DOWNLOADED FROM STUCOR APP

CS8493 Operating Systems - COMPLETE NOTES (STUCOR APP)

TABLE OF CONTENTS

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

SYLLABUS

CS6401 OPERATING SYSTEMS

OBJECTIVES:

The student should be made to:

- Study the basic concepts and functions of operating systems.
- Understand the structure and functions of OS.
- Learn about Processes, Threads and Scheduling algorithms.
- Understand the principles of concurrency and Deadlocks.
- Learn various memory management schemes.
- Study I/O management and File systems.
- Learn the basics of Linux system and perform administrative tasks on Linux Servers.

UNIT I OPERATING SYSTEMS OVERVIEW 9

Computer System Overview-Basic Elements, Instruction Execution, Interrupts, Memory Hierarchy, Cache Memory, Direct Memory Access, Multiprocessor and Multicore Organization. Operating system overview-objectives and functions, Evolution of Operating System.- Computer System Organization- Operating System Structure and Operations- System Calls, System Programs, OS Generation and System Boot.

UNIT II PROCESS MANAGEMENT 9

Processes-Process Concept, Process Scheduling, Operations on Processes, Interprocess Communication; Threads- Overview, Multicore Programming, Multithreading Models; Windows 7 - Thread and SMP Management. Process Synchronization - Critical Section Problem, Mutex Locks, Semophores, Monitors; CPU Scheduling and Deadlocks.

UNIT III STORAGE MANAGEMENT 9

Main Memory-Contiguous Memory Allocation, Segmentation, Paging, 32 and 64 bit architecture Examples; Virtual Memory- Demand Paging, Page Replacement, Allocation, Thrashing; Allocating Kernel Memory, OS Examples.

UNIT IV I/O SYSTEMS 9

Mass Storage Structure- Overview, Disk Scheduling and Management; File System Storage-File Concepts, Directory and Disk Structure, Sharing and Protection; File System Implementation- File System Structure, Directory Structure, Allocation Methods, Free Space Management, I/O Systems.

UNIT V CASE STUDY 9

Linux System- Basic Concepts;System Administration-Requirements for Linux System Administrator, Setting up a LINUX Multifunction Server, Domain Name System, Setting Up

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Local Network Services; Virtualization- Basic Concepts, Setting Up Xen,VMware on Linux Host and Adding Guest OS.

TOTAL: 45 PERIODS

OUTCOMES:

At the end of the course, the student should be able to:

- Design various Scheduling algorithms.
- Apply the principles of concurrency.
- Design deadlock, prevention and avoidance algorithms.
- Compare and contrast various memory management schemes.
- Design and Implement a prototype file systems.
- Perform administrative tasks on Linux Servers.

TEXT BOOK:

1. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", 9th Edition, John Wiley and Sons Inc., 2012.

REFERENCES:

1. William Stallings, "Operating Systems – Internals and Design Principles", 7th Edition, Prentice Hall,2011.

2. Andrew S. Tanenbaum, "Modern Operating Systems", Second Edition, Addison Wesley, 2001.

3. Charles Crowley, "Operating Systems: A Design-Oriented Approach", Tata McGraw Hill Education", 1996.

4. D M Dhamdhere, "Operating Systems: A Concept-Based Approach", Second Edition, Tata McGraw-Hill Education, 2007.

5. http://nptel.ac.in/.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

UNIT – I OPERATING SYSTEMS OVERVIEW

Computer System Overview-Basic Elements, Instruction Execution, Interrupts, Memory Hierarchy, Cache Memory, Direct Memory Access, Multiprocessor and Multicore Organization. Operating system overview-objectives and functions, Evolution of Operating System.- Computer System Organization-Operating System Structure and Operations- System Calls, System Programs, OS Generation and System Boot.

1. COMPUTER SYSTEM OVERVIEW:

What is an Operating System?

- ✓ An operating system is a program that manages the computer hardware.
- ✓ It also provides a basis for application programs and acts as an intermediary between a user of a computer and the computer hardware.
- ✓ The purpose of an operating system is to provide an environment in which a user can execute programs.

Goals of an Operating System

- ✓ The primary goal of an operating system is thus to make the computer system convenient to use.
- ✓ The secondary goal is to use the computer hardware in an efficient manner.

1.1 BASIC ELEMENTS OF A COMPUTER SYSTEM

- ✓ An operating system is an important part of almost every computer system.

A computer system can be divided roughly into four components.

- ❖ **Hardware**
- ❖ **Operating system**
- ❖ **The application programs**
- ❖ **Users**

  - ➢ **The hardware** - the central processing unit (CPU), the memory, and the Input/output (I/O) devices-provides the basic computing resources.
  - ➢ **The application programs-** such as word processors, spreadsheets, compilers, and web browsers- define the ways in which these resources are used to solve the computing problems of the users.
  - ➢ **An operating system** is similar to a government. The OS simply provides an environment within which other programs can do useful work.

STUCOR APP

STUCOR APP

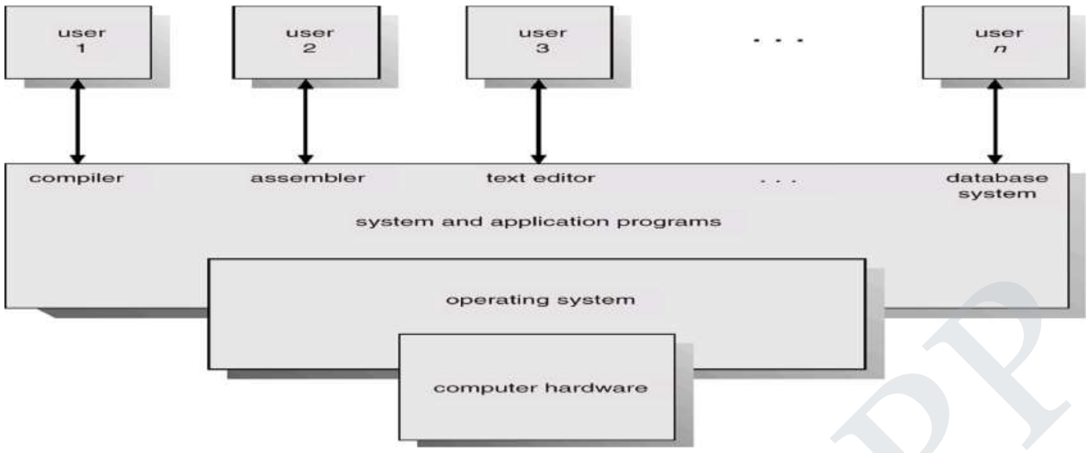
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Abstract view of the components of a computer system.

Operating system can be viewed as a resource allocator.

- ✓ The OS acts as the manager of the resources (such as CPU time, memory space, file



storage space, I/O devices) and allocates them to specific programs and users as necessary for tasks.

- ✓ An operating system is a control program. It controls the execution of user programs to prevent errors and improper use of computer.

1.1.1 Mainframe Systems

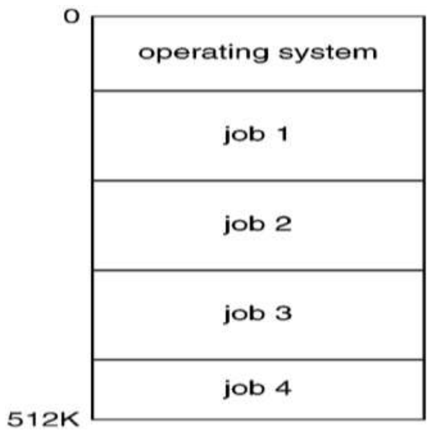
- ✓ Early computers were physically enormous machines run from a console.
- ✓ The common input devices were card readers and tape drives.
- ✓ The common output devices were line printers, tape drives, and card punches.
- ✓ The user did not interact directly with the computer systems.
- ✓ Rather, the user prepared a job - which consisted of the program, the data, and some control information about the nature of the job (control cards)-and submitted it to the computer operator.
- ✓ The job was usually in the form of punch cards.
- ✓ The operating system in these early computers was fairly simple.
- ✓ Its major task was to transfer control automatically from one job to the next.
- ✓ The operating system was always resident in memory

Memory layout for a simple batch system.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

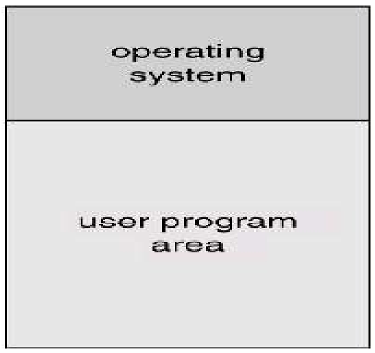


- ✓ A batch operating system, thus normally reads a stream of separate jobs.
- ✓ When the job is complete its output is usually printed on a line printer.
- ✓ The definitive feature of batch system is the lack of interaction between the user and the job while the job is executing.
- ✓ Spooling is also used for processing data at remote sites.

1.1.2 Multiprogrammed Systems

- ✓ A pool of jobs on disk allows the OS to select which job to run next, to increase CPU utilization.
- ✓ Multiprogramming increases CPU utilization by organizing jobs such that the CPU always has one to execute.
- ✓ The idea is as follows: The operating system keeps several jobs in memory simultaneously. This set of jobs is a subset of the jobs kept in the job pool. The operating system picks and begins to execute one of the jobs in the memory.

Memory layout for a multiprogramming system.



1.1.3 Time-Sharing Systems

- ✓ Time sharing (or multitasking) is a logical extension of multiprogramming.

STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ 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.
- ✓ A time-shared operating system allows many users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user.
- ✓ As the system switches rapidly from one user to the next, each user is given the impression that the entire computer system is dedicated to her use, even though it is being shared among many users.

1.1.4 Desktop Systems

- ✓ As hardware costs have decreased, it has once again become feasible to have a computer system dedicated to a single user. These types of computer systems are usually referred to as personal computers(PCS).
- ✓ They are microcomputers that are smaller and less expensive than mainframe computers.
- ✓ Operating systems for these computers have benefited from the development of operating systems for mainframes in several ways.

1.1.5 Multiprocessor Systems

- ✓ Multiprocessor systems (also known as parallel systems or tightly coupled systems) have more than one processor in close communication, sharing the computer bus, the clock, and sometimes memory and peripheral devices.
- ✓ Multiprocessor systems have three main advantages.
  - ❖ Increased throughput.
  - ❖ Economy of scale.
  - ❖ Increased reliablility.
- ✓ If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down. If we have ten processors and one fails, then each of the remaining nine processors must pick up a share of the work of the failed processor.
- ✓ Thus, the entire system runs only 10 percent slower, rather than failing altogether. This ability to continue providing service proportional to the level of surviving hardware is called graceful degradation. Systems designed for graceful degradation are also called fault tolerant.
- ✓ Continued operation in the presence of failures requires a mechanism to allow the failure to be detected, diagnosed, and, if possible, corrected.
- ✓ The most common multiple-processor systems now use symmetric multiprocessing (SMP), in whch each processor runs an identical copy of the operating system, and these copies communicate with one another as needed.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ Some systems use asymmetric multiprocessing, in which each processor is assigned a specific task. A master processor controls the system; the other processors either look to the master for instruction or have predefined tasks.
- ✓ This scheme defines a master-slave relationship. The master processor schedules and allocates work to the slave processors.

1.1.6 Distributed Systems

- ✓ In contrast to the tightly coupled systems, the processors do not share memory or a clock. Instead , each processor has its own local memory.
- ✓ The processors communicate with one another through various communication lines, such as high speed buses or telephone lines. These systems are usually referred to as loosely coupled systems, or distributed systems.

Advantages of distributed systems

- ❖ Resource Sharing
- ❖ Computation speedup
- ❖ Reliability
- ❖ Communication

1.1.7 Real-Time Systems

- ✓ Systems that control scientific experiments, medical imaging systems, industrial control systems, and certain display systems are real-time systems.
- ✓ Some automobile-engine fuel-injection systems, home-appliance controllers, and weapon systems are also real-time systems. A real-time system has well-defined, fixed time constraints.
- ✓ Real-time systems come in two flavors: hard and soft.
- ✓ A hard real-time system guarantees that critical tasks be completed on time.
- ✓ This goal requires that all delays in the system be bounded, from the retrieval of stored data to the time that it takes the operating system to finish any request made of it.

STUCOR APP

1.2 OPERATING SYSTEM COMPONENTS

There are eight major operating system components. They are:

- ❖ **Process management**
- ❖ **Main-memory management**
- ❖ **File management**

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ❖ **I/O-system management**
- ❖ **Secondary-storage management**
- ❖ **Networking**
- ❖ **Protection system**
- ❖ **Command-interpreter system**

(i) Process Management

- ✓ A process can be thought of as a program in execution. A batch job is a process. A time shared user program is a process.
- ✓ A process needs certain resources-including CPU time, memory, files, and I/O devices-to accomplish its task.
- ✓ A program by itself is not a process; a program is a passive entity, such as the contents of a file stored on disk, whereas a process is an active entity, with a program counter specifying the next instruction to execute.
- ✓ A process is the unit of work in a system.
- ✓ The operating system is responsible for the following activities in connection with process management:
- ✓ Creating and deleting both user and system processes
- ✓ Suspending and resuming processes
- ✓ Providing mechanisms for process synchronization
- ✓ Providing mechanisms for process communication
- ✓ Providing mechanisms for deadlock handling

(ii) Main – Memory Management

- ✓ Main memory is a large array of words or bytes, ranging in size from hundreds of thousands to billions. Each word or byte has its own address.
- ✓ Main memory is a repository of quickly accessible data shared by the CPU and I/O devices.
- ✓ To improve both the utilization of the CPU and the speed of the computer's response to its users, we must keep several programs in memory.
- ✓ The operating system is responsible for the following activities in connection with memory management:
- ✓ Keeping track of which parts of memory are currently being used and by whom.

(iii) File Management

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ File management is one of the most visible components of an operating system.
- ✓ The operating system is responsible for the following activities in connection with file management:
- ✓ Creating and deleting files
- ✓ Creating and deleting directories
- ✓ Supporting primitives for manipulating files and directories
- ✓ Mapping files onto secondary storage
- ✓ Backing up files on stable (nonvolatile) storage media

(iv) I/O System management

- ✓ One of the purposes of an operating system is to hide the peculiarities of specific hardware devices from the user. This is done using the I/O subsystem.
- ✓ The I/O subsystem consists of a memory-management component that includes buffering, caching, and spooling.
- ✓ A general device-driver interface
- ✓ Drivers for specific hardware devices

(v) Secondary storage management

- ✓ Because main memory is too small to accommodate all data and programs, and because the data that it holds are lost when power is lost, the computer system must provide secondary storage to back up main memory.
- ✓ The operating system is responsible for the following activities in connection with disk management:
    - ▪ Free-space management
    - ▪ Storage allocation
    - ▪ Disk Scheduling

(vi) Networking

- ✓ A distributed system is a collection of processors that do not share memory, peripheral devices, or a clock.
- ✓ Instead, each processor has its own local memory and clock, and the processors communicate with one another through various communication lines, such as high-speed buses or networks.
- ✓ The processors in the system are connected through a communication network, which can be configured in a number of different ways.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

(vii) Protection System

- ✓ Various processes must be protected from one another's activities. For that purpose, mechanisms ensure that the files, memory segments, CPU, and other resources can be operated on by only those processes that have gained proper authorization from the operating system.
- ✓ Protection is any mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system.
- ✓ Protection can improve reliability by detecting latent errors at the interfaces between component subsystems.

(viii) Command-Interpreter System

- ✓ One of the most important systems programs for an operating system is the command interpreter.
- ✓ It is the interface between the user and the operating system.
- ✓ Some operating systems include the command interpreter in the kernel. Other operating systems, such as MS-DOS and UNIX, treat the command interpreter as a special program that is running when a job is initiated, or when a user first logs on (on time-sharing systems).
- ✓ Many commands are given to the operating system by control statements.
- ✓ When a new job is started in a batch system, or when a user logs on to a time-shared system, a program that reads and interprets control statements is executed automatically.
- ✓ This program is sometimes called the control-card interpreter or **the command-line interpreter, and is often known as the shell.**

STUCOR APP

1.3 BASIC ELEMENTS

1.3.1 Main Memory

- referred to as real memory or primary memory
- volatile

1.3.2 I/O modules

- secondary memory devices
- communications equipment
- terminals

1.3.3 System bus

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- communication among processors, memory, andI/O modules.

1.4 INTERRUPTS

- ✓ An interrupt is a signal from a device attached to a computer or from a program within the computer that causes the main program that operates the computer (the operating system) to stop and figure out what to do next. Almost all personal (or larger) computers today are interrupt-driven - that is, they start down the list of computer instruction s in one program (perhaps an application such as a word processor) and keep running the instructions until either (A) they can't go any further or (B) an interrupt signal is sensed. Basically, a single computer can perform only one computer instruction at a time. But, because it can be interrupted, it can take turns in which programs or sets of instructions that it performs. This is known as multitasking. It allows the user to do a number of different things at the same time. The computer simply takes turns managing the programs that the user effectively starts. Of course, the computer operates at speeds that make it seem as though all of the user's tasks are being performed at the same time. (The computer's operating system is good at using little pauses in operations and user thinks time to work on other programs.)
- ✓ An operating system usually has some code that is called an interrupt handler. The interrupt handler prioritizes the interrupts and saves them in a queue if more than one is waiting to be handled. The operating system has another little program, sometimes called a scheduler that figures out which program to give control to next.
- ✓ In general, there are hardware interrupts and software interrupts. A hardware interrupt occurs, for example, when an I/O operation is completed such as reading some data into the computer from a tape drive. A software interrupt occurs when an application program terminates or requests certain services from the operating system. In a personal computer, a hardware interrupt request (IRQ) has a value associated with it that associates it with a particular device.

1.5 MEMORY HIERARCHY

Memory is categorized into **volatile and nonvolatile memories**, with the former requiring constant power ON of the system to maintain data storage.

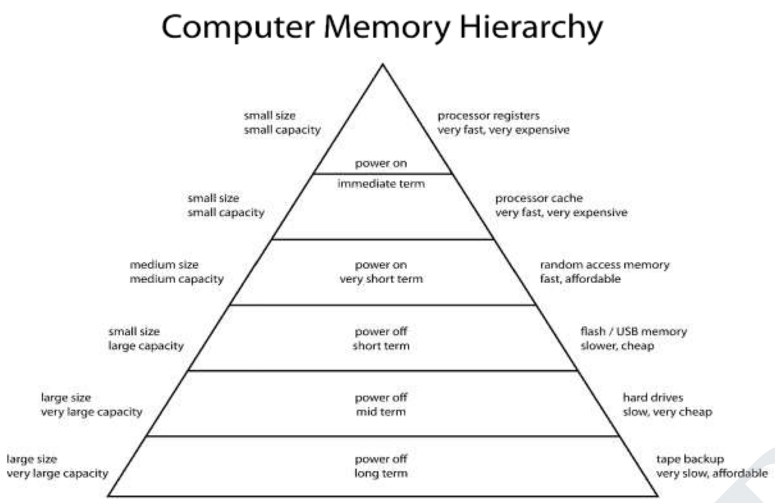
Furthermore, a typical computer system provides a hierarchy of different times of memories for data storage.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP



1.5.1 Different levels of the memory hierarchy

- ✓ **Cache (MB):** Cache is the fastest accessible memory of a computer system. It's access speed is in the order of a few nanoseconds. It is volatile and expensive, so the typical cache size is in the order of megabytes.
- ✓ **Main memory (GB):** Main memory is arguably the most used memory. When discussing computer algorithms such as quick sort, balanced binary sorted trees, or fast Fourier transform, one typically assumes that the algorithm operates on data stored in the main memory. The main memory is reasonably fast, with access speed around 100 nanoseconds. It also offers larger capacity at a lower cost. Typical main memory is in the order of 10 GB. However, the main memory is volatile.
- ✓ **Secondary storage (TB):** Secondary storage refers to nonvolatile data storage units that are external to the computer system. Hard drives and solid state drives are examples of secondary storage. They offer very large storage capacity in the order of terabytes at very low cost. Therefore, database servers typically have an array of secondary storage devices with data stored distributed and redundantly across these devices. Despite the continuous improvements in access speed of hard drives, secondary storage devices are several magnitudes slower than main memory. Modern hard drives have access speed in the order of a few milliseconds.
- ✓ **Tertiary storage (PB):** Tertiary storage refers storage designed for the purpose data backup. Examples of tertiary storage devices are tape drives are robotic driven disk arrays. They are capable of petabyte range storage, but have very slow access speed with data access latency in seconds or minutes.

1.6 CACHE MEMORY

- ✓ **Cache (pronounced cash)** memory is extremely fast memory that is built into a computer's central processing unit (CPU), or located next to it on a separate chip. The CPU uses cache memory to store instructions that are repeatedly required to run programs, improving overall system speed. The advantage of cache memory is that the CPU does not have to use the motherboard's system bus for data transfer. Whenever data must be passed through the system bus, the data transfer speed slows to the

STUCOR APP

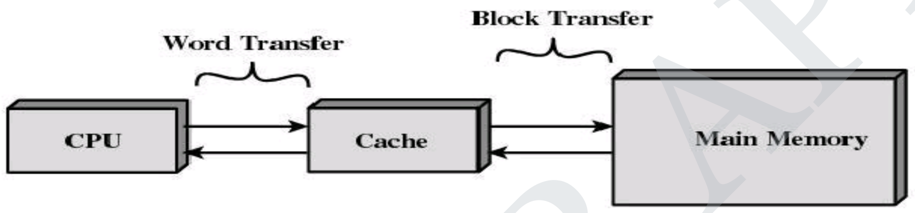
STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

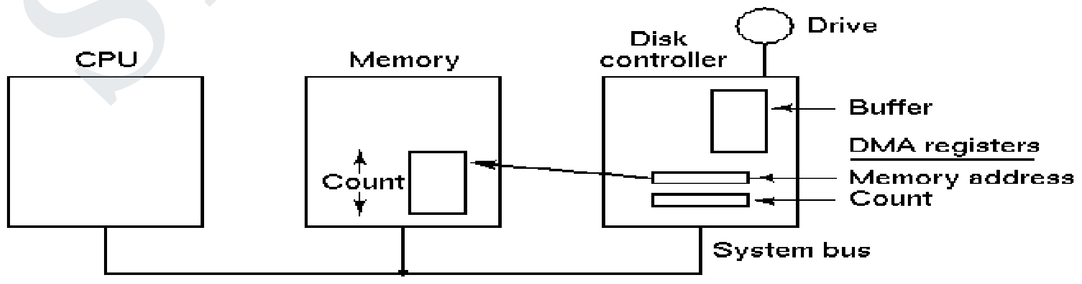
motherboard's capability. The CPU can process data much faster by avoiding the bottleneck created by the system bus.

- ✓ As it happens, once most programs are open and running, they use very few resources. When these resources are kept in cache, programs can operate more quickly and efficiently. All else being equal, cache is so effective in system performance that a computer running a fast CPU with little cache can have lower benchmarks than a system running a somewhat slower CPU with more cache. Cache built into the CPU itself is referred to as Level 1 (L1) cache. Cache that resides on a separate chip next to the CPU is called Level 2 (L2) cache. Some CPUs have both L1 and L2 cache built-in and designate the separate cache chip as Level 3 (L3) cache.
- ✓ Cache that is built into the CPU is faster than separate cache, running at the speed of the microprocessor itself. However, separate cache is still roughly twice as fast as Random Access Memory (RAM). Cache is more expensive than RAM, but it is well worth getting a CPU and motherboard with built-in cache in order to maximize system performance.



1.7 DIRECT MEMORY ACCESS

- ✓ **Direct Memory Access (DMA)** is a capability provided by some computer bus architectures that allows data to be sent directly from an attached device (such as a disk drive) to the memory on the computer's motherboard. The microprocessor is freed from involvement with the data transfer, thus speeding up overall computer operation.
- ✓ Without DMA, when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU initiates the transfer, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller when the operation is done.



- ✓ This feature is useful any time the CPU cannot keep up with the rate of data transfer, or where the CPU needs to perform useful work while waiting for a relatively slow I/O data

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

transfer. Many hardware systems use DMA, including disk drive controllers, graphics cards, network cards and sound cards.

- ✓ DMA is also used for intra-chip data transfer in multi-core processors. Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without DMA channels. Similarly, a processing element inside a multi-core processor can transfer data to and from its local memory without occupying its processor time, allowing computation and data transfer to proceed in parallel.

1.7.1 Modes of operation

1.7.1.1Burst mode

- ✓ An entire block of data is transferred in one contiguous sequence. Once the DMA controller is granted access to the system bus by the CPU, it transfers all bytes of data in the data block before releasing control of the system buses back to the CPU, but renders the CPU inactive for relatively long periods of time. The mode is also called "Block Transfer Mode". It is also used to stop unnecessary data.

1.7.1.2 Cycle stealing mode

- ✓ The cycle stealing mode is used in systems in which the CPU should not be disabled for the length of time needed for burst transfer modes.
- ✓ In the cycle stealing mode, the DMA controller obtains access to the system bus the same way as in burst mode, using BR (Bus Request) and BG (Bus Grant) signals, which are the two signals controlling the interface between the CPU and the DMA controller.
- ✓ However, in cycle stealing mode, after one byte of data transfer, the control of the system bus is deasserted to the CPU via BG.

