses the stop button on the browser, all threads loading the page are canceled.

Que 3.17. **Discuss First Come First Serve (FCFS) scheduling algorithm. Give its advantages and disadvantages.**

Answer

1. The simplest scheduling algorithm is the First Come First Served (FCFS) algorithm, *i.e.*, the process, which requests the CPU first is allocated the CPU first.
2. The FCFS algorithm is simply realized with a FIFO queue. It functions as follows :
   a. When a process enters the ready queue, its Process Control Block (PCB) is linked onto the tail of the queue.
   b. As soon as the CPU is free, it is allocated to the process at the head of the ready queue.
   c. The running process is then removed from the ready queue.

Advantages of First Come First Served (FCFS) algorithm :

1. Better for long processes.
2. Simple method (*i.e.*, minimum overhead on processor).
3. No starvation.

Disadvantages of First Come First Served (FCFS) algorithm :

1. Convoy effect occurs. Even very small process should wait for its turn to come to utilize the CPU.

2. Short process behind long process results in lower CPU utilization.
3. Throughput is not emphasized.

Que 3.18. **Discuss Shortest Job First (SJF) scheduling algorithm. Also, write its advantages and disadvantages.**

Answer

1. SJF algorithm associates with each process the length of the process's next CPU burst.
2. When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
3. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.
4. The appropriate term for this scheduling method would be the shortest-next-CPU-burst algorithm, because scheduling depends on the length of the next CPU burst of a process, rather that it's total length.
5. The SJF algorithm can be either preemptive or non-preemptive.
6. The choice arises when a new process arrives at the ready queue while a previous process is still executing.
7. The next CPU burst of the newly arrived process may be shorter than what is left of the currently executing process.
8. A preemptive SJF algorithm will preempt the currently executing process, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst.
9. Preemptive SJF scheduling is sometimes called shortest remaining time first scheduling.

Advantages of Shortest Job First (SJF) algorithm :

1. It gives superior turnaround time performance to shortest process next because a short job is given immediate preference to a running longer job.
2. Throughput is high.

Disadvantages of Shortest Job First (SJF) algorithm :

1. Elapsed time (*i.e.*, execution-completed-time) must be recorded, it results an additional overhead on the processor.
2. Starvation may be possible for the longer processes.

Que 3.19. **Discuss priority scheduling algorithm. What are its advantages and disadvantages ?**

Answer

1. Priority scheduling is a non pre-emptive algorithm and one of the most common scheduling algorithms in batch systems.

2. Each process is assigned a priority. Process with highest priority is to be executed first and so on.
3. Processes with same priority are executed on FCFS basis.
4. Priority can be decided based on memory requirements, time requirements or any other resource requirements.

Advantages of priority scheduling algorithm :

1. The priority of a process can be selected based on memory requirement, time requirement or user preference.
2. For example, a high end game will have better graphics that means the process which updates the screen in a game will have higher priority so as to achieve better graphics performance.
3. Priority scheduling provides a good mechanism where the relative importance of each process may be precisely defined.

Disadvantages of priority scheduling algorithm :

1. If high priority processes use up a lot of CPU time, lower priority processes may starve and be postponed indefinitely.
2. The situation where a process never gets scheduled to run is called starvation.
3. Another problem with priority scheduling is deciding which process gets which priority level assigned to it.

Que 3.20. **Discuss Round Robin scheduling algorithm. Give its advantages and disadvantages.**

Answer

1. The Round Robin (RR) scheduling algorithm is designed especially for time sharing systems.
2. It is similar to FCFS scheduling, but pre-emption is added to enable the system to switch between processes.
3. The ready queue is treated as a circular queue.
4. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.
5. A small unit of time is called a time quantum or time slice.
6. A time quantum is generally from 10 to 100 milliseconds in length.
7. To implement RR scheduling, we again treat the ready queue as a FIFO queue of processes.
8. New processes are added to the tail of the ready queue.
9. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.

Advantages of Round Robin (RR) algorithm :

1. Round-Robin is effective in a general-purpose, time sharing system or transaction processing system.
2. Fair treatment for all the processes.
3. Overhead on processor is low.
4. Good response time for short processes.

Disadvantages of Round Robin (RR) algorithm :

1. Care must be taken in choosing quantum value.
2. Processing overhead is there in handling clock interrupt.
3. Throughput is low if time quantum is too small.

Que 3.21. **What is CPU scheduling? Discuss the multilevel feedback queue scheduling.** **AKTU 2012-13, Marks 10**

OR

Explain the functioning of multilevel feedback queue scheduling. **AKTU 2013-14, Marks 10**

Answer

CPU scheduling : Refer Q. 3.1, Page 3–2B, Unit-3.

Multilevel feedback queue scheduling :

1. Multilevel Feedback Queue (MFQ) scheduling algorithm overcomes the problem of multilevel queue scheduling algorithm.
2. MFQ means to separate processes with different CPU burst time. If a process uses too much CPU time, it will be moved to a lower priority queue.
3. MFQ allows a process to move between the queues.
4. MFQ implements two or more scheduling queues.

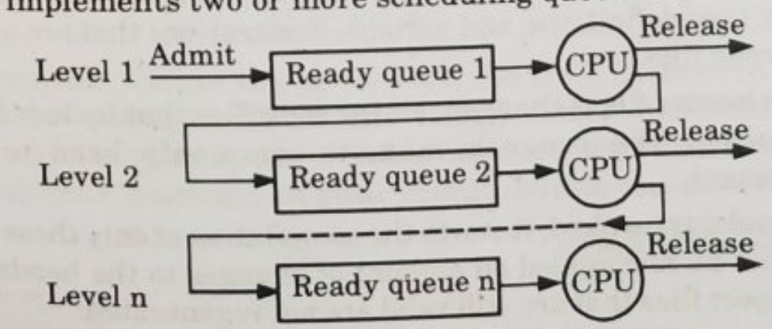


Fig. 3.21.1. Multilevel feedback queue.

