TED (10)-3071 Reg. No. …….

(REVISION-2010) Signature ……..……

FORTH SEMESTER DIPLOMA EXAMINATION IN ENGINEERING/ TECHNOLIGY- OCTOBER, 2014

OPERATING SYSTEM

(Common to CT, CM and IF)

[*Time:* 3 hours

(Maximum marks: 100)

Marks

PART –A
(Maximum marks: 10)

I. Answer all questions in a sentence

1. Define operating system

   An operating system (OS) is software that manages computer hardware and software resources and provides common services for computer programs. The operating system is an essential component of the system software in a computer system.

2. What is short term scheduling?

   Short term scheduling concerns with the allocation of CPU time to process in order to meet some pre-defined system performance objectives

3. Define memory management unit

   Memory management is the functionality of an operating system which handles or manages primary memory. It checks how much memory is to be allocated to processes.

4. Explain about opening a file

   It is a file operation. Most systems require that the programmer open a file explicitly with the *open()* system call before that file can be used.

www.indiapoly.com

5. What is bounded waiting?

   Bounded waiting means no process should wait for a resource for infinite amount of time.

PART – B

II. Answer *any five* questions. Each question carries 6 marks

1. What is goals of an operating system
   – Efficiency
   – Robustness
   – Scalability
   – Extensibility
   – Portability
   – Security
   – Protection
   – Interactivity
   – Usability

2. Explain context switch

   A context switch is the computing process of storing and restoring state (context) of a CPU so that execution can be resumed from the same point at a later time. This enables multiple processes to share a single CPU. The context switch is an essential feature of a multitasking operating system. Context switches are usually computationally intensive and much of the design of operating systems is to optimize the use of context switches. A context switch can mean a register context switch, a task context switch, a thread context switch, or a process context switch. What constitutes the context is determined by the processor and the operating system. Switching from one process to another requires a certain amount of time for doing the administration - saving and loading registers and memory maps, updating various tables and list etc

3. Explain fragmentation

www.gradinpoly.com

As processes are loaded and removed from memory, the free memory space is broken into little pieces is called fragmentation. There are two type of fragmentation

External Fragmentation – total memory space exists to satisfy a request, but it is not contiguous

- 50-percent rule: for first fit, given N allocated blocks, another 0.5N will be lost to fragmentation
    - 1/3 of memory may be unusable!!!

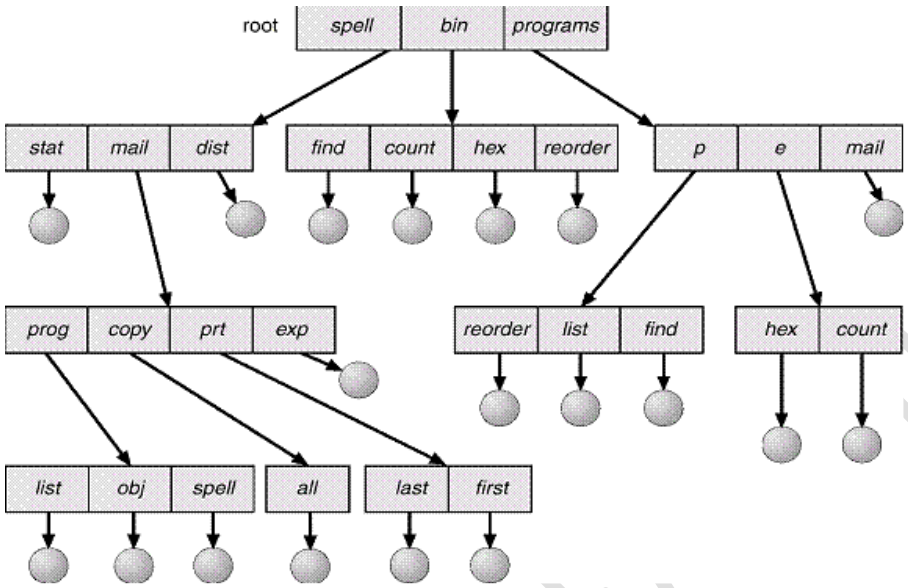
Internal Fragmentation – allocated memory may be slightly larger than requested memory; the extra bytes can-not be used by other processes

4. Explain file attributes
    - Read - Only allows a file to be read, but nothing can be written to the file.
    - Archive - Tells Windows Backup to backup the file.
    - System - System file.
    - Hidden - File will not be shown when doing a regular dir from DOS.
    - Volume Label: Every disk volume can be assigned an identifying label, either when it is formatted, or later through various tools
    - Directory: This is the bit that differentiates between entries that describe files and those that describe subdirectories within the current directory.

