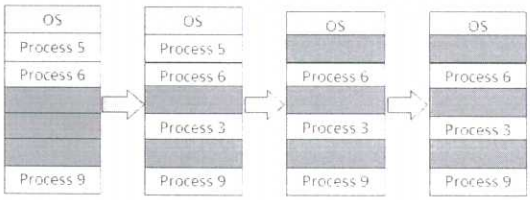
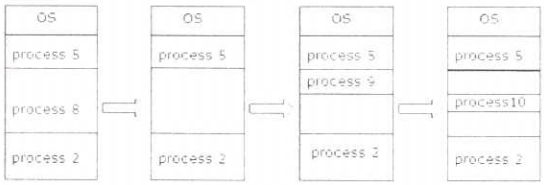
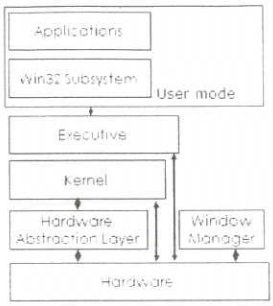
	Operating Systems- Rev 2015 TED 4134 ANSWER KEY (A)		
Q no	**Scoring Indicators**	**split score**	**Total score**
I	**PART A**		
1	It is a translator Translates assembly language program into machine language	2	2
2	Thread is the smallest sequence of programmed instructions that can be managed independently	2	2
3	address generated by the CPU; also referred to as virtual address	2	2
4	Sequential Access Direct Access/ Random Access Indexed Access	Any 2*1	2
5	VMWare Virtual Box Etc.	Any 2 * 1	2
II	**PART B**		
1	**Compiler** • Is a translator program • Translates high level language program into its equivalent machine language • Entire program is translated into machine language before execution • Object program is stored into a separate file **Interpreter** • Is a translator program • Translates high level language program into its equivalent machine language • Each high level language statement is translated and executed before translating the next statement • Object program is not stored into file	3+3	6
2	• A real time system is used when rigid time requirements have been placed on the operation of a processor or the flow of data • Real time system has well defined fixed time constraints • Processing must be done within the defined constraints, or the system will fail **Two flavors** *Hard Real Time System* - Guarantees that critical task be completed on time *Soft Real Time System* – Critical task gets priority over other task, and retains that priority until it completes.	4+1+1	6
3	**Preemptive scheduling** is used when a process switches from running state to ready state or from waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining **Non-preemptive Scheduling** does not interrupt a process running CPU in middle of the execution. Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.	3+3	6
4	I/O-bound process – spends more time doing I/O than computations, many short CPU bursts. CPU-bound process – spends more time doing computations; few very long CPU bursts.	3+3	6
5	**fixed-sized partitions** Divide memory into several fixed-sized partitions Each partition may contain exactly one process	3+3	6

1/10

	When the process terminates, the partition becomes available for another process When a partition is free, a process is loaded into it  **variable-sized partitions** Operating system keeps a table indicating which parts of memory are available and which are occupied When process arrives and needs memory, OS search for a hole large enough for this process(Hole – block of available memory) If find one, OS allocate only as much memory as is needed  		
6	**Segmentation** / **Paging** User view of memory / System view of memory View memory as a collection of variable-sized segments, with no necessary ordering among segments / View memory as a collection of fixed-sized frames The addresses specify both the segment name and the offset within the segment / user specified only a single address. Hardware partitioned the address into a page number and offset User specifies the address / Address in invisible to user	Any 3*2	6
7	• Name - symbolic file-name, in human-readable form • Identifier - This unique tag, it is the non-human-readable name for the file • Type - for systems that support multiple types • Location - pointer to a device and to file location on device • Size - current size of the file • Protection - controls who can read, write, execute • Time, Date and user identification • data for protection, security and usage monitoring	Any 3*2	6
	PART C		
III (a)	**Windows** • Is a series of operating systems produced by Microsoft • Support graphical user interfaces (GUIs) • Windows offers a single, monolithic user interface • Windows has two major series, a desktop oriented series (Windows 9x), network oriented series (Windows NT). • For embedded systems, Windows CE • Use pre-emptive multitasking and multithreading to perform several actions at the same time.	4.5+4.5	9

2/10



Linux

- Linux is multi-user, multi-tasking operating system
- Linux system is described as kernel and shell
- Kernel controls hard wares, CPU, memory, hard disk, network card etc.
- Shell is an interface between user and kernel
- Shell interprets input commands and passes them to kernel
- Data, directory, process, hard disk etc (almost everything) are expressed as a file
- Files are put in a directory
- All directories are in a hierarchical

(b)

Sr. No. Multiprocessing

1 Multiprocessing refers to processing of multiple processes at same time by multiple CPUs.
2 It utilizes multiple CPUs.
3 It permits parallel processing.
4 Less time taken to process the jobs.
5 It facilitates much efficient utilization of devices of the computer system.
6 Usually more expensive.

