OSSP-S-2017

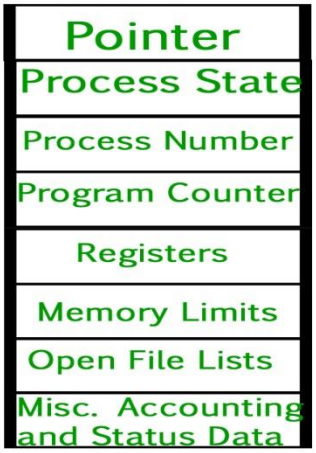
1. Define multitasking.

Ans: Multitasking, in an operating system, is allowing a user to perform more than one computer task at a time. The operating system is able to keep track of processing in these tasks and go from one to the other without losing information.

a. What is PCB? Explain various component of PCB.

Ans: A process control block is a data structure used by computer operating systems to store all the information about a process.

A process control block (PCB) contains information about the process, i.e. registers, quantum, priority, etc. The process table is an array of PCB's, that means logically contains a PCB for all of the current processes in the system.



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 known as process ID or PID 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.
- Open files list – This information includes the list of files opened for a process.

Miscellaneous accounting and status data – This field includes information about the amount of CPU used, time constraints, jobs or process number, etc.

The process control block stores the register content also known as execution content of the processor when it was blocked from running. This execution content architecture enables the operating system to restore a process's execution context when the process returns to the running state. When the process made transitions from one state to another, the operating system update its information in the process's PCB. The operating system maintains pointers to each process's PCB in a process table so that it can access the PCB quickly.

b. Differentiate between preemptive and non-preemptive cpu scheduling algorithm. Explain all cpu scheduling algorithm.

Ans: Preemptive Scheduling is defined as the scheduling which is done when the process changes from running state to ready state or from waiting for the state to ready state. In this, the resources are allocated to execute the process for a certain period.

Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state. In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state.

Some cpu scheduling algorithms are: First Come First Serve (FCFS), Shortest-Job-First (SJF) Scheduling, Priority Scheduling, Round Robin Scheduling.

First Come First Serve (FCFS):

- The process which requests the CPU first gets the CPU allocation first.
- It offers non-preemptive and pre-emptive scheduling algorithm.
- Jobs are always executed on a first-come, first-serve basis
- It is easy to implement and use.
- However, this method is poor in performance, and the general wait time is quite high.

Shortest-Job-First (SJF) Scheduling

- Here he process with the shortest execution time should be selected for execution next
- It is associated with each job as a unit of time to complete.
- In this method, when the CPU is available, the next process or job with the shortest completion time will be executed first.
- It is Implemented with non-preemptive policy.
- This algorithm method is useful for batch-type processing, where waiting for jobs to complete is not critical.
- It improves job output by offering shorter jobs, which should be executed first, which mostly have a shorter turnaround time.

Priority Scheduling:

- Priority scheduling is a method of scheduling processes based on priority.

- In this method, the scheduler selects the tasks to work as per the priority.
- Priority scheduling also helps OS to involve priority assignments.
- The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis.
- Priority can be decided based on memory requirements, time requirements, etc.

Round Robin Scheduling:

- Here each person gets an equal share of something in turn.
- It is mostly used for scheduling algorithms in multitasking.
- This algorithm method helps for starvation free execution of processes.
- Round robin is a hybrid model which is clock-driven
- Time slice should be minimum, which is assigned for a specific task to be processed. However, it may vary for different processes.
- It is a real time system which responds to the event within a specific time limit.

2. Define semaphore?

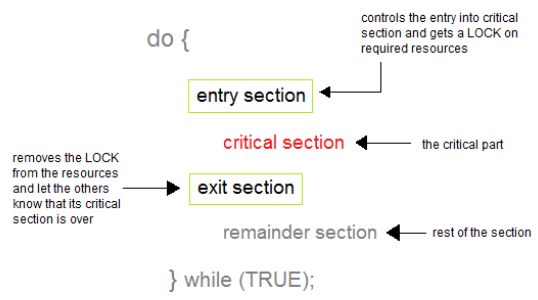
a. Ans: Semaphore is simply a variable that is non-negative and shared between threads. A semaphore is a signaling mechanism, and a thread that is waiting on a semaphore can be signaled by another thread. It uses two atomic operations, 1)wait, and 2) signal for the process synchronization.

a. What is process synchronization? Explain critical section problem in the process synchronization.