5. Explain Tree structured directory

Tree structured Directory: Once we have seen how to view a two-level directory as a two-level tree, the natural generalization is to extend the directory structure to a tree of arbitrary height. This generalization allows users to create their own subdirectories and to organize their files accordingly. A tree is the most common directory structure. The tree has a root directory, and every file in the system has a unique path name. A directory is simply another file, but it is treated in a special way. All directories have the same internal format.

www.madhu...com



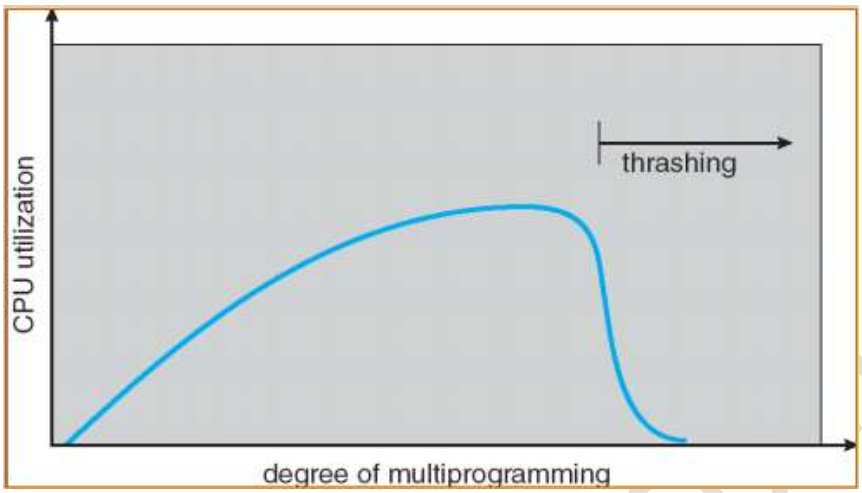
6. Explain trashing

Thrashing occurs when a system spends more time processing page faults than executing transactions. While processing page faults is necessary to in order to appreciate the benefits of virtual memory, thrashing has a negative effect on the system.

As the page fault rate increases, more transactions need processing from the paging device. The queue at the paging device increases, resulting in increased service time for a page fault. While the transactions in the system are waiting for the paging device, CPU utilization, system throughput and system response time decrease, resulting in below optimal performance of a system. Thrashing becomes a greater threat as the degree of multiprogramming of the system increases.

Thrashing is computer activity that makes little or no progress, usually because memory or other resources have become exhausted or too limited to perform needed operations. When this happens, a pattern typically develops in which a request is made of the operating system by a process or program, the operating system tries to find resources by taking them from some other process, which in turn makes new requests that can't be satisfied. In a virtual storage system (an operating system that manages its logical storage or memory in units called pages), thrashing is a condition in which excessive paging operations are taking place.

www.madinoly.com



7. Explain dead lock prevention

For a deadlock to occur, each of the four necessary conditions must hold simultaneously. Deadlock prevention is a set of methods for ensuring that at least one of the four necessary conditions cannot hold.

Mutual Exclusion: Removing the mutual exclusion condition means that no process may have exclusive access to a resource. Mutual-exclusion conditions must hold for non-sharable resources and the sharable resources do not require mutually exclusive access. We cannot prevent deadlocks by denying the mutual-exclusion condition: Some resources are intrinsically non-sharable occur.

Hold and Wait: The "hold and wait" conditions may be removed by ensuring that, whenever a process requests a device, it does not hold any other devices.

- One protocol requires each process to request and be allocated all its resources before it begins execution.
- Another protocol requires the processes to release all their allocated resources before requesting additional resources.
- Disadvantages: Low resource utilization Starvation is possible.

No pre-emption: The third necessary condition is that there is no preemption of resources that have already been allocated. To ensure that this condition does not hold, two protocols are there.

www.madinpoly.com

- If a process is holding some resources and requests another resources that cannot be immediately allocated to it, then all the resources currently being held by the process are preempted (released).
- If a process requests some resources, first check whether they are available.
- If they are available, allocate them.
- If they are not available and are allocated to some other processes waiting for additional resources, preempt the requested resources from the waiting process and allocate to the requesting process.
- If the resources are not available and not held by a waiting process, the requesting process must wait. While waiting, some of the resources may be preempted if some other process requests them.

Circular wait condition: To ensure that the circular wait condition never holds is to impose a total ordering for all resources types.

Let R={ R1, R2,...,Rn} be the set of resource types. Define a one-to-one function: R→N, where N is a set of natural numbers. I.e. assign a unique integer number to each resource type.

- Each process requests resources only in an increasing order of enumeration.

Whenever a process requests an instance of resource type Rj, it has released any resources Ri such that F(Ri) ≥ F(Rj).

PART – C

(Answer *one* full question from each unit. Each question carries 15 marks)

UNIT – I

III.

a) Explain the following 15

i. Linker

ii. Batch System

iii. UNIX os

www.madinpoly.com

i. A linker or link editor is a computer program that takes one or more object files generated by a compiler and combines them into a single executable file, library file, or another object file. Linkers can take objects from a collection called a *library*. Some linkers do not include the whole library in the output; they only include its symbols that are referenced from other object files or libraries. Libraries exist for diverse purposes, and one or more system libraries are usually linked in by default.