5. For example, each process may start at the top level queue. If the process is completed within a given time slice, it departs the system.
6. Processes that need more than one time slice may be reassigned by the operating system to a lower priority queue, which gets a lower percentage of the processor time.

Multilevel feedback queue scheduler is defined by the following parameters.

1. The number of queues.
2. Scheduling algorithm for each queue.
3. Method used to determine when to denote a process to a lower priority queue.
4. Method used to determine when to upgrade a process to a higher priority queue.
5. Method used to determine which queue a process will enter when that process needs service.

Que 3.22. Explain the following scheduling algorithms :

i. Multilevel feedback queue scheduling

ii. Multiprocessor scheduling **AKTU 2014-15, Marks 10**

Answer

i. Multilevel feedback queue scheduling : Refer Q. 3.21, Page 3-21B, Unit-3.

ii. Multiprocessor scheduling :

1. On a multiprocessor, scheduling is two dimensional. The scheduler has to decide which process to run and which CPU to run it on. This extra dimension greatly complicates scheduling on multiprocessors.
2. Another complicating factor is that in some systems, all the processes are unrelated whereas in others they come in groups.
3. An example of the former situation is a timesharing system in which independent users start up independent processes. The processes are unrelated and each one can be scheduled without regard to the other ones.
4. Large systems often consist of some number of header files containing macros, type definitions, and variable declarations that are used by the actual code files.
5. When a header file is changed, all the code files that include it must be recompiled. The program make is commonly used to manage development.
6. When make is invoked, it starts the compilation of only those code files that must be recompiled on account of changes to the header or code files. Object files that are still valid are not regenerated.
7. The original version of make did its work sequentially, but newer versions designed for multiprocessors can start up all the compilations at once.

Que 3.23. Consider a variant of Round Robin scheduling algorithm where the entries in the ready queue are pointers to the

processes. What would be the effect of putting two pointers to the same process in the ready queue ? What would be advantages and disadvantages of this scheme ? **AKTU 2012-13, Marks 05**

Answer

1. Process appears twice in the ready queue and is scheduled twice as often as other processes.
2. In effect, that process will have increased its priority since by getting time more often it is receiving preferential treatment. It will double the time given to that process.
3. The advantage of this scheme is that more important jobs could be given more time. It will provide higher priority with minimal modification to scheduler.
4. The disadvantage of this scheme is that the shorter job will suffer because important jobs will execute with more time. There will be overhead for managing pointers. It may also increase overhead if same process runs back-to-back.

Que 3.24. Consider the set of the processes given in the table and the following scheduling algorithms :

i. Round Robin (Quantum = 1)

ii. Round Robin (Quantum = 2)

iii. Shortest Remaining Job First

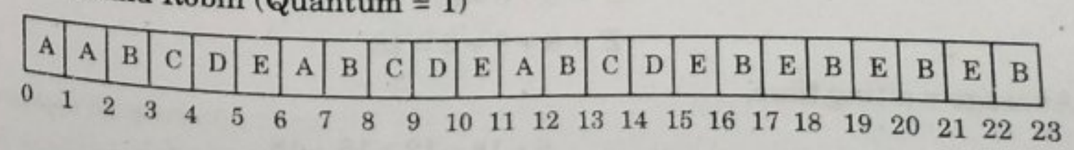
Process Id	Arrival Time	Execution Time
A	0	4
B	2	7
C	3	3
D	3.5	3
E	4	5

If there is tie within the processes, the tie is broken in the favour of the oldest process. Draw the Gantt chart and find the average waiting time, response time and turnaround time for the algorithms. Comment on your result. Which one is better and why ?

AKTU 2013-14, Marks 10

Answer

i. Round Robin (Quantum = 1)



Waiting time for process,

$$A = 0 + 4 + 4 = 8$$
$$B = 2 + 4 + 4 + 3 + 1 + 1 + 1 = 16$$
$$C = 3 + 4 + 4 = 11$$
$$D = 4 + 4 + 4 = 12$$
$$E = 5 + 4 + 4 + 1 + 1 + 1 = 16$$

$$\text{Average waiting time} = \frac{8 + 16 + 11 + 12 + 16}{5} = 12.6$$

Turnaround time for process,

$$A = 12$$
$$B = 21$$
$$C = 11$$
$$D = 11.5$$
$$E = 18$$

Average turnaround time,

$$= \frac{12 + 21 + 11 + 11.5 + 18}{5} = 14.7$$

Response time for process,

$$A = 0$$
$$B = 0$$
$$C = 0$$
$$D = 4 - 3.5 = 0.5$$
$$E = 5 - 1 = 4$$

Average response time,

$$= \frac{0.5 + 4}{5} = 0.9$$

ii. Round Robin (Quantum = 2)