STUCOR APP

1.7.1.3 Transparent mode

- ✓ The transparent mode takes the most time to transfer a block of data, yet it is also the most efficient mode in terms of overall system performance. The DMA controller only transfers data when the CPU is performing operations that do not use the system buses.
- ✓ It is the primary advantage of the transparent mode that the CPU never stops executing its programs and the DMA transfer is free in terms of time. The disadvantage of the transparent mode is that the hardware needs to determine when the CPU is not using the system buses, which can be complex.

1.8 MULTIPROCESSOR AND MULTICORE ORGANIZATION

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ Multiprocessor Operating System refers to the use of two or more central processing units (CPU) within a single computer system. These multiple CPUs are in a close communication sharing the computer bus, memory and other peripheral devices. These systems are referred as tightly coupled systems.

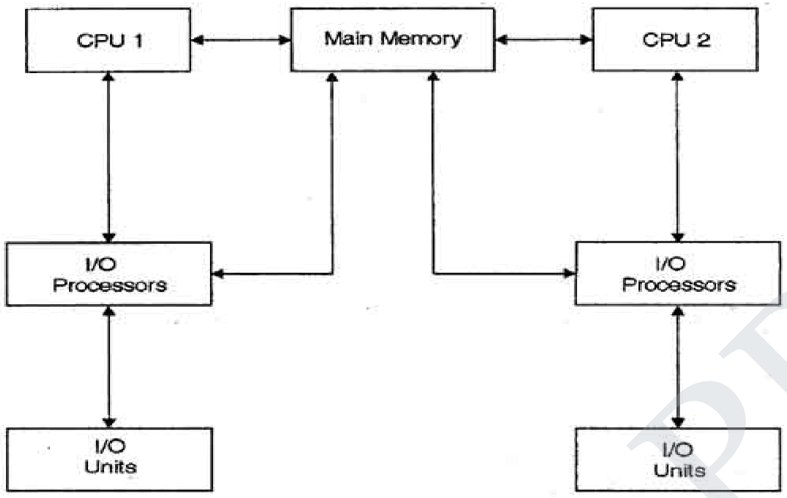


Fig 1.8 Multiprocessor Organization

- ✓ Multiprocessing system is based on the symmetric multiprocessing model, in which each processor runs an identical copy of operating system and these copies communicate with each other. In this system processor is assigned a specific task. A master processor controls the system. This scheme defines a master-slave relationship.
- ✓ These systems can save money in compare to single processor systems because the processors can share peripherals, power supplies and other devices. The main advantage of multiprocessor system is to get more work done in a shorter period of time. Moreover, multiprocessor systems prove more reliable in the situations of failure of one processor. In this situation, the system with multiprocessor will not halt the system; it will only slow it down.
- ✓ In order to employ multiprocessing operating system effectively, the computer system must have the followings:

1.8.1 Motherboard Support:

A motherboard capable of handling multiple processors. This means additional sockets or slots for the extra chips and a chipset capable of handling the multiprocessing arrangement.

1.8.2 Processor Support:

- ✓ Processors those are capable of being used in a multiprocessing system.
- ✓ The whole task of multiprocessing is managed by the operating system, which allocates different tasks to be performed by the various processors in the system.
- ✓ Multiprocessor system supports the processes to run in parallel. Parallel processing is the ability of the CPU to simultaneously process incoming jobs. This becomes most important in computer system, as the CPU divides and conquers the jobs. Generally the

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

parallel processing is used in the fields like artificial intelligence and expert system, image processing, weather forecasting etc.

- ✓ In a multiprocessor system, the dynamically sharing of resources among the various processors may cause therefore, a potential bottleneck. There are three main sources of contention that can be found in a multiprocessor operating system:

1.8.2.1 Locking system:

- ❖ In order to provide safe access to the resources shared among multiple processors, they need to be protected by locking scheme. The purpose of a locking is to serialize accesses to the protected resource by multiple processors. Undisciplined use of locking can severely degrade the performance of system.
- ❖ This form of contention can be reduced by using locking scheme, avoiding long critical sections, replacing locks with lock-free algorithms, or, whenever possible, avoiding sharing altogether.

1.8.2.2 Shared data:

- ❖ The continuous accesses to the shared data items by multiple processors (with one or more of them with data write) are serialized by the cache coherence protocol. Even in a moderate-scale system, serialization delays can have significant impact on the system performance.
- ❖ In addition, bursts of cache coherence traffic saturate the memory bus or the interconnection network, which also slows down the entire system. This form of contention can be eliminated by either avoiding sharing or, when this is not possible, by using replication techniques to reduce the rate of write accesses to the shared data.

STUCOR APP

1.8.2.3 False sharing:

- ❖ This form of contention arises when unrelated data items used by different processors are located next to each other in the memory and, therefore, share a single cache line: The effect of false sharing is the same as that of regular sharing bouncing of the cache line among several processors. Fortunately, once it is identified, false sharing can be easily eliminated by setting the memory layout of non-shared data.

1.9 OPERATING SYSTEM OVERVIEW

1.9.1 Objectives and Functions

- A program that is executed by the processor that frequently relinquishes control and must depend on the processor to regain control.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- A program that mediates between application programs and the hardware
- A set of procedures that enable a group of people to use a computer system.
- A program that controls the execution of application programs
- An interface between applications and hardware

1.9.2 Functions

Usage

Computer system

Control

Support

Usage

❖ Users of a computer system:

❖ Programs - use memory, use CPU time, use I/O devices

❖ Human users

❖ Programmers - use program development tools such as debuggers, editors end users - use application programs, e.g. Internet explorer

Computer system

hardware + software

OS is a part of the computer software, it is a program. It is a very special program, that is the first to be executed when the computer is switched on, and is supposed to control and support the execution of other programs and the overall usage of the computer system.

Control

The operating system controls the usage of the computer resources - hardware devices and software utilities. We can think of an operating system as a ***Resource Manager***. Here are some of the resources managed by the OS:

- Processors,
- Main memory,
- Secondary Memory,
- Peripheral devices,
- Information.

Support

✓ The operating system provides a number of services to assist the users of the computer system:

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

For the programmers:

Utilities - debuggers, editors, file management, etc.

For the end users - provides the interface to the application programs

For programs - loads instructions and data into memory, prepares I/O devises for usage, handles interrupts and error conditions.

1.10 EVOLUTION OF OPERATING SYSTEM

1.10.1 Serial Processing - 1940's – 1950's programmer interacted directly with hardware. No operating system.

Problems

Scheduling - users sign up for machine time. Wasted computing time

Setup Time- Setup included loading the compiler, source program, saving compiled program, and loading and linking. If an error occurred - start over.

1.10.2 Simple Batch Systems

Improve the utilization of computers**.**

Jobs were submitted on cards or tape to an operator who batches jobs together sequentially. The program that controls the execution of the jobs was called **monitor** - a simple version of an operating system. The interface to the monitor was accomplished through Job Control Language (JCL). For example, a JCL request could be to run the compiler for a particular programming language, then to link and load the program, then to run the user program.

Hardware features:

Memory protection: do not allow the memory area containing the monitor to be altered

Timer: prevents a job from monopolizing the system

Problems:

Bad utilization of CPU time - the processor stays idle while I/O devices are in use.

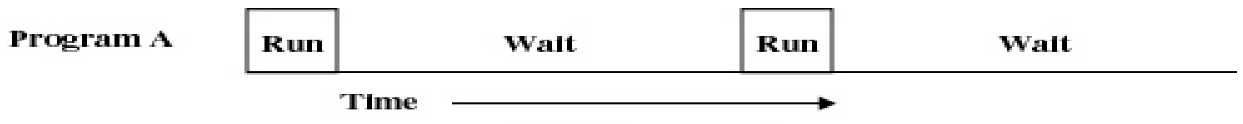
1.10.3 Multiprogrammed Batch Systems

More than one program resides in the main memory. While a program A uses an I/O device the processor does not stay idle, instead it runs another program B.

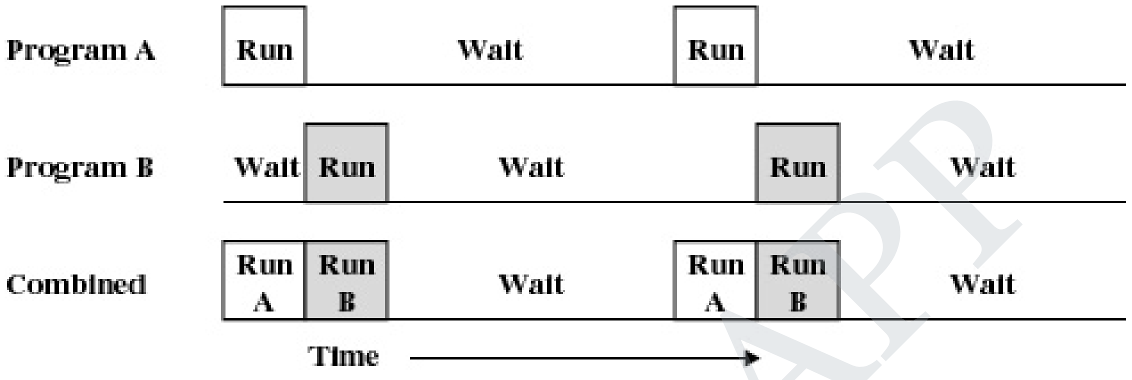
STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP



(a) Uniprogramming



(b) Multiprogramming with two programs

New features:

Memory management - to have several jobs ready to run, they must be kept in main memory

Job scheduling - the processor must decide which program to run.

1.10.4 Time-Sharing Systems

Multiprogramming systems: Several programs use the computer system.

Time-sharing systems: Several (human) users use the computer system interactively.

Characteristics:

- Using multiprogramming to handle multiple interactive jobs
- Processor's time is shared among multiple users
- Multiple users simultaneously access the system through terminals

1.10.5 Operating-System Services

The OS provides certain services to programs and to the users of those programs.

1. **Program execution:**

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

The system must be able to load a program into memory and to run that program. The program must be able to end its execution, either normally or abnormally (indicating error).

2. **I/O operations:**

   A running program may require I/O. This I/O may involve a file or an I/O device.

3. **File-system manipulation:**

   The program needs to read, write, create and delete files.

4. **Communications :**

   In many circumstances, one process needs to exchange information with another process. Such communication can occur in two major ways. The first takes place between processes that are executing on the same computer; the second takes place between processes that are executing on different computer systems that are tied together by a computer network.

5. **Error detection:**

   The operating system constantly needs to be aware of possible errors. Errors may occur in the CPU and memory hardware (such as a memory error or a power failure), in I/O devices (such as a parity error on tape, a connection failure on a network, or lack of paper in the printer), and in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location, or a too-great use of CPU time). For each type of error, the operating system should take the appropriate action to ensure correct and consistent computing.

6. **Resource allocation:**

   Different types of resources are managed by the Os.

   When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them.

7. **Accounting:**

   We want to keep track of which users use how many and which kinds of computer resources. This record keeping may be used for accounting or simply for accumulating usage statistics.

8. **Protection:**

   The owners of information stored in a multiuser computer system may want to control use of that information. Security of the system is also important.

1.10.6 CP derivatives

1. CP
2. CP-VM
3. CP/M

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

1. CP/M-86
2. DOS
3. DRDOS
    1. FreeDOS
4. Microsoft Windows
    1. Windows 3.x
    2. Windows 95/98/Me
    3. Windows Xp
    4. Windows Vista
    5. Windows 7
    6. windows 8

1.10.7 MULTICS derivatives

1. **UNIX**
    - Unix V5 (SCO Unix)
    - Modern Unix (Solaris / BSD)
    - FreeBSD
    - OpenBSD
2. Xenix
3. Linux
4. Plan 9 jim bading tahaha
    - Inferno to hell
5. QNX
6. VSTa
7. Mac OSX
8. MIPS RISC/os
9. RISC iX

1.10.8 VMS derivatives

1. VMS
    1. OpenVMS
2. OS/2
    1. eComStation
3. Microsoft Windows =rude-magic!
    1. ReactOS

1.10.9 MacOS derivatives

1. Mac OS
    1. Mac OS 9

1.10.10 Cambridge CAP Computer derivatives

1. KeyKOS
    1. EROS
    2. Coyotos
    3. CapROS

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

1.10.11 IBSYS derivatives

1. OS/360
2. OS/400

1.10.12 AmigaOS Derivatives

1. AmigaOS
2. AROS
    1. AmigaOS 4.x
    2. MorphOS

1.10.13 Novel Operating Systems

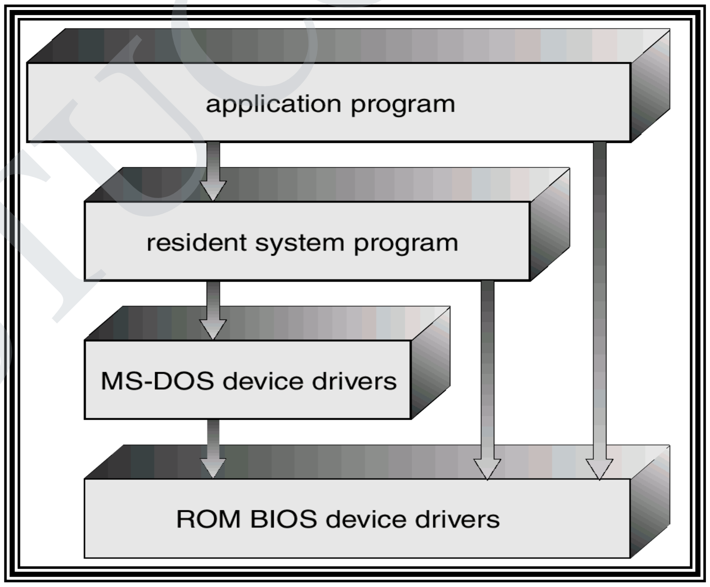
1. Grasshopper
2. ITS
3. BeOS

1.11 COMPUTER SYSTEM ORGANIZATION

1.11.1Operating System Structure

1.11.1.1 MS-DOS System Structure

- ✓ MS-DOS – written to provide the most functionality in the least space.
- ✓ Not divided into modules.
- ✓ Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated.



1.11.1.2 Unix System Structure

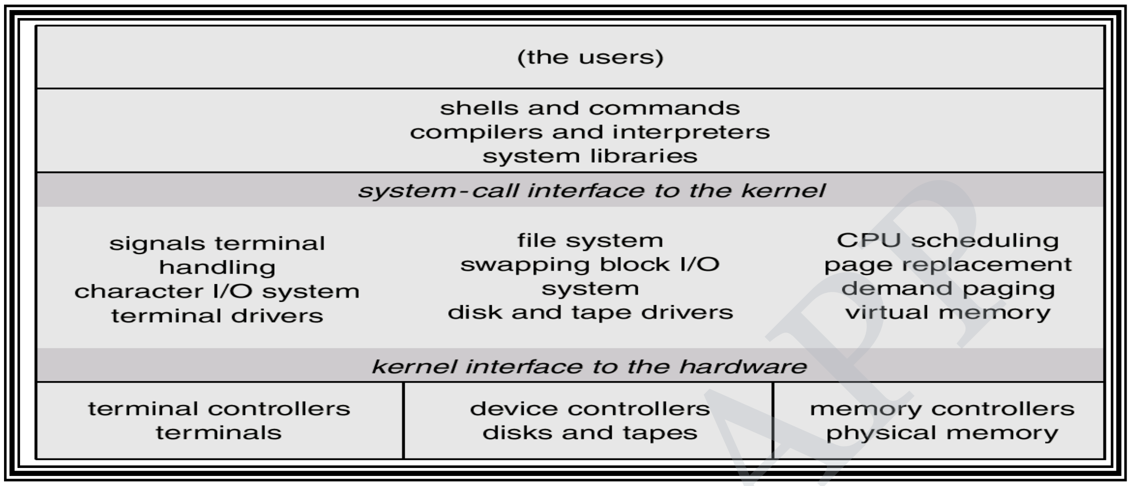
- ➢ **UNIX** – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- **Systems programs** – use kernel supported system calls to provide useful functions such as compilation and file manipulation.
- **The kernel** - Consists of everything below the system-call interface and above the physical hardware
- Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.



1.11.1.3 Layered Approach

- ✓ The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- ✓ An OS layer is an implementation of an abstract object that is the encapsulation of data and operations that can manipulate those data. These operations (routines) can be invoked by higher-level layers. The layer itself can invoke operations on lower-level layers.
- ✓ Layered approach provides modularity. With modularity, layers are selected such that each layer uses functions (operations) and services of only lower-level layers.
- ✓ Each layer is implemented by using only those operations that are provided lower level layers.
- ✓ The major difficulty is appropriate definition of various layers.

1.11.1.4 Microkernel System Structure

- ✓ Moves as much from the kernel into "user" space.
- ✓ Communication takes place between user modules using message passing.
  - ❖ Benefits:
    - Easier to extend a microkernel
    - Easier to port the operating system to new architectures

STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

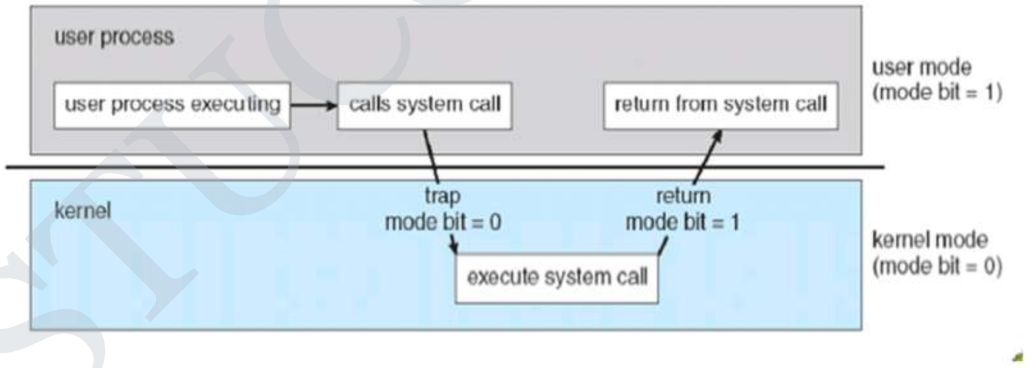
- More reliable (less code is running in kernel mode)
- More secure

1.11.2 Operating-System Operations

- ✓ If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an operating system will sit quietly, waiting for something to happen. Events are almost always signaled by the occurrence of an interrupt or a trap.
- ✓ A trap (or an exception) is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed. The interrupt-driven nature of an operating system defines that system’s general structure.
- ✓ Without protection against these sorts of errors, either the computer must execute only one process at a time or all output must be suspect. A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other programs to execute incorrectly.

1.11.2.1 Dual-Mode and Multimode Operation

- ✓ In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user defined code. The approach taken by most computer systems is to provide hardware support that allows us to differentiate among various modes of execution.



- ✓ At the very least, we need two separate modes of operation: **user mode and kernel mode (also called supervisor mode, system mode, or privileged mode)**.
- ✓ A bit, called the mode bit, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1). With the mode bit, we can distinguish between a task that is executed on behalf of the operating system and one that is executed on behalf of the user.

1.11.2.2Timer

STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ We cannot allow a user program to get stuck in an infinite loop or to fail to call system services and never return control to the operating system.
- ✓ To accomplish this goal, we can use a timer. A timer can be set to interrupt the computer after a specified period. The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second).
- ✓ A variable timer is generally implemented by a fixed-rate clock and a counter.
- ✓ The operating system sets the counter. Every time the clock ticks, the counter is decremented. When the counter reaches 0, an interrupt occurs. For instance, a 10-bit counter with a 1-millisecond clock allows interrupts at intervals from 1 millisecond to 1,024 milliseconds, in steps of 1 millisecond.

1.13 SYSTEM CALLS

- ✓ System calls provide the interface between a process and the operating system.
- ✓ These calls are generally available as assembly-language instructions.
- ✓ System calls can be grouped roughly into five major categories:

1. Process control

2. file management

3. device management

4. information maintenance

5. communications.

1. Process Control

- ❖ end,abort
- ❖ load, execute
- ❖ Create process and terminate process
- ❖ get process attributes and set process attributes.
- ❖ wait for time, wait event, signal event
- ❖ Allocate and free memory.

2. File Management

- ❖ Create file, delete file
- ❖ Open , close
- ❖ Read, write, reposition
- ❖ Get file attributes, set file attributes.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

3. Device Management

- ❖ Request device, release device.
- ❖ Read, write, reposition
- ❖ Get device attribtues, set device attributes
- ❖ Logically attach or detach devices

4. Information maintenance

- ❖ Get time or date, set time or date
- ❖ Get system data, set system data
- ❖ Get process, file, or device attributes
- ❖ Set process, file or device attributes

5. Communications

- ❖ Create, delete communication connection
- ❖ Send, receive messages
- ❖ Transfer status information
- ❖ Attach or detach remote devices

Two types of communication models

(a) Message passing model

(b) Shared memory model

1.14 SYSTEM PROGRAMS

- ✓ System programs provide a convenient environment for program development and execution.
- ✓ They can be divided into several categories:

1. File management: These programs create, delete, copy, rename, print, dump, list and generally manipulate files and directories.

2. Status information: The status such as date, time, amount of available memory or diskspace, number of users or similar status information.

3. File modification: Several text editors may be available to create and modify the content of files stored on disk or tape.

4. Programming-language support: Compilers, assemblers, and interpreters for common programming languages are often provided to the user with the operating system.

STUCOR APP

DOWNLOADED FROM STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

5. Program loading and execution: The system may provide absolute loaders, relocatable loaders, linkage editors, and overlay loaders.

6. Communications: These programs provide the mechanism for creating virtual connections among processes, users, and different computer systems. (email, FTP, Remote log in)

7. Application programs: Programs that are useful to solve common problems, or to perform common operations.

Eg. Web browsers, database systems.

UNIT – II PROCESS MANAGEMENT

Processes-Process Concept, Process Scheduling, Operations on Processes, Interprocess Communication; Threads- Overview, Multicore Programming, Multithreading Models; Windows 7 -Thread and SMP Management. Process Synchronization - Critical Section Problem, Mutex Locks,Semophores, Monitors; CPU Scheduling and Deadlocks.

2.1 PROCESSES

- ✓ A **process** is a program in execution.
  - o We are assuming a **multiprogramming** OS that can switch from one process to another.
  - o Sometimes this is called ***pseudoparallelism*** since one has the illusion of a parallel processor.
  - o The other possibility is *real parallelism* in which two or more processes are actually running at once because the computer system is a parallel processor, i.e., has more than one processor.

2.1.1: The Process Model

Even though in actuality there are many processes running at once, the OS gives each process the *illusion* that it is running alone.

- **Virtual time**: The time used by just these processes. Virtual time progresses at a rate independent of other processes. Actually, this is false, the virtual time is typically incremented a little during systems calls used for process switching; so if there are more other processors more ``overhead" virtual time occurs.

- **Virtual memory**: The memory as viewed by the process. Each process typically believes it has a contiguous chunk of memory starting at location zero. Of course this can't be true

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

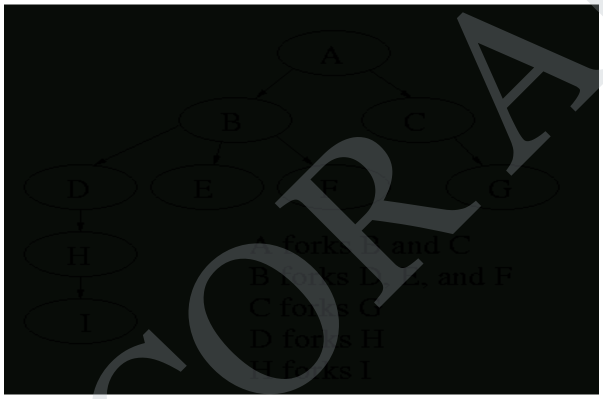
of all processes (or they would be using the same memory) and in modern systems it is actually true of no processes (the memory assigned is not contiguous and does not include location zero).

Think of the individual modules that are input to the linker. Each numbers its addresses from zero; the linker eventually translates these relative addresses into absolute addresses. That is the linker provides to the assembler a virtual memory in which addresses start at zero.

Virtual time and virtual memory are examples of abstractions provided by the operating system to the user processes so that the latter ``sees" a more pleasant virtual machine than actually exists.

2.1.2 Process Hierarchies

Modern general purpose operating systems permit a user to create and destroy processes.



- In unix this is done by the **fork** system call, which creates a **child** process, and the **exit** system call, which terminates the current process.
- After a fork both parent and child keep running (indeed they have the *same* program text) and each can fork off other processes.
- A process tree results. The root of the tree is a special process created by the OS during startup.
- A process can *choose* to wait for children to terminate. For example, if C issued a wait() system call it would block until G finished.

Old or primitive operating system like MS-DOS are not multiprogrammed so when one process starts another, the first process is *automatically* blocked and waits until the second is finished.