Sr. No. Multiprogramming

1 Multiprogramming keeps several programs in main memory at the same time and execute them concurrently utilizing single CPU.
2 It utilizes single CPU.
3 Context switching takes place.
4 More Time taken to process the jobs.
5 Less efficient than multiprocessing.
6 Such systems are less expensive.

3+3 | 6

IV (a)

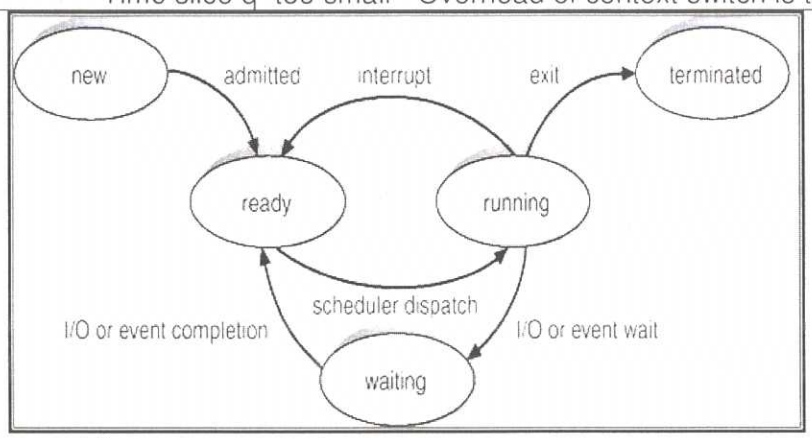
A program that acts as an intermediary between a user of a computer and the computer hardware

Functions of operating systems are:

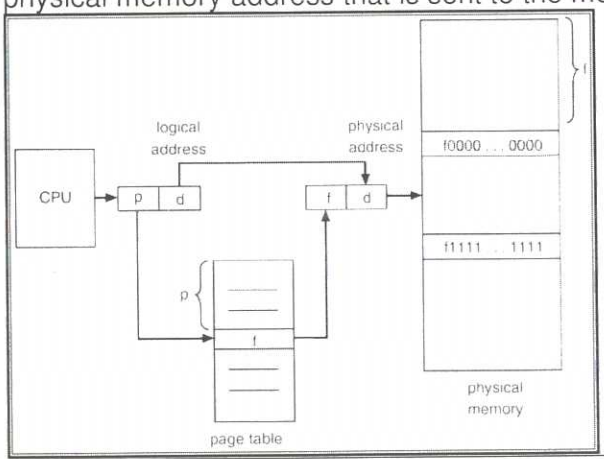
- Manages and Interacts with Computer Hardware
- Resource management
- Provides and Manages System Security
- Provides the System Interface
- Provides the Interface for Application Software

Def 1 M + funs. With expl. Any 4*2 | 9

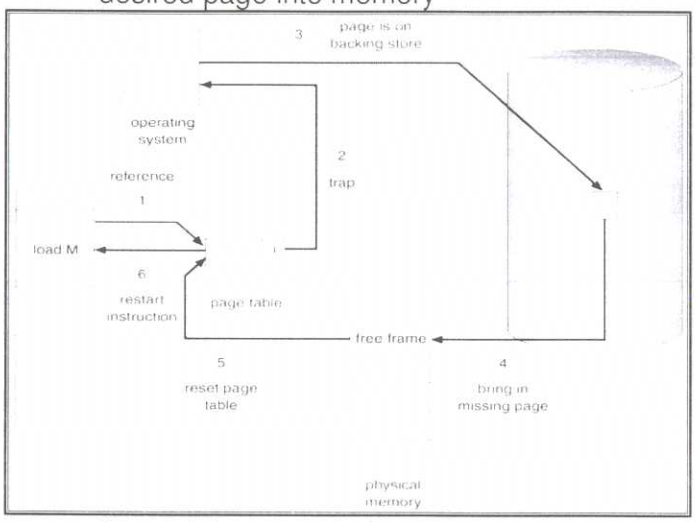
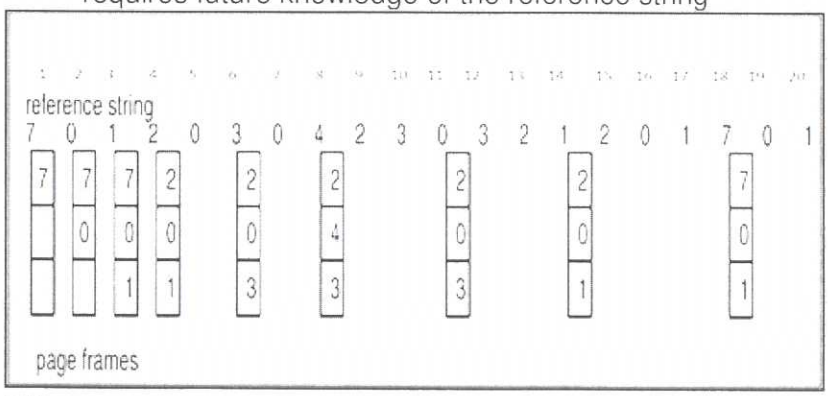
3/10