A	B	C	D	E	A	B	C	D	E	B	E	B

0 2 4 6 8 10 12 14 15 16 18 20 21 22

Waiting time for process,

$$A = 8$$
$$B = 2 + 8 + 6 + 2 = 18$$
$$C = 4 + 8 = 12$$
$$D = 8 + 8 = 16$$
$$E = 8 + 8 + 2 = 18$$

Average waiting time,

$$= \frac{8 + 18 + 12 + 16 + 18}{5} = 14.4$$

Turnaround time for process,

$$A = 12$$
$$B = 20$$
$$C = 11$$
$$D = 12.5$$
$$E = 17$$

Average turnaround time,

$$= \frac{12 + 20 + 11 + 12.5 + 17}{5} = 14.5$$

Response time for process,

$$A = 0$$
$$B = 0$$
$$C = 4 - 3 = 1$$
$$D = 6 - 3.5 = 2.5$$
$$E = 8 - 4 = 4$$

Average response time,

$$= \frac{7.5}{5} = 1.5$$

iii. Shortest Remaining Job First

A	A	A	C	D	E	B

0 2 3 4 7 10 15 22

Waiting time for process,

$$A = 0$$
$$B = 15$$
$$C = 4$$
$$D = 7$$
$$E = 10$$

Average waiting time,

$$= \frac{0 + 15 + 4 + 7 + 10}{5} = 7.2$$

Turnaround time for process,

$$A = 4$$
$$B = 20$$
$$C = 4$$
$$D = 6.5$$
$$E = 6$$

Average turnaround time,

$$= \frac{4 + 20 + 4 + 6.5 + 6}{5} = 8.1$$

Response time for process,

$$A = 0$$
$$B = 15 - 2 = 13$$
$$C = 4 - 3 = 1$$
$$D = 7 - 3.5 = 3.5$$
$$E = 10 - 4 = 6$$

$$\text{Average response time} = \frac{23.5}{5} = 4.7$$

SJF gives the minimum average waiting time as compared to RR algorithm. So, it is the best scheduling algorithm.

Que 3.25. **Consider the processes, CPU burst time and arrival time given below :**

Processes	CPU burst time	Arrival time
P_1	8	0
P_2	4	1
P_3	9	2
P_4	5	3

Draw the Gantt chart and calculate the following by using SRTF CPU scheduling algorithm,

i. Average waiting time

ii. Average turnaround time

AKTU 2015-16, Marks 10

Answer

Processes	Arrival time	Burst time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Gantt Chart :

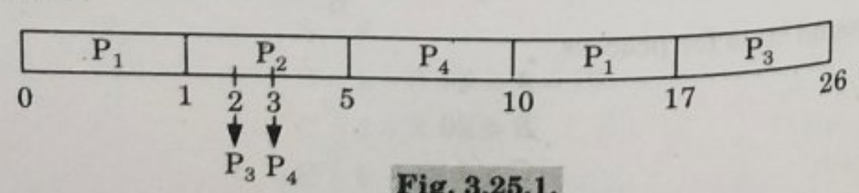


Fig. 3.25.1.

Waiting time,

$P_1 = 10 - 1 - 0 = 9$

$P_2 = 1 - 1 = 0$

$P_3 = 17 - 2 = 15$

$P_4 = 5 - 3 = 2$

$$\text{Average waiting time} = \frac{9 + 0 + 15 + 2}{4} = \frac{26}{4} = 6.5$$

Turnaround time (TAT),

$P_1 = 17 - 0 = 17$

$P_2 = 5 - 1 = 4$

$P_3 = 26 - 2 = 24$

$P_4 = 10 - 3 = 7$

$$\text{Average turnaround time} = \frac{17 + 4 + 24 + 7}{4} = \frac{52}{4} = 13$$

Que 3.26. **List various performance criteria for scheduling algorithms. Five processes A, B, C, D, E require CPU burst of 3, 5, 2, 5 and 5 units respectively. Their arrival times in the system are 0, 1, 3, 9 and 12 respectively. Draw Gantt Chart and compute the average turnaround time and average waiting time of these processes for the Shortest Job First (SJF) and Shortest Remaining Time First (SRTF) scheduling algorithms.**

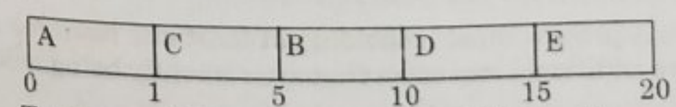
AKTU 2011-12, Marks 10

Answer

Performance criteria for scheduling algorithm : Refer Q. 3.1, Page 3–2B, Unit-3.

Numerical :

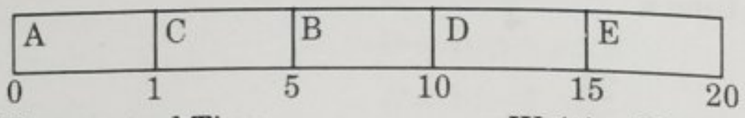
SJF :



Turnaround Time	Waiting Time
A = 3	0
B = 9	4
C = 2	0
D = 6	1

E = 8	3
Average turnaround time	(3 + 9 + 2 + 6 + 8)/5 = 5.6 ms
Average waiting time	(0 + 4 + 0 + 1 + 3)/5 = 1.6 ms

SRTF :



Turnaround Time	Waiting Time
A = 3	0
B = 9	4
C = 2	0
D = 6	1
E = 8	3
Average turnaround time	(3 + 9 + 2 + 6 + 8)/5 = 5.6 ms
Average waiting time	(0 + 4 + 0 + 1 + 3)/5 = 1.6 ms

Que 3.27. What is deadlock ? What are the necessary conditions for deadlock ?

OR

What are the necessary conditions to hold a deadlock in a system ?

AKTU 2013-14, Marks 05

OR

What is a deadlock ? Discuss the necessary conditions for deadlock with examples.

AKTU 2016-17, Marks 7.5

Answer

1. A deadlock is a situation where a group of processes are permanently blocked as a result of each process having acquired a subset of the resources needed for its completion and waiting for release of the remaining resource held by other in the same group, thus making it impossible for any of the processes to proceed.
2. Resource managers and other operating system, processes can be involved in a deadlock situation.

A deadlock situation can arise if the following four conditions holds simultaneously in a system :

1. **Mutual exclusion :** At least one resource must be held in a non-sharable mode; that is, only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.
2. **Hold and wait :** A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.
3. **No pre-emption :** Resources cannot be pre-empted; *i.e.*, a resource can be released only voluntarily by the process holding it, after that process has completed its task.