Ans: Process Synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.

It is specially needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or data at the same time.

This can lead to the inconsistency of shared data. So the change made by one process not necessarily reflected when other processes accessed the same shared data. To avoid this type of inconsistency of data, the processes need to be synchronized with each other.



To synchronize processes, the process is logically devided into four segments: Entry section, critical section, exit section and remainder section. A critical section is a segment of code which can be accessed by a signal process at a specific point of time. The section consists of shared data resources that required to be accessed by other processes.

- The entry to the critical section is handled by the wait() function, and it is represented as P().
- The exit from a critical section is controlled by the signal() function, represented as V().

In the critical section, only a single process can be executed. Other processes, waiting to execute their critical section, need to wait until the current process completes its execution.

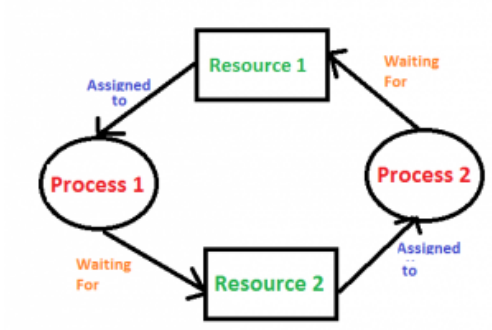
The critical section need to must enforce all three rules:

- Mutual Exclusion: Mutual Exclusion is a special type of binary semaphore which is used for controlling access to the shared resource. It includes a priority inheritance mechanism to avoid extended priority inversion problems. Not more than one process can execute in its critical section at one time.
- Progress: This solution is used when no one is in the critical section, and someone wants in. Then those processes not in their reminder section should decide who should go in, in a finite time.
- Bound Waiting: When a process makes a request for getting into critical section, there is a specific limit about number of processes can get into their critical section. So, when the limit is reached, the system must allow request to the process to get into its critical section.

So binary semaphore , mutex , monitor are used to achive process synchronization among different process.

b. What is deadlock? Explain necessary condition for deadlock in a system having process and resources.

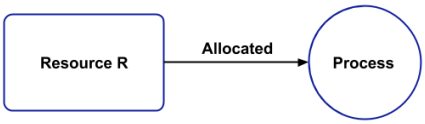
1. Ans: A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function. Here resource 1 and resource 2 have single instances. There is a cycle R1 → P1 → R2 → P2. So, Deadlock is detected.



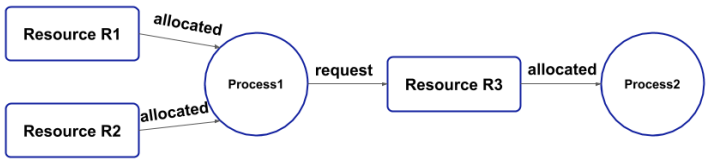
Necessary condition for deadlock: Necessary Conditions of Deadlock

There are four different conditions that result in Deadlock. These four conditions are also known as Coffman conditions and these conditions are not mutually exclusive. Let's look at them one by one.

- Mutual Exclusion: A resource can be held by only one process at a time. In other words, if a process P1 is using some resource R at a particular instant of time, then some other process P2 can't hold or use the same resource R at that particular instant of time. The process P2 can make a request for that resource R but it can't use that resource simultaneously with process P1.

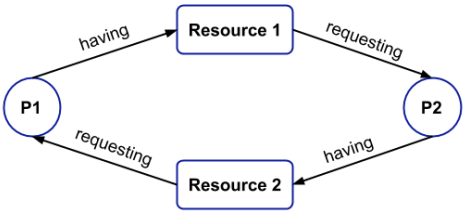


Hold and Wait: A process can hold a number of resources at a time and at the same time, it can request for other resources that are being held by some other process. For example, a process P1 can hold two resources R1 and R2 and at the same time, it can request some resource R3 that is currently held by process P2.



- No preemption: A resource can't be preempted from the process by another process, forcefully. For example, if a process P1 is using some resource R, then some other process P2 can't forcefully take that resource. If it is so, then what's the need for various scheduling algorithm. The process P2 can request for the resource R and can wait for that resource to be freed by the process P1.

- Circular Wait: Circular wait is a condition when the first process is waiting for the resource held by the second process, the second process is waiting for the resource held by the third process, and so on. At last, the last process is waiting for the resource held by the first process. So, every process is waiting for each other to release the resource and no one is releasing their own resource. Everyone is waiting here for getting the resource. This is called a circular wait.