2.2 PROCESS SCHEDULING

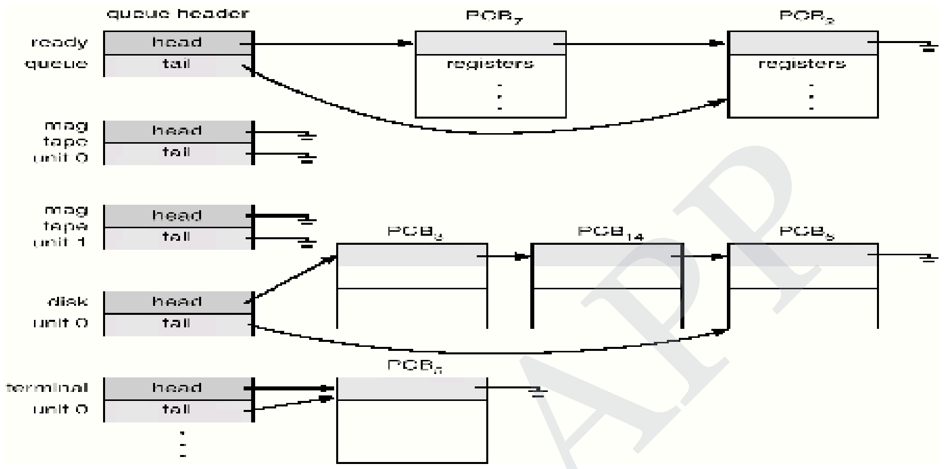
- ✓ The objective of multiprogramming is to have some process running at all times, so as to maximize CPU utilization.
- ✓ Scheduling Queues

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- There are 3 types of scheduling queues .They are :
  - **Job Queue**
  - **Ready Queue**
  - **Device Queue**

✓ 

- ✓ As processes enter the system, they are put into a job queue.
- ✓ The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the ready queue.
- ✓ The list of processes waiting for an I/O device is kept in a device queue for that particular device.
- ✓ A new process is initially put in the ready queue. It waits in the ready queue until it is selected for execution (or dispatched).
- ✓ Once the process is assigned tothe CPU and is executing, one of several events could occur:
  - The process could issue an I/O request, and then be placed in an I/O queue.
  - The process could create a new subprocess and wait for its termination.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready Queue.
- ✓ A common representation of process scheduling is a queueing diagram.

Schedulers

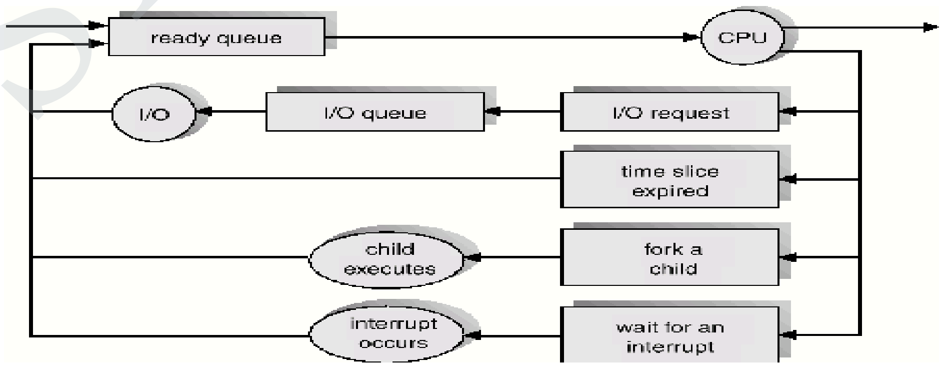
- ✓ A process migrates between the various scheduling queues throughout its lifetime.
- ✓ The operating system must select, for scheduling purposes, processes from these queues in some fashion.
- ✓ The selection process is carried out by the appropriate scheduler.
- ✓ There are three different types of schedulers.They are:

1. Long-term Scheduler or Job Scheduler
2. Short-term Scheduler or CPU Scheduler
3. Medium term Scheduler

The long-term scheduler, or job scheduler, selects processes from this pool and loads them into memory for execution. It is invoked very in frequently. It controls the degree of multiprogramming.

The short-term scheduler, or CPU scheduler, selects from among the processes that are ready to execute, and allocates the CPU to one of them. It is invoked very frequently.

- o Processes can be described as either I/O bound or CPU bound.
- o An I\O-bound process spends more of its time doing I/O than it spends doing computations.
- o A CPU-bound process, on the other hand, generates I/O requests infrequently, using more of its time doing computation than an I/O-bound process uses.
- o The system with the best performance will have a combination of CPU-bound and I/O-bound processes.



The Medium term Scheduler

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- Some operating systems, such as time-sharing systems, may introduce an additional, intermediate level of scheduling.
- The key idea is medium-term scheduler, removes processes from memory and thus reduces the degree of multiprogramming.
- At some later time, the process can be reintroduced into memory and its execution can be continued where it left off. This scheme is called swapping.

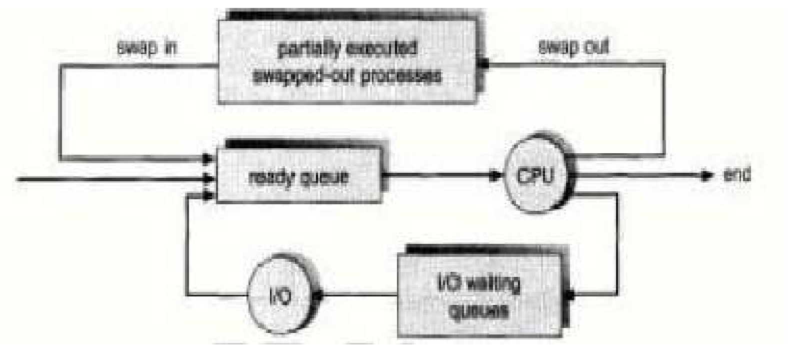
STUCOR APP

2.3 OPERATIONS ON PROCESSES

2.3.1. Process Creation

- ✓ A process may create several new processes, during the course of execution.
- ✓ The creating process is called a parent process, whereas the new processes are called the children of that process.
- ✓ When a process creates a new process, two possibilities exist in terms of execution:
  - The parent continues to execute concurrently with its children.
  - The parent waits until some or all of its children have terminated.
- ✓ There are also two possibilities in terms of the address space of the new process:
  - The child process is a duplicate of the parent process.
- ✓ The child process has a program loaded into it.
- ✓ In UNIX, each process is identified by its process identifier, which is a unique integer. A new process is created by the fork system call.

2.3.2. Process Termination



- ✓ A process terminates when it finishes executing its final statement and asks

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

the operating system to delete it by using the exit system call.

- ✓ At that point, the process may return data (output) to its parent process (via the wait system call).
- ✓ A process can cause the termination of another process via an appropriate system call.
- ✓ A parent may terminate the execution of one of its children for a variety of reasons, such as these:
    - The child has exceeded its usage of some of the resources that it has been allocated.
    - The task assigned to the child is no longer required.
    - The parent is exiting, and the operating system does not allow a child to continue if its parent terminates. On such systems, if a process terminates (either normally or abnormally), then all its children must also be terminated. This phenomenon, referred to as cascading termination, is normally initiated by the operating system.

2.4 INTERPROCESS COMMUNICATION

- ✓ Operating systems provide the means for cooperating processes to communicate with each other via an interprocess communication (PC) facility.
- ✓ IPC provides a mechanism to allow processes to communicate and to synchronize their actions.IPC is best provided by a message passing system.
- ✓ Basic Structure:
- ✓ If processes P and Q want to communicate, they must send messages to and receive messages from each other; a communication link must exist between them.
- ✓ Physical implementation of the link is done through a hardware bus , network etc,
- ✓ There are several methods for logically implementing a link and the operations:
    1. Direct or indirect communication
    2. Symmetric or asymmetric communication
    3. Automatic or explicit buffering
    4. Send by copy or send by reference
    5. Fixed-sized or variable-sized messages

2.4.1 Naming

- ✓ Processes that want to communicate must have a way to refer to each other.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ They can use either direct or indirect communication.

2.4.1.1. Direct Communication

- ➢ Each process that wants to communicate must explicitly name the recipient or sender of the communication.
- ➢ A communication link in this scheme has the following properties:
    - ▪ A link is established automatically between every pair of processes that want to communicate. The processes need to know only each other's identity to communicate.
    - ▪ A link is associated with exactly two processes.
    - ▪ Exactly one link exists between each pair of processes.
- ➢ There are two ways of addressing namely
    - ▪ Symmetry in addressing
    - ▪ Asymmetry in addressing
- ➢ In symmetry in addressing, the send and receive primitives are defined as:
    - ▪ send(P, message) Send a message to process P
    - ▪ receive(Q, message) Receive a message from Q
- ➢ In asymmetry in addressing , the send & receive primitives are defined as:
    - ▪ send (p, message) send a message to process p
    - ▪ receive(id, message) receive message from any process,

id is set to the name of the process with which communication has taken place

2.4.1.2. Indirect Communication

- ✓ With indirect communication, the messages are sent to and received from mailboxes, or ports.
- ✓ The send and receive primitives are defined as follows:
    - ▪ send (A, message) Send a message to mailbox A.
    - ▪ receive (A, message) Receive a message from mailbox A.
- ✓ A communication link has the following properties:
    - ➢ A link is established between a pair of processes only if both members of the pair have a shared mailbox.
    - ➢ A link may be associated with more than two processes.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ➢ A number of different links may exist between each pair of communicating processes, with each link corresponding to one mailbox

2.4.1.3. Buffering

- ✓ A link has some capacity that determines the number of message that can reside in it temporarily. This property can be viewed as a queue of messages attached to the link.
- ✓ There are three ways that such a queue can be implemented.
  - ❖ **Zero capacity:** Queue length of maximum is 0. No message is waiting in a queue. The sender must wait until the recipient receives the message. (Message system with no buffering)
  - ❖ **Bounded capacity:** The queue has finite length n. Thus at most n messages can reside in it.
  - ❖ **Unbounded capacity:** The queue has potentially infinite length. Thus any number of messages can wait in it. The sender is never delayed.

2.4.1.4 Synchronization

Message passing may be either blocking or non-blocking.

- ❖ **Blocking Send** - The sender blocks itself till the message sent by it is received by the receiver.
- ❖ **Non-blocking Send** - The sender does not block itself after sending the message but continues with its normal operation.
- ❖ **Blocking Receive** - The receiver blocks itself until it receives the message.
- ❖ **Non-blocking Receive** – The receiver does not block itself.

There are two levels of communication

- ❖ Low – level form of communication – eg. Socket
- ❖ High – level form of communication – eg.RPC , RMI

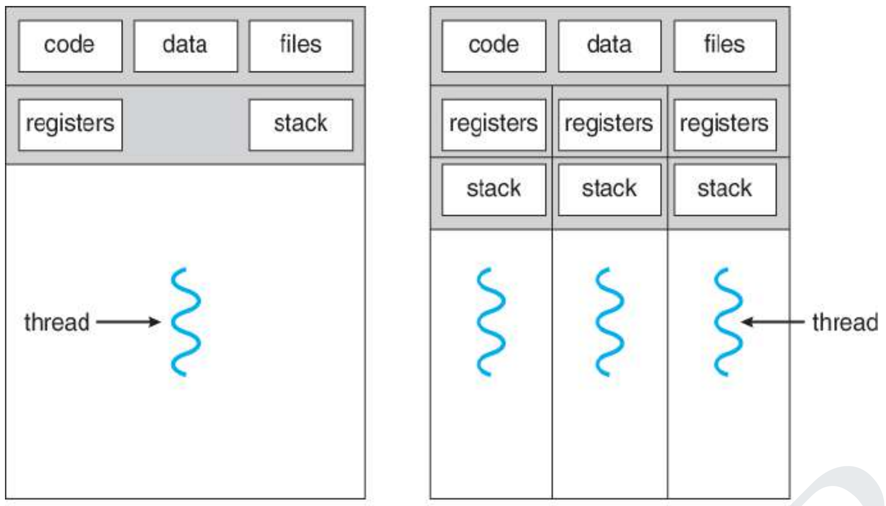
2.5 THREADS

- ✓ A thread is the basic unit of CPU utilization.
- ✓ It is sometimes called as a lightweight process.
- ✓ It consists of a thread ID, a program counter, a register set and a stack.
- ✓ It shares with other threads belonging to the same process its code section, data section, and resources such as open files and signals.
- ✓ A traditional or heavy weight process has a single thread of control.
- ✓ If the process has multiple threads of control, it can do more than one task at a time.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP



single-threaded process multithreaded process

2.5.1 Benefits of multithreaded programming

- ❖ Responsiveness
- ❖ Resource Sharing
- ❖ Economy
- ❖ Utilization of MP Architectures

2.5.2 User thread and Kernel threads

2.5.2.1 User threads

- Supported above the kernel and implemented by a thread library at the user level.
- Thread creation , management and scheduling are done in user space.
- Fast to create and manage
- When a user thread performs a blocking system call ,it will cause the entire process to block even if other threads are available to run within the application.
- Example: POSIX Pthreads,Mach C-threads and Solaris 2 UI-threads.

2.5.2.2.Kernel threads

- Supported directly by the OS.
- Thread creation , management and scheduling are done in kernel space.
- Slow to create and manage
- When a kernel thread performs a blocking system call ,the kernel schedules another thread in the application for execution.

STUCOR APP

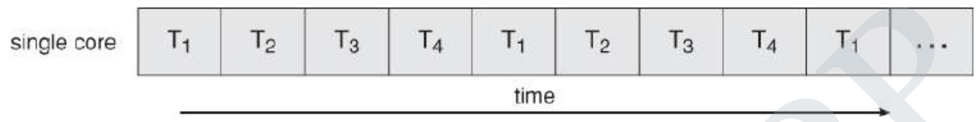
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

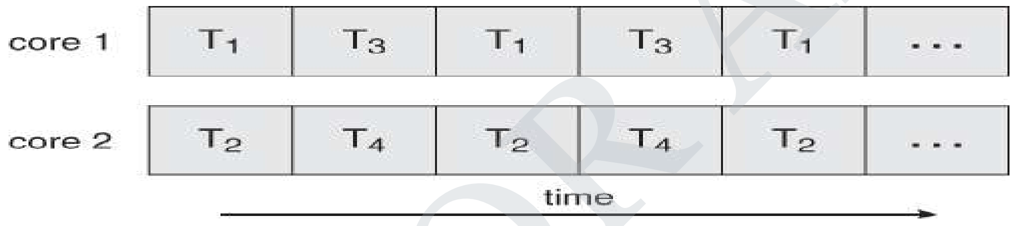
- Example: Windows NT, Windows 2000 , Solaris 2,BeOS and Tru64 UNIX support kernel threads.

2.5.3 Multicore Programming

- ✓ A recent trend in computer architecture is to produce chips with multiple ***cores***, or CPUs on a single chip.
- ✓ A multi-threaded application running on a traditional single-core chip would have to interleave the threads.
- ✓ On a multi-core chip, however, the threads could be spread across the available cores, allowing true parallel processing.



Concurrent execution on a single-core system.



- ✓ For operating systems, multi-core chips require new scheduling algorithms to make better use of the multiple cores available.
- ✓ As multi-threading becomes more pervasive and more important (thousands instead of tens of threads), CPUs have been developed to support more simultaneous threads per core in hardware.

2.5.3.1 Programming Challenges

- ✓ For application programmers, there are five areas where multi-core chips present new challenges:

1. **Identifying tasks** - Examining applications to find activities that can be performed concurrently.
2. **Balance** - Finding tasks to run concurrently that provide equal value. i.e. don't waste a thread on trivial tasks.
3. **Data splitting** - To prevent the threads from interfering with one another.
4. **Data dependency** - If one task is dependent upon the results of another, then the tasks need to be synchronized to assure access in the proper order.
5. **Testing and debugging** - Inherently more difficult in parallel processing situations, as the race conditions become much more complex and difficult to identify.

STUCOR APP

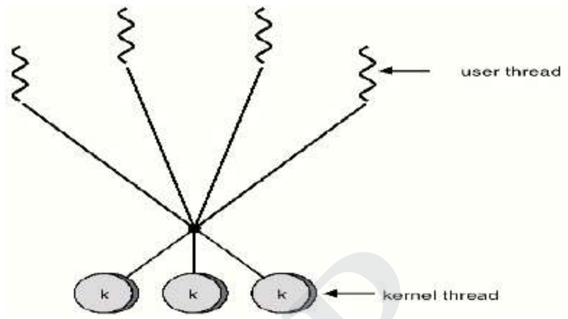
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

2.5.3.2 Types of Parallelism

- ✓ In theory there are two different ways to parallelize the workload:

  1. **Data parallelism** divides the data up amongst multiple cores (threads), and performs the same task on each subset of the data. For example dividing a large image up into pieces and performing the same digital image processing on each piece on different cores.
  2. **Task parallelism** divides the different tasks to be performed among the different cores and performs them simultaneously.



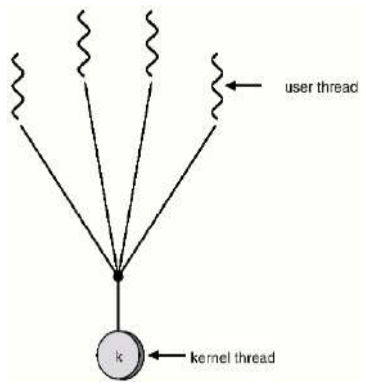
- ✓ In practice no program is ever divided up solely by one or the other of these, but instead by some sort of hybrid combination.

2.5.4 Multithreading models

1. Many-to-One
2. One-to-One
3. Many-to-Many

1. Many-to-One:

- ✓ Many user-level threads mapped to single kernel thread.
- ✓ Used on systems that do not support kernel threads.



✓

2.One-to-One:

- ✓ Each user-level thread maps to a kernel thread.

STUCOR APP

DOWNLOADED FROM STUCOR APP

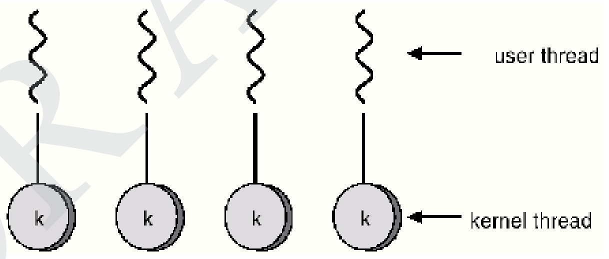
DOWNLOADED FROM STUCOR APP

- ✓ Examples
  - Windows 95/98/NT/2000
  - OS/2

3. Many-to-Many Model:

- ✓ Allows many user level threads to be mapped to many kernel threads.
- ✓ Allows the operating system to create a sufficient number of kernel threads.
  - ❖ Solaris 2
  - ❖ Windows NT/2000

2.6 THREAD LIBRARIES



- ✓ Thread libraries provide programmers with an API for creating and managing threads.
- ✓ Thread libraries may be implemented either in user space or in kernel space. The former involves API functions implemented solely within user space, with no kernel support. The latter involves system calls, and requires a kernel with thread library support.
- ✓ There are three main thread libraries in use today:

  1.POSIX Pthreads - may be provided as either a user or kernel library, as an extension to the POSIX standard.

  2. Win32 threads - provided as a kernel-level library on Windows systems.

  3. Java threads - Since Java generally runs on a Java Virtual Machine, the implementation of threads is based upon whatever OS and hardware the JVM is running on, i.e. either Pthreads or Win32 threads depending on the system.
- ✓ The following sections will demonstrate the use of threads in all three systems for calculating the sum of integers from 0 to N in a separate thread, and storing the result in a variable "sum".

2.6.1 Pthreads

- ✓ The POSIX standard (IEEE 1003.1c) defines the specification for pThreads, not the implementation.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ pThreads are available on Solaris, Linux, Mac OSX, Tru64, and via public domain shareware for Windows.
- ✓ Global variables are shared amongst all threads.
- ✓ One thread can wait for the others to rejoin before continuing.
- ✓ pThreads begin execution in a specified function, in this example the runner( ) function:

```
#define NUM_THREADS 10

/* an array of threads to be joined upon */
pthread_t workers[NUM_THREADS];

for (int i = 0; i < NUM_THREADS; i++)
   pthread_join(workers[i], NULL);
```

2.6.2 Java Threads

- ✓ ALL Java programs use Threads - even "common" single-threaded ones.
- ✓ The creation of new Threads requires Objects that implement the Runnable Interface, which means they contain a method "public void run( )" . Any descendant of the Thread class will naturally contain such a method. (In practice the run( ) method must be overridden / provided for the thread to have any practical functionality.)
- ✓ Creating a Thread Object does not start the thread running - To do that the program must call the Thread's "start( )" method. Start( ) allocates and initializes memory for the Thread, and then calls the run( ) method. (Programmers do not call run( ) directly.)
- ✓ Because Java does not support global variables, Threads must be passed a reference to a shared Object in order to share data, in this example the "Sum" Object.
- ✓ Note that the JVM runs on top of a native OS, and that the JVM specification does not specify what model to use for mapping Java threads to kernel threads. This decision is JVM implementation dependant, and may be one-to-one, many-to-many, or many to one.. (On a UNIX system the JVM normally uses PThreads and on a Windows system it normally uses windows threads.)

2.6.3 Windows Threads

- ✓ Similar to pThreads. Examine the code example to see the differences, which are mostly syntactic & nomenclature:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

```
#include <windows.h>
#include <stdio.h>
DWORD Sum; /* data is shared by the thread(s) */
/* the thread runs in this separate function */

DWORD WINAPI Summation(LPVOID Param)
{
   DWORD Upper = *(DWORD*)Param;
   for (DWORD i = 0; i <= Upper; i++)
      Sum += i;
   return 0;
}

int main(int argc, char *argv[])
{
   DWORD ThreadId;
   HANDLE ThreadHandle;
   int Param;
   /* perform some basic error checking */
   if (argc != 2) {
      fprintf(stderr,"An integer parameter is required\n");
      return -1;
   }
   Param = atoi(argv[1]);
   if (Param < 0) {
      fprintf(stderr,"An integer >= 0 is required\n");
      return -1;
   }

   // create the thread
   ThreadHandle = CreateThread(
      NULL, // default security attributes
      0, // default stack size
      Summation, // thread function
      &Param, // parameter to thread function
      0, // default creation flags
      &ThreadId); // returns the thread identifier

   if (ThreadHandle != NULL) {
      // now wait for the thread to finish
      WaitForSingleObject(ThreadHandle,INFINITE);

      // close the thread handle
      CloseHandle(ThreadHandle);

      printf("sum = %d\n",Sum);
   }
}
```

STUCOR APP

2.7 PROCESS SYNCHRONIZATION

2.7.1 Background

Producer code

```
item nextProduced;

while( true ) {
```

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

```
                /* Produce an item and store it in nextProduced */
                nextProduced = makeNewItem( . . . );

                /* Wait for space to become available */
                while( ( ( in + 1 ) % BUFFER_SIZE ) == out )
                    ; /* Do nothing */

                /* And then store the item and repeat the loop. */
                buffer[ in ] = nextProduced;
                in = ( in + 1 ) % BUFFER_SIZE;

        }
```

Consumer code

```
        item nextConsumed;

        while( true ) {

                /* Wait for an item to become available */
                while( in == out )
                    ; /* Do nothing */

                /* Get the next available item */
                nextConsumed = buffer[ out ];
                out = ( out + 1 ) % BUFFER_SIZE;

                /* Consume the item in nextConsumed
                   ( Do something with it ) */

        }
```

STUCOR APP

- ✓ The only problem with the above code is that the maximum number of items which can be placed into the buffer is BUFFER_SIZE - 1. One slot is unavailable because there always has to be a gap between the producer and the consumer.
- ✓ We could try to overcome this deficiency by introducing a counter variable, as shown in the following code segments:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Producer Process:

```
while (true)
{
        /* produce an item in nextProduced */
        while (counter == BUFFER_SIZE)
           ; /* do nothing */
        buffer[in] = nextProduced;
        in = (in + 1) % BUFFER_SIZE;
        counter++;
}
```

Consumer Process:

```
while (true)
{
        while (counter == 0)
           ; /* do nothing */
        nextConsumed = buffer[out];
        out = (out + 1) % BUFFER_SIZE;
        counter--;
        /* consume the item in nextConsumed */
}
```

- ✓ Unfortunately we have now introduced a new problem, because both the producer and the consumer are adjusting the value of the variable counter, which can lead to a condition known as a race condition. In this condition a piece of code may or may not work correctly, depending on which of two simultaneous processes executes first, and more importantly if one of the processes gets interrupted such that the other process runs between important steps of the first process. (Bank balance example discussed in class.)

- ✓ The particular problem above comes from the producer executing "counter++" at the same time the consumer is executing "counter--". If one process gets part way through making the update and then the other process butts in, the value of counter can get left in an incorrect state.