4. **Circular wait :** A set $\{P_0, P_1, ..., P_n\}$ of waiting processes must exist such that P_0 is waiting for a resource held by P_1, P_1 is waiting for a resource held by P_2, ..., P_{n-1} is waiting for a resource held by P_n, and P_n is waiting for a resource held by P_0.

Que 3.28. How deadlock are prevented ?

OR

Define and explain the deadlock prevention.

OR

What are the approaches that can be used for prevention of deadlock ?

AKTU 2013-14, Marks 05

Answer

Deadlock prevention is an approach which ensures that system will never enter in deadlock state.

These are following approaches of deadlock prevention :

1. **Mutual exclusion :**
   a. Mutual exclusion condition must hold for non-sharable resources.
   b. If access to a resource requires mutual exclusion, then mutual exclusion must be supported by the operating system.
2. **Hold and wait :**
   a. The hold and wait condition can be eliminated by forcing a process to release all resources held by it, whenever it requests a resource that is not available.
   b. For example, process copies data from a floppy disk to a hard disk, sort a disk file and then prints the results to a printer.
   c. If all the resources must be requested at the beginning of the process, then the process must initially request the floppy disk, hard disk and a printer.
   d. It will hold the printer for its entire execution, even though it needs the printer only at the end.
3. **No pre-emption :**
   a. If a process holding certain resources is denied a further request. That process must release its original resources and if necessary request them again, together with additional resource.
   b. If a process requests a resource that is currently held by another process, the operating system may preempt the second process and require it to release its resources.
   c. Pre-emption is possible for certain types of resources, such as CPU and main memory.
4. **Circular wait :**
   a. One way to prevent the circular wait condition is by linear ordering of different types of system resources.

b. In this, system resources are divided into different classes.

c. If a process has been allocated resources of type R, then it may subsequently request only those resource types following R in the ordering.

Que 3.29. What is deadlock avoidance ? Define all states of a system.

Answer

1. Deadlock avoidance allows the three necessary conditions but makes judicious choices to assure that the deadlock point is never reached.
2. Deadlock avoidance therefore allows more concurrency than prevention does.
3. Deadlock avoidance requires additional information about how resources are to be requested.
4. With deadlock avoidance, a decision is made dynamically whether the current resource allocation request could, if granted, potentially lead to a deadlock.

Two approaches are used to avoid the deadlock :

1. Do not start a process if its demands might lead to deadlock.
2. Do not grant an incremental resource request to a process if this allocation might lead to deadlock.

System can be in one of the following states :

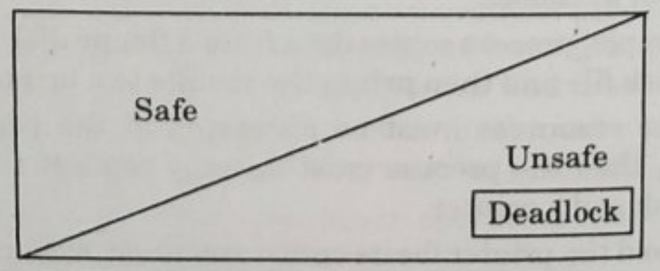


Fig. 3.29.1. Relationship between safe, unsafe and a deadlock state.

1. **Safe state :** Such a state occurs when the system can allocate resources to each process (up to its maximum) in some order and avoid a deadlock. This state will be characterized by a safe sequence.
2. **Unsafe state :** If the system did not follow the safe sequence of resource allocation from the beginning and it is now in a situation, which may lead to a deadlock, then it is in an unsafe state.
3. **Deadlock state :** If the system has some circular wait condition existing for some processes, then it is in deadlock.

Que 3.30. Describe resource-allocation graph.

Answer

1. Resource-allocation graph is used to describe the deadlock. It is also called system resource allocation graph.
2. Graph consists of a set of vertices (V) and set of edges (E).
3. All the active processes in the system denoted by $P = \{P_1, P_2, ... P_n\}$ and set consisting of all resource type in the system is denoted by $R = \{R_1, R_2, R_3, ... R_m\}$.
4. Request edge is an edge from process to resource and denoted by $P_i \rightarrow R_j$.
5. An assignment edge is an edge from resource to process and denoted by $R_j \rightarrow P_i$.
6. Holding of resource by process is denoted by assignment edge.
7. Requesting of resource by process is denoted by request edge.
8. For representing process and resource in the resource allocation graph is shown by square and circle. Each process is represented by circle and resource by square. Dot within the square represents the number of instance.
9. Consider the following system which consists of three processes *i.e.*, P_1, P_2 and P_3 and four resources *i.e.*, R_1, R_2, R_3 and R_4. Resource R_1 and R_3 have one instance, R_2 has two instance and R_4 has three instances.

i. Process P_1 is holding an instance of resource type R_2 and is waiting for an instance of resource type R_1.

ii. Process P_2 is holding an instance of R_1 and R_2 and waiting for an instance of resource type R_3.

iii. Process P_3 is holding an instance of R_3.

$$P = \{P_1, P_2, P_3\}$$
$$R = \{R_1, R_2, R_3, R_4\}$$
$$E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1, \rightarrow P_2, R_2 \rightarrow P_2, R_2, \rightarrow P_1, R_3 \rightarrow P_3\}$$

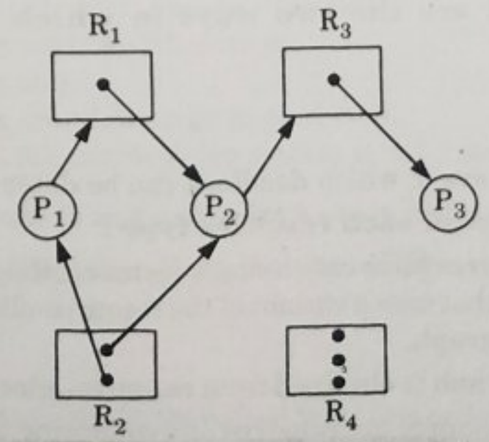


Fig. 3.30.1. Resource-allocation graph.