(b)	Is a program that allocate memory and loads program into memory for execution. Also perform relocation and linking of program modules **Functions** • **Allocation**: allocate space in memory for the programs • **Linking:** Resolve symbolic references between object files • Combines two or more separate object programs • **Relocation**: Adjust all address dependent locations, such as address constants, to correspond to the allocated space • Modifies the object program so that it can be loaded at an address different from the location originally specified • **Loading:** Physically place the machine instructions and data into memory	2M + 4*1	6
V (a)	**Shortest-Job-First(SJF) Scheduling** • Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time. • Two Schemes: • 1: Non-preemptive • Once CPU is given to the process it cannot be preempted until it completes its CPU burst. • 2: Preemptive • If a new CPU process arrives with CPU burst length less than remaining time of current executing process, preempt. Also called Shortest-Remaining-Time-First (SRTF). • SJF is optimal - gives minimum average waiting time for a given set of processes. **Round Robin (RR)** Each process gets a small unit of CPU time • Time quantum usually 10-100 milliseconds. • After this time has elapsed, the process is preempted and added to the end of the ready queue. n processes, time quantum = q • Each process gets 1/n CPU time in chunks of at most q time units at a time. • No process waits more than (n-1)q time units. Performance • Time slice q too large - FIFO behavior • Time slice q too small - Overhead of context switch is too expensive.	4.5 + 4.5	9
(b)	 new: The process is being created. running: Instructions are being executed. waiting: The process is waiting for some event to occur. ready: The process is waiting to be assigned to a process. terminated: The process has finished execution.	Fig 3 +3	6
VI (a)	Consider a system consisting of N process • All competing to use shared data. • Each process has a code segment, called critical section, in which the	Expl 3+ 3*2	9

4/10

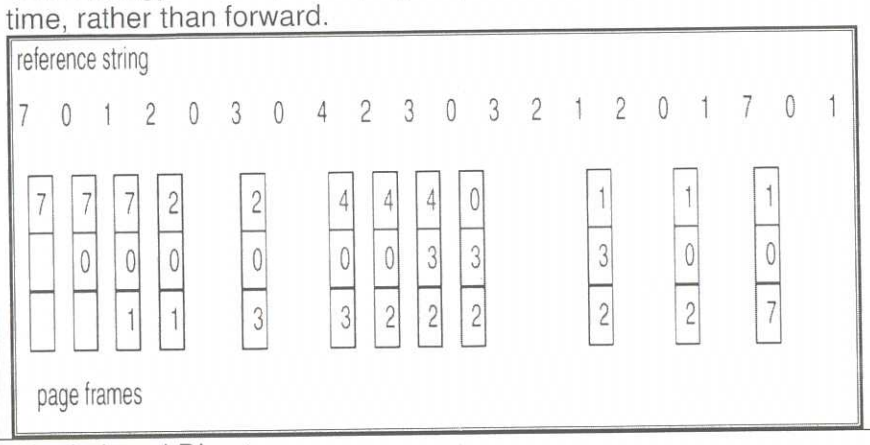
	shared data is accessed. • Problem – ensure that when one process is executing in its critical section, no other process is allowed to execute in its critical section **Mutual Exclusion** • If process Pi is executing in its critical section, then no other processes can be executing in their critical sections. **Progress** • If no process is executing in its critical section and there exists some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely. **Bounded Waiting** • A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.		
(b)	A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set. • A process is deadlocked if it is waiting for an event that will never occur. **Mutual Exclusion:** • Only once process at a time can use the resource. **Hold and Wait:** • Processes hold resources already allocated to them while waiting for other resources. **No preemption:** • Resources are released by processes holding them only after that process has completed its task. **Circular wait:** • A circular chain of processes exists in which each process waits for one or more resources held by the next process in the chain	Expl 2+ 4*1	6
VII (a)	• Memory management scheme that permits the physical-address space of a process to be non-contiguous • Physical memory is broken into fixed-sized blocks: frames • Logical memory is also broken into same size blocks: pages • When a process is to be executed, its pages are loaded into any available memory frames • Every address generated by the CPU is divided into two parts: a page number (p) and a page offset (d) • Page number is used as an index into a page table • Page table contains the base address of each page in physical memory • This base address is combined with the page offset to define the physical memory address that is sent to the memory unit  	Fig 5 + expl 4	9
(b)	STRATEGIES USED TO SELECT MEMORY PARTITIONS 1. First Fit 2. Best Fit 3. Worst Fit **FIRSTFIT** - Allocate the first hole that is big enough.	3*2	6