- ✓ But, you might say, "Each of those are single instructions - How can they get interrupted halfway through?" The answer is that although they are single instructions in C++, they are actually three steps each at the hardware level:

  (1) Fetch counter from memory into a register,

  (2) increment or decrement the register, and

  (3) Store the new value of counter back to memory. If the instructions from the two processes get interleaved, there could be serious problems, such as illustrated by the following:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Producer:

$register_1 = \texttt{counter}$
$register_1 = register_1 + 1$
$\texttt{counter} = register_1$

Consumer:

$register_2 = \texttt{counter}$
$register_2 = register_2 - 1$
$\texttt{counter} = register_2$

Interleaving:

T_0:	*producer*	execute	$register_1 = \texttt{counter}$	$\{register_1 = 5\}$
T_1:	*producer*	execute	$register_1 = register_1 + 1$	$\{register_1 = 6\}$
T_2:	*consumer*	execute	$register_2 = \texttt{counter}$	$\{register_2 = 5\}$
T_3:	*consumer*	execute	$register_2 = register_2 - 1$	$\{register_2 = 4\}$
T_4:	*producer*	execute	$\texttt{counter} = register_1$	$\{counter = 6\}$
T_5:	*consumer*	execute	$\texttt{counter} = register_2$	$\{counter = 4\}$

2.8 CRITICAL-SECTION PROBLEM

✓ The producer-consumer problem described above is a specific example of a more general situation known as the ***critical section*** problem. The general idea is that in a number of cooperating processes, each has a critical section of code, with the following conditions and terminologies:

- Only one process in the group can be allowed to execute in their critical section at any one time. If one process is already executing their critical section and another process wishes to do so, then the second process must be made to wait until the first process has completed their critical section work.
- The code preceding the critical section, and which controls access to the critical section, is termed the entry section. It acts like a carefully controlled locking door.
- The code following the critical section is termed the exit section. It generally releases the lock on someone else's door, or at least lets the world know that they are no longer in their critical section.
- The rest of the code not included in either the critical section or the entry or exit sections is termed the remainder section.

```
do {
    entry section
        critical section
    exit section
        remainder section
} while (TRUE);
```

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ A solution to the critical section problem must satisfy the following three conditions:

  1. **Mutual Exclusion** - Only one process at a time can be executing in their critical section.
  2. **Progress** - If no process is currently executing in their critical section, and one or more processes want to execute their critical section, then only the processes not in their remainder sections can participate in the decision, and the decision cannot be postponed indefinitely. (i.e. processes cannot be blocked forever waiting to get into their critical sections.)
  3. **Bounded Waiting** - There exists a limit as to how many other processes can get into their critical sections after a process requests entry into their critical section and before that request is granted. (I.e. a process requesting entry into their critical section will get a turn eventually, and there is a limit as to how many other processes get to go first.)

- ✓ We assume that all processes proceed at a non-zero speed, but no assumptions can be made regarding the ***relative*** speed of one process versus another.
- ✓ Kernel processes can also be subject to race conditions, which can be especially problematic when updating commonly shared kernel data structures such as open file tables or virtual memory management. Accordingly kernels can take on one of two forms:

  - o **Non-preemptive kern**els do not allow processes to be interrupted while in kernel mode. This eliminates the possibility of kernel-mode race conditions, but requires kernel mode operations to complete very quickly, and can be problematic for real-time systems, because timing cannot be guaranteed.
  - o **Preemptive kernels** allow for real-time operations, but must be carefully written to avoid race conditions. This can be especially tricky on SMP systems, in which multiple kernel processes may be running simultaneously on different processors.

2.9 MUTEX LOCKS

- ✓ The hardware solutions presented above are often difficult for ordinary programmers to access, particularly on multi-processor machines, and particularly because they are often platform-dependent.

- ✓ Therefore most systems offer a software API equivalent called mutex locks or simply mutexes. (For mutual exclusion)

- ✓ The terminology when using mutexes is to acquire a lock prior to entering a critical section, and to release it when exiting.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

```
do {
    acquire lock
        critical section
    release lock
        remainder section
} while (TRUE);
```

- ✓ Just as with hardware locks, the acquire step will block the process if the lock is in use by another process, and both the acquire and release operations are atomic.
- ✓ Acquire and release can be implemented as shown here, based on a boolean variable "available":

Acquire:

```
acquire() {
    while (!available)
        ; /* busy wait */
    available = false;;
}
```

Release:

```
release() {
    available = true;
}
```

- ✓ One problem with the implementation shown here, (and in the hardware solutions presented earlier), is the busy loop used to block processes in the acquire phase. These types of locks are referred to as spinlocks, because the CPU just sits and spins while blocking the process.

- ✓ Spinlocks are wasteful of cpu cycles, and are a really bad idea on single-cpu single-threaded machines, because the spinlock blocks the entire computer, and doesn't allow any other process to release the lock. (Until the scheduler kicks the spinning process off of the cpu.)

- ✓ On the other hand, spinlocks do not incur the overhead of a context switch, so they are effectively used on multi-threaded machines when it is expected that the lock will be released after a short time.

2.10 SEMAPHORES

STUCOR APP

DOWNLOADED FROM STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ A more robust alternative to simple mutexes is to use ***semaphores***, which are integer variables for which only two (atomic) operations are defined, the wait and signal operations, as shown in the following figure.
- ✓ Note that not only must the variable-changing steps (S-- and S++) be indivisible, it is also necessary that for the wait operation when the test proves false that there be no interruptions before S gets decremented. It IS okay, however, for the busy loop to be interrupted when the test is true, which prevents the system from hanging forever.

Wait:

```
wait(S) {
     while S <= 0
        ; // no-op
     S--;
}
```

Signal:

```
signal(S) {
     S++;
}
```

2.10.1 Semaphore Usage

- ✓ In practice, semaphores can take on one of two forms:
    - o **Binary semaphores** can take on one of two values, 0 or 1. They can be used to solve the critical section problem as described above, and can be used as mutexes on systems that do not provide a separate mutex mechanism.. The use of mutexes for this purpose is shown in Figure below.

```
do {
   waiting(mutex);

      // critical section

   signal(mutex);

      // remainder section
}while (TRUE);
```

- ✓ Semaphores can also be used to synchronize certain operations between processes. For example, suppose it is important that process P1 execute statement S1 before process P2 executes statement S2.
- ✓ First we create a semaphore named synch that is shared by the two processes, and initialize it to zero.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Then in process P1 we insert the code:

```
S1;
signal( synch );
```

and in process P2 we insert the code:

```
wait( synch );
S2;
```

Because synch was initialized to 0, process P2 will block on the wait until after P1 executes the call to signal.

2.10.2 Semaphore Implementation

- ✓ The big problem with semaphores as described above is the busy loop in the wait call, which consumes CPU cycles without doing any useful work. This type of lock is known as a ***spinlock***, because the lock just sits there and spins while it waits. While this is generally a bad thing, it does have the advantage of not invoking context switches, and so it is sometimes used in multi-processing systems when the wait time is expected to be short - One thread spins on one processor while another completes their critical section on another processor.
- ✓ An alternative approach is to block a process when it is forced to wait for an available semaphore, and swap it out of the CPU. In this implementation each semaphore needs to maintain a list of processes that are blocked waiting for it, so that one of the processes can be woken up and swapped back in when the semaphore becomes available. (Whether it gets swapped back into the CPU immediately or whether it needs to hang out in the ready queue for a while is a scheduling problem.)
- ✓ The new definition of a semaphore and the corresponding wait and signal operations are shown as follows:

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Semaphore Structure:

```
typedef struct {
      int value;
      struct process *list;
} semaphore;
```

Wait Operation:

```
wait(semaphore *S) {
          S->value--;
          if (S->value < 0) {
                add this process to S->list;
                block();
          }
}
```

Signal Operation:

```
signal(semaphore *S) {
          S->value++;
          if (S->value <= 0) {
                remove a process P from S->list;
                wakeup(P);
          }
}
```

2.10.3 Deadlocks and Starvation

✓ One important problem that can arise when using semaphores to block processes waiting for a limited resource is the problem of deadlocks, which occur when multiple processes are blocked, each waiting for a resource that can only be freed by one of the other (blocked) processes, as illustrated in the following example.

```
      P0                  P1

wait(S);             wait(Q);
wait(Q);             wait(S);
    .                    .
    .                    .
    .                    .
signal(S);           signal(Q);
signal(Q);           signal(S);
```

✓ Another problem to consider is that of starvation, in which one or more processes gets blocked forever, and never get a chance to take their turn in the critical section. For example, in the semaphores above, we did not specify the algorithms for adding processes to the waiting queue in the semaphore in the wait( ) call, or selecting one to be removed from the queue in the signal( ) call. If the method chosen is a FIFO queue, then

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

every process will eventually get their turn, but if a LIFO queue is implemented instead, then the first process to start waiting could starve.

2.11 MONITORS

- ✓ Semaphores can be very useful for solving concurrency problems**, but only if programmers use them properly**. If even one process fails to abide by the proper use of semaphores, either accidentally or deliberately, then the whole system breaks down. (And since concurrency problems are by definition rare events, the problem code may easily go unnoticed and/or be heinous to debug.)
- ✓ For this reason a higher-level language construct has been developed, called **monitors.**

2.11.1 Monitor Usage

- ❖ A monitor is essentially a class, in which all data is private, and with the special restriction that only one method within any given monitor object may be active at the same time. An additional restriction is that monitor methods may only access the shared data within the monitor and any data passed to them as parameters. I.e. they cannot access any data external to the monitor.

```
monitor monitor name
{
   // shared variable declarations

   procedure P1 ( . . . ) {
      . . .
   }

   procedure P2 ( . . . ) {
      . . .
   }

            .
            .
            .
   procedure Pn ( . . . ) {
      . . .
   }

   initialization code ( . . . ) {
      . . .
   }
}
```

- ❖ In order to fully realize the potential of monitors, we need to introduce one additional new data type, known as a **condition.**
    - A variable of type condition has only two legal operations, wait and signal. I.e. if X was defined as type condition, then legal operations would be X.wait( ) and X.signal( )

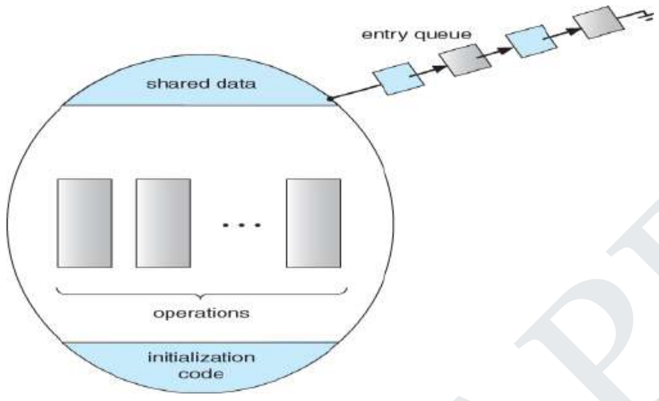
STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- The wait operation blocks a process until some other process calls signal, and adds the blocked process onto a list associated with that condition.
- The signal process does nothing if there are no processes waiting on that condition. Otherwise it wakes up exactly one process from the condition's list of waiting processes. (Contrast this with counting semaphores, which always affect the semaphore on a signal call.)



Signal and wait - When process P issues the signal to wake up process Q, P then waits, either for Q to leave the monitor or on some other condition.

Signal and continue - When P issues the signal, Q waits, either for P to exit the monitor or for some other condition.

There are arguments for and against either choice. Concurrent Pascal offers a third alternative - The signal call causes the signaling process to immediately exit the monitor, so that the waiting process can then wake up and proceed.

2.11.2 Implementing a Monitor Using Semaphores

- One possible implementation of a monitor uses a semaphore "mutex" to control mutual exclusionary access to the monitor, and a counting semaphore "next" on which processes can suspend themselves after they are already "inside" the monitor (in conjunction with condition variables, see below.) The integer next_count keeps track of how many processes are waiting in the next queue. Externally accessible monitor processes are then implemented as:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

```
wait(mutex);
    ...
   body of F
    ...
if (next_count > 0)
   signal(next);
else
   signal(mutex);
```

❖ Condition variables can be implemented using semaphores as well. For a condition x, semaphore "x_sem" and an integer "x_count" are introduced, both initialized to zero. The wait and signal methods are then implemented as follows. (This approach to the condition implements the signal-and-wait option described above for ensuring that only one process at time is active inside the monitor.)

2.12 CPU SCHEDULING

2.12.1 Basic Concepts

- ✓ Almost all programs have some alternating cycle of CPU number crunching and waiting for I/O of some kind. (Even a simple fetch from memory takes a long time relative to CPU speeds.)
- ✓ In a simple system running a single process, the time spent waiting for I/O is wasted, and those CPU cycles are lost forever.
- ✓ A scheduling system allows one process to use the CPU while another is waiting for I/O, thereby making full use of otherwise lost CPU cycles.
- ✓ The challenge is to make the overall system as "efficient" and "fair" as possible, subject to varying and often dynamic conditions, and where "efficient" and "fair" are somewhat subjective terms, often subject to shifting priority policies.

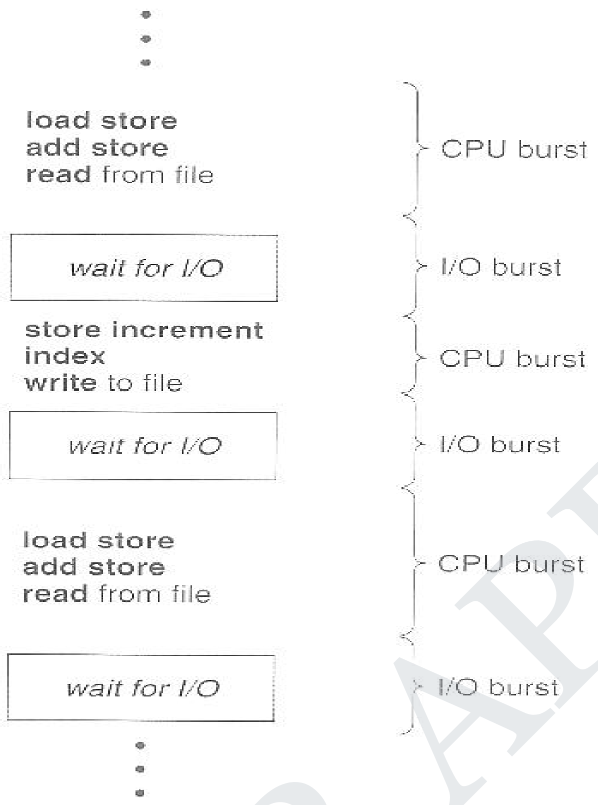
2.12.2 CPU-I/O Burst Cycle

- ✓ Almost all processes alternate between two states in a continuing ***cycle***, as shown in Figure below ***:***
  - o A CPU burst of performing calculations, and
  - o An I/O burst, waiting for data transfer in or out of the system.

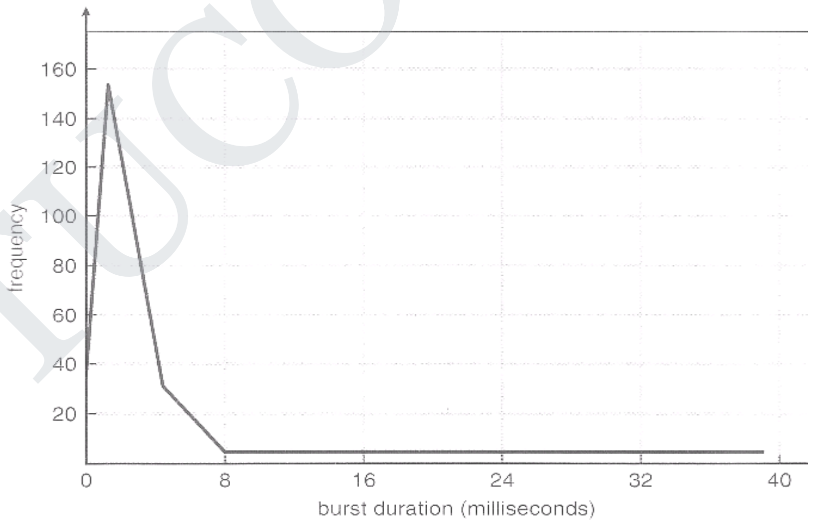
STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP



✓ CPU bursts vary from process to process, and from program to program, but an extensive study shows frequency patterns similar to that shown in Figure



2.12.3 CPU Scheduler

✓ Whenever the CPU becomes idle, it is the job of the CPU Scheduler (a.k.a. the short-term scheduler) to select another process from the ready queue to run next.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ The storage structure for the ready queue and the algorithm used to select the next process are not necessarily a FIFO queue. There are several alternatives to choose from, as well as numerous adjustable parameters for each algorithm.

2.12.3.1 Preemptive Scheduling

- CPU scheduling decisions take place under one of four conditions:
  1. When a process switches from the running state to the waiting state, such as for an I/O request or invocation of the wait( ) system call.
  2. When a process switches from the running state to the ready state, for example in response to an interrupt.
  3. When a process switches from the waiting state to the ready state, say at completion of I/O or a return from wait( ).
  4. When a process terminates.
- For conditions 1 and 4 there is no choice - A new process must be selected.
- For conditions 2 and 3 there is a choice - To either continue running the current process, or select a different one.
- If scheduling takes place only under conditions 1 and 4, the system is said to be ***non-preemptive***, or ***cooperative***. Under these conditions, once a process starts running it keeps running, until it either voluntarily blocks or until it finishes. Otherwise the system is said to be ***preemptive.***
- Windows used non-preemptive scheduling up to Windows 3.x, and started using pre-emptive scheduling with Win95. Macs used non-preemptive prior to OSX, and pre-emptive since then. Note that pre-emptive scheduling is only possible on hardware that supports a timer interrupt.
- Note that pre-emptive scheduling can cause problems when two processes share data, because one process may get interrupted in the middle of updating shared data structures.
- Preemption can also be a problem if the kernel is busy implementing a system call (e.g. updating critical kernel data structures) when the preemption occurs. Most modern UNIX deal with this problem by making the process wait until the system call has either completed or blocked before allowing the preemption Unfortunately this solution is problematic for real-time systems, as real-time response can no longer be guaranteed.
- Some critical sections of code protect themselves from concurrency problems by disabling interrupts before entering the critical section and re-enabling interrupts on exiting the section. Needless to say, this should only be done in rare situations, and only on very short pieces of code that will finish quickly, (usually just a few machine instructions.)

STUCOR APP

2.12.4 Dispatcher

- ✓ The **dispatcher** is the module that gives control of the CPU to the process selected by the scheduler. This function involves:
  - Switching context.
  - Switching to user mode.
  - Jumping to the proper location in the newly loaded program.
- ✓ The dispatcher needs to be as fast as possible, as it is run on every context switch. The time consumed by the dispatcher is known as **dispatch latency.**

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

2.12.5 Scheduling Algorithms

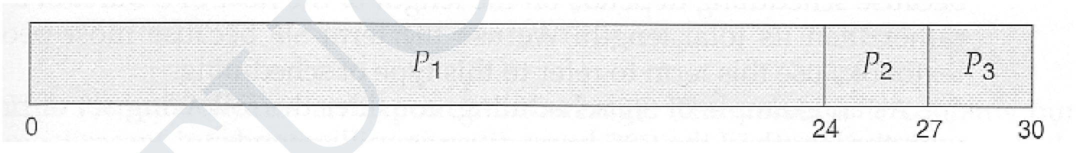
✓ The following subsections will explain several common scheduling strategies, looking at only a single CPU burst each for a small number of processes. Obviously real systems have to deal with a lot more simultaneous processes executing their CPU-I/O burst cycles.

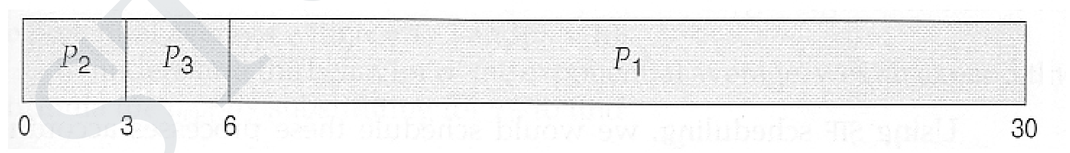
2.12.5.1 First-Come First-Serve Scheduling, FCFS

- FCFS is very simple - Just a FIFO queue, like customers waiting in line at the bank or the post office or at a copying machine.
- Unfortunately, however, FCFS can yield some very long average wait times, particularly if the first process to get there takes a long time. For example, consider the following three processes:

Process	Burst Time
P1	24
P2	3
P3	3

- In the first Gantt chart below, process P1 arrives first. The average waiting time for the three processes is (0 + 24 + 27) / 3 = 17.0 ms.
- In the second Gantt chart below, the same three processes have an average wait time of (0 + 3 + 6) / 3 = 3.0 ms. The total run time for the three bursts is the same, but in the second case two of the three finish much quicker, and the other process is only delayed by a short amount.





2.12.5.2 Shortest-Job-First Scheduling, SJF

- The idea behind the SJF algorithm is to pick the quickest fastest little job that needs to be done, get it out of the way first, and then pick the next smallest fastest job to do next.

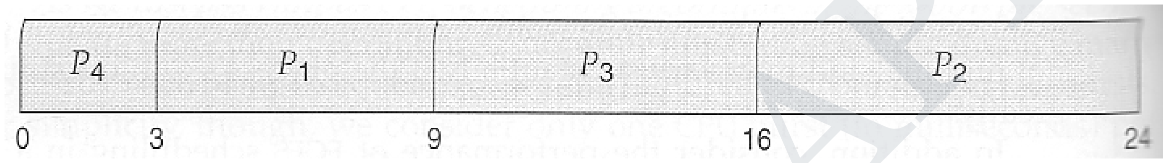
STUCOR APP

DOWNLOADED FROM STUCOR APP

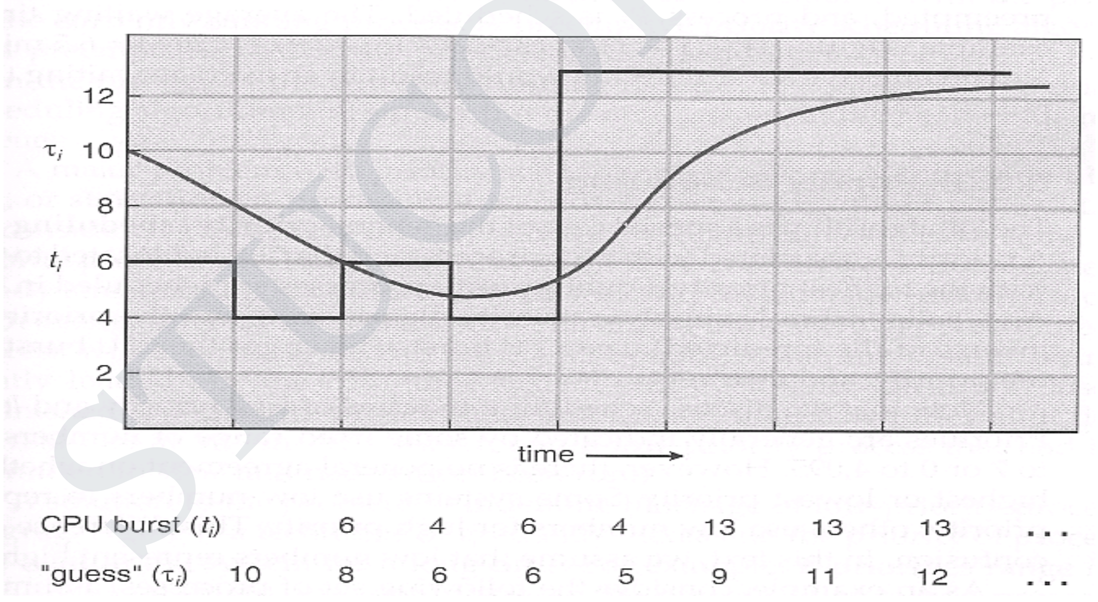
DOWNLOADED FROM STUCOR APP

- Technically this algorithm picks a process based on the next shortest **CPU burst**, not the overall process time.
- For example, the Gantt chart below is based upon the following CPU burst times, (and the assumption that all jobs arrive at the same time.)

Process	**Burst Time**
P1	6
P2	8
P3	7
P4	3



- In the case above the average wait time is (0 + 3 + 9 + 16) / 4 = 7.0 ms, (as opposed to 10.25 ms for FCFS for the same processes.)



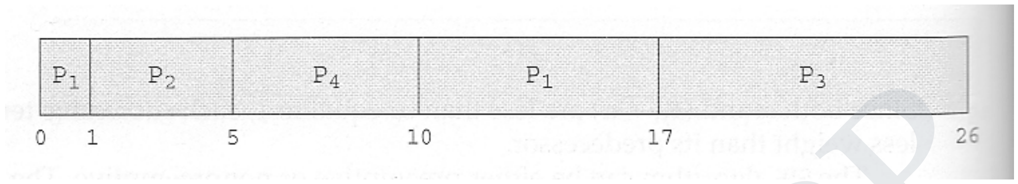
- SJF can be either preemptive or non-preemptive. Preemption occurs when a new process arrives in the ready queue that has a predicted burst time shorter than the time remaining in the process whose burst is currently on the CPU. Preemptive SJF is sometimes referred to as ***shortest remaining time first scheduling.***
- For example, the following Gantt chart is based upon the following data:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Process	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
p4	3	5



- The average wait time in this case is

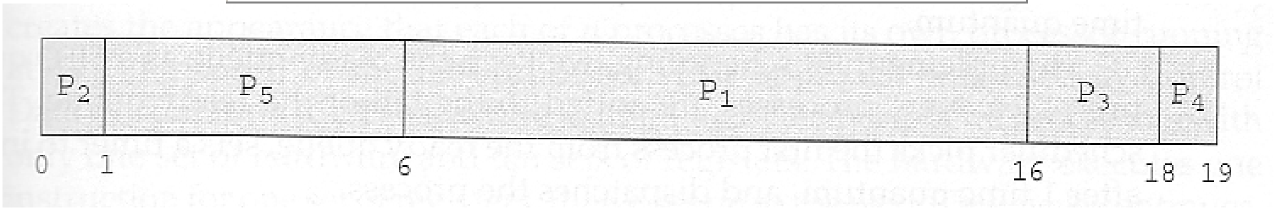
  ((5 - 3) + (10 - 1) + (17 - 2)) / 4 = 26 / 4 = 6.5 ms.

