M. L. V. Govt. Textile Engineering College, Bhilwara Pur Road, Pratap Nagar
Marks 10 Subject(Operating System) (V Semester 2017-2018) Time 1 hr
All questions carry equal marks

Q.1 Explain process control block.

Ans: While creating a process the operating system performs several operations. To identify these process, it must identify each process, hence it assigns a process identification number (PID) to each process. As the operating system supports multi-programming, it needs to keep track of the all the processes. For this task, the process control block (PCB) is used to track the process's execution status. Each block of memory contains information about the process state, program counter, stack pointer, status of opened files, scheduling algorithms, etc. All these information is required and must be saved when the process is switched from one state to another. When the process made transitions from one state to another, the operating system must update information in the process's PCB.

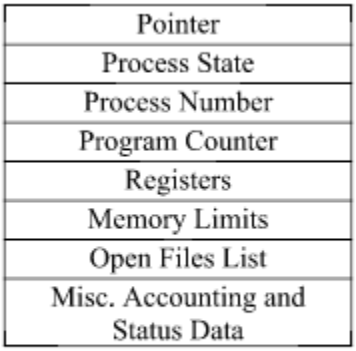


Figure 1: Process Control Block

- **Pointer –** It is a stack pointer which is required to be saved when the process is switched from one state to another to retain the current position of the process.
- **Process state –** It stores the respective state of the process.
- **Process number –** Every process is assigned with a unique id is known as processed which stores the process identifier.
- **Program counter –** It stores the counter which contains the address of the next instruction that is to be executed for the process.
- **Register –** These are the CPU registers which includes: accumulator, base, registers and general purpose registers.
- **Memory limits –** This field contains the information about memory management system used by operating system. This may include the page tables, segment tables etc.

Q.2 Explain thread .Briefly explain user level and kernel level threads.

Ans :A thread is a path of execution within a process. Also, a process can contain multiple threads. Thread is also known as lightweight process. The idea is achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS word uses multiple threads, one thread to format the text, other thread to process inputs etc.

Process vs Thread:

The typical difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces.

Threads are not independent of one other like processes as a result threads shares with other threads

their code section, data section and OS resources like open files and signals. But, like process, a thread has its own program counter (PC), a register set, and a stack space.
User level To make concurrency cheaper, the execution aspect of process is separated out into threads. As such, the OS now manages threads and processes. All thread operations are implemented in the kernel and the OS schedules all threads in the system. OS managed threads are called kernel-level threads or light weight processes
Kernel level: Kernel-Level threads make concurrency much cheaper than process because, much less state to allocate and initialize. However, for fine-grained concurrency, kernel-level threads still suffer from too much overhead. Thread operations still require system calls. Ideally, we require thread operations to be as fast as a procedure call. Kernel-Level threads have to be general to support the needs of all programmers, languages, runtimes, etc. For such fine grained concurrency we need still "cheaper" threads. To make threads cheap and fast, they need to be implemented at user level. User-Level threads are managed entirely by the run-time system (user-level library).The kernel knows nothing about user-level threads and manages them as if they were single-threaded processes.User-Level threads are small and fast, each thread is represented by a PC,register,stack, and small thread control block. Creating a new thread, switiching between threads, and synchronizing threads are done via procedure call. i.e no kernel involvement. User-Level threads are hundred times faster than Kernel-Level threads.

Q.3 What is CPU scheduler. Explain FCFS scheduling algorithm.

Ans:**CPU scheduling** is a process which allows one process to use the CPU while the execution of another process is on hold(in waiting state) due to unavailability of any resource like I/O etc, thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair.

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.

FCFS Scheduling Algorithm

Given n processes with their burst times, the task is to find average waiting time and average turn around time using FCFS scheduling algorithm.
First in, first out (FIFO), also known as first come, first served (FCFS), is the simplest scheduling algorithm. FIFO simply queues processes in the order that they arrive in the ready queue.
In this, the process that comes first will be executed first and next process starts only after the previous gets fully executed.
Here we are considering that arrival time for all processes is 0.

FCFS Example

Process	Duration	Order	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0

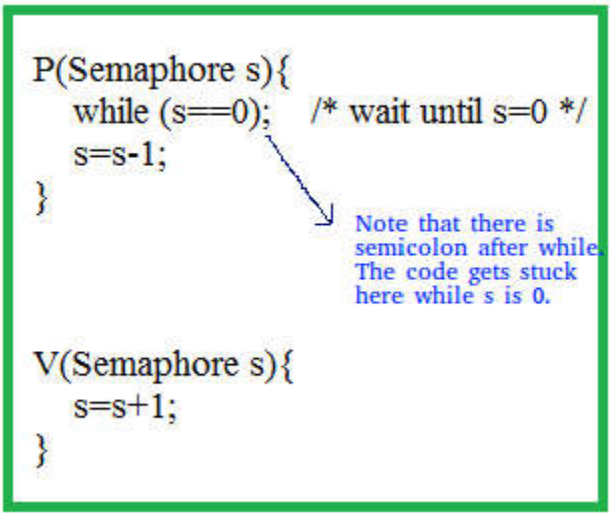
The final schedule (Gantt chart):

P1 waiting time: 0
P2 waiting time: 24
P3 waiting time: 27

The average waiting time:
(0+24+27)/3 = 17

Q.4 Explain semaphore. E xplain producer consumer problem.