5/10

	BESTFIT: Allocate the smallest hole that is big enough **WORST FIT**: Allocate the largest hole		
VIII (a)	• to distinguish between pages that are in memory and pages that are on the disk • Valid-invalid bit scheme can be used for this purpose • When this bit is set to "valid," indicates the associated page is both legal and in memory • If the bit is set to "invalid,", indicates the page either is not valid, or is valid but is currently on the disk. • Access to a page marked invalid causes a **page-fault** • Paging hardware, in translating the address through the page table, will notice that the invalid bit is set, causing a trap to the operating system • This trap is the result of the operating system's failure to bring the desired page into memory  • Check internal table for this process to determine whether the reference was a valid or invalid memory access • If the reference was invalid, we terminate the process • If it was valid, and is not brought in, now page in • Find a free frame. • Read the desired page into the newly allocated frame. • Modify the internal tables	2 + Fig 4 + 3	9
(b)	Optimal page-replacement algorithm • Replace the page that will not be used for the longest period of time • Use of this page-replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames • Optimal page-replacement algorithm is difficult to implement, because it requires future knowledge of the reference string  LRU Recent past is used as an approximation of the near future, then replace the page that has not been used for the longest period of time LRU (Least Recently Used) replacement associates with each page the time of that page's last use When a page must be replaced, LRU chooses that page that has not been used	3+3	6

6/10

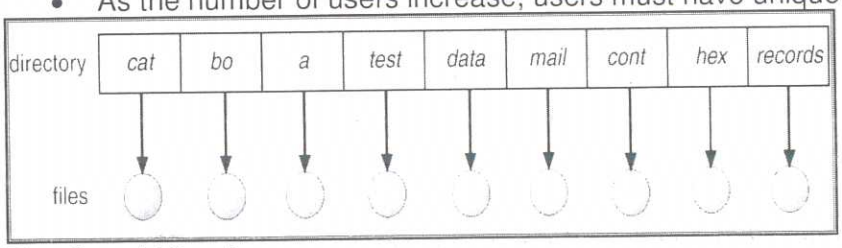
for the longest period of time
This strategy is the optimal page-replacement algorithm looking backward in time, rather than forward.



IX (a) Single Level Directory
Two Level Directory
Tree Structured Directory

Single Level Directory

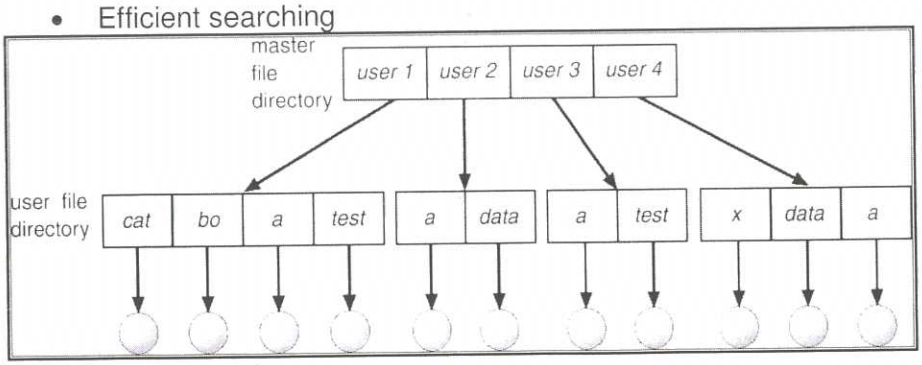
- A single directory for all users (all the files are in the same directory)
- Naming Problem and Grouping Problem
- As the number of files increases, difficult to remember unique names
- As the number of users increase, users must have unique names.