The linker also takes care of arranging the objects in a program's address space. This may involve *relocating* code that assumes a specific base address to another base. Since a compiler seldom knows where an object will reside, it often assumes a fixed base location (for example, zero). Relocating machine code may involve re-targeting of absolute jumps, loads and stores.

ii. **Batch processing system**: In a batch system, more processes are submitted than can be executed immediately. These processes are spooled to a mass-storage device (typically a disk), where they are kept for later execution. Finally, the operator retrieves the output of all these jobs and returns them to the concerned users. Batch processing is most suitable for tasks where a large amount of data has to be processed on a regular basis. Batch jobs are set up so they can be run to completion without manual intervention, so all input data is preselected through scripts or command-line parameters. This is in contrast to "online" or interactive programs which prompt the user for such input. A program takes a set of data files as input, processes the data, and produces a set of output data files. This operating environment is termed as "batch processing" because the input data are collected into batches of files and are processed in batches by the program.

iii. UNIX was meant to be a programmer's workbench to be used for developing software to be run on multiple platforms more than to be used to run application software. The system grew larger as the operating system started spreading in the

academic circle, as users added their own tools to the system and shared them with colleagues

UNIX was designed to be portable, multi-tasking and multi-user in a time-sharing configuration. Unix systems are characterized by various concepts: the use of plain text for storing data; a hierarchical file system; treating devices and certain types of inter-process communication (IPC) as files; and the use of a large number of software tools, small programs that can be strung together through a command using pipes, as opposed to using a single monolithic program that includes all of the same functionality. These concepts are collectively known as the "Unix philosophy." Brian Kernighan and Rob Pike summarize this in *The UNIX Programming Environment* as "the idea that the power of a system comes more from the relationships among programs than from the programs themselves.

OR

IV.

a) Explain time sharing system 8

Time sharing system

Timesharing refers to the allocation of computer resources in a time-dependent fashion to several programs simultaneously. In time-sharing systems, the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running. Time sharing requires an interactive (or hands-on) computer system, which provides direct communication between the user and the system. By allowing a large number of users to interact concurrently with a single computer, time-sharing dramatically lowered the cost of providing computing capability, made it possible for individuals and organizations to use a computer without owning one, and promoted the interactive use of computers and the development of new interactive applications.

- Time sharing, or multitasking, is a logical extension of multiprogramming.
- Multiple jobs are executed simultaneously by switching the CPU between them.
- Each job gets a predetermined “time slice”
- Time slice is defined by the OS, for sharing CPU time between processes.

www.madinpc...

- At the end of time slice current job is set aside and a new one starts
- In this, the CPU time is shared by different processes, so it is called as "Time sharing Systems".
- Examples: Multics, Unix, etc.,

b) Compare compiler and interpreter 7

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

UNIT – II

V.

a) Explain resource allocation graph 8

Deadlocks can be described more precisely in terms of a directed graph called a system resource-allocation graph. This graph consists of a set of vertices V and a set of edges E. The set of vertices V is partitioned into two different types of nodes: P - {Pi,

Pi,,.., P,,\, the set consisting of all the active processes in the system, and R = {R[, R2, •••/ Rm}, the set consisting of all resource types in the system.

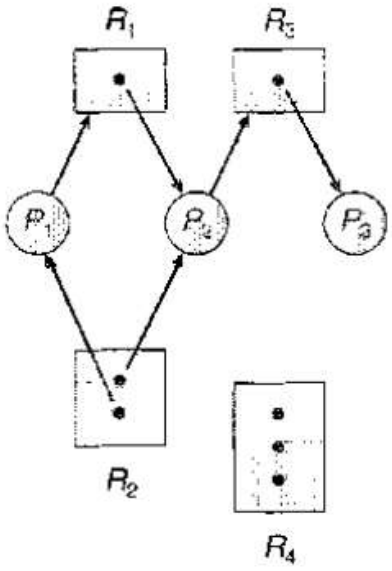
The resource-allocation graph shown in Figure depicts the following situation.

The sets P, *R,* and £:

- *P={PhP2/P?,}*
- *R=* {/?!, *RZ,R3,* R;}
- £ = {p, _>*Ru P2* _> R3/ R, _>*p2f R2* _> P2/ *R2* _> p.,, *R3* ->P3 }

Resource instances:

- One instance of resource type R1
- Two instances of resource type R2
- One instance of resource type *R3*
- Three instances of resource type *R4*