3. What do you mean by turnaround time?

Ans: Turnaround time (TAT) is the time interval from the time of submission of a process to the time of the completion of the process. It can also be considered as the sum of the time periods spent waiting to get into memory or ready queue, execution on CPU and executing input/output.

a. Explain banker's algorithm with example.

Ans: The Banker algorithm , is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation of predetermined maximum possible amounts of all resources, and

then makes an "safty-state" check to test for possible deadlock conditions for all other pending activities, before deciding whether allocation should be allowed to continue.

The Data structures used by the Banker's Algorithm are:

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 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 P_i may request at most 'k' instances of resource type 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 P_i is currently allocated 'k' instances of resource type 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 P_i currently need 'k' instances of resource type R_j for its execution.
- 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 consists 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, 3, 4….n

2) Find an i such that both
a) Finish[i] = false

b) $Need_i$ <= 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 a safe state

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 $Request_i$ <= $Need_i$
Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.

2) If $Request_i$ <= 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 Pi by modifying the state as
follows:
Available = Available – Requesti
$Allocation_i$ = $Allocation_i$ + $Request_i$
$Need_i$ = $Need_i$– $Request_i$

Example:

Considering a system with five processes P_0 through P_4 and three resources of type A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t_0 following snapshot of the system has been taken:

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Question1. What will be the content of the Need matrix?

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

So, the content of Need Matrix is:

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Here m=types of resources=3 and n=number of processes=5

Work=available

So work=[3,3,2]

Lets set finish for all processes are false. So finish=[false,false,false,false,false]

Now consider for process P0 :

$Need_3$=[7,4,3]

Since Need0[7,4,3]> work[3,3,2], so P0 must wait.

Set finish[0]=false.

For process P1:

Need1=[1,2,2]

Since need1[1,2,2]<work[3,3,2] so P1 must keep in the safe sequence.

So work=work+allocation1 =[3,3,2]+[2,0,0]=[5,3,2]

Set finish[1]= true.

For process P2:

Need2=[6,0,0]

Since need2[6,0,0]>work[5,3,2], so P2 must wait.

For process P3:

Need3=[0,1,1]

Since need3[0,1,1]<work[5,3,2] so P3 must keep in the safe sequence.

So work=work+allocation3 =[5,3,2]+[2,1,1]=[7,4,3]

Set finish[3]= true.

For process P4:

Need4=[4,3,1]

Since need4[4,3,1]<work[7,4,3] so P4 must keep in the safe sequence.

So work=work+allocation4 =[7,4,3]+[0,0,2]=[7,4,5]

Set finish[4]= true.

For process P0:

Need0=[7,4,3]

Since need0[7,4,3]<work[7,4,5] so P0 must keep in the safe sequence.

So work=work+allocation0 =[7,4,5]+[0,1,0]=[7,5,5]

Set finish[0]= true.

For process P2:

Need2=[6,0,0]

Since need2[6,0,0,]<work[7,5,5] so P2 must keep in the safe sequence.

So work=work+allocation2 =[7,5,5]+[3,0,2]=[10,5,7]

Set finish[2]= true.

Since finish[i]=true for 0<=i<=4, hence the sequence (P1,P3,P4,P0,P2) is the safe sequence and the system is safe

b. What is scheduler? Explain different types of file access method used in os.

Ans: scheduler schedule the processes which are migrated from ready state to running state .

There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.

SequentialAccess–
It is the simplest access method. Information in the file is processed in order, one record after the other. for example, editor and compiler usually access the file in this fashion. Read and write make up the bulk of the operation on a file. A read operation -*read next*- read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the write *write next* append to the end of the file and advance to the newly written material.

- Data is accessed one record right after another record in an order.
- When we use read command, it move ahead pointer by one
- When we use write command, it will allocate memory and move the pointer to the end of the file

- Such a method is reasonable for tape.

DirectAccess–
Another method is *direct access method* also known as *relative access method*. A filed-length logical record that allows the program to read and write record rapidly in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. There is no restriction on the order of reading and writing for a direct access file.

A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.

Indexsequentialmethod–
It is the other method of accessing a file which is built on the top of the direct access method. These methods construct an index for the file. The index, like an index in the back of a book, contains the pointer to the various blocks. To find a record in the file, we first search the index and then by the help of pointer we access the file directly.