Two Level Directory

Introduced to remove naming problem between users

- First Level contains list of user directories
- Second Level contains user files
- Need to specify Path name
- Can have same file names for different users.
- System files kept in separate directory or Level 1.
- Efficient searching

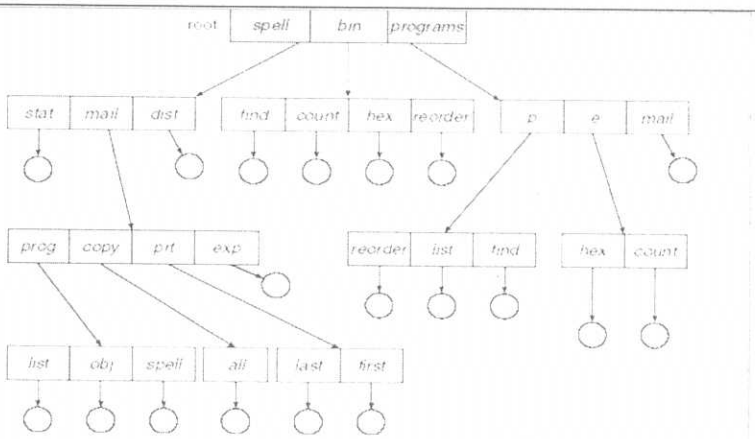


Tree Structured Directory

- Arbitrary depth of directories
  Leaf nodes are files, interior nodes are directories.
- Efficient Searching
- Grouping Capability
- Current Directory (working directory)
- MS-DOS, Linux uses a tree structured directory

3+3+3 | 9

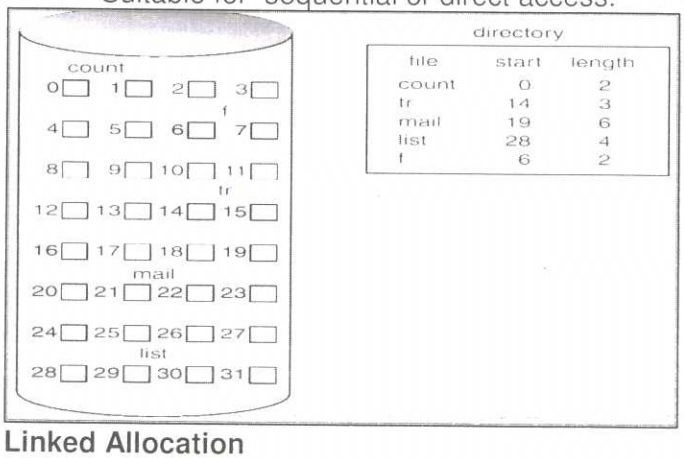
7/10



(b)

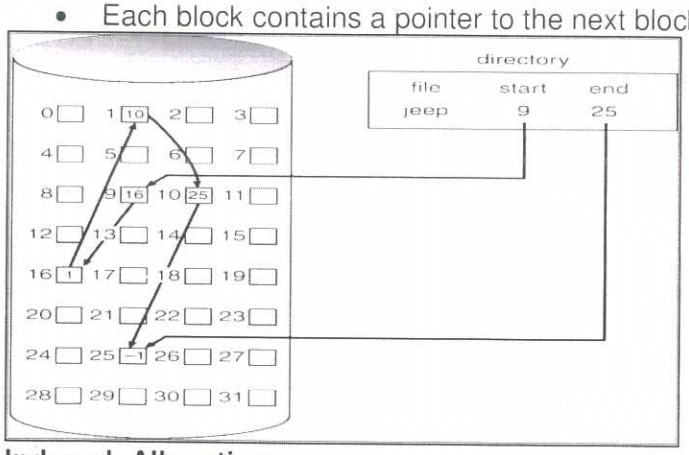
Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Simple - only starting address and length are required.
- Suitable for sequential or direct access.



Linked Allocation

- Each file is a linked list of disk blocks
- Blocks may be scattered anywhere on the disk.
- The directory contains a pointer to the first and last blocks of a file.
- Each block contains a pointer to the next block.