Ans: Semaphore is a simply a variable. This variable is used to solve critical section problem and to achieve process synchronization in the multi processing environment.
The two most common kinds of semaphores are counting semaphores and binary semaphores. Counting semaphore can take non-negative integer values and Binary semaphore can take the value 0 & 1.



Semaphores are commonly use for two purposes: to share a common memory space and to share access to files. Semaphores are one of the techniques for interprocess communication (IPC). The C programming language provides a set of interfaces or "functions" for managing semaphores.

Producer-Consumer Problem Using Semaphores

The Solution to producer-consumer problem uses three semaphores, namely, full, empty and mutex.

The semaphore 'full' is used for counting the number of slots in the buffer that are full. The 'empty' for counting the number of slots that are empty and semaphore 'mutex' to make sure that the producer and consumer do not access modifiable shared section of the buffer simultaneously.

Initialization

- Set full buffer slots to 0.
  i.e., semaphore Full = 0.
- Set empty buffer slots to N.
  i.e., semaphore empty = N.
- For control access to critical section set mutex to 1.
  i.e., semaphore mutex = 1.

```
Producer ( )
WHILE (true)
        produce-Item ( );
    P (empty);
    P (mutex);
    enter-Item ( )
    V (mutex)
    V (full);

Consumer ( )
WHILE (true)
    P (full)
    P (mutex);
    remove-Item ( );
    V (mutex);
    V (empty);
    consume-Item (Item)
```

M. L. V. Govt. Textile Engineering College, Bhilwara Pur Road, Pratap Nagar
Marks 10 Subject (Operating System) (V Semester 2017-2018) Time 1 hr

Q.1Explain bankers algorithm.

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined 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.

Available :

- It is a 1-d array of size **'m'** indicating the number of available resources of each type.
- Available[j] = k means there are **'k'** instances of resource type $\mathbf{R_j}$

Max :

- It is a 2-d array of size '**n*m'** that defines the maximum demand of each process in a system.
- Max[i, j] = k means process $\mathbf{P_i}$ may request at most **'k'** instances of resource type $\mathbf{R_j}$.

Allocation :

- It is a 2-d 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 $\mathbf{P_i}$ is currently allocated **'k'** instances of resource type $\mathbf{R_j}$

Need :

- It is a 2-d array of size **'n*m'** that indicates the remaining resource need of each process.
- Need [i, j] = k means process $\mathbf{P_i}$ currently allocated **'k'** instances of resource type $\mathbf{R_j}$
- 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 consist of Safety algorithm and Resource request algorithm

Safety Algorithm

The algorithm for finding out whether or not a system is in a safe state can be described as follows:

1) Let Work and Finish be vectors of length '**m**' and '**n**' respectively.

   Initialize: Work = Available

   Finish [i] = false; for i=1, 2, ..., n

2) Find an i such that both

   a) Finish [i] = false

   b) $Need_i \leq work$

      If no such i exists goto step (4)

3) Work = Work + $Allocation_i$

   Finish [i] = true

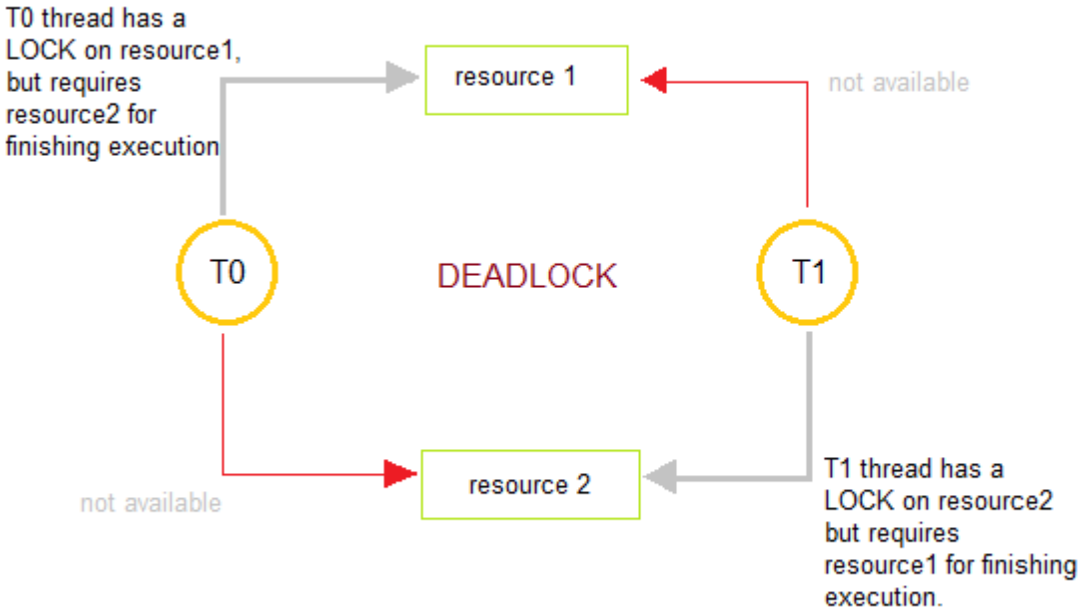
   goto step (2)

4) If Finish [i] = true for all i,

   then the system is in safe state.