Resource allocation graph

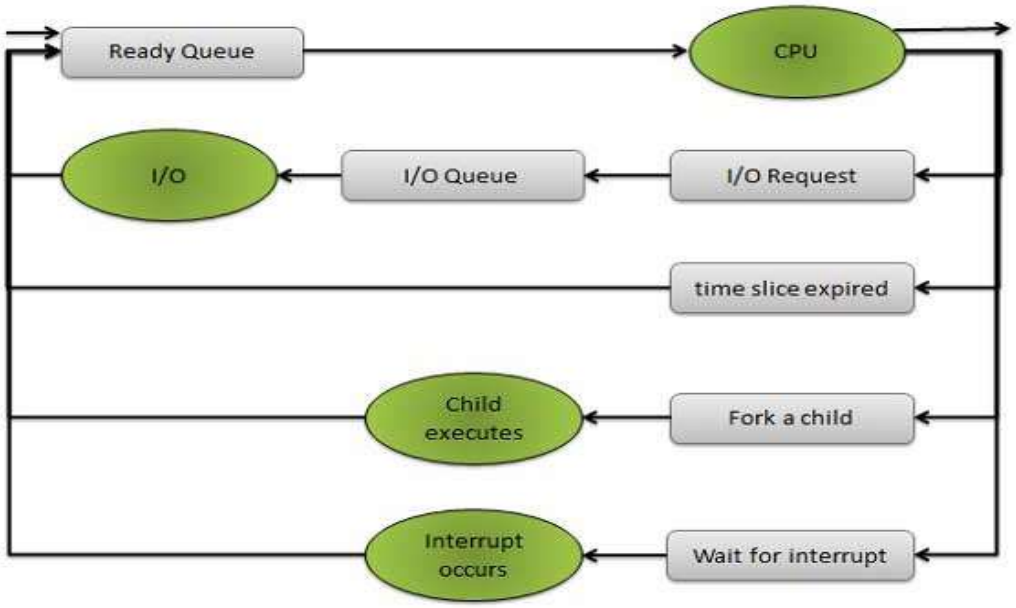
Process states:

- Process P1is holding an instance of resource type R2 and is waiting for an instance of resource type R1.
- Process Pn is holding an instance of R1 and an instance of R2 and is waiting for an instance of R3.
- Process P3 is holding an instance of R3.

b) Explain process scheduling queue 7

www.[illegible]oly.com

Scheduling queues refers to queues of processes or devices. When the process enters into the system, then this process is put into a job queue. This queue consists of all processes in the system. The operating system also maintains other queues such as device queue. Device queue is a queue for which multiple processes are waiting for a particular I/O device. Each device has its own device queue.



Queues are of two types

- Ready queue

  Set of all process residing in main memory, ready and waiting to execute
- Device queue

  Set of all process waiting for an I/O device

A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.

- The process could issue an I/O request and then it would be placed in an I/O queue.
- The process could create new sub process and will wait for its termination.
- The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

OR

VI.

a) Explain producer-consumer problem 6

The producer–consumer problem (also known as the bounded-buffer problem) is a classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer used as a queue. The producer's job is to generate a piece of data, put it into the buffer and start again. At the same time, the consumer is consuming the data (i.e., removing it from the buffer) one piece at a time. The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

The solution for the producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer. The solution can be reached by means of inter-process communication, typically using semaphores. An inadequate solution could result in a deadlock where both processes are waiting to be awakened. The problem can also be generalized to have multiple producers and consumers.

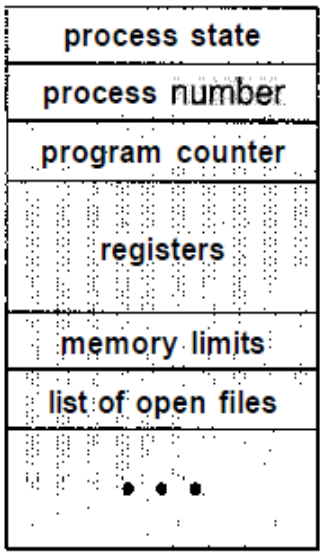
b) Explain PCB with the help of diagram 9

Each process is represented in the operating system by a **process control block (PCB)** also called a *task control block.* A PCB is shown in Figure. It contains many pieces ofinformation associated with a specific process, including these:

- **Process state.** The state may be new, ready, running, and waiting, halted, and so on.
- **Program counter.** The counter indicates the address of the next instruction to be executed for this process.
- 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. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward

- **CPU-scheduling information.** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- **Accounting information.** This information includes the amount of CPU nd real time used, time limits, account members, job or process numbers, and so on.
- **I/O status information.** This information includes the list of I/O devices allocated to the process, a list of open files, and so on.



Process control block (PCB).

UNIT – III

VII.

a) Explain difference between physical address and logical address 7

- An address generated by the CPU is commonly referred to as a logical a logical address. The set of all logical addresses generated by a program is known as logical address space. Whereas, an address seen by the memory unit- that is, the one loaded into the memory-address register of the memory- is commonly referred to as physical address.