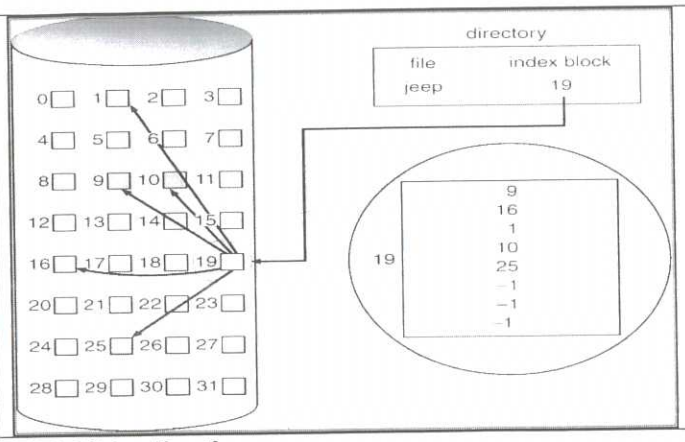


Indexed Allocation

- All pointers are stored into one location - The index block.
- Need index table.
- Supports sequential, direct and indexed access.
- For a new file all the pointers in the index block are set to NIL

2+2+2 | 6

8/10



X (a)

Full Virtualization

- In full virtualization, the guest operating system is unaware that it is in a virtualized environment
- The hardware is virtualized by the host operating system
- The guest can issue commands to what it thinks is actual hardware, but really are just simulated hardware devices created by the host.

Para Virtualization

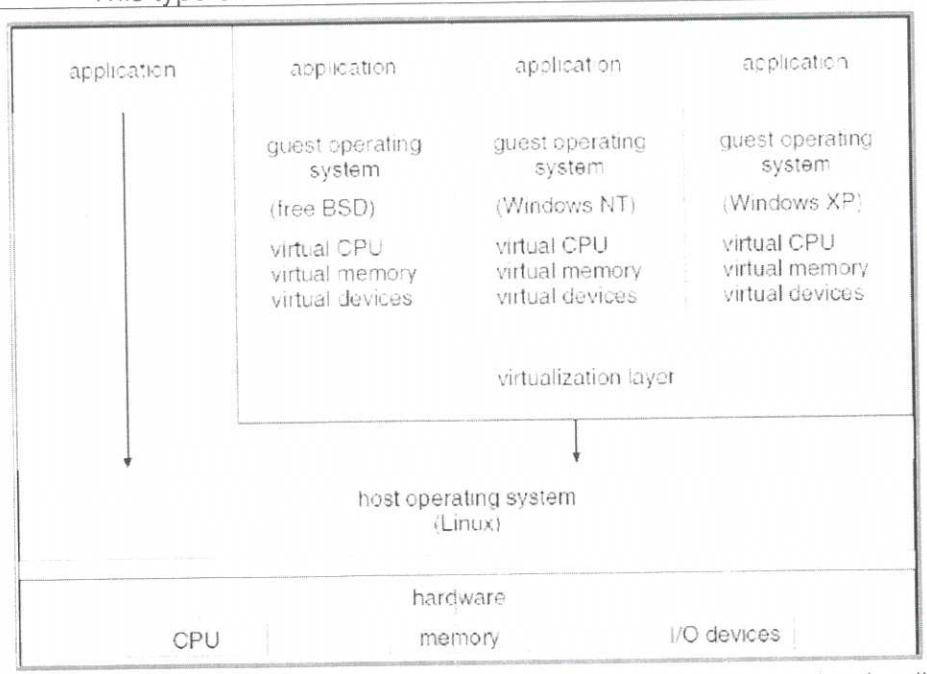
- In para virtualization the guest operating system is aware that it is a guest and it has drivers for it
- Instead of issuing hardware commands, simply issues commands directly to the host operating system.

Partial Virtualization

- The virtual machine simulates multiple instances of an underlying hardware environment.
- The entire operating systems cannot run in the virtual machine but many applications can run.
- This type of virtualization is far easier to execute than full virtualization

3*3 | 9

(b)



- VMware Server is feature-packed with the following market-leading capabilities:
- Support for any standard x86 hardware
- Support for a wide variety of Linux and Windows host operating systems, including64-bit operating systems
- Support for a wide variety of Linux, NetWare, Solaris x86, and Windows guest operating systems, including 64-bit operating systems
- Support for Virtual SMP(VMware Virtual SMP is a utility that allows a single virtual machine to use two or more processors simultaneously), enabling a single virtual machine to span multiple physical processors

3+3 | 6

9/10

	• Quick and easy virtual machine creation with a virtual machine wizard • Virtual machine monitoring and management with a user-friendly remote console		

10/10