Key points:

- It is built on top of Sequential access.
- It control the pointer by using index.

4. What is page fault?

Ans: A page fault is a type of exception raised by computer hardware when a running program accesses a memory page that is not currently mapped by the memory management unit (MMU) into the virtual address space of a process.

a. Differentiate between paging and segmentation.

Ans:

Sr. No.	Key	Paging	Segmentation
0	Definition	Paging is a memory management technique in which process address space is broken into blocks of the same size called pages	Segmentation is a me management technique in which job is divided into several segmer different sizes, one for each mo that contains pieces that per related functions.
1	Memory Size	In Paging, a process address space is broken into fixed sized blocks called pages.	In Segmentation, a process ad space is broken in varying sized b called sections.

Sr. No.	Key	Paging	Segmentation
2	Accountability	Operating System divides the memory into pages.	Compiler is responsible to calc the segment size, the virtual ad and actual address.
3	Size	Page size is determined by available memory.	Section size is determined by the u
4	Speed	Paging technique is faster in terms of memory access.	Segmentation is slower than pagin
5	Fragmentation	Paging can cause internal fragmentation as some pages may go underutilized.	Segmentation can cause ext fragmentation as some memory l may not be used at all.
6	Logical Address	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical ad is divided into section number section offset.
7	Table	During paging, a logical address is divided into page number and page offset.	During segmentation, a logical ad is divided into section number section offset.
8	Data Storage	Page table stores the page data.	Segmentation table stores segmentation data.

b. What do mean by file organization? Explain different types of file access method used in OS.

Ans: File Organization refers to the logical relationships among various records that constitute the file, particularly with respect to the means of identification and access to any specific record. That is Storing the files in certain order is called file Organization.

There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.

SequentialAccess–
It is the simplest access method. Information in the file is processed in order, one record after the other. for example, editor and compiler usually access the file in this fashion. Read and write make up the bulk of the operation on a file. A read operation -*read next*- read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the write *write next* append to the end of the file and advance to the newly written material.

- Data is accessed one record right after another record in an order.
- When we use read command, it move ahead pointer by one
- When we use write command, it will allocate memory and move the pointer to the end of the file
- Such a method is reasonable for tape.

DirectAccess–
Another method is *direct access method* also known as *relative access method*. A filed-length logical record that allows the program to read and write record rapidly in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. There is no restriction on the order of reading and writing for a direct access file.

A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.

Indexsequentialmethod–
It is the other method of accessing a file which is built on the top of the direct access method. These methods construct an index for the file. The index, like an index in the back of a book, contains the pointer to the various blocks. To find a record in the file, we first search the index and then by the help of pointer we access the file directly.

Key points:

- It is built on top of Sequential access.
- It control the pointer by using index.

5. What is spooling?

Ans: Spooling refers to placing data into an intermediate storage area, where it is held until the user is ready to process it. This storage area is called a spool. Spool- simultaneous peripheral operation online. This technique used by operating system to enable device sharing among user processes.

a. Differentiate between contiguous and non-contigeous memory allocation used by OS.

Ans:

BASIS	CONTIGUOUS MEMORY ALLOCATION	NONCONTIGUOUS MEMORY ALLOCATION

Basic	Allocates consecutive blocks of memory to a process	Allocates separate blocks of memory to a process.
Overhead	Contiguous memory allocation does not have the overhead of address translation while	Noncontiguous memory allocation has overhead of address translation while execution of a process.
Execution rate	A process executes fatser in contiguous memory allocation	A process executes quite slower comparatively in noncontiguous memory allocation.
solution	The memory space must be divided into the fixed-sized partition and each partition is allocated to a single process only.	Divide the process into several blocks and place them in different parts of the memory according to the availability of memory space available.
table	A table is maintained by operating system which maintains the list of available and occupied partition in the memory space	A table has to be maintained for each process that carries the base addresses of each block which has been acquired by a process in memory

b. What is compiler? Explain each phase of a compiler.