- The compile-time and load-time address-binding methods generate identical logical and physical addresses. However, in the execution-time address-binding scheme, the logical and physical-address spaces differ
- The user program never sees the physical addresses. The programs creates a pointer to a logical address, say 346, stores it in memory, manipulate it, compares it to other logical addresses- all as the number 346. Only when a logical address is used as memory address, it is relocated relative to the base/relocation register. The memory-mapping hardware device called the memory- management unit (MMU) converts logical addresses into physical addresses.
- Logical addresses range from 0 to max. User program that generates logical address thinks that the process runs in locations 0 to max.

b) Explain the following:

i. FIFO

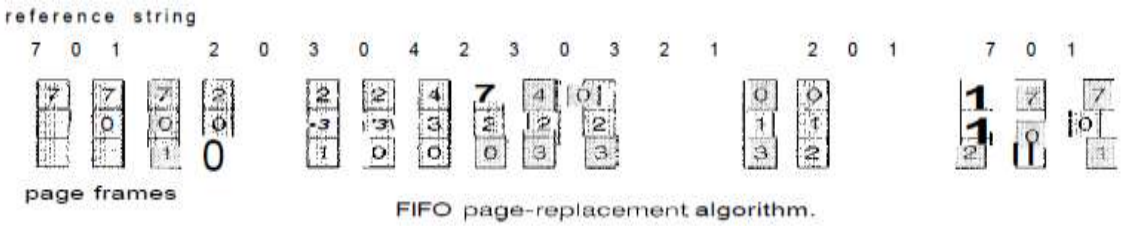
ii. Multiple partition allocation 8

i. **FIFO Page Replacement**

For our example reference string, our three frames are initially empty. The first three references (7,0,1) cause page faults and are brought into these empty frames. The next reference (2) replaces page 7, because page 7 was brought in first. Since 0 is the next reference and 0 is already in memory, we have no fault for this reference. The first reference to 3 results in replacement of page 0, since it is now first in line. Because of this replacement, the next reference, to 0, will fault. Page 1 is then replaced by page 0. This process continues as shown in Figure. Every time a fault occurs, we show which pages are in our three frames. There are 15 faults altogether.

The FIFO page-replacement algorithm is easy to understand and program. However, its performance is not always good. On the one hand, the page replaced may be an initialization module that was used a long time ago and is no longer needed. On the other hand, it could contain a heavily used variable that was initialized early and is in constant use.

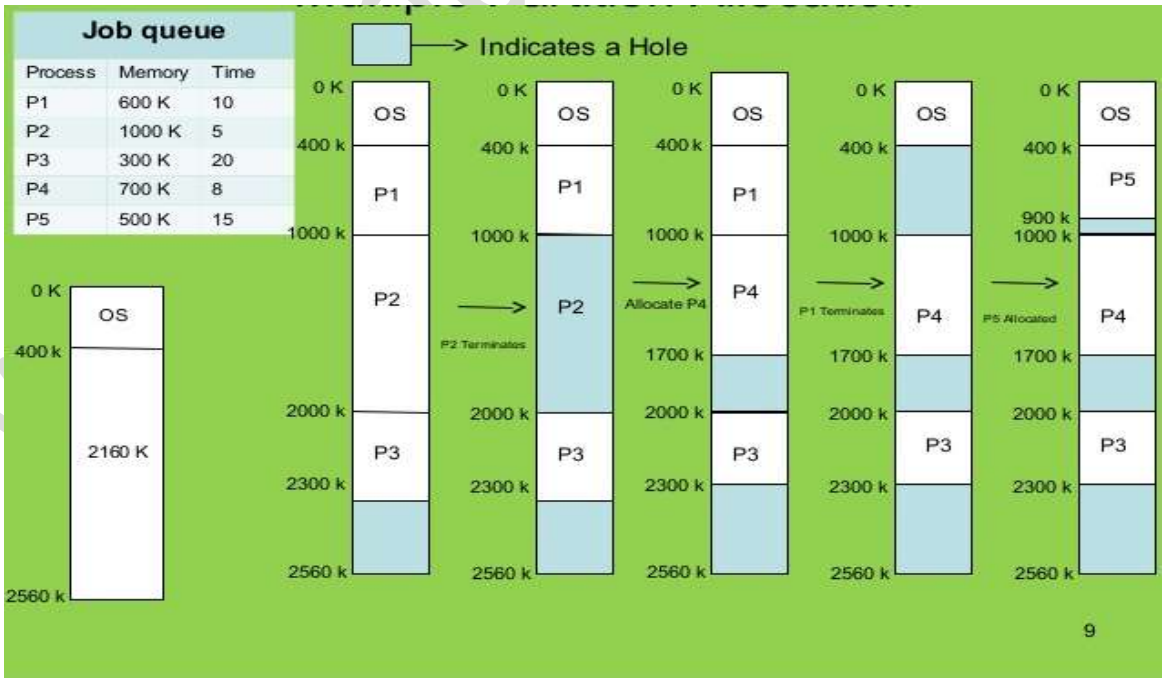
www.madinpoly.com



FIFO page-replacement algorithm.

ii.