Que 3.31. **Define deadlock detection. Write an algorithm for deadlock detection.**

Answer

1. Deadlock detection is the process of actually determining that a deadlock exists and identifying the processes and resources involved in the deadlock.
2. The deadlock detection algorithm is invoked when the allocation cannot be granted immediately and this condition occurs frequently.
3. The basic idea is to check allocation against resource availability for all possible allocation sequences to determine if the system is in deadlocked state.

Deadlock detection algorithm :

Step 1 : Let work and finish be vector of length m and n respectively.
Initialize
Work = Available
If $\text{Allocation}_i \neq 0$, then Finish[i] = false
Otherwise, Finish[i] = true For i = 1, 2, ... n

Step 2 : Find an index i such that both
Finish[i] = false
$\text{Request}_i \leq \text{Work}$

Step 3 : If no such i exists, go to step 4.

Step 4 : $\text{Work} = \text{Work} + \text{Allocation}_i$
Finish[i] = true
Go to step 2

Step 5 : If [i] = false, for some i, $1 \leq i \leq n$ then system is in deadlock state.
Moreover, if Finish [i] = false, then P_i is deadlocked.

Que 3.32. **What are the two ways in which deadlock can be detected ?**

Answer

Following are the ways in which deadlock can be detected :

i. **Single instance of each resource type :**
   1. If all resources have only a single instance, then deadlock detection algorithm that uses a variant of the resource-allocation graph, called a wait-for graph.
   2. Wait-for graph is obtained from resource-allocation graph.
   3. Nodes of resource are removed and collapsing the appropriate edge *i.e.*, assignment and request edge.

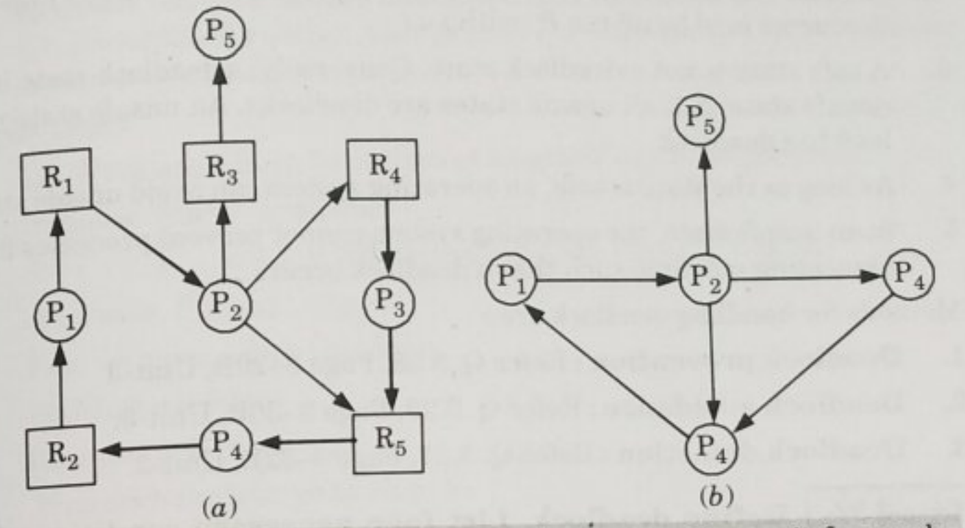


Fig. 3.32.1. Resource-allocation graph with corresponding wait-for graph.

   4. The assumptions made in the wait-for graph are as follows :
      a. An edge from P_i to P_j in a wait-for graph implies that process P_i is waiting for process P_j to release a resource that P_i needs.
      b. An edge $P_i \rightarrow P_j$ exists in a wait-for graph if and only if the corresponding resource-allocation graph contains two edges $P_i \rightarrow R_q$ and $R_q \rightarrow P_i$ for some resource R_q.
   5. If WFG contains cycle then the system is deadlocked otherwise is in safe state.

ii. **Several instances of a resource type :**
   1. Unmark all active processes from Allocation, Max and Available in accordance with the system state.
   2. Find an unmarked process i such that
      Max $\leq$ Available
      If found, mark process i, update Available
      Available : = Available + Allocation
      and repeat this step.
      If no process is found, then go to next step.
   3. If all processes are marked, the system is not deadlocked.

Que 3.33. **"A safe state is not a deadlock state but a deadlock state is an unsafe state". Explain. Discuss the different methods for handling deadlocks.**

Answer

1. A safe state is a state in which there is at least one order in which all the processes can be run to completion without resulting in a deadlock.
2. A sequence of processes $< P_1, P_2 ..... P_n >$ is a safe sequence for the current allocation state if, for each P_i the resources that P_j can still

request can be satisfied by the currently available resources plus resources held by all the P_j, with $j < i$.

3. A safe state is not a deadlock state. Conversely, a deadlock state is an unsafe state. Not all unsafe states are deadlocks. An unsafe state may lead to a deadlock.
4. As long as the state is safe, an operating system can avoid unsafe states.
5. In an unsafe state, the operating system cannot prevent processes from requesting resource such that a deadlock occurs.

Methods for handling deadlock are :

1. **Deadlock prevention :** Refer Q. 3.28, Page 3–29B, Unit-3.
2. **Deadlock avoidance :** Refer Q. 3.29, Page 3–30B, Unit-3.
3. **Deadlock detection :** Refer Q. 3.31, Page 3–32B, Unit-3.

Que 3.34. Define deadlock. List four necessary conditions for occurrence of deadlock. A system contains 6 units of resource, and processes that use the resource. What is the maximum value of for which the system will be deadlock free if the maximum requirement of each process is 3 ?

AKTU 2012-13, Marks 05

Answer

Deadlock and its four necessary conditions : Refer Q. 3.27, Page 3-29B, Unit-3.

Maximum value of n :

The maximum value of n for which the system is guaranteed to be deadlock free is 2. Two processes can never lead to deadlock as the peak time demand of $(3 + 3 = 6)$ resources can be satisfied. But 3 processes can lead to deadlock if each process holds 2 resources and then demand on more.

Que 3.35. Write an algorithm for detection of deadlock in a system having several instances of multiple resource types.

AKTU 2012-13, Marks 10

Answer

The algorithm uses several time-varying data structures which are as follows :

1. **Available :** A vector of length m indicates the number of available resources of each type.
2. **Allocation :** An $n \times m$ matrix defines the number of resources of each type currently allocated to each process.

3. **Request :** An $n \times m$ matrix indicates the current request of each process. If it Request[i][j] equals k, then process P_i is requesting k more instances of resource type R_j.

Algorithm :

1. Let Work and Finish be vectors of length m and n, respectively.
   Initialize Work = Available.
   For $i = 0, 1, ..., n - 1$, if Allocation$_i \neq 0$, then Finish[i] = false;
   otherwise, Finish[i] = true.
2. Find an index i such that both
   Finish[i] = = false
   Request$_i \leq$ Work
   If no such i exists, go to step (4).
3. Work = Work + Allocation$_i$
   Finish[i] = true
   Go to step (2).
4. If Finish[i] = = false for some i, $0 \leq i < n$ then the system is in a deadlock state.
   Moreover if Finish[i] == false then process P_i is deadlock.
   This algorithm requires an order of $m \times n^2$ operations to detect whether the system is in deadlock state.

Que 3.36. Write the Banker's algorithm and how it can be used to avoid can be used to avoid deadlock.

OR

Describe Banker's algorithm for safe allocation.

AKTU 2016-17, Marks 7.5

OR

Write and explain Banker's algorithm for avoidance of deadlock.

AKTU 2011-12, Marks 10

Answer

Banker's algorithm :

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for pre-determined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

Following data structures are used to implement the Banker's algorithm :

Let 'n' be the number of processes in the system and 'm' be the number of resources types.

1. **Available :** It is a 1D array of size 'm' indicating the number of available resources of each type.

   Available[j] = k means there are 'k' instances of resource type R_j

2. **Max :** It is a 2D array of size '$n*m$' that defines the maximum demand of each process in a system.

   Max[i, j] = k means process P_i may request at most 'k' instances of resource type R_j

3. **Allocation :** It is a 2D array of size '$n*m$' that defines the number of resources of each type currently allocated to each process.

   Allocation[i, j] = k means process P_i is currently allocated 'k' instances of resource type R_j

4. **Need :** It is a 2D array of size '$n*m$' that indicates the remaining resource need of each process.

   Need [i, j] = k means process P_i currently allocated 'k' instances of resource type R_j

   Need [i, j] = Max [i, j] – Allocation [i, j]

   $Allocation_i$ specifies the resources currently allocated to process P_i and $Need_i$ specifies the additional resources that process P_i may still request to complete its task.

Banker's algorithm is used to avoid deadlock by following algorithms :

a. Safety algorithm :

1. Let work and finish be vector of length m and n respectively. Initialize Work = Available and Finish [i] = False for i = 1, 2, 3, 4, ... n.
2. Find an i such that both
   a. Finish [i] = False
   b. Need [i] ≤ Work

   If no such i exist, goto step 4.
3. Work = Work + $Allocation_i$

   Finish [i] = True

   Goto step 2
4. If Finish [i] = True for all i, then the system is in a safe state.

b. Resource request algorithm : Let $Request_i$ be the request array for process P_i. $Request_i$ [j] = k means process P_i wants k instances of resource type R_j. When a request for resources is made by process P_i, the following actions are taken :

1. If $Request_i \le Need_i$

   Goto step 2; otherwise, raise an error condition, since the process has exceeded its maximum claim.
2. If $Request_i \le$ Available

   Goto step 3; otherwise, P_i, must wait, since the resources are not available.
3. Have the system pretend to have allocated the requested resources to process P_i by modifying the state as follows :

   Available = Available – $Request_i$

   $Allocation_i$ = $Allocation_i$ + $Request_i$

   $Need_i$ = $Need_i$ – $Request_i$

Que 3.37. Describe Banker's algorithm for deadlock avoidance. Consider a system with three processes and three resources. The snapshot of a system at time t_0 is given below :

Processes	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	2	2	3	3	6	8	7	7	10
P_1	2	0	3	4	3	3			
P_2	1	2	4	3	4	4			

i. Is the current allocation in safe state ?

ii. Would the following requests be granted in the current state :

a. Process P_2 requests (1, 0, 0)

b. Process P_1 requests (1, 0, 0)

AKTU 2014-15, Marks 10

Answer

Banker's algorithm : Refer Q. 3.36, Page 3–35B, Unit-3.

i. First find the available resources in the system

Available = Number of instance - sum of allocation

Process	Current allocation		
	A	B	C
P_0	2	2	3
P_1	2	0	3
P_2	1	2	4
	5	4	10

Available = (7 7 10) – (5 4 10)

Available resources = (2 3 0)

Content of need matrix is :

Process	Need		
P_0	1	4	5
P_1	2	3	0
P_2	2	2	0

Safe sequence $< P_2, P_1, P_0 >$.

The system is in safe state.

ii. a. Process P_2 request (1, 0, 0), this request is less than need. Need for process P_2 is (2, 2, 0), available resource is (2, 3, 0) and request is (1, 0, 0).

Request < Available, (1, 0, 0) < (2, 3, 0)

After allocating (1, 0, 0) to process P_2 the need becomes as follows :

Process	Need		
P_0	1	4	5
P_1	2	3	0
P_2	1	2	0

And available resource is (1, 3, 0). We see that system is in safe state with safe sequence $<P_2, P_1, P_0>$. Hence request of P_2 (1, 0, 0) will be granted.

b. Process P_1 request (1, 0, 0), this request is less than need. Need for process P_1 is (2, 3, 0). Available resource is (1, 3, 0) and request is (1, 0, 0).

Request <Available, (1, 0, 0) < (1, 3, 0)

After allocating (1, 0, 0) to process P_1 the need becomes as follows :

Process	Need		
P_0	1	4	5
P_1	1	3	0
P_2	1	2	0

And available resource is (0, 3, 0). Need of any process is never satisfied after granting the P_1 request (1, 0, 0). So, the system will be blocked. Therefore, request of P_1 (1, 0, 0) cannot be granted.

Que 3.38. **Consider the following snapshot of the system :**

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Answer the following questions using the banker's algorithm :

i. What is the content of the matrix need ?

ii. Is the system in a safe state ? If yes then find the safe sequence.

iii. If a request from process P_1 arrives for (0, 4, 2, 0) can the request be granted immediately ?

AKTU 2015-16, Marks 10

Answer

i. The need matrix can be calculated according to Banker's algorithm.

$Need_i = Max_i - Allocation_i$

Need matrix

	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

ii. Yes, system is in safe state. This can be seen as below :

Since, we have available = (1, 5, 2, 0) which can be allocated to P_0 or P_2 to satisfy the need or request

According to safety algorithm,

	A	B	C	D					
Work =	1	5	2	0	(Available)				
	A	B	C	D		A	B	C	D
Need $[P_0]$ =	0	0	0	0	<	1	5	2	0 (Available)

Hence, process P_0 will release the allocated resources. Hence,

Work = Work + allocation

	A	B	C	D		A	B	C	D
=	1	5	2	0	+	0	0	1	2
	A	B	C	D					
=	1	5	3	2	(Available)				

This can satisfy the request of process P_2. Since

	A	B	C	D		A	B	C	D
Need $[P_2]$ =	1	0	0	2	<	1	5	3	2 (Available)

Hence, process P_2 will also release the resource. Hence

Work = Work + Allocation

	A	B	C	D		A	B	C	D
=	1	5	3	2	+	1	3	5	4
=	2	8	8	6 (Available)					

This (Available) can satisfy the request process P_3.
Hence, resources will be allocated to it

Work = Work + Allocation

	A	B	C	D		A	B	C	D
=	2	8	8	6	+	0	6	3	2
	A	B	C	D					
=	2	14	11	8 (Available)					

This (Available) resource, can be allocated to Process P_4. Since

	A	B	C	D		A	B	C	D
Need $[P_4]$ =	0	6	4	2	<	2	14	11	8

Hence, after completion processes P_4 will also release the resource.

Work = Work + Allocation

	A	B	C	D		A	B	C	D
=	2	14	11	8	+	0	0	1	4
	A	B	C	D					
=	2	14	12	12					

These resource, can be allocated to process P_1

	A	B	C	D		A	B	C	D
Since, Need $[P_1]$	0	7	5	0	<	2	14	12	12

Hence safe sequence will be $\langle P_0, P_2, P_3, P_4, P_1 \rangle$

iii. Since process P_1 requests for the resources (0, 4, 2, 0)
Now, the matrixes will be

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	1	0	0
P_1	1	4	2	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Here, Available

	A	B	C	D		A	B	C	D		A	B	C	D
=	1	5	2	0	–	0	4	2	0	=	1	1	0	0

Now the need matrix can be represented as

$\text{Need}_i = \text{Max}_i - \text{Allocation}_i$

A	B	C	D
0	0	0	0
0	3	3	0
1	0	0	2
0	0	2	0
0	6	4	2

Now again calculate the safe sequence as in previous section of this question. The safe sequence will be

$< P_0, P_2, P_3, P_1, P_4 >$

Que 3.39. In a system, n processes share m resource units that can be reserved and released only one at a time. The maximum need of each process does not exceed m and sum of all maximum need is less than $m + n$. Show that a deadlock cannot occur.

AKTU 2011-12, Marks 05

Answer

i. $\sum_{i=1}^{n} \text{Max}_i < m + n$

ii. $\text{Max}_i \geq 1$ for all i
Proof : $\text{Need}_i = \text{Max}_i - \text{Allocation}_i$

iii. If there exists a deadlock state then :

$\sum_{i=1}^{n} \text{Allocation}_i = m$

Use (i) to get : $\sum \text{Need}_i + \sum \text{Allocation}_i = \sum \text{Max}_i < m + n$

Use (iii) to get : $\sum \text{Need}_i + m < m + n$

Rewrite to get : $\sum_{i=1}^{n} \text{Need}_i < n$

This implies that there exists a process P_i such that $\text{Need}_i = 0$. Since $\text{Max}_i \geq 1$ it follows that P_i has at least one resource that it can release. Hence, the system cannot be in a deadlock state.

Que 3.40. Describe the Banker's algorithm for safe allocation. Consider a system with five processes and three resource types and at time T_0 the following snapshot of the system has been taken :

Process Id	Allocated			Maximum			Available		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	1	1	2	4	3	3	3	1	0
P_2	2	1	2	3	2	2			
P_3	4	0	1	9	0	2			
P_4	0	2	0	7	5	3			
P_5	1	1	2	11	2	3			

i. **Determine the total amount of resources of each type.**
ii. **Compute the need matrix.**
iii. **Determine if the state is safe or not using Banker's algorithm.**
iv. **Would the following request be granted in the current state?**
a. $P_1 < 3, 3, 1 >$
b. $P_2 < 2, 1, 0 >$

AKTU 2013-14, Marks 10

Answer

Banker's algorithm for safe allocation :

When a new process enters the system, it must declare the maximum number of instances of each resource type that it may need. This number may not exceed the total number of resources in the system. When a user requests, a set of resources will leaves the system in a safe state. If it will, the resources are allocated; otherwise, the process must wait until some other process releases enough resources.

Safety algorithm : Refer Q. 3.36, Page 3–35B, Unit-3.

Numerical :

i. Total resource = total allocated (Sum of columns of allocation + available)

$$= [8\ 5\ 7] + [3\ 1\ 0] = [11\ 6\ 7]$$

ii. Need = Maximum – Allocation

A	B	C	–	A	B	C		Need
4	3	3		1	1	2	=	3 2 1
3	2	2		2	1	2	=	1 1 0
9	0	2		4	0	1	=	5 0 1
7	5	3		0	2	0	=	7 3 3
11	2	3		1	1	2	=	10 1 1

iii. Need is compared with Available. If need ≤ available, then resources are allocated to that process and process will release the resource.

If Need is greater than Available, next process need is taken for comparison. Need for process P_1 is (321) and Available is (310).

Need > Available → False

So, system will move to next process.

Need for P_2 is (110) and available is (310)

Need ≤ Available → True

Request for P_2 is granted.

Available = Available + Allocation

= 310 + 212 = 522

Next Process P_3 needs 501 and available is 522.

Available = 522 + 401 = 923

So, request is granted 501 ≤ 522 → true

Next process P_4 needs 733 and available is 923.

So, request is granted.

Now Available = 923 + 020 = 943

Next process P_5 needs (10, 1, 1) and available is 943.

Need > Available

Hence, no sequence is possible thus the state is unsafe.

iv. a. $P_1 < 3, 3, 1 >$

Need > Available → 331 ≮ 310

Hence, resource will not be granted.

b. $P_2 < 2,1, 0 >$

Need < Available → True

210 < 310

Hence, the request will be granted.

Que 3.41. **Differentiate between deadlock and starvation.**

AKTU 2011-12, Marks 2.5

Answer

Basis for Comparison	Deadlock	Starvation
Basic	Deadlock is where no process proceeds, and get blocked.	Starvation is where low priority processes get blocked, and high priority process proceeds.
Arising condition	The occurrence of mutual exclusion, hold and wait, no preemption and circular wait simultaneously.	Enforcement of priorities, uncontrolled resource management.
Other name	Circular wait.	Lifelock.
Resources	In deadlocked, requested resources are blocked by the other processes.	In starvation, the requested resources are continuously used by high priority processes.
Prevention	Avoiding mutual exclusion, hold and wait, and circular wait and allowing preemption.	Ageing.

Que 3.42. **Discuss the techniques to recover from deadlock.**

Answer

There are two options for recovering from a deadlock :

1. **Process termination :** To eliminate deadlocks by aborting a process, we use one of two methods.
   a. **Abort all deadlocked processes :** This method clearly will break the deadlock cycle, but the deadlocked processes may have computed for a long time, and the results of these partial computations must be discarded and probably will have to be recomputed later.
   b. **Abort one process at a time until the deadlock cycle is eliminated :** This method incurs considerable overhead, since after each process is aborted, a deadlock-detection algorithm must be invoked to determine whether any processes are still deadlocked.
2. **Resource pre-emption :** To eliminate deadlocks using resource pre-emption, we successively preempt some resources from processes and give these resources to other processes until the deadlock cycle is broken.
   a. **Selecting a victim :**
      i. Which resources and which processes are to be preempted? As in process termination, we must determine the order of pre-emption to minimize cost.
      ii. Cost factors may include such parameters as the number of resources a deadlocked process is holding and the amount of time the process has thus far consumed during its execution.
   b. **Rollback :**
      i. If we preempt a resource from a process, then it cannot continue with its normal execution.
      ii. It is missing some needed resource.
      iii. We must roll back the process to some safe state and restart it from that state.
      iv. Since, in general, it is difficult to determine what a safe state is, the simplest solution is a total rollback, abort the process and then restart it.
      v. Although it is more effective to roll back the process only as far as necessary to break the deadlock, this method requires the system to keep more information about the state of all running processes.
   c. **Kill the process.**

VERY IMPORTANT QUESTIONS

Following questions are very important. These questions may be asked in your SESSIONALS as well as UNIVERSITY EXAMINATION.

Q. 1. Explain performance criteria of CPU scheduling.

Ans. Refer Q. 3.1.

Q. 2. What is CPU scheduling? Write the difference between pre-emptive and non pre-emptive scheduling.

Ans. Refer Q. 3.1 and Q. 3.3.

Q. 3. Explain process transition diagram.

Ans. Refer Q. 3.5.

Q. 4. Discuss Process Control Block (PCB).

Ans. Refer Q. 3.7.

Q. 5. Write short notes on process identification information.

Ans. Refer Q. 3.10.

Q. 6. Write short notes on following :
i. **FCFS scheduling**
ii. **SJF scheduling**
iii. **Round robin scheduling**

Ans.
i. Refer Q. 3.17. ii. Refer Q. 3.18. iii. Refer Q. 3.20.

Q. 7. Write the difference between following :
1. **Thread and process**
2. **User level thread and kernel level thread**
3. **Long term, short term and mid term scheduler**

Ans.
1. Refer Q. 3.15. 2. Refer Q. 3.13. 3. Refer Q. 3.8.

Q. 8. Explain Banker's algorithm for avoidance of deadlock.

Ans. Refer Q. 3.36.

Q. 9. Explain the following :
1. **Deadlock prevention**
2. **Deadlock avoidance**
3. **Deadlock detection**

Ans.
1. Refer Q. 3.28.
2. Refer Q. 3.29.
3. Refer Q. 3.31.

☺☺☺