Ans: The compilation process is a sequence of various phases. Each phase takes input from its previous stage, has its own representation of source program, and feeds its output to the next phase of the compiler. Different phases of a compiler are

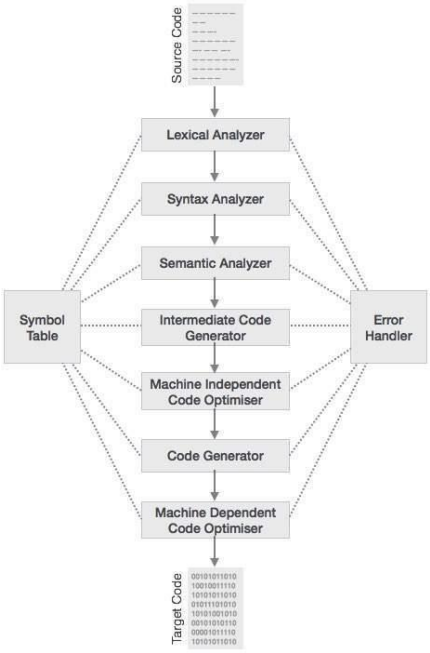
Lexical Analysis

The first phase of scanner works as a text scanner. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. Lexical analyzer represents these lexemes in the form of tokens as:

```
<token-name, attribute-value>
```

Syntax Analysis

The next phase is called the syntax analysis or parsing. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree). In this phase, token arrangements are checked against the source code grammar, i.e. the parser checks if the expression made by the tokens is syntactically correct.



Semantic Analysis

Semantic analysis checks whether the parse tree constructed follows the rules of language. For example, assignment of values is between compatible data types, and adding string to an integer. Also, the semantic analyzer keeps track of identifiers, their types and expressions; whether identifiers are declared before use or not etc. The semantic analyzer produces an annotated syntax tree as an output.

Intermediate Code Generation

After semantic analysis the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language. This intermediate code should be generated in such a way that it makes it easier to be translated into the target machine code.

Code Optimization

The next phase does code optimization of the intermediate code. Optimization can be assumed as something that removes unnecessary code lines, and arranges the sequence of statements in order to speed up the program execution without wasting resources (CPU, memory).

Code Generation

In this phase, the code generator takes the optimized representation of the intermediate code and maps it to the target machine language. The code generator translates the intermediate code into a sequence of (generally) re-locatable machine code. Sequence of instructions of machine code performs the task as the intermediate code would do.

Symbol Table

It is a data-structure maintained throughout all the phases of a compiler. All the identifier's names along with their types are stored here. The symbol table makes it easier for the compiler to quickly search the identifier record and retrieve it. The symbol table is also used for scope management.

6. What is swapping?

Ans: Sw*apping* is a mechanism in which a process can be *swapped* temporarily out of main memory to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from the secondary storage to main memory.

a. Find average turnaround time , average response time, average waiting time of all processing using FCFS scheduling algorithm(assume the arrival time of all processes are same)

Process	CPU burst time
P1	3msec
P2	6msec
P3	9msec
P4	12msec
P5	15msec

Ans: The Gantt chart will be

0_____3_____9_____18_____30_____45

P1 P2 P3 P4 P5

Average waiting time:

For P1=0msec

For P2=3msec

For P3=9msec

For P4=18msec

For P5=30 msec

Total waiting time=0+3+9+18+30=60msec

Average waiting time=60/5=12msec.

Average turnaround time:

Turnaround time is the total amount of time spent by the process from coming in the ready state for the first time to its completion.For for P1=3+0=3msec

For P2=6+3=9msec

For P3=9+9=18msec

For P4=12+18=30msec

For P5=15+30=45 msec

Total turnaround time= 3+9+18+30+45=105msec

Average turnaround time=105/5=21msec

average response time:

Response time is the time spent when the process is in the ready state and gets the CPU for the first time.

for P1=0msec

For P2=3-0=3msec

For P3=9-0=9msec

For P4=18-0=18msec

For P5=30-0=30msec
Total response time= 0+3+9+18+30=50msec
Average response time=50/5=10 msec.

b. Explain the function of I/O traffic controller, I/O scheduler, and I/O device controller.

Ans: see previous question for solution.

7. What is virtual memory?

Ans: Virtual memory is a memory management capability of an operating system which uses hardware and software to allow a computer to compensate for physical memory shortages, by temporarily transferring data from random access memory (RAM) to disk storage.

a. Explain the different page replacement algorithm used in demand paging.

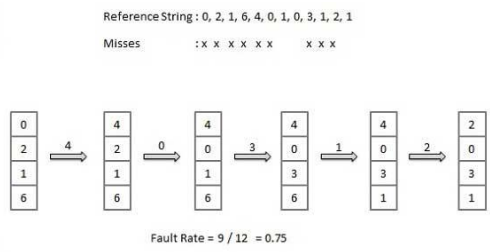
Ans: Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated so that minimum number of page misses occure. Paging happens whenever a page fault occurs. When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion.