- Multiple partitions may be used to isolate critical data into a separate partition, on the theory that if the main partitions file system gets "hosed" the data can be recovered
- Operating system maintains information about
    - Allocated partitions
    - Free partitions
- Operating system maintains a table of this memory
- Space is allocated based on this table
- Adjacent free spaces merged to get largest holes
- Consider an example consisting of 5 processes from P1 o P5, this process occupy some space in memory and corresponding burst times for their execution as shown in the table

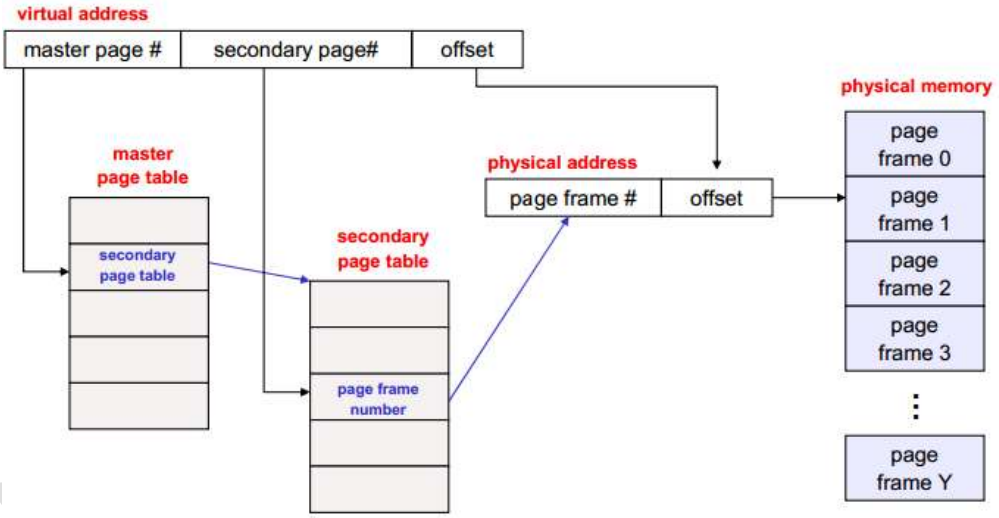


OR

VIII.

a) Explain two level paging 8

- With two-level PT's, virtual addresses have 3 parts:
  – Master page number, secondary page number, offset
  – Master PT maps master PN to secondary PT
  – Secondary PT maps secondary PN to page frame number
  – offset + PFN = physical address
- Example:
  – 4KB pages, 4 bytes/PTE
    • How many bits in offset? need 12 bits for 4KB
  – Want master PT in one page: 4KB/4 bytes = 1024 PTE
    • Hence, 1024 secondary page tables
  – So: master page number = 10 bits, offset = 12 bits
    • With a 32 bit address, that leaves 10 bits for secondary PN

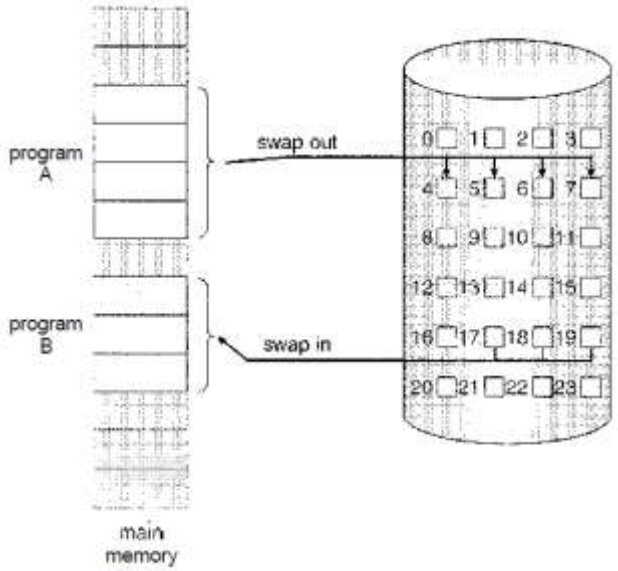


b) Explain demand paging 7

Consider a program that starts with a list of available options from which the user is to select. Loading the entire program into memory results in loading the executable code for all options, regardless of whether an option is ultimately selected by the user or not. An alternative strategy is to initially load pages only as they are needed. This technique is known as demand paging and is commonly used in virtual memory systems. With demand-paged virtual memory, pages are only loaded when they are demanded during

program execution; pages that are never accessed are thus never loaded into physical memory.

A demand-paging system is similar to a paging system with swapping where processes reside in secondary memory (usually a disk). When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory, however, we use a lazy swapper. A lazy swapper never swaps a page into memory unless that page will be needed. Since we are now viewing a process as a sequence of pages, rather than as one large contiguous address space, use of the term swapper is technically incorrect. A swapper manipulates entire processes, whereas a pager is concerned with the individual pages of a process. We thus use pager, rather than swapper, in connection with demand paging.



Transfer of a paged memory to contiguous disk space.

