Assignment 2 19UCS122

Q1. What is the process? Explain the process concept in detail.

→

1. A process can be thought of as a program in execution.
2. A process will need certain resources — such as CPU time, memory, files, and IO devices — to accomplish its task.
3. A process is the unit of work in most systems. Systems consist of a collection of processes: operating-system processes execute system code
4. User processes execute user code.
5. All these processes may execute concurrently.
6. To execute the program, OS needs to take the program in the main memory.
7. OS has to define a data structure in the main memory for execution that is the process.
8. Traditionally a process contained only a single thread of control as it ran, most modern operating systems now support processes that have multiple threads.

19UCS122

9. The operating system is responsible for several important aspects of process and thread management
10. The creation and deletion of both user and system processes; the scheduling of processes; and the provision of mechanisms for synchronization, communication, and deadlock handling for processes.
11. Early computers allowed only one program to be executed at a time. This program had complete control of the system and had access to all the system's resources.

Process Concept :

1. A process is a program in execution.
2. A process is more than the program code, which is sometimes known as the text section.
3. It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers.

19UCS122

4. A process generally also includes the process stack, which contains temporary data.
5. A process may also include a heap, which is the memory that is dynamically allocated during process runtime

Q2.

→ With the help of a suitable diagram explain PCB

Process Control Block :

1. Each process is represented in the operating system by a process control block [PCB] – also called a task control block.
2. It contains many pieces of information associated with a specific process.
3. Process state. The state may be new, ready, running, waiting, halted, and so on.
4. Program counter. The counter indicates the address of the next instruction to be executed for this process

19UCS122

5. CPU registers. The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information.

6. CPU-scheduling information. This information includes a process priority

7. Memory-management information. This information may include such items as the value of the base and limit registers and the page tables, or the segment tables, depending on the memory system used by the operating system.

8. Accounting information. This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.

9. IO status information. This information includes the list of IO devices allocated to the process, a list of open files, and so on.

process state
process number
program counter
registers
memory limits
list of open files
...

Figure: Process control block (PCB).

Q3. Explain process states using a process state diagram

->

Process State :

As a process executes, it changes state.

1. New. The process is being created.

2. Running. Instructions are being executed.

3. Waiting. The process is waiting for some event to occur (such as an IO 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

Q4. What is a scheduler? Explain different types of schedulers in OS

→

Schedulers

A process migrates among the various scheduling queues throughout its lifetime.

Schedulers are special system software which handles process scheduling in various ways.

19UCS122

Their main task is to select the jobs to be submitted into the system and to decide which process to run.

There are three types of schedulers available:
1. Long Term Scheduler
2. Short Term Scheduler
3. Medium Term Scheduler

Long Term Scheduler
1. Long-term scheduler runs less frequently.
2. It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing
3. Long Term Schedulers decide which program must get into the job queue.
4. From the job queue, the Job Processor, selects processes and loads them into the memory for execution.
5. Primary aim of the Job Scheduler is to maintain a good degree of Multiprogramming.

19UCS122

Short Term Scheduler

1. Short-term scheduling involves selecting one of the processes from the ready queue and scheduling them for execution. This is done by the short-term scheduler. A scheduling algorithm is used to decide which process will be scheduled for execution next by the short-term scheduler.

2. The short-term scheduler executes much more frequently than the long-term scheduler as a process may execute only for a few milliseconds.

3. The choices of the short-term scheduler are very important. If it selects a process with a long burst time, then all the processes after that will have to wait for a long time in the ready queue. This is known as starvation and it may happen if a wrong decision is made by the short-term scheduler.

19UCS122

Medium Term Scheduler

1. Medium-term scheduling involves swapping out a process from the main memory. The process can be swapped in later from the point it stopped executing. This can also be called suspending and resuming the process and is done by the medium-term scheduler.

2. At some later time, the process can be reintroduced into memory and its execution can be continued where it left off.

3. This scheme is called swapping. The process is swapped out and is later swapped in, by the medium-term scheduler.

Q5. Explain the various types of queues with the help of queueing diagram.

->

Scheduling Queues

19UCS122

1. Process scheduling is the activity 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.
2. As processes enter the system, they are put into a job queue, which consists of all processes in the system.
3. The processes that are residing in the main memory and are ready and waiting to execute are kept on a list called the ready queue.
4. This queue is generally stored as a linked list. A ready-queue header contains pointers to the first and final PCBs in the list.
5. Each PCB includes a pointer field that points to the next PCB in the ready queue
6. The system also includes other queues. The list of processes waiting for a particular IO device is called a device queue. Each device has its own device queue

19UCS122

7. All processes, upon entering the system, are stored in the job queue.
8. Processes in the Ready state are placed in the Ready Queue.
9. Processes waiting for a device to become available are placed in Device Queues.
10. There are unique device queues available for each IO device.
11. A new process is initially put in the Ready queue.