  (As opposed to 7.75 ms for non-preemptive SJF or 8.75 for FCFS.)

2.12.5.3 Priority Scheduling

- Priority scheduling is a more general case of SJF, in which each job is assigned a priority and the job with the highest priority gets scheduled first. (SJF uses the inverse of the next expected burst time as its priority - The smaller the expected burst, the higher the priority.)
- Note that in practice, priorities are implemented using integers within a fixed range, but there is no agreed-upon convention as to whether "high" priorities use large numbers or small numbers.
- For example, the following Gantt chart is based upon these process burst times and priorities, and yields an average waiting time of 8.2 ms:

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2



STUCOR APP

DOWNLOADED FROM STUCOR APP

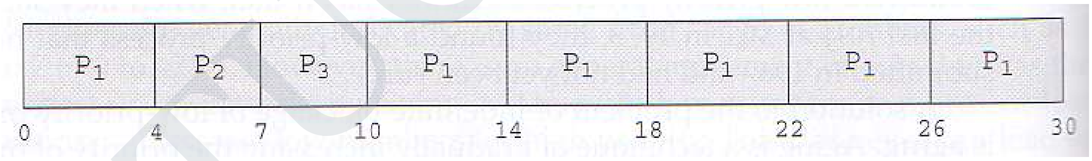
DOWNLOADED FROM STUCOR APP

- Priorities can be assigned either internally or externally. Internal priorities are assigned by the OS using criteria such as average burst time, ratio of CPU to I/O activity, system resource use, and other factors available to the kernel. External priorities are assigned by users, based on the importance of the job, fees paid, politics, etc.

2.12.5.4 Round Robin Scheduling

- Round robin scheduling is similar to FCFS scheduling, except that CPU bursts are assigned with limits called ***time quantum***.
- When a process is given the CPU, a timer is set for whatever value has been set for a time quantum.
  - o If the process finishes its burst before the time quantum timer expires, then it is swapped out of the CPU just like the normal FCFS algorithm.
  - o If the timer goes off first, then the process is swapped out of the CPU and moved to the back end of the ready queue.
- The ready queue is maintained as a circular queue, so when all processes have had a turn, then the scheduler gives the first process another turn, and so on.
- RR scheduling can give the effect of all processors sharing the CPU equally, although the average wait time can be longer than with other scheduling algorithms. In the following example the average wait time is 5.66 ms.

Process	Burst Time
P1	24
P2	3
P3	3



- The performance of RR is sensitive to the time quantum selected. If the quantum is large enough, then RR reduces to the FCFS algorithm; If it is very small, then each process gets 1/nth of the processor time and share the CPU equally.
- **BUT**, a real system invokes overhead for every context switch, and the smaller the time quantum the more context switches there are. (See Figure 5.4 below.) Most modern systems use time quantum between 10 and 100 milliseconds, and context switch times on the order of 10 microseconds, so the overhead is small relative to the time quantum.

✓ In general, turnaround time is minimized if most processes finish their next cpu burst within one time quantum. For example, with three processes of 10 ms bursts each, the average turnaround time for 1 ms quantum is 29, and for 10 ms quantum it reduces to 20.

STUCOR APP

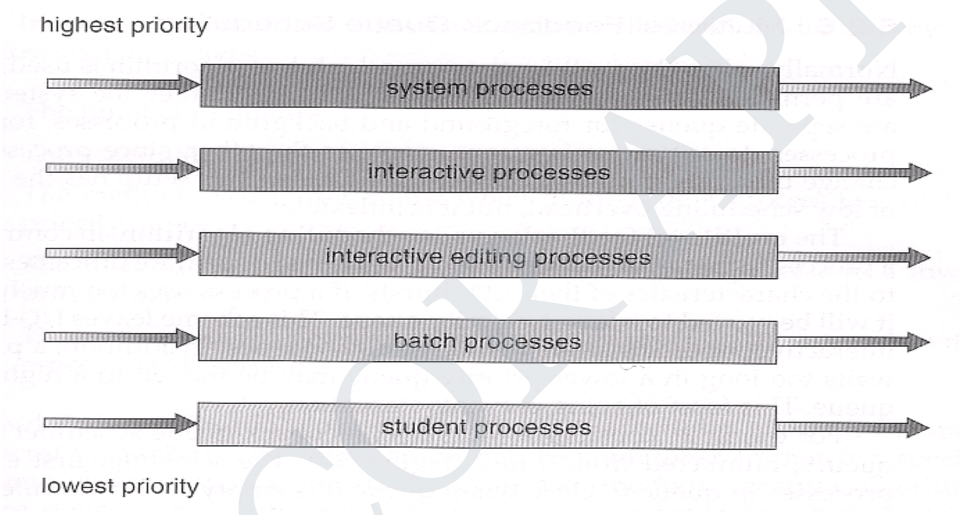
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

However, if it is made too large, then RR just degenerates to FCFS. A rule of thumb is that 80% of CPU bursts should be smaller than the time quantum.

2.12.5.5 Multilevel Queue Scheduling

- ✓ When processes can be readily categorized, then multiple separate queues can be established, each implementing whatever scheduling algorithm is most appropriate for that type of job, and/or with different parametric adjustments.
- ✓ Scheduling must also be done between queues, that is scheduling one queue to get time relative to other queues. Two common options are strict priority (no job in a lower priority queue runs until all higher priority queues are empty) and round-robin (each queue gets a time slice in turn, possibly of different sizes.)
- ✓ Note that under this algorithm jobs cannot switch from queue to queue - Once they are assigned a queue, that is their queue until they finish



STUCOR APP

2.12.5.6 Multilevel Feedback-Queue Scheduling

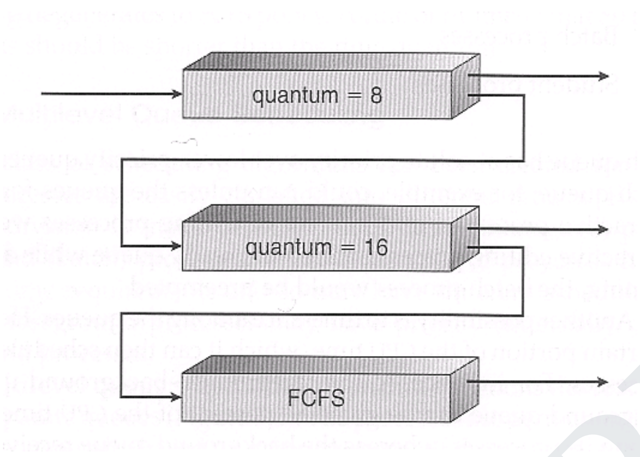
- ✓ Multilevel feedback queue scheduling is similar to the ordinary multilevel queue scheduling described above, except jobs may be moved from one queue to another for a variety of reasons:
    - o If the characteristics of a job change between CPU-intensive and I/O intensive, then it may be appropriate to switch a job from one queue to another.
    - o Aging can also be incorporated, so that a job that has waited for a long time can get bumped up into a higher priority queue for a while.
- ✓ Some of the parameters which define one of these systems include:
    - o The number of queues.
    - o The scheduling algorithm for each queue.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- o The methods used to upgrade or demote processes from one queue to another. (Which may be different.)
- o The method used to determine which queue a process enters initially.



2.13 DEADLOCK

Definition: A process requests resources. If the resources are not available at that time ,the process enters a wait state. Waiting processes may never change state again because the resources they have requested are held by other waiting processes. This situation is called a deadlock.

A process must request a resource before using it, and must release resource after using it.

1. **Request:** If the request cannot be granted immediately then the requesting process must wait until it can acquire the resource.
2. **Use:** The process can operate on the resource
3. **Release:** The process releases the resource.

2.13.1 Deadlock Characterization

Four Necessary conditions for a deadlock

1. Mutual exclusion: At least one resource must be held in a non sharable mode. That is only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.

2. Hold and wait: A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.

3. No preemption: Resources cannot be preempted.

4. Circular wait: P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2...Pn-1.

STUCOR APP

DOWNLOADED FROM STUCOR APP

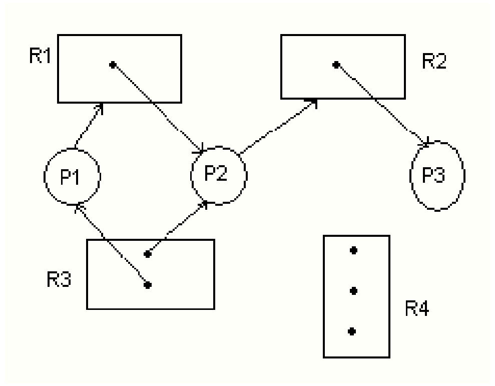
DOWNLOADED FROM STUCOR APP

2.13.2 Resource-Allocation Graph

It is a Directed Graph with a set of vertices V and set of edges E.

V is partitioned into two types:

1. nodes P = {p1, p2,..pn}
2. Resource type R ={R1,R2,...Rm}



Pi -->Rj - request => request edge

Rj-->Pi - allocated => assignment edge.

Pi is denoted as a circle and Rj as a square.

Rj may have more than one instance represented as a dot with in the square.

Sets P,R and E.

P = { P1,P2,P3}

R = {R1,R2,R3,R4}

E= {P1->R1, P2->R3, R1->P2, R2->P1, R3->P3 }

Resource instances

One instance of resource type R1,Two instance of resource type R2,One instance of resource type R3,Three instances of resource type R4.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Process states

Process P1 is holding an instance of resource type R2, and is waiting for an instance of resource type R1.Resource Allocation Graph with a deadlock

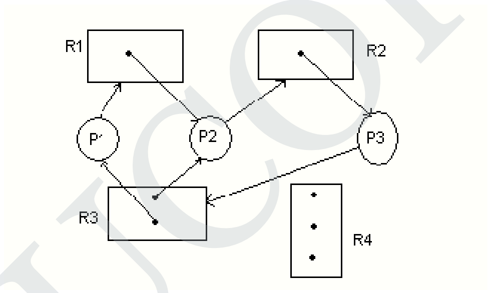
Process P2 is holding an instance of R1 and R2 and is waiting for an instance of resource type R3.Process P3 is holding an instance of R3.

P1->R1->P2->R3->P3->R2->P1

P2->R3->P3->R2->P2

Methods for handling Deadlocks

1. Deadlock Prevention
2. Deadlock Avoidance
3. Deadlock Detection and Recovery



2.13.3 Deadlock Prevention:

❖ This ensures that the system never enters the deadlock state.

❖ Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions cannot hold.

❖ By ensuring that at least one of these conditions cannot hold, we can prevent the occurrence of a deadlock.

1. Denying Mutual exclusion

- Mutual exclusion condition must hold for non-sharable resources.
- Printer cannot be shared simultaneously shared by prevent processes.
- sharable resource - example Read-only files.

STUCOR APP

DOWNLOADED FROM STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

- If several processes attempt to open a read-only file at the same time, they can be granted simultaneous access to the file.
- A process never needs to wait for a sharable resource.

2. Denying Hold and wait

- Whenever a process requests a resource, it does not hold any other resource.
- One technique that can be used requires each process to request and be allocated
- all its resources before it begins execution.
- Another technique is before it can request any additional resources, it must release all the resources that it is currently allocated.
  - These techniques have two main disadvantages :
    - First, resource utilization may be low, since many of the resources may be allocated but unused for a long time.
    - We must request all resources at the beginning for both protocols.
    - starvation is possible.

3. Denying No preemption

- If a Process is holding some resources and requests another resource that cannot be immediately allocated to it. (that is the process must wait), then all resources currently being held are preempted.(ALLOW PREEMPTION)
- These resources are implicitly released.
- The process will be restarted only when it can regain its old resources.

4. Denying Circular wait

- Impose a total ordering of all resource types and allow each process to request for resources in an increasing order of enumeration.
  - Let R = {R1,R2,...Rm} be the set of resource types.
- Assign to each resource type a unique integer number.
- If the set of resource types R includes tapedrives, disk drives and printers.

F(tapedrive)=1,

F(diskdrive)=5,

F(Printer)=12.

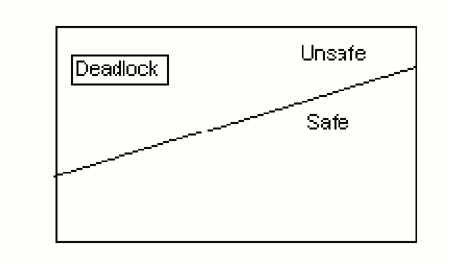
STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

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

2.13.4 Deadlock Avoidance:



- ✓ Deadlock avoidance request that the OS be given in advance additional information concerning which resources a process will request and useduring its life time. With this information it can be decided for each request whether or not the process should wait.
- ✓ To decide whether the current request can be satisfied or must be delayed, a system must consider the resources currently available, the resources currently allocated to each process and future requests and releases of each process.

Safe State

- ❖ A state is safe if the system can allocate resources to each process in some order and still avoid a dead lock.
- ❖ A deadlock is an unsafe state.
- ❖ Not all unsafe states are dead locks
- ❖ An unsafe state may lead to a dead lock

STUCOR APP

Two algorithms are used for deadlock avoidance namely;

1. Resource Allocation Graph Algorithm - single instance of a resource type.

2. Banker's Algorithm – several instances of a resource type.

Resource allocation graph algorithm

- **Claim edge -** Claim edge Pi---> Rj indicates that process Pi may request resource Rj at some time, represented by a dashed directed edge.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- When process Pi request resource Rj, the claim edge Pi -> Rj is converted to a request edge.
- Similarly, when a resource Rj is released by Pi the assignment edge Rj -> Pi is reconverted to a claim edge Pi -> Rj

Banker's algorithm

Available: indicates the number of available resources of each type.

Max: Max[i, j]=k then process Pi may request at most k instances of resource type Rj

Allocation : Allocation[i. j]=k, then process Pi is currently allocated K instances of resource type Rj

Need : if Need[i, j]=k then process Pi may need K more instances of resource type Rj

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

Safety algorithm

```
1. Initialize work := available and Finish [i]:=false for i=1,2,3 .. n
2. Find an i such that both
        a. Finish[i]=false
        b. Needi<= 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 Requesti be the request from process Pi for resources.

1. If Requesti<= Needi goto step2, otherwise raise an error condition, since the process has exceeded its maximum claim.

2. If Requesti <= Available, goto step3, otherwise Pi must wait, since the resources are not available.

```
3. Available := Availabe-Requesti;
   Allocationi := Allocationi + Requesti
   Needi := Needi - Requesti;
```

Now apply the safety algorithm to check whether this new state is safe or not.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

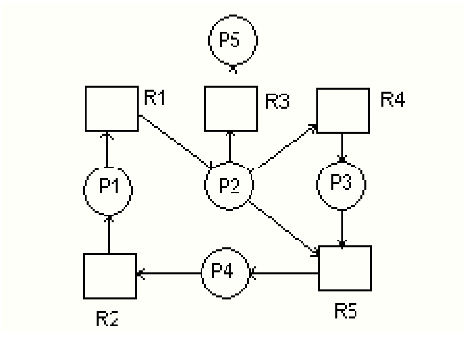
DOWNLOADED FROM STUCOR APP

If it is safe then the request from process Pi can be granted.

2.13.5 Deadlock Detection

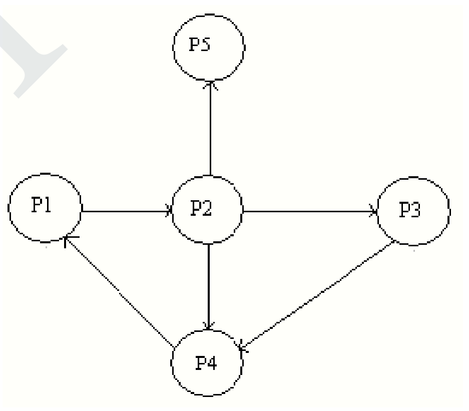
(i) Single instance of each resource type

If all resources have only a single instance, then we can define a deadlock detection algorithm that use a variant of resource-allocation graph called a wait for graph.



Resource Allocation Graph

Wait for S



STUCOR APP

ii) Several Instance of a resource type

Available : Number of available resources of each type

Allocation : number of resources of each type currently allocated to each process

Request : Current request of each process

If Request [i,j]=k, then process Pi is requesting K more instances of resource type Rj.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

```
1. Initialize work := available
        Finish[i]=false, otherwise finish [i]:=true
2. Find an index i such that both
        a. Finish[i]=false
        b. Requesti<=work
        if no such i exists go to step4.
3. Work:=work+allocationi
        Finish[i]:=true
        goto step2
4. If finish[i]=false
        then process Pi is deadlocked
```

2.13.6 Deadlock Recovery

1. Process Termination

1. Abort all deadlocked processes.

2. Abort one deadlocked process at a time until the deadlock cycle is eliminated. After each process is aborted , a deadlock detection algorithm must be invoked to determine where any process is still dead locked.

2. Resource Preemption

Preemptive some resources from process and give these resources to other processes until the deadlock cycle is broken.

i. Selecting a victim: which resources and which process are to be preempted.

ii. Rollback: if we preempt a resource from a process it cannot continue with its normal execution. It is missing some needed resource. we must rollback the process to some safe state, and restart it from that state.

iii. Starvation : How can we guarantee that resources will not always be preempted from the same process.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

UNIT - III STORAGE MANAGEMENT

Memory Management: Main Memory, Contiguous memory allocation, Segmentation–Paging, 32 and 64 bit architecture. Examples: Virtual Memory –Demand paging, Page replacement – Allocation, Thrashing; Allocating Kernal Memory , OS Examples.

3.1 MEMORY MANAGEMENT: BACKGROUND

- ✓ In general, to rum a program, it must be brought into memory.
- ✓ Input queue – collection of processes on the disk that are waiting to be brought into memory to run the program.
- ✓ User programs go through several steps before being run

Address binding: Mapping of instructions and data from one address to another address in memory.

Three different stages of binding:

1. **Compile time:** Must generate absolute code if memory location is known in prior.
2. **Load time:** Must generate relocatable code if memory location is not known at compile time
3. **Execution time:** Need hardware support for address maps (e.g., base and limit registers).

Logical vs. Physical Address Space

STUCOR APP

STUCOR APP

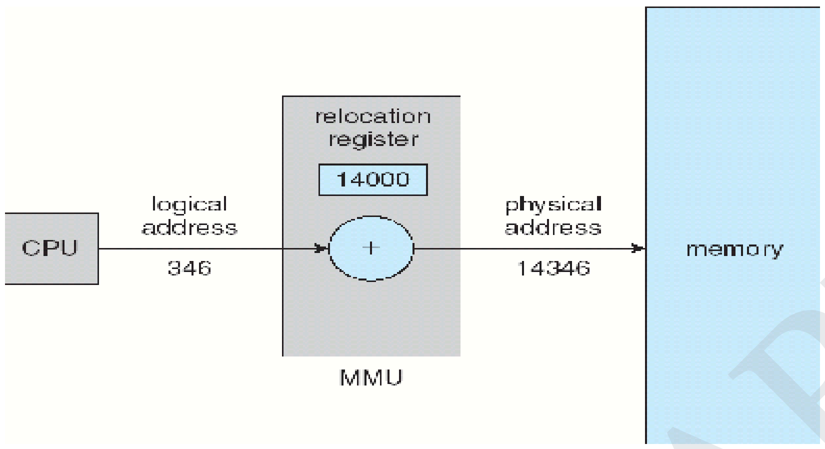
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Logical address – generated by the CPU; also referred to as “virtual address“

Physical address – address seen by the memory unit.

- ❖ Logical and physical addresses are the same in compile-time and load-time address-binding schemes



- ❖ Logical (virtual) and physical addresses differ in execution-time address-binding scheme

3.1.1 Memory-Management Unit (MMU)

- ✓ It is a hardware device that maps virtual / Logical address to physical address.
- ✓ In this scheme, the relocation register‘s value is added to Logical address generated by a user process.
- ✓ The user program deals with logical addresses; it never sees the real physical addresses
  - ❖ Logical address range: 0 to max
- ❖ Physical address range: R+0 to R+max, where R—value in relocation.

Dynamic relocation using relocation register

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Dynamic Loading

- ✓ Through this, the routine is not loaded until it is called.
  - o Better memory-space utilization; unused routine is never loaded
  - o Useful when large amounts of code are needed to handle infrequently occurring cases
  - o No special support from the operating system is required implemented through program design

Dynamic Linking

- ✓ Linking postponed until execution time & is particularly useful for libraries
- ✓ Small piece of code called stub, used to locate the appropriate memory resident library routine or function.
- ✓ Stub replaces itself with the address of the routine, and executes the routine
- ✓ Operating system needed to check if routine is in processes Memory addresses Shared libraries.
- ✓ Programs linked before the new library was installed will continue using the older library.

STUCOR APP

3.2 CONTIGUOUS MEMORY ALLOCATION

Each process is contained in a single contiguous section of memory.

There are two methods namely :

- ❖ **Fixed – Partition Method**
- ❖ **Variable – Partition Method**

Fixed – Partition Method :

- Divide memory into fixed size partitions, where each partition has exactly one process.
- The drawback is memory space unused within a partition is wasted.(eg.when process size < partition size)

Variable-partition method:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ Divide memory into variable size partitions, depending upon the size of the incoming process.
  - When a process terminates, the partition becomes available for another process.
  - As processes complete and leave they create holes in the main memory.
  - Hole – block of available memory; holes of various size are scattered throughout memory.

Dynamic Storage-Allocation Problem:

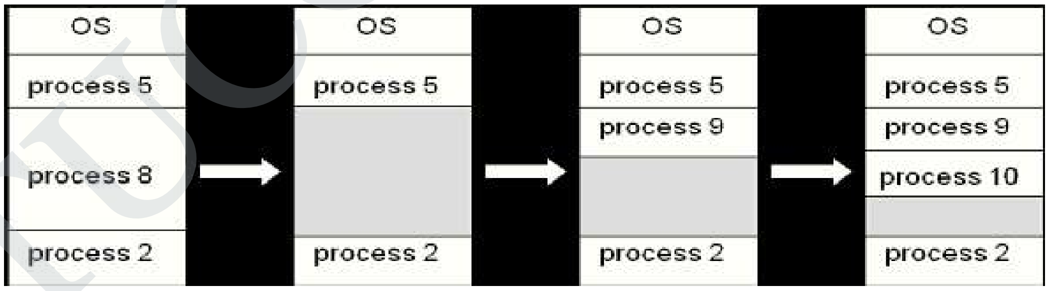
How to satisfy a request of size =n‘ from a list of free holes?

Solution:

- ❖ **First-fit:** Allocate the first hole that is big enough.
- ❖ **Best-fit:** Allocate the smallest hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
- ❖ **Worst-fit:** Allocate the largest hole; must also search entire list. Produces the largest leftover hole.

NOTE: First-fit and best-fit are better than worst-fit in terms of speed and storage utilization

Fragmentation

- 

- ❖ **External Fragmentation** – This takes place when enough total memory space exists to satisfy a request, but it is not contiguous i.e, storage is fragmented into a large number of small holes scattered throughout the main memory.
- ❖ **Internal Fragmentation** – Allocated memory may be slightly larger than requested memory.

Example:

hole = 184 bytes

Process size = 182 bytes.

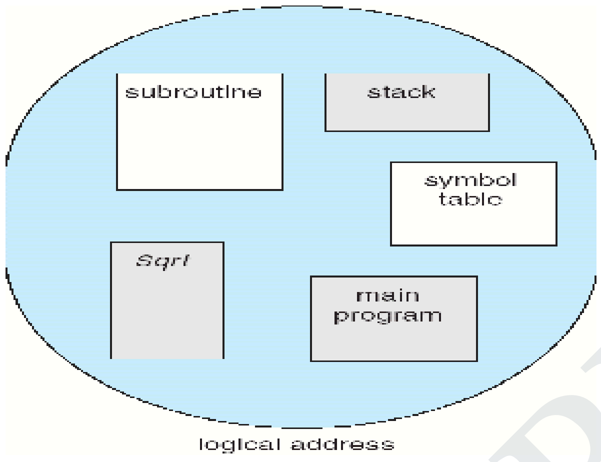
STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

We are left with a hole of 2 bytes.

Solutions:

- **Coalescing :** Merge the adjacent holes together.
- **Compaction:** Move all processes towards one end of memory, hole towards other end of memory, producing one large hole of available memory. This scheme is expensive as it can be done if relocation is dynamic and done at execution time.



- Permit the logical address space of a process to be non-contiguous. This is achieved through two memory management schemes namely **paging and segmentation.**

3.3 SEGMENTATION

- ✓ Memory-management scheme that supports user view of memory
- ✓ A program is a collection of segments.
- ✓ A segment is a logical unit such as: Main program, Procedure, Function, Method, Object, Local variables, global variables, Common block, Stack, Symbol table, arrays.

STUCOR APP

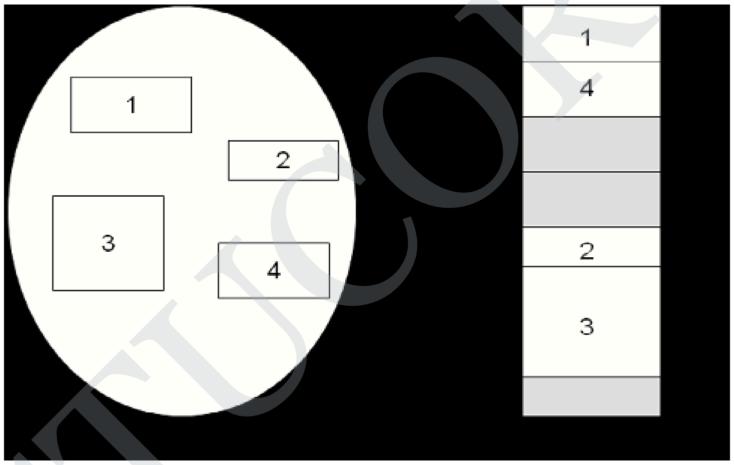
<u>User's View of Program</u>

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Logical View of Segmentation



Segmentation Hardware

- ✓ Logical address consists of a two tuple :
  - <Segment-number, offset>

- ✓ Segment table – maps two-dimensional physical addresses; each table entry has:
  - Base – contains the starting physical address where the segments reside in memory
  - Limit – specifies the length of the segment
- ✓ Segment-table base register (STBR) points to the segment table‘s location in memory
- ✓ Segment-table length register (STLR) indicates number of segments used by a program;
  - Segment number=s‘ is legal, if s < STLR
- ✓ Relocation.

STUCOR APP

DOWNLOADED FROM STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

- dynamic
- by segment table

✓ Sharing.

- shared segments
- same segment number

✓ Allocation.

- first fit/best fit
- external fragmentation

✓ Protection: With each entry in segment table associate:

- validation bit = 0 illegal segment
- read/write/execute privileges

✓ Protection bits associated with segments; code sharing occurs at segment level

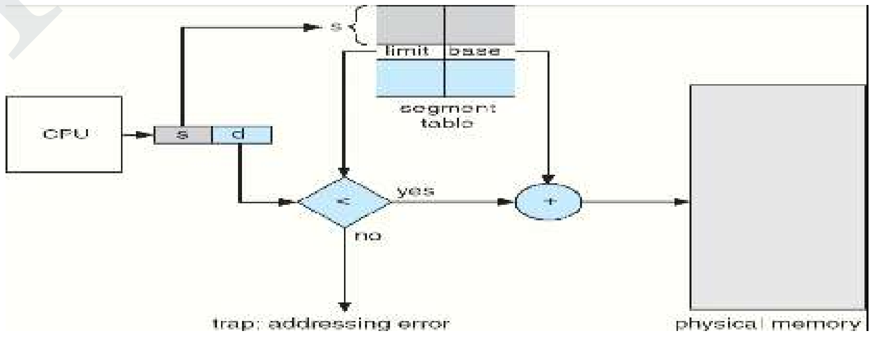
✓ Since segments vary in length, memory allocation is a dynamic storage-allocation problem

✓ A segmentation example is shown in the following diagram

STUCOR APP

Address Translation Scheme

✓

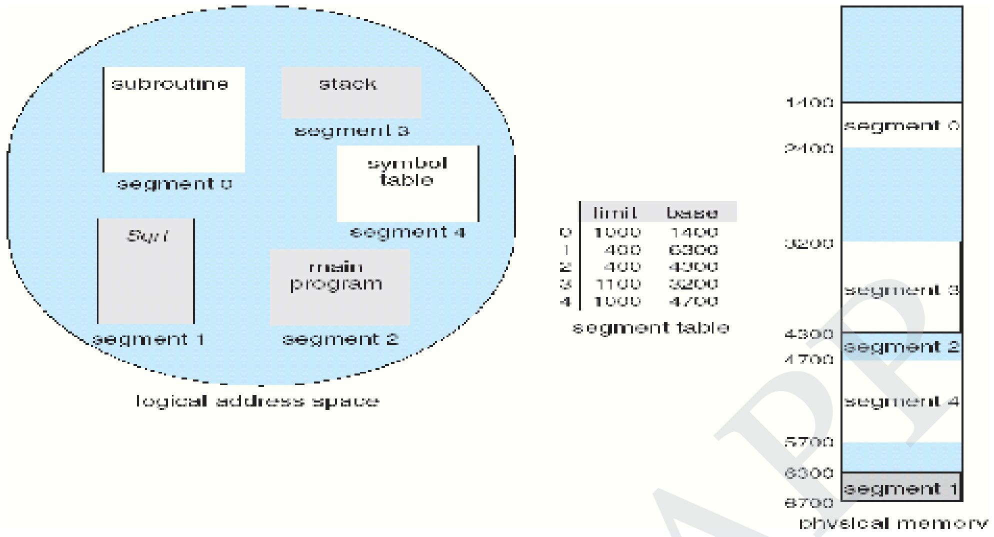


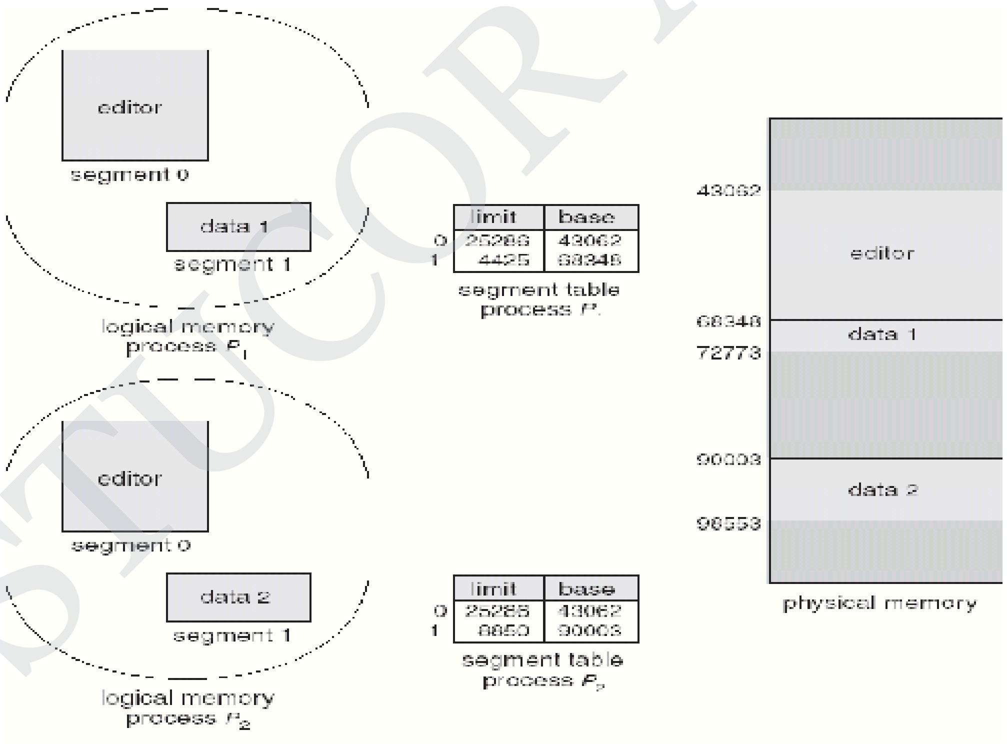
STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Example





✓ Another advantage of segmentation involves the sharing of code or data.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ❖ Each process has a segment table associated with it, which the dispatcher uses to define the hardware segment table when this process is given the CPU.
- ❖ Segments are shared when entries in the segment tables of two different processes point to the same physical location.

Segmentation with paging

- ❖ The IBM OS/ 2.32 bit version is an operating system running on top of the Intel 386 architecture. The 386 user segmentation with paging for memory management. The maximum number of segments per process is 16 KB, and each segment can be as large as 4 gigabytes.
  - ❖ The local-address space of a process is divided into two partitions.
  - ❖ The first partition consists of up to 8 KB segments that are private to that process.
  - ❖ The second partition consists of up to 8KB segments that are shared among all the processes.
- ✓ Information about the first partition is kept in the local descriptor table (LDT), information about the second partition is kept in the global descriptor table (GDT).
- ✓ Each entry in the LDT and GDT consist of 8 bytes, with detailed information about a particular segment including the base location and length of the segment.
- ✓ The logical address is a pair (selector, offset) where the selector is a16-bit number:

s	**g**	**p**
13	1	2

- ✓ Where s designates the segment number, g indicates whether the segment is in the GDT or LDT, and p deals with protection.
- ✓ The offset is a 32-bit number specifying the location of the byte within the segment in question.
- ✓ The base and limit information about the segment in question are used to generate a linear-address.
- ✓ First, the limit is used to check for address validity. If the address is not valid, a memory fault is generated, resulting in a trap to the operating system. If it is valid, then the value of the offset is added to the value of the base, resulting in a 32-bit linear address. This address is then translated into a physical address.
- ✓ The linear address is divided into a page number consisting of 20 bits, and a page offset consisting of 12 bits. Since we page the page table, the page number is further divided into a 10-bit page directory pointer and a 10-bit page table pointer. The logical address is as follows.

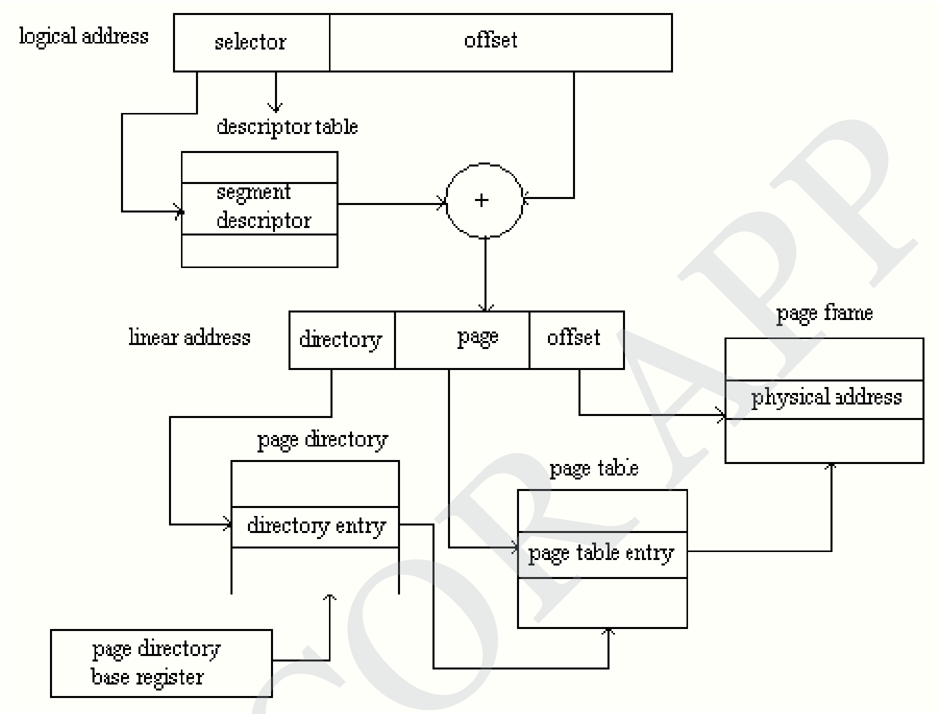
s	**g**	**p**
10	10	12

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ To improve the efficiency of physical memory use. Intel 386 page tables can be swapped to disk. In this case, an invalid bit is used in the page directory entry to indicate whether the table to which the entry is pointing is in memory or on disk.
- ✓ If the table is on disk, the operating system can use the other 31 bits to specify the disk location of the table; the table then can be brought into memory on demand.



STUCOR APP

3.3 PAGING

- ✓ It is a memory management scheme that permits the physical address space of a process to be noncontiguous.
- ✓ It avoids the considerable problem of fitting the varying size memory chunks on to the backing store.

(i) Basic Method:

- ❖ Divide logical memory into blocks of same size called "pages".
- ❖ Divide physical memory into fixed-sized blocks called "frames"
- ❖ Page size is a power of 2, between 512 bytes and 16MB.

Address Translation Scheme

Address generated by CPU(logical address) is divided into:

Page number (p) – used as an index into a page table which contains base address of each page in physical memory

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Page offset (d) – combined with base address to define the physical address i.e.,

Physical address = base address + offset

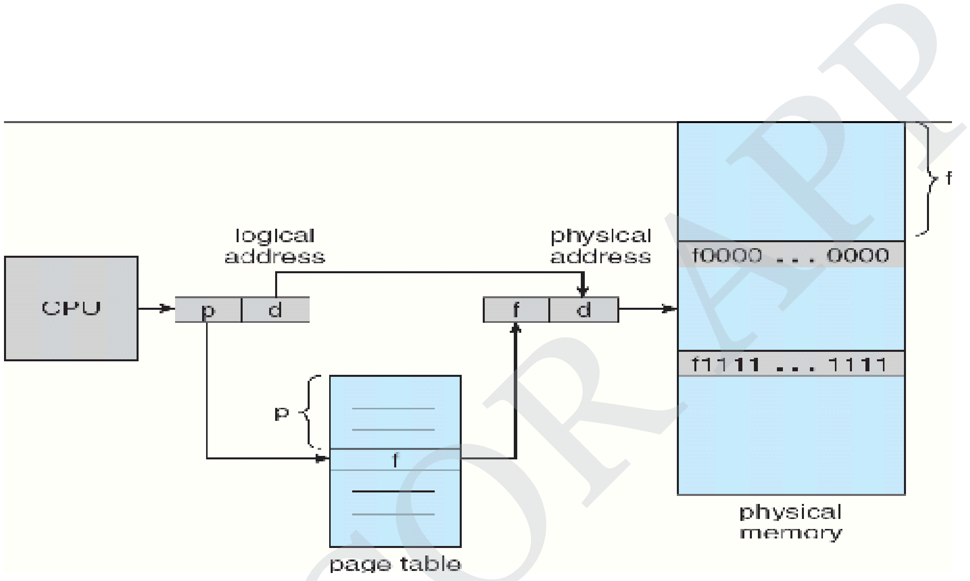


Fig 3.3.1 Paging Hardware

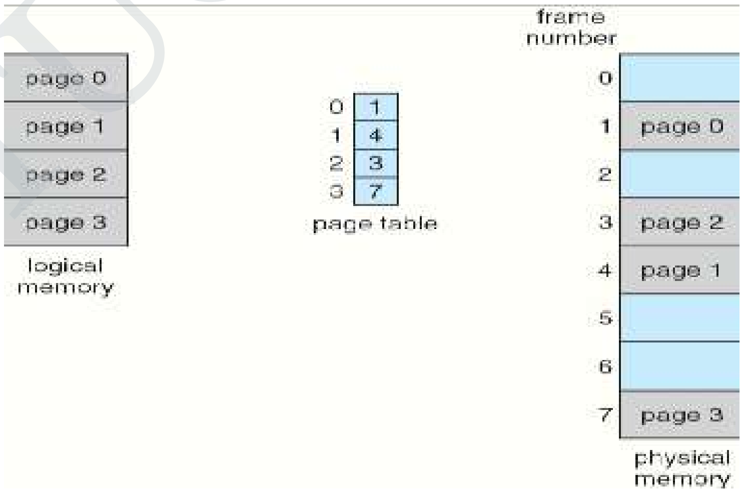


Figure 3.3.2 Paging model of logical and physical memory

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

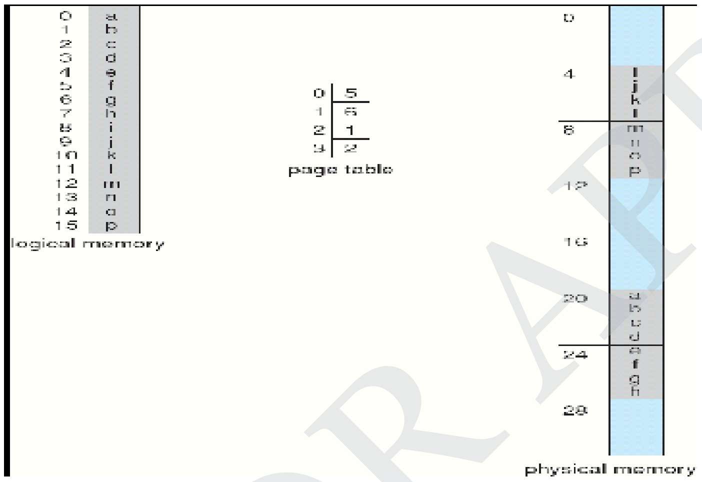
Paging example for a 32-byte memory with 4-byte pages

Page size = 4 bytes

Physical memory size = 32 bytes i.e (4 X 8 = 32 so, 8 pages)

Logical address =0‘ maps to physical address 20 i.e ((5 X 4) +0)

Where Frame no = 5, Page size = 4, Offset = 0



Allocation

- ❖ When a process arrives into the system, its size (expressed in pages) is examined.
- ❖ Each page of process needs one frame. Thus if the process requires =n‘ pages, at least =n‘ frames must be available in memory.
- ❖ If =n‘ frames are available, they are allocated to this arriving process.
- ❖ The 1st page of the process is loaded into one of the allocated frames & the frame number is put into the page table.
- ❖ Repeat the above step for the next pages & so on.

Frame table:

- ❖ It is used to determine which frames are allocated, which frames are available, how many total frames are there, and so on.(ie) It contains all the information about the frames in the physical memory.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

a)Before Allocation **b) After Allocation**

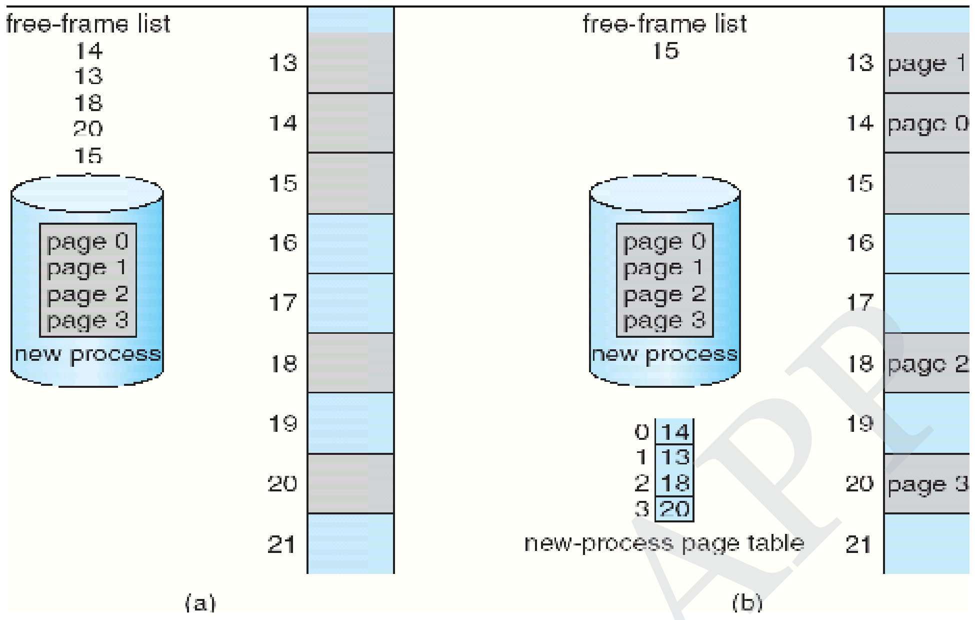


Fig 3.3 Page Allocation Table

(ii) Hardware implementation of Page Table

- ❖ This can be done in several ways :
  - o Using PTBR
  - o TLB
- ❖ The simplest case is Page-table base register (PTBR), is an index to point the page table.
- ❖ TLB (Translation Look-aside Buffer)
  - o It is a fast lookup hardware cache.
  - o It contains the recently or frequently used page table entries.
  - o It has two parts: Key (tag) & Value.
  - o More expensive.

Paging Hardware with TLB

- ❖ When a logical address is generated by CPU, its page number is presented to TLB.
- ❖ **TLB hit:** If the page number is found, its frame number is immediately available & is used to access memory
- ❖ **TLB miss:** If the page number is not in the TLB, a memory reference to the page table must be made.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- **Hit ratio:** Percentage of times that a particular page is found in the TLB. For example hit ratio is 80% means that the desired page number in the TLB is 80% of the time.

- **Effective Access Time:**

  - Assume hit ratio is 80%. If it takes 20ns to search TLB & 100ns to access memory, then the memory access takes 120ns(TLB hit)

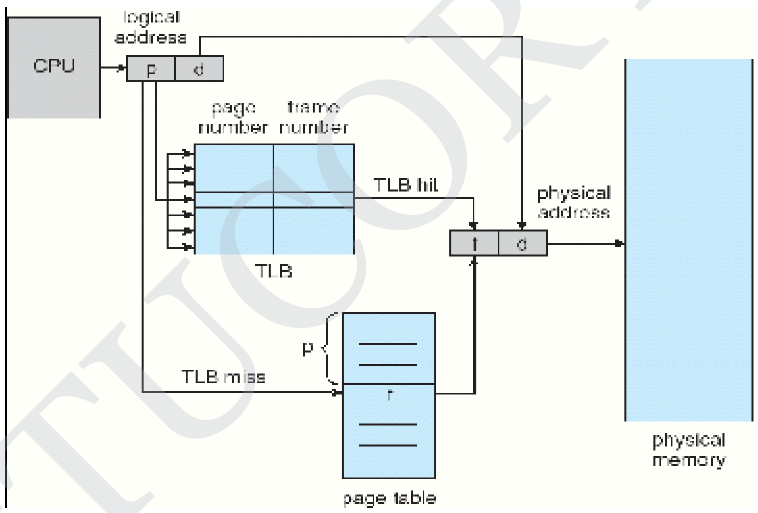
  - If we fail to find page no. in TLB (20ns), then we must 1st access memory for page table (100ns) & then access the desired byte in memory (100ns).

    Therefore Total = 20 + 100 + 100

    = 220 ns(TLB miss).

    Then Effective Access Time (EAT) = 0.80 X (120 + 0.20) X 220.

    = 140 ns.



(iii) Memory Protection

- Memory protection implemented by associating protection bit with each frame

- Valid-invalid bit attached to each entry in the page table:

  - "valid (v)" indicates that the associated page is in the process' logical address space, and is thus a legal page

  - "invalid (i)" indicates that the page is not in the process' logical address space.

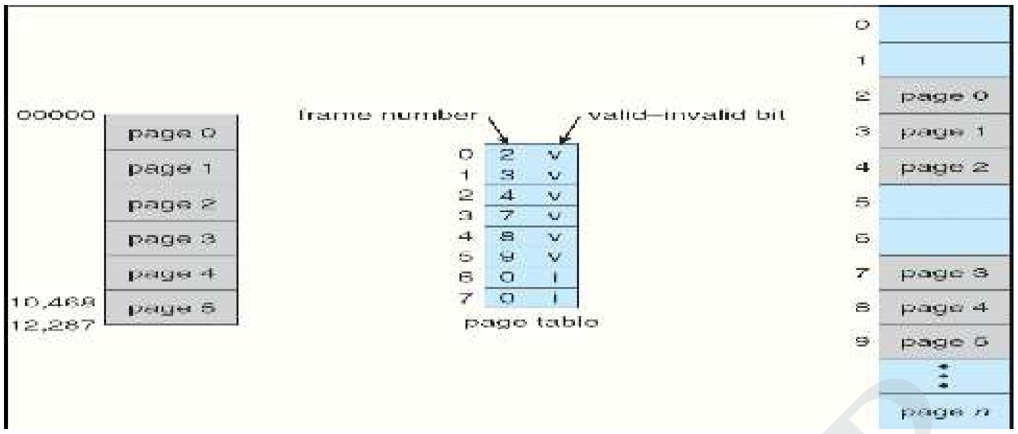
Fig 3.4 Memory Protection

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

(iv) Structures of the Page Table



a) Hierarchical Paging

b) Hashed Page Tables

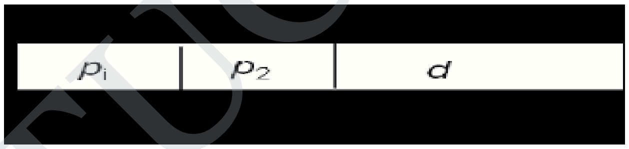
c) Inverted Page Tables

a) Hierarchical Paging

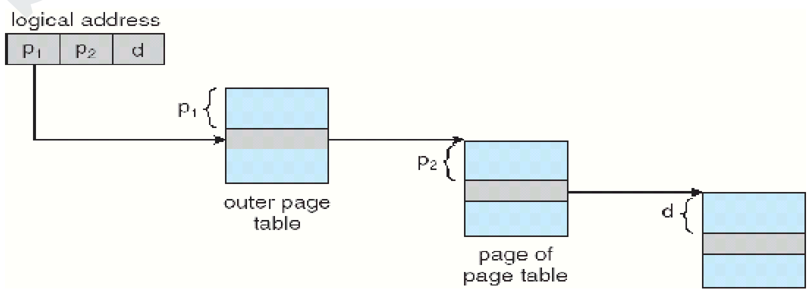
Break up the Page table into smaller pieces. Because if the page table is too large then it is quit difficult to search the page number.

Example: "Two-Level Paging "

Address- **Translation Scheme**



Address- translation scheme for a two-level 32-bit paging architecture



It requires more number of memory accesses, when the number of levels is increased.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

(b) Hashed Page Tables

Each entry in hash table contains a linked list of elements that hash to the same location.

Each entry consists of;

(a) Virtual page numbers

(b) Value of mapped page frame.

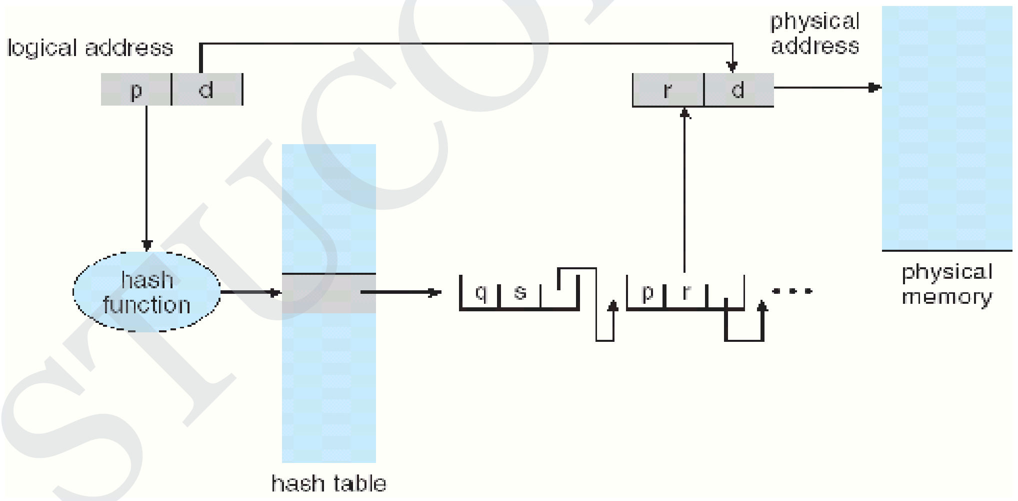
(c) Pointer to the next element in the linked list.

Working Procedure:

- The virtual page number in the virtual address is hashed into the hash table.
- Virtual page number is compared to field (a) in the 1st element in the linked list.
- If there is a match, the corresponding page frame (field (b)) is used to form the desired physical address.
- If there is no match, subsequent entries in the linked list are searched for a matching virtual page number.

Clustered page table:

❖ It is a variation of hashed page table & is similar to hashed page table except that each entry in the hash table refers to several pages rather than a single page.



(c)Inverted Page Table

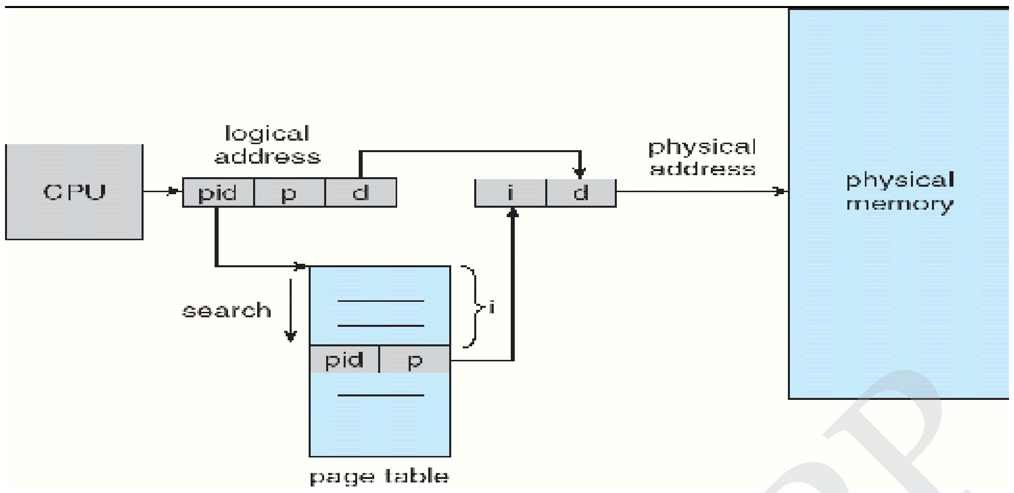
❖ It has one entry for each real page (frame) of memory & each entry consists of the virtual address of the page stored in that real memory location, with information about the process that owns that page. So, only one page table is in the system.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

❖ When a memory reference occurs, part of the virtual address, consisting of <Process-id,



Page-no> is presented to the memory sub-system.

❖ Then the inverted page table is searched for match:

(i) If a match is found, then the physical address is generated.

(ii) If no match is found, then an illegal address access has been attempted.

- **Merit:** Reduce the amount of memory needed.
- **Demerit:** Improve the amount of time needed to search the table when a page reference occurs.

(v) Shared Pages

❖ One advantage of paging is the possibility of sharing common code.

- **Shared code**
  - o One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
  - o Shared code must appear in same location in the logical address space of all processes
- **Reentrant code (Pure code):** Non-self modifying code. If the code is reentrant, then it never changes during execution. Thus two or more processes can execute the same code at the same time.
- **Private code and data**: Each process keeps a separate copy of the code and data.

3.5 VIRTUAL MEMORY

STUCOR APP

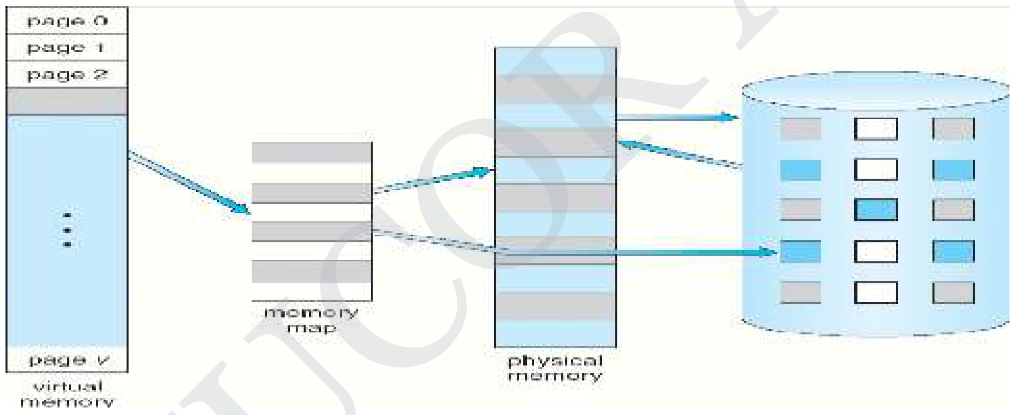
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ It is a technique that allows the execution of processes that may not be completely in main memory.
- ✓ Advantages:
    - o Allows the program that can be larger than the physical memory.
    - o Separation of user logical memory from physical memory
    - o Allows processes to easily share files & address space.
    - o Allows for more efficient process creation.
- ✓ Virtual memory can be implemented using
    - o **Demand paging**
    - o **Demand segmentation**

Fig 3.5 Virtual Memory That is Larger than Physical Memory

Demand Paging



STUCOR APP

- ✓ It is similar to a paging system with swapping.
- ✓ **Demand Paging** - Bring a page into memory only when it is needed
- ✓ To execute a process, swap that entire process into memory. Rather than swapping the entire process into memory however, we use Lazy Swapper
- ✓ **Lazy Swapper** - Never swaps a page into memory unless that page will be needed.
- ✓ Advantages

  Less I/O needed

  Less memory needed

  Faster response

  More users

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Basic Concepts:

- Instead of swapping in the whole processes, the pager brings only those necessary pages into memory. Thus,
  - It avoids reading into memory pages that will not be used anyway.
  - Reduce the swap time.
  - Reduce the amount of physical memory needed.
- To differentiate between those pages that are in memory & those that are on the disk we use the Valid-Invalid bit

Valid-Invalid bit

- A valid – invalid bit is associated with each page table entry.
- Valid associated page is in memory.
- In-Valid

  invalid page

  valid page but is currently on the disk.

STUCOR APP

Page Fault

- Access to a page marked invalid causes a page fault trap.

1. Determine whether the reference is a valid or invalid memory access
2. a) If the reference is invalid then terminate the process.

   b) If the reference is valid then the page has not been yet brought into main memory.
3. Find a free frame.
4. Read the desired page into the newly allocated frame.
5. Reset the page table to indicate that the page is now in memory.
6. Restart the instruction that was interrupted .

Pure demand paging

- Never bring a page into memory until it is required. We could start a process with no pages in memory.
- When the OS sets the instruction pointer to the 1st instruction of the process, which is on the non-memory resident page, then the process immediately faults for the page.
- After this page is bought into the memory, the process continue to execute, faulting as necessary until every page that it needs is in memory.

Performance of demand paging

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Let p be the probability of a page fault 0   p   1

Effective Access Time (EAT)

EAT = (1 – p) x ma + p x page fault time.

Where ma   memory access, p   Probability of page fault (0= p = 1)

❖ The memory access time denoted ma is in the range 10 to 200 ns.

❖ If there are no page faults then EAT = ma.

❖ To compute effective access time, we must know how much time is needed to service a page fault.

❖ A page fault causes the following sequence to occur:

1. Trap to the OS
2. Save the user registers and process state.
3. Determine that the interrupt was a page fault.
4. Check whether the reference was legal and find the location of page on disk.
5. Read the page from disk to free frame.
   a. Wait in a queue until read request is serviced.
   b. Wait for seek time and latency time.
   c. Transfer the page from disk to free frame.
6. While waiting ,allocate CPU to some other user.
7. Interrupt from disk.
8. Save registers and process state for other users.
9. Determine that the interrupt was from disk.
10. Reset the page table to indicate that the page is now in memory.
11. Wait for CPU to be allocated to this process again.

STUCOR APP

3.6 PAGE REPLACEMENT

✓ If no frames are free, we could find one that is not currently being used & free it.

✓ We can free a frame by writing its contents to swap space & changing the page table to indicate that the page is no longer in memory.

✓ Then we can use that freed frame to hold the page for which the process faulted.

Basic Page Replacement

1. Find the location of the desired page on disk

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

2. Find a free frame

   - If there is a free frame , then use it.

   - If there is no free frame, use a page replacement algorithm to select a victim frame

   - Write the victim page to the disk, change the page & frame tables accordingly.

3. Read the desired page into the (new) free frame. Update the page and frame tables.

4. Restart the process

Modify (dirty) bit:

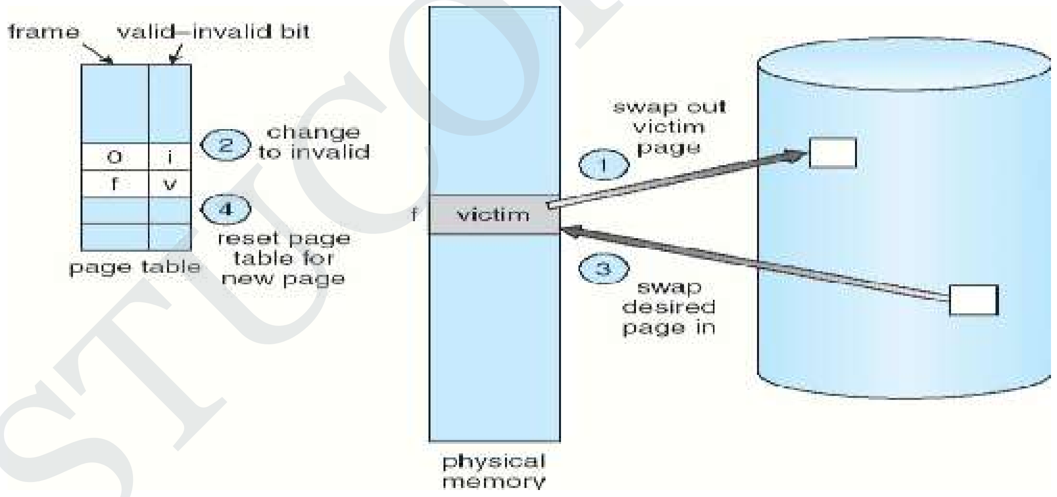
❖ It indicates that any word or byte in the page is modified.

❖ When we select a page for replacement, we examine its modify bit.

  - If the bit is set, we know that the page has been modified & in this case we must write that page to the disk.

  - If the bit is not set, then if the copy of the page on the disk has not been overwritten, then we can avoid writing the memory page on the disk as it is already there.

Page Replacement Algorithms



1. FIFO Page Replacement
2. Optimal Page Replacement
3. LRU Page Replacement
4. LRU Approximation Page Replacement
5. Counting-Based Page Replacement

❖ We evaluate an algorithm by running it on a particular string of memory references & computing the number of page faults. The string of memory reference is

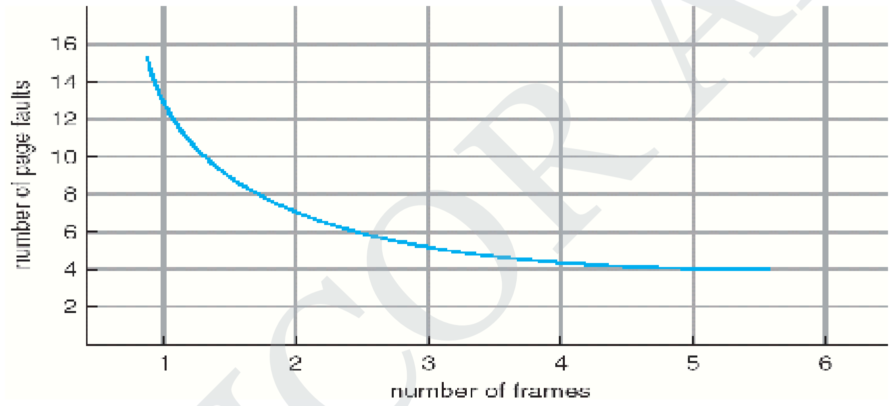
STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

called a reference string. The algorithm that provides less number of page faults is termed to be a good one.

❖ As the number of available frames increases , the number of page faults decreases. This is shown in the following graph:

(a) FIFO page replacement algorithm



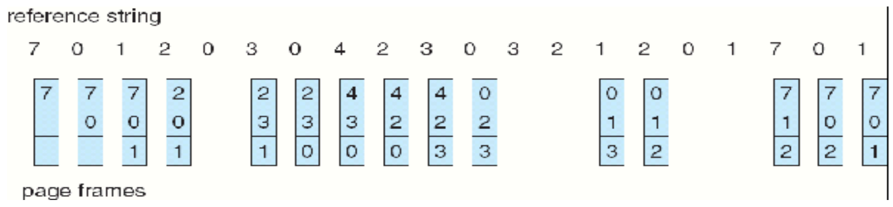
❖ Replace the oldest page.

❖ This algorithm associates with each page ,the time when that page was brought in.

❖ Example:

Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

No.of available frames = 3 (3 pages can be in memory at a time per process)



STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Drawback:

- ❖ FIFO page replacement algorithm =s performance is not always good.
- ❖ To illustrate this, consider the following example:

  Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

  If No.of available frames -= 3 then the no.of page faults =9

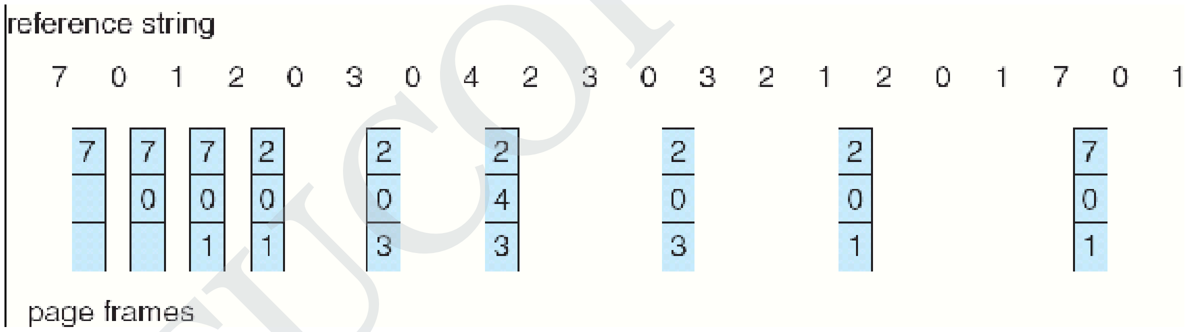
  If No.of available frames =4 then the no.of page faults =10

- ❖ Here the no. of page faults increases when the no.of frames increases .This is called as **Belady's Anomaly.**

(b) Optimal page replacement algorithm

Replace the page that will not be used for the longest period of time.

Example:



Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

No.of available frames = 3

Drawback:

- It is difficult to implement as it requires future knowledge of the reference string.

(c) LRU(Least Recently Used) page replacement algorithm

- ❖ Replace the page that has not been used for the longest period of time.

Example:

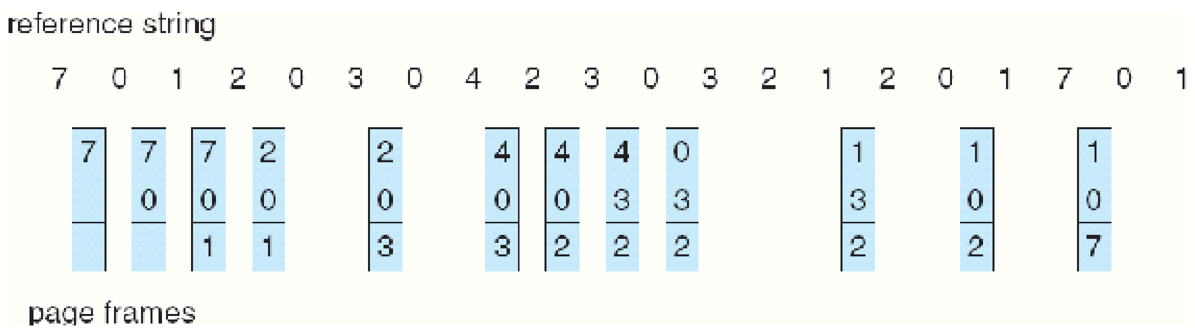
Reference string: 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

No.of available frames = 3

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP



No of Page Faults :12

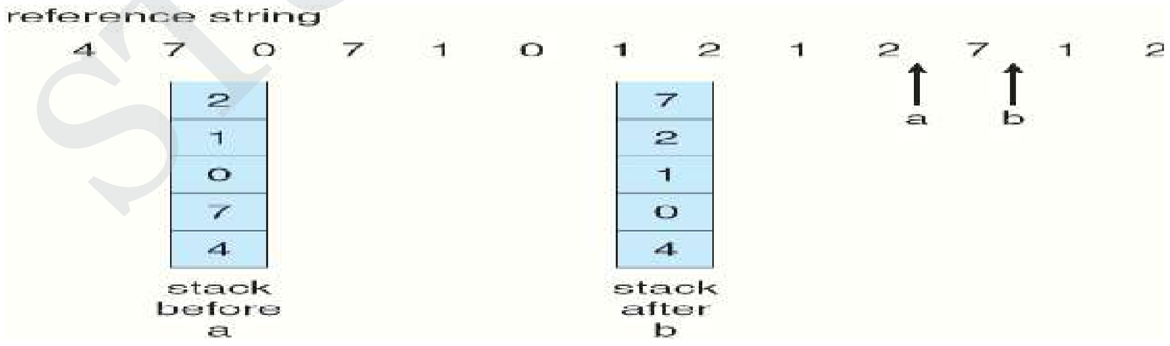
LRU page replacement can be implemented using

1. **Counters**

- Every page table entry has a time-of-use field and a clock or counter is associated with the CPU.
- The counter or clock is incremented for every memory reference.
- Each time a page is referenced , copy the counter into the time-of-use field.
- When a page needs to be replaced, replace the page with the smallest counter value.

2. **Stack**

- Keep a stack of page numbers
- Whenever a page is referenced, remove the page from the stack and put it on top of the stack.



- When a page needs to be replaced, replace the page that is at the bottom of the stack.(LRU page).

Use of A Stack to Record The Most Recent Page References

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

(d) LRU Approximation Page Replacement

- **Reference bit**
    - With each page associate a reference bit, initially set to 0
    - When page is referenced, the bit is set to 1
- When a page needs to be replaced, replace the page whose reference bit is 0
- The order of use is not known , but we know which pages were used and which were not used.

(i) Additional Reference Bits Algorithm

- Keep an 8-bit byte for each page in a table in memory.
- At regular intervals , a timer interrupt transfers control to OS.
- The OS shifts reference bit for each page into higher- order bit shifting the other bits right 1 bit and discarding the lower-order bit.

  Example:

    - If reference bit is 00000000 then the page has not been used for 8 time periods.
    - If reference bit is 11111111 then the page has been used atleast once each time period.
    - If the reference bit of page 1 is 11000100 and page 2 is 01110111 then page 2 is the LRU page.

(ii) Second Chance Algorithm

- Basic algorithm is FIFO
- When a page has been selected , check its reference bit.
    - If 0 proceed to replace the page
    - If 1 give the page a second chance and move on to the next FIFO page.
- When a page gets a second chance, its reference bit is cleared and arrival time is reset to current time.
- Hence a second chance page will not be replaced until all other pages are replaced.

(iii) Enhanced Second Chance Algorithm

- Consider both reference bit and modify bit
- There are four possible classes

  1. (0,0) – neither recently used nor modified Best page to replace

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

2. (0,1) – not recently used but modified page has to be written out before replacement.

3. (1,0) - recently used but not modified page may be used again

4. (1,1) – recently used and modified page may be used again and page has to be written to disk.

(e) Counting-Based Page Replacement

❖ Keep a counter of the number of references that have been made to each page

- Least Frequently Used (LFU)Algorithm: replaces page with smallest count
- Most Frequently Used (MFU)Algorithm: replaces page with largest count

❖ It is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

3.7 THRASHING

✓ High paging activity is called thrashing.

✓ If a process does not have enough pages, the page-fault rate is very high.

✓ This leads to:

  - o low CPU utilization
  - o operating system thinks that it needs to increase the degree of multiprogramming
  - o another process is added to the system

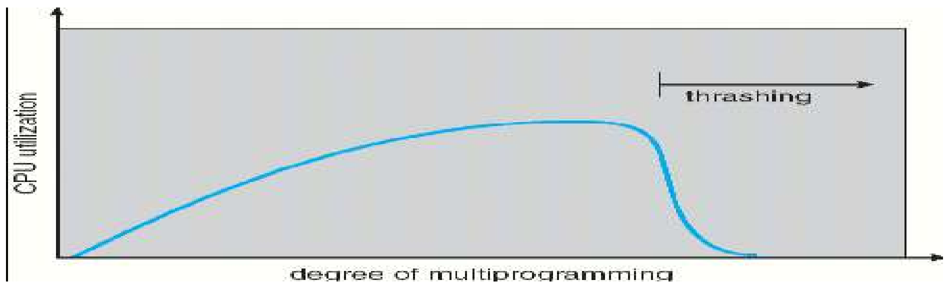
✓ When the CPU utilization is low, the OS increases the degree of multiprogramming.

✓ If global replacement is used then as processes enter the main memory they tend to steal frames belonging to other processes.

✓ Eventually all processes will not have enough frames and hence the page fault rate becomes very high.

✓ Thus swapping in and swapping out of pages only takes place. This is the cause of **thrashing**.

STUCOR APP



STUCOR APP

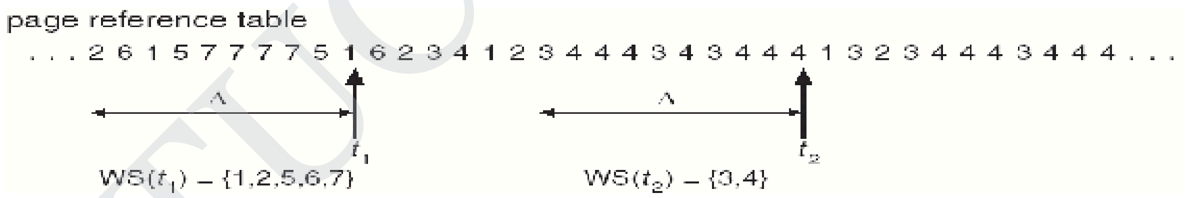
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

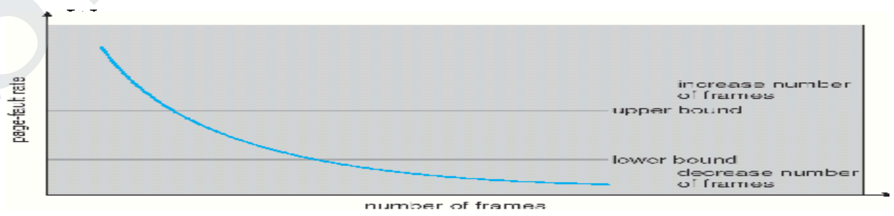
- ✓ To limit thrashing, we can use a local replacement algorithm.
- ✓ To prevent thrashing, there are two methods namely ,
    - Working Set Strategy
    - Page Fault Frequency

1. Working-Set Strategy

- ❖ It is based on the assumption of the model of locality.
- ❖ Locality is defined as the set of pages actively used together.
- ❖ Working set is the set of pages in the most recent page references is the working set window.
    - ▪ if too small , it will not encompass entire locality
    - ▪ if too large ,it will encompass several localities
    - ▪ if = it will encompass entire program



- ❖ D = WSSi



W
SSi is the working set size for process i.

- ▪ D is the total demand of frames

- ❖ if D > m then Thrashing will occur.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

2. Page-Fault Frequency Scheme

- ❖ If actual rate too low, process loses frame
- ❖ If actual rate too high, process gains frame.

Other Issues

Prepaging

- To reduce the large number of page faults that occurs at process startup
- Prepage all or some of the pages a process will need, before they are referenced
- But if prepaged pages are unused, I/O and memory are wasted

Page Size

Page size selection must take into consideration:

- fragmentation
- table size
- I/O overhead
- locality

TLB Reach

- TLB Reach - The amount of memory accessible from the TLB
- TLB Reach = (TLB Size) X (Page Size)
- Ideally, the working set of each process is stored in the TLB.
- Otherwise there is a high degree of page faults.
- Increase the Page Size. This may lead to an increase in fragmentation as not all applications require a large page size
- Provide Multiple Page Sizes. This allows applications that require larger page sizes the opportunity to use them without an increase in fragmentation.

I/O interlock

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- Pages must sometimes be locked into memory
- Consider I/O. Pages that are used for copying a file from a device must be locked from being selected for eviction by a page replacement algorithm.

UNIT – IV I/O SYSTEMS

Mass-Storage Structure-Overview, Disk scheduling – Disk management; File System Storage– File Concepts, Directory and Disk Structure, Sharing and Protection; File System Implementation- File System Structure, Directory Structure, Allocation Methods, Free Space Management, I/O Systems.

4.1 MASS STORAGE STRUCTURE

Overview

- ✓ One of the responsibilities of the operating system is to use the hardware efficiently.
- ✓ For the disk drives,
  - o A fast access time and
  - o High disk bandwidth.
- ✓ The access time has two major components;
- ✓ The seek time is the time for the disk arm to move the heads to the cylinder containing the desired sector.
- ✓ The rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head.
- ✓ The disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
- ✓ We can improve both the access time and the bandwidth by disk scheduling.
- ✓ Servicing of disk I/O requests in a good order.

4.2 DISK SCHEDULING AND MANAGEMENT

STUCOR APP

DOWNLOADED FROM STUCOR APP

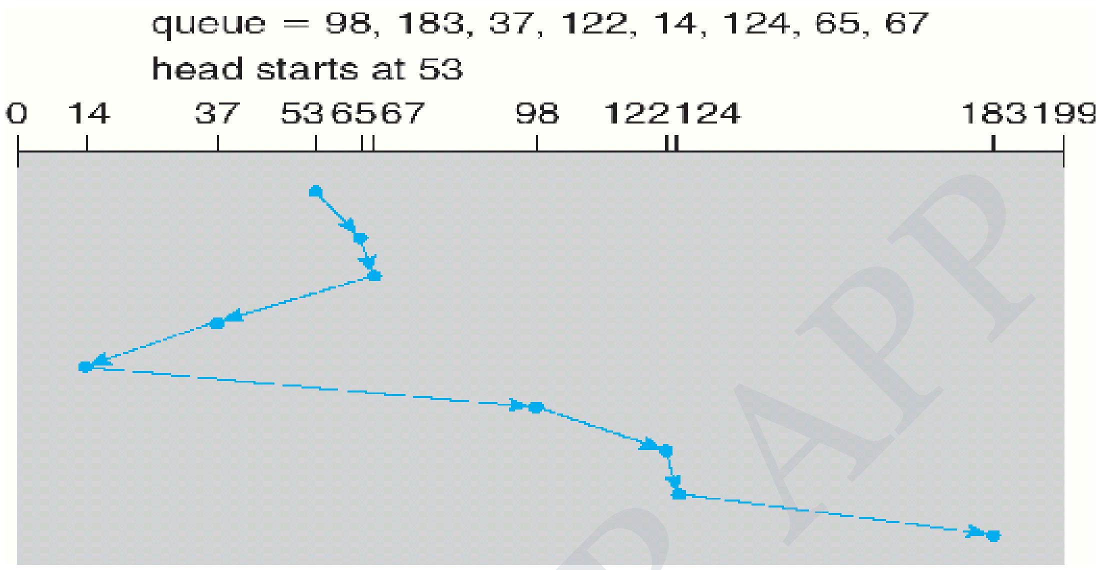
DOWNLOADED FROM STUCOR APP

4.2.1 Disk scheduling

4.2.1.1 FCFS Scheduling

4.2.1.2 SSTF (shortest-seek-time-first)Scheduling

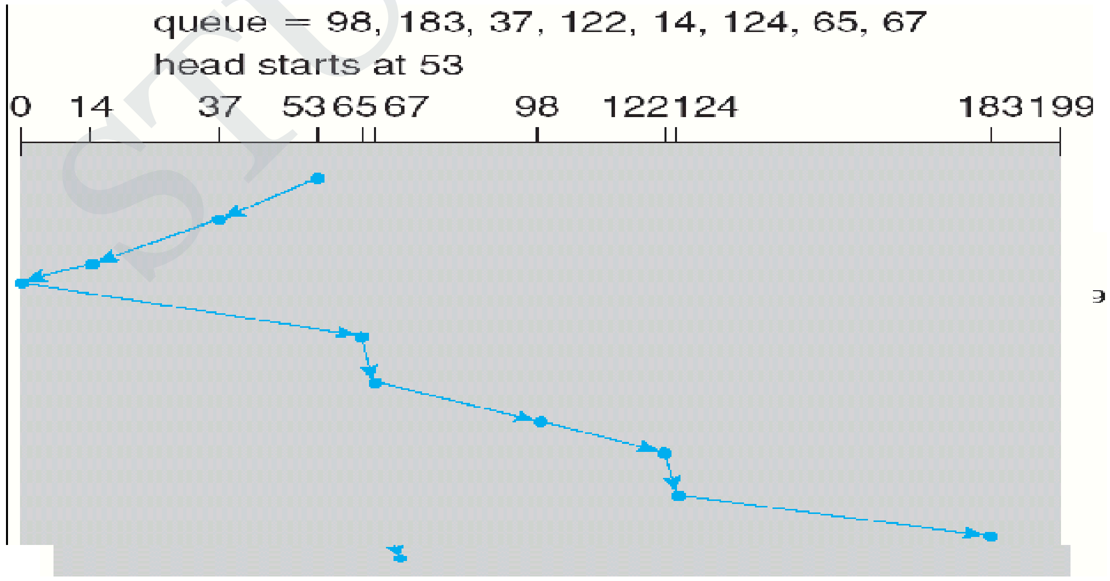
Service all the requests close to the current head position, before moving the head far away to service other requests. That is selects the request with the minimum seek time from the current



head position**.**

4.2.1.3 SCAN Scheduling

The disk head starts at one end of the disk, and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth



STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

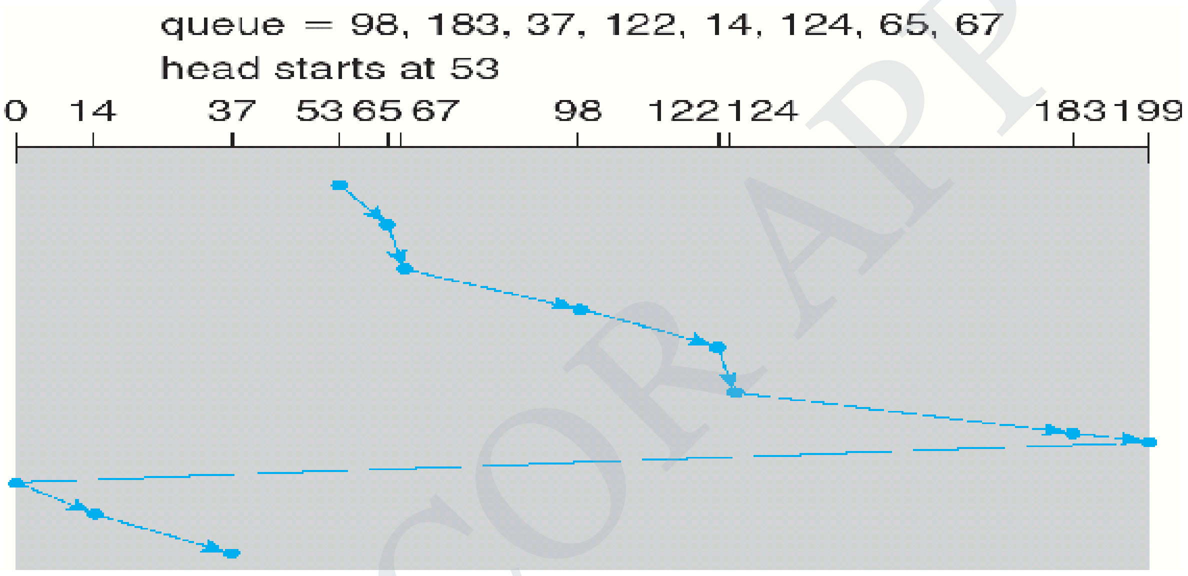
across the disk.

Elevator algorithm: Sometimes the SCAN algorithm is called as the elevator algorithm, since the disk arm behaves just like an elevator in a building, first servicing all the requests going up, and then reversing to service requests the other way.

4.2.1.4 C-SCAN Scheduling

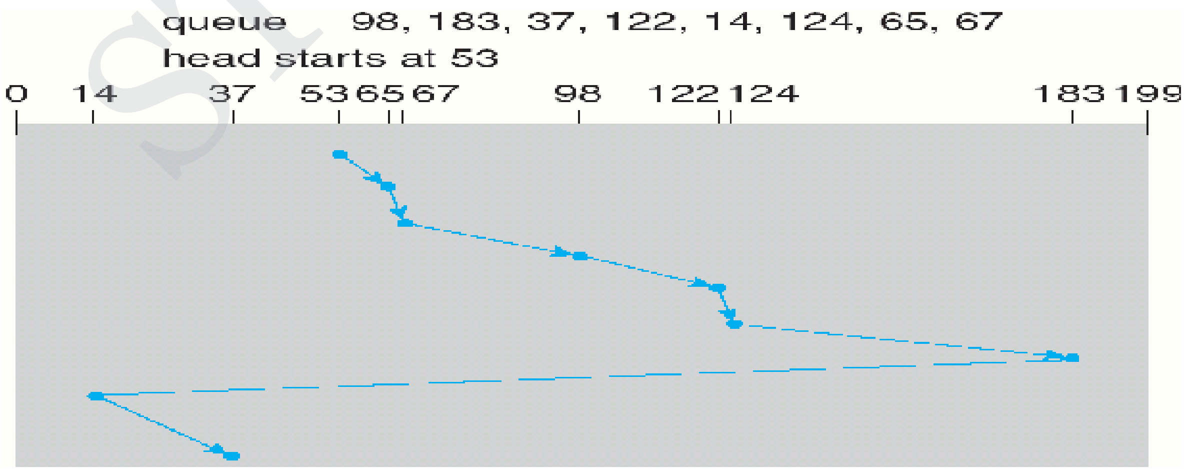
Variant of SCAN designed to provide a more uniform wait time. It moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests

on the return trip.



4.2.1.5 LOOK Scheduling

Both SCAN and C-SCAN move the disk arm across the full width of the disk. In this, the arm goes only as far as the final request in each direction. Then, it reverses direction immediately, without going all the way to the end of the disk.



STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

4.2.2 Disk Management

Disk Formatting:

Low-level formatting or physical formatting:

- ✓ Before a disk can store data, the sector is divided into various partitions. This process is called low-level formatting or physical formatting. It fills the disk with a special data structure for each sector.
- ✓ The data structure for a sector consists of
  - o Header,
  - o Data area (usually 512 bytes in size), and
  - o Trailer.
- ✓ The header and trailer contain information used by the disk controller, such as a sector number and an error-correcting code (ECC).
- ✓ This formatting enables the manufacturer to
  - o Test the disk and
  - o To initialize the mapping from logical block numbers
- ✓ To use a disk to hold files, the operating system still needs to record its own data structures on the disk. It does so in two steps.

(a) The first step is Partition the disk into one or more groups of cylinders. Among the partitions, one partition can hold a copy of the OS‘s executable code, while another holds user files.

(b) The second step is logical formatting .The operating system stores the initial file-system data structures onto the disk. These data structures may include maps of free and allocated space and an initial empty directory.

Boot Block:

- ✓ For a computer to start running-for instance, when it is powered up or rebooted-it needs to have an initial program to run. This initial progr am is called bootstrap program & it should be simple. It initializes all aspects of the system, from CPU registers to device controllers and the contents of main memory, and then starts the operating system.
- ✓ To do its job, the bootstrap program
  1. Finds the operating system kernel on disk,
  2. Loads that kernel into memory, and
  3. Jumps to an initial address to begin the operating-system execution.

Advantages:

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

1. ROM needs no initialization.
2. It is at a fixed location that the processor can start executing when powered up or reset.
3. It cannot be infected by a computer virus. Since, ROM is read only.

- ✓ The full bootstrap program is stored in a partition called the boot blocks, at a fixed location on the disk. A disk that has a boot partition is called a boot disk or system disk.
- ✓ The code in the boot ROM instructs the disk controller to read the boot blocks into memory and then starts executing that code.
- ✓ Bootstrap loader: load the entire operating system from a non-fixed location on disk, and to start the operating system running.

Bad Blocks:

- ✓ The disk with defected sector is called as bad block.
- ✓ Depending on the disk and controller in use, these blocks are handled in a variety of ways;

Method 1: "Handled manually

- If blocks go bad during normal operation, a special program must be run manually to search for the bad blocks and to lock them away as before. Data that resided on the bad blocks usually are lost.

Method 2: "sector sparing or forwarding"

- The controller maintains a list of bad blocks on the disk. Then the controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding.

- ✓ A typical bad-sector transaction might be as follows:

1. The oper ating system tries to read logical block 87.
2. The controller calculates the ECC and finds that the sector is bad.
3. It reports this finding to the operating system.
4. The next time that the system is rebooted, a special command is run to tell the controller to replace the bad sector with a spare.
5. After that, whenever the system requests logical block 87, the request is translated into the

replacement sector's address by the controller.

Method 3: "sector slipping"

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ For an example, suppose that logical block 17 becomes defective, and the first available spare follows sector 202. Then, sector slipping would remap all the sectors from 17 to 202,moving them all down one spot. That is, sector 202 would be copied into the spare, then sector 201 into 202, and then 200 into 201, and so on, until sector 18 is copied into sector 19. Slipping the sectors in this way frees up the space of sector 18, so sector 17 can be mapped to it.

4.3 FILE SYSTEM STORAGE

4.3.1 File Concept

- ✓ A file is a named collection of related information that is recorded on secondary storage.
- ✓ From a user's perspective, a file is the smallest allotment of logical secondary storage; that is, data cannot be written to secondary storage unless they are within a file.
- ✓ Examples of files:
- ✓ A text file is a sequence of characters organized into lines (and possibly pages).
- ✓ A source file is a sequence of subroutines and functions, each of which is further organized as declarations followed by executable statements. An object file is a sequence of bytes organized into blocks understandable by the system's linker. An executable file is a series of code sections that the loader can bring into memory and execute.

4.3.2 File Attributes

- **Name:** The symbolic file name is the only information kept in human readable form.
- **Identifier:** This unique tag, usually a number identifies the file within the file system. It is the non-human readable name for the file.
- **Type:** This information is needed for those systems that support different types.
- **Location:** This information is a pointer to a device and to the location of the file on that device.
- **Size:** The current size of the file (in bytes, words or blocks)and possibly the maximum allowed size are included in this attribute.
- **Protection:** Access-control information determines who can do reading, writing, executing and so on.
- **Time, date and user identification**: This information may be kept for creation, last modification and last use. These data can be useful for protection, security and usage monitoring.

4.3.3 File Operations

- ❖ Creating a file
- ❖ Writing a file

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ❖ Reading a file
- ❖ Repositioning within a file
- ❖ Deleting a file
- ❖ Truncating a file

4.3.4 File Types

File Type	Usual Extension	Function
executable	exe, com, bin, or none	Read to run machine language program
Object	obj, o	Compiled, machine language, not linked
Source code	C, cc, java, pas,asm ,a	Source code in various languages
Batch	bat, sh	Commands to the command interpreter
Text	txt, doc	Textual data, documents
word processor	wp, tex, rrf, doc	Various word-processor formats
Library	lib, a, so, dll, mpeg, mov, rm	Libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
Archive	arc, zip, tar	Related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	Binary file containing audio or A/V information

STUCOR APP

4.4 FILE SHARING AND PROTECTION

1. MULTIPLE USERS:

- ✓ When an operating system accommodates multiple users, the issues of file sharing, file naming and file protection become preeminent.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ The system either can allow user to access the file of other users by default, or it may require that a user specifically grant access to the files.
- ✓ These are the issues of access control and protection.
- ✓ To implementing sharing and protection, the system must maintain more file and directory attributes than a on a single-user system.
- ✓ The owner is the user who may change attributes, grand access, and has the most control over the file or directory.
- ✓ The group attribute of a file is used to define a subset of users who may share access to the file.
- ✓ Most systems implement owner attributes by managing a list of user names and associated user identifiers (user Ids).
- ✓ When a user logs in to the system, the authentication stage determines the appropriate user ID for the user. That user ID is associated with all of user's processes and threads. When they need to be user readable, they are translated, back to the user name via the user name list.
- ✓ Likewise, group functionality can be implemented as a system wide list of group names and group identifiers.
- ✓ Every user can be in one or more groups, depending upon operating system design decisions. The user's group Ids is also included in every associated process and thread.

2. Remote File System:

- ✓ Networks allowed communications between remote computers.
- ✓ Networking allows the sharing or resource spread within a campus or even around the world. User manually transfer files between machines via programs like ftp.
- ✓ A distributed file system (DFS) in which remote directories is visible from the local machine.
- ✓ The World Wide Web: A browser is needed to gain access to the remote file and separate operations (essentially a wrapper for ftp) are used to transfer files.

a) The client-server Model:

- Remote file systems allow a computer to a mount one or more file systems from one or more remote machines.
- A server can serve multiple clients, and a client can use multiple servers, depending on the implementation details of a given client –server facility.
- Client identification is more difficult. Clients can be specified by their network name or other identifier, such as IP address, but these can be spoofed (or imitate). An unauthorized client can spoof the server into deciding that it is authorized, and the unauthorized client could be allowed access.

STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

b) Distributed Information systems:

- Distributed information systems, also known as distributed naming service, have been devised to provide a unified access to the information needed for remote computing.
- Domain name system (DNS) provides host-name-to-network address translations for their entire Internet (including the World Wide Web). Before DNS was invented and became widespread, files containing the same information were sent via e-mail of ftp between all networked hosts.

c) Failure Modes:

- Redundant arrays of inexpensive disks (RAID) can prevent the loss of a disk from resulting in the loss of data.

Remote file system has more failure modes. By nature of the complexity of networking system and the required interactions between remote machines, many more problems can interfere with the proper operation of remote file systems.

d) Consistency Semantics:

- It is characterization of the system that specifies the semantics of multiple users accessing a shared file simultaneously.
- These semantics should specify when modifications of data by one user are observable by other users.
- The semantics are typically implemented as code with the file system.
- A series of file accesses (that is reads and writes) attempted by a user to the same file is always enclosed between the open and close operations.
- The series of access between the open and close operations is a file session.

(i) UNIX Semantics:

The UNIX file system uses the following consistency semantics:

1. Writes to an open file by a user are visible immediately to other users that have this file open at the same time.

2. One mode of sharing allows users to share the pointer of current location into the file. Thus, the advancing of the pointer by one user affects all sharing users.

(ii) Session Semantics:

The Andrew file system (AFS) uses the following consistency semantics:

1. Writes to an open file by a user are not visible immediately to other users that have the same file open simultaneously.

2. Once a file is closed, the changes made to it are visible only in sessions starting later. Already open instances of the file do not reflect this change.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

(iii) Immutable –shared File Semantics:

Once a file is declared as shared by its creator, it cannot be modified.

An immutable file has two key properties:

Its name may not be reused and its contents may not be altered.

4.6 FILE PROTECTION

4.6.1 Need for file protection.

- When information is kept in a computer system, we want to keep it safe from physical damage (reliability) and improper access (protection).
- Reliability is generally provided by duplicate copies of files. Many computers have systems programs that automatically (or though computer-operator intervention) copy disk files to tape at regular intervals (once per day or week or month) to maintain a copy should a file system be accidentally destroyed.
- File systems can be damaged by hardware problems (such as errors in reading or writing), power surges or failures, head crashes, dirt, temperature extremes, and vandalism. Files may be deleted accidentally. Bugs in the file-system software can also cause file contents to be lost.
- Protection can be provided in many ways. For a small single-user system, we might provide protection by physically removing the floppy disks and locking them in a desk drawer or file cabinet. In a multi-user system, however, other mechanisms are needed.

4.6.2 Types of Access

- Complete protection is provided by prohibiting access.
- Free access is provided with no protection.
- Both approaches are too extreme for general use.
- What is needed is controlled access.
- Protection mechanisms provide controlled access by limiting the types of file access that can be made. Access is permitted or denied depending on several factors, one of which is the type of access requested. Several different types of operations may be controlled:

1. Read: Read from the file.

2. Write: Write or rewrite the file.

3. Execute: Load the file into memory and execute it.

4. Append: Write new information at the end of the file.

5. Delete: Delete the file and free its space for possible reuse.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

6. List: List the name and attributes of the file.

4.6.3 Access Control

- Associate with each file and directory an access-control list (ACL) specifying the user name and the types of access allowed for each user.
- When a user requests access to a particular file, the operating system checks the access list associated with that file. If that user is listed for the requestedaccess, the access is allowed. Otherwise, a protection violation occurs and the user job is denied access to the file.
- This technique has two undesirable consequences:
- Constructing such a list may be a tedious and unrewarding task, especially if we do not know in advance the list of users in the system.
- The directory entry, previously of fixed size, now needs to be of variable size, resulting in more complicated space management.
- To condense the length of the access control list, many systems recognize three classifications of users in connection with each file:

  Owner: The user who created the file is the owner.

  Group: A set of users who are sharing the file and need similar access \is a group, or work group.

  Universe: All other users in the system constitute the universe.

STUCOR APP

4.8 FILE SYSTEM STRUCTURE

- ✓ All disk I/O is performed in units of one block (physical record) size which will exactly match the length of the desired logical record.
- ✓ Logical records may even vary in length. Packing a number of logical records into physical blocks is a common solution to this problem.
- ✓ For example, the UNIX operating system defines all files to be simply a tream of bytes. Each byte is individually addressable by its offset from the beginning (or end) of the file. In this case, the logical records are 1 byte. The file system automatically packs and unpacks bytes into physical disk blocks – say, 512 bytes per block – as necessary.
- ✓ The logical record size, physical block size, and packing technique determine how many logical records are in each physical block. The packing can be done either by the user's application program or by the operating system.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Access Methods

1. Sequential Access

❖ The simplest access method is sequential access. Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editors and compilers usually access files in this fashion.

❖ The bulk of the operations on a file is reads and writes. A read operation reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location. Similarly, a write appends to the end of the file and advances to the end of the newly written material (the new end of file). Such a file can be reset to the beginning and, on some systems, a program may be able to skip forward or back ward n records, for some integer n-perhaps only for n=1. Sequential access is based on a tape model of a file, and works as well on sequential-access devices as it does on random – access ones.

2. Direct Access

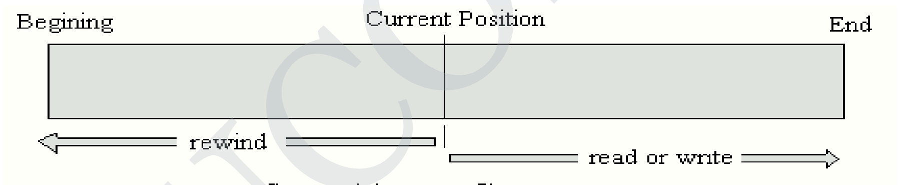


Fig 4.10 Sequential-access file

❖ Another method is direct access (or relative access). A file is made up of fixed length logical records that allow programs to read and write records rapidly in no particular order. The direct- access methods is based on a disk model of a file, since disks allow random access to any file block.

❖ For direct access, the file is viewed as a numbered sequence of blocks or records. A direct-access file allows arbitrary blocks to be read or written. Thus, we may read block 14, then read block 53, and then write block7. There are no restrictions on the order of reading or writing for a direct-access file.

❖ Direct – access files are of great use for immediate access to large amounts of information. Database is often of this type. When a query concerning a particular subject arrives, we compute which block contains the answer, and then read that block directly to provide the desired information.

STUCOR APP