UNIT – IV

IX.

a) Explain different file organization concepts 7

Sequential File Organization

Every file record contains a data field (attribute) to uniquely identify that record. In sequential file organization mechanism, records are placed in the file in the some sequential order based on the unique key field or search key. Practically, it is not possible

to store all the records sequentially in physical form. This is the most common structure for large files that are typically processed in their entirety, and it's at the heart of the more complex schemes. In this scheme, all the records have the same size and the same field format, with the fields having fixed size as well. The records are sorted in the file according to the content of a field of a scalar type, called ``key''. The key must identify uniquely a records, hence different record have different keys. This organization is well suited for batch processing of the entire file, without adding or deleting items: this kind of operation can take advantage of the fixed size of records and file; moreover, this organization is easily stored both on disk and tape.

Indexed file organization

Indexes can be obviously built for each field that uniquely identifies a record (or set of records within the file), and whose type is amenable to ordering. Multiple indexes hence provide a high degree of flexibility for accessing the data via search on various attributes; this organization also allows the use of variable length records (containing different fields).

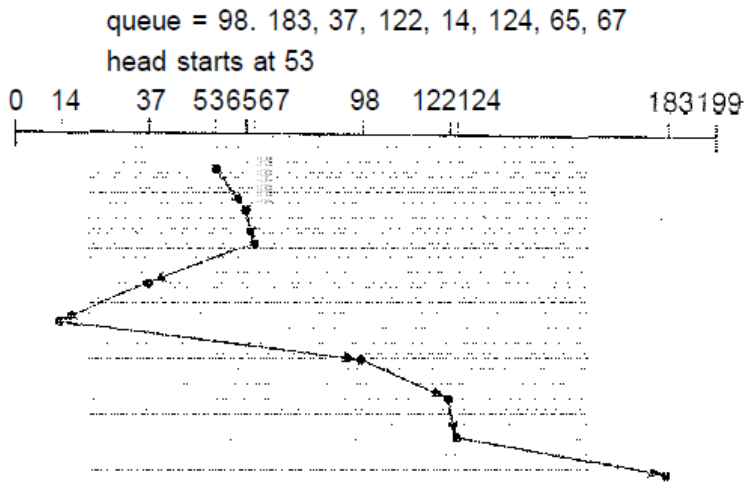
It should be noted that when multiple indexes are used the concept of sequentiality of the records within the file is useless: each attribute (field) used to construct an index typically imposes an ordering of its own. For this very reason is typically not possible to use the ``sparse'' type of indexing previously described. Two types of indexes are usually found in the applications: the exhaustive type, which contains an entry for each record in the main file, in the order given by the indexed key, and the partial type, which contain an entry for all those records that contain the chosen key field

b) Explain the following:
   i. SSTF
   ii. C-SCAN 8

i.

It seems reasonable to service all the requests close to the current head position before moving the head far away to service other requests. This assumption is the basis for the **shortest-seek-time-first (SSTF) algorithm.** The SSTF algorithm selects the request with the minimum seek time from the current head position. Since seek time

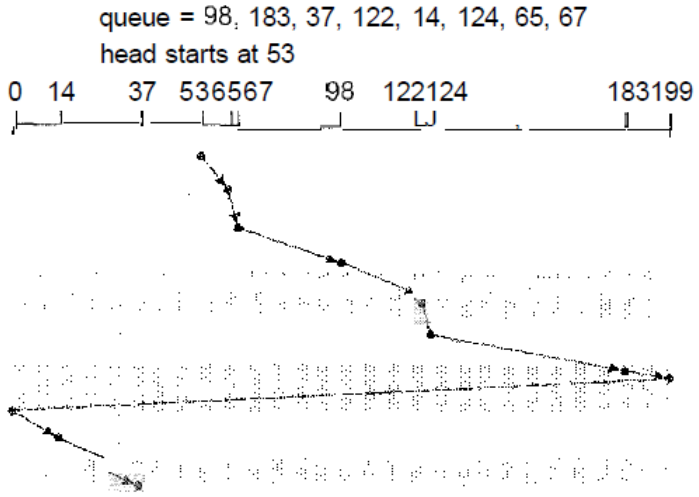
increases with the number of cylinders traversed by the head, SSTF chooses the pending request closest to the current head position. For our example request queue, the closest request to the initial head position (53) is at cylinder 65. Once we are at cylinder 65, the next closest request is at cylinder 67. From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next. Continuing, we service the request at cylinder 14, then 98,122, 124, and finally 183. This scheduling method results in a total head movement of only 236 cylinders—little more than one-third of the distance needed for FCFS scheduling of this request queue. This algorithm gives a substantial improvement in performance.



SSTF disk scheduling.

ii.

Circular SCAN (C-SCAN) scheduling is a variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip. The C-SCAN scheduling algorithm essentially treats the cylinders as a circular list that wraps around from the final cylinder to the first one.