All these algorithm are evaluated by running these algorithm on a particular string of memory reference and computing the number of page faults.

Let the memory reference is 0,2,1,6,4,0,1,0,3,1,2,1.

First In First Out (FIFO) algorithm

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

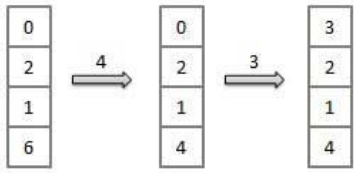


Optimal Page algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x



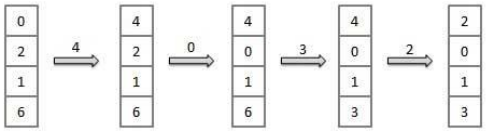
Fault Rate = 6 / 12 = 0.50

Least Recently Used (LRU) algorithm

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x



Fault Rate = 8 / 12 = 0.67

Page Buffering algorithm

- To get a process start quickly, keep a pool of free frames.
- On page fault, select a page to be replaced.
- Write the new page in the frame of free pool, mark the page table and restart the process.
- Now write the dirty page out of disk and place the frame holding replaced page in free pool.

b. Explain the function of various types of OS.

Ans: The various functions of operating system are as follows:

1. Process Management:

A program does nothing unless their instructions are executed by a CPU.A process is a program in execution. A time shared user program such as a complier is a process. A word processing program being run by an individual user on a pc is a process.

A system task such as sending output to a printer is also a process. A process needs certain resources including CPU time, memory files & I/O devices to accomplish its task.

These resources are either given to the process when it is created or allocated to it while it is running. The OS is responsible for the following activities of process management.

Creating & deleting both user & system processes.

Suspending & resuming processes.

Providing mechanism for process synchronization.

Providing mechanism for process communication.

Providing mechanism for deadlock handling.

2. Main Memory Management:

The main memory is central to the operation of a modern computer system. Main memory is a large array of words or bytes ranging in size from hundreds of thousand to billions. Main memory stores the quickly accessible data shared by the CPU & I/O device. The central processor reads instruction from main memory during instruction fetch cycle & it both reads &writes data from main memory during the data fetch cycle. The main memory is generally the only large storage device that the CPU is able to address & access directly. For example, for the CPU to process data from disk. Those data must first be transferred to main memory by CPU generated E/O calls. Instruction must be in memory for the CPU to execute them. The OS is responsible for the following activities in connection with memory management.

Keeping track of which parts of memory are currently being used & by whom.

Deciding which processes are to be loaded into memory when memory space becomes available.

Allocating &deallocating memory space as needed.

3. File Management:

File management is one of the most important components of an OS computer can store information on several different types of physical media magnetic tape, magnetic disk & optical disk are the most common media. Each medium is controlled by a device such as disk drive or tape drive those has unique characteristics. These characteristics include access speed, capacity, data transfer rate & access method (sequential or random).For convenient use of computer system the OS provides a uniform logical view of information storage. The OS abstracts from the physical properties of its storage devices to define a logical storage unit the file. A file is collection of related information defined by its creator. The OS is responsible for the following activities of file management.

Creating & deleting files.

Creating & deleting directories.

Supporting primitives for manipulating files & directories.

Mapping files into secondary storage.

Backing up files on non-volatile media.

4. I/O System Management:

One of the purposes of an OS is to hide the peculiarities of specific hardware devices from the user. For example, in UNIX the peculiarities of I/O devices are hidden from the bulk of the OS itself by the I/O subsystem. The I/O subsystem consists of:

A memory management component that includes buffering, catching & spooling.

A general device- driver interfaces drivers for specific hardware devices. Only the device driver knows the peculiarities of the specific device to which it is assigned.

5. Secondary Storage Management:

The main purpose of computer system is to execute programs. These programs with the data they access must be in main memory during execution. As the main memory is too small to accommodate all data & programs & because the data that it holds are lost when power is lost. The computer system must provide secondary storage to back-up main memory. Most modern computer systems are disks as the storage medium to store data & program. The operating system is responsible for the following activities of disk management.

Free space management.

Storage allocation.

Disk scheduling

Because secondary storage is used frequently it must be used efficiently.