Q2) Explain deadlock .

Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.



Deadlocks can be avoided by avoiding at least one of the four conditions, because all this four conditions are required simultaneously to cause deadlock.

1. **Mutual Exclusion**

   Resources shared such as read-only files do not lead to deadlocks but resources, such as printers and tape drives, requires exclusive access by a single process.

2. **Hold and Wait**

   In this condition processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.

3. **No Preemption**

   Preemption of process resource allocations can avoid the condition of deadlocks, where ever possible.

4. **Circular Wait**

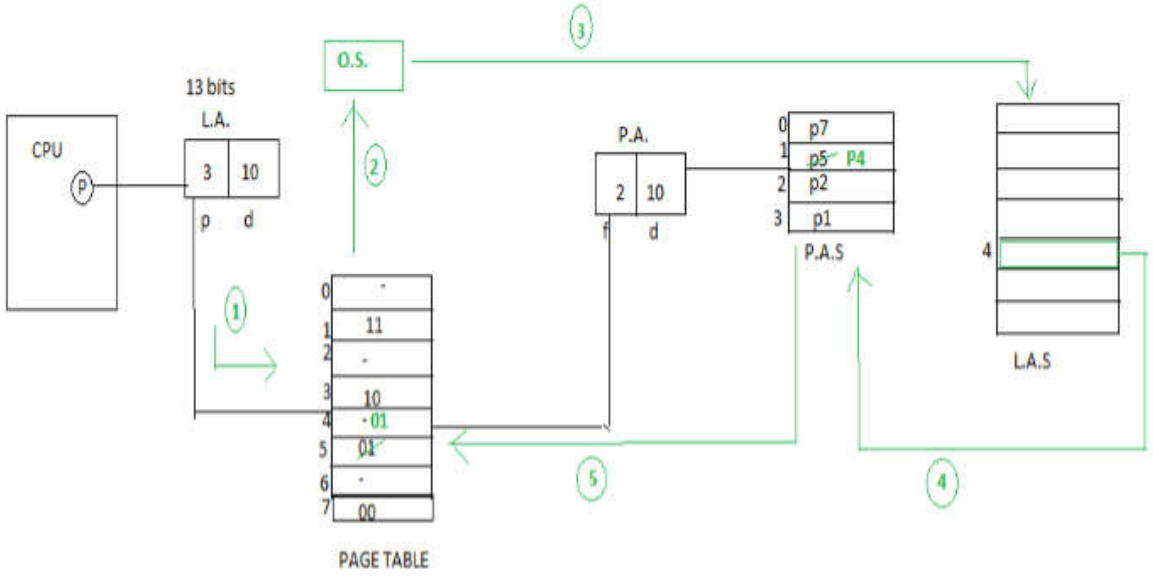
   Circular wait can be avoided if we number all resources, and require that processes request resources only in strictly increasing(or decreasing) order.

Q.3 What is demand paging. Briefly explain segmentation.

Ans: The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging.

The process includes the following steps:

1. If CPU try to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault.
2. The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.
3. The OS will search for the required page in the logical address space.
4. The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision making of replacing the page in physical address space.
5. The page table will updated accordingly.
6. The signal will be sent to the CPU to continue the program execution and it will place the process back into ready state.



Segmentation: A Memory Management technique in which memory is divided into variable sized chunks which can be allocated to processes. Each chunk is called a **Segment**.

A table stores the information about all such segments and is called **Segment Table.**

Segment Table: It maps two dimensional Logical address into one dimensional Physical address.
It's each table entry has

- **Base Address:** It contains the starting physical address where the segments reside in memory.
- **Limit:** It specifies the length of the segment.

Advantages of Segmentation:

- No Internal fragmentation.
- Segment Table consumes less space in comparison to Page table in paging.

Disadvantage of Segmentation:

- As processes are loaded and removed from the memory, the free memory space is broken into little pieces, causing External fragmentation.

Q4) Explain page replacement algorithm .Briefly explain FIFO algorithm.

In a operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults.

Page Fault – A page fault is a type of interrupt, raised by the hardware when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

First In First Out (FIFO) –
This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

For example-1, consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots.

Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**
when 3 comes, it is already in memory so —> **0 Page Faults.**
Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.**
Finally 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**