OR

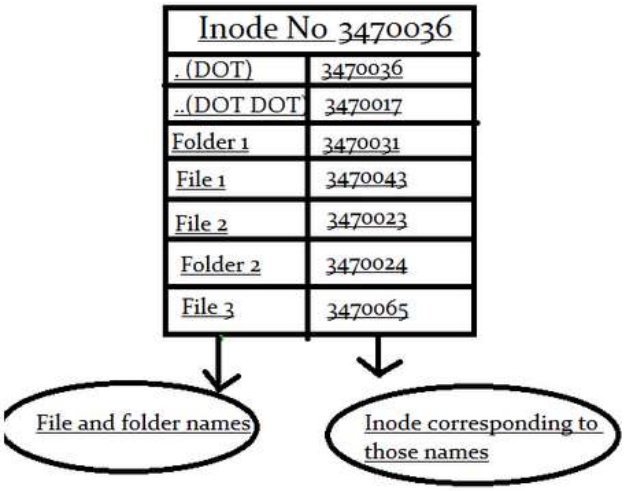
X.

a) Explain inode. 7

Inode is a data structure that keeps track of all the information about a file. You keep your information in a file and the OS stores the information about a file in an inode. Information about files is sometimes called metadata. We can say that an inode is metadata of the data.

Whenever you need to access a file, OS first looks for the exact and unique inode in a table called inode table. In fact, the application or the user who accesses a file reaches the file with the help of the inode number provided by the inode table. To get to a particular file by its name, the OS needs an inode number corresponding to that file. Now, to reach an inode number you don't need to know the file name. In fact, by knowing the inode number of a file you can access the data stored in that file.

An interesting fact is that the total number of inodes is created when a file system is set up. This means there is a limit to the number of inodes you can have in your file system. After that limit is reached, you won't be able to create any more files, even if you have space left on the partition



INODE STRUCTURE OF A DIRECTORY

b) Explain the following 8

i. Contiguous allocation

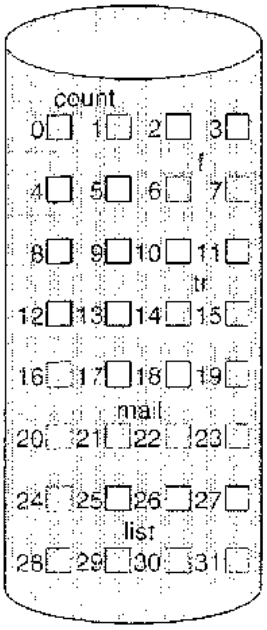
ii. Linked allocation

Contiguous Allocation

Contiguous allocation requires that each file occupy a set of contiguous blockson the disk. Disk addresses define a linear ordering on the disk. With thisordering, assuming that only one job is accessing the disk, accessing block*b* +1 after block *b* normally requires no head movement. When head movementis needed (from the last sector of one cylinder to the first sector of the nextcylinder), the head need only move from one track to the next. Thus, the numberof disk seeks required for accessing contiguously allocated files is minimal, asis seek time when a seek is finally needed.

Contiguous allocation of a file is defined by the disk address and length (inblock units) of the first block. If the file is *n* blocks long and starts at location*b,* then it occupies blocks *b*, *b* + 1, *b* + *2*, ...,*b* + *n* — 1. The directory entry foreach file indicates the address of the starting block and the length of the areaallocated for this file

www.madinpoly.com

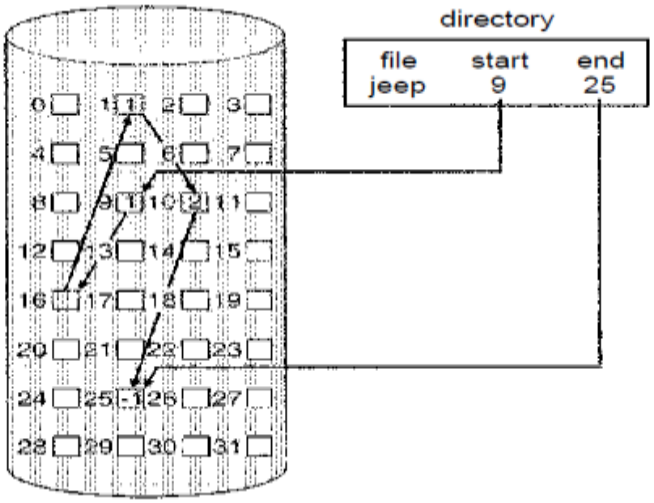


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Linked Allocation

Linked allocation solves all problems of contiguous allocation. With linkedallocation, each file is a linked list of disk blocks; the disk blocks may bescattered anywhere on the disk. The directory contains a pointer to the firstand last blocks of the file. For example, a file of five blocks might start at block9 and continue at block 16, then block 1, then block 10, and finally block 25. Each block contains a pointer to the next block. These pointersare not made available to the user. Thus, if each block is 512 bytes in size, anda disk address (the pointer) requires 4 bytes, then the user sees blocks of 508bytes.



www.madinpoint.com