#Once the process is assigned to the CPU and is executing, one of the following several events can occur:

1. The process could issue an IO request, and then be placed in the IO queue.
2. The process could create a new sub-process and wait for its termination.
3. The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue

19UCS122

Q6. Write a note on :

a. Context Switching

b. Shared memory model

c. Message passing model

d. RPC

e. Scheduling criteria

f. Real-time CPU scheduling

->

a. Context Switching :

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

2. The context of a process is represented in the Process Control Block (PCB) of a process; it includes the value of the CPU registers, the process state, and memory management information.

3. When a context switch occurs, the Kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run.

19UCS122

4. Context Switching involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier. This is a feature of a multitasking operating system and allows a single CPU to be shared by multiple processes.

Context Switching Triggers

1. There are three major triggers for context switching. These are given as follows:

2. Multitasking: In a multitasking environment, a process is switched out of the CPU so another process can be run. The state of the old process is saved and the state of the new process is loaded.

3. Interrupt Handling: The hardware switches a part of the context when an interrupt occurs. This happens automatically. Only some of the context is changed to minimize the time required to handle the interrupt.

19UCS122

4. User and Kernel Mode Switching: A context switch may take place when a transition between the user mode and kernel mode is required in the operating system.

b. Shared Memory Model :

Shared-Memory Systems:-

1. Inter-process communication using shared memory requires communicating processes to establish a region of shared memory.
2. Typically, a shared-memory region resides in the address space of the process creating the shared memory segment.
3. Other processes that wish to communicate using this shared-memory segment must attach it to their address space

c. Message-Passing Model

Message-Passing Systems:-

1. Message passing provides a mechanism to allow

the process to communicate and to synchronize their actions without sharing the same address space.

2. It is particularly useful in a distributed environment, where the communicating processes may reside on different computers connected by a network

3. An Internet chat program could be designed so that chat participants communicate with one another by exchanging messages

d. RPC:

Remote Procedure Calls:-

1. The RPC was designed as a way to abstract the procedure-call mechanism for use between systems with network connections.

2. Its similar in many respects to the IPC mechanism however because we are dealing with an environment in which the processes are executing

on separate systems, we must use a message-based communication scheme to provide remote service.

3. In contrast to IPC messages, the messages exchanged in RPC communication are well structured and are thus no longer just packets of data.

4. Each message is addressed to an RPC daemon listening to a port on the remote system, and each contains an identifier specifying the function to execute and the parameters to pass to that function.

5. The function is then executed as requested, and any output is sent back to the requester in a separate message.

e. Scheduling criteria :

1. Different CPU-scheduling algorithms have different properties.

19UCS122

2. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms.
3. Many criteria have been suggested for comparing CPU-scheduling algorithms.

4. The criteria include... ...
5. CPU utilization. We want to keep the CPU as busy as possible.
6. Throughput. If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput. For long processes, this rate may be one process per hour; for short transactions, it may be ten processes per second.
7. Turnaround time. The interval from the time of submission of a process to the time of completion is the turnaround time.
8. Turnaround time = periods spent waiting to get into memory + waiting in the ready queue + executing on the CPU + and doing I/O.

19UCS122

9. Waiting time:- Waiting time is the sum of the periods spent waiting in the ready queue.
10. The CPU-scheduling algorithm affects only the amount of time that a process spends waiting in the ready queue
11. Response time.:-the time from the submission of a request until the first response is produced

f. Real-time CPU Scheduling :

1. CPU scheduling for real-time operating systems involves
special issues.
2. We can distinguish between soft real-time systems and hard real-time systems.
3. Soft real-time systems provide no guarantee as to when the acritical real-time process will be scheduled.
4. They guarantee only that the process will be given preference over noncritical processes.

19UCS122

5. Hard real-time systems have stricter requirements.
A
the task must be serviced by its deadline; service
after the
deadline has expired is the same as no service at all

#Minimizing Latency:

The system is typically waiting for an event in real time to occur.

Events may arise either in software — as when a timer expires — or in hardware — as when a remote-controlled vehicle detects that it is approaching an obstruction

When an event occurs, the system must respond to and service it as quickly as possible.

Event latency: The amount of time that elapses from when an event occurs to when it is serviced.

#Two types of latencies affect the performance of real-time

19UCS122

1. Interrupt latency refers to the period of time from the
the arrival of an interrupt at the CPU to the start of the routine that services the interrupt

2. Dispatch latency

The amount of time required for the scheduling dispatcher to stop one process and start another is known as dispatch latency

The most effective technique for keeping dispatch latency low is to provide preemptive kernels.

The conflict phase of dispatch latency has two components:
1. Preemption of any process running in the kernel
2. Release by low-priority processes of resources needed by a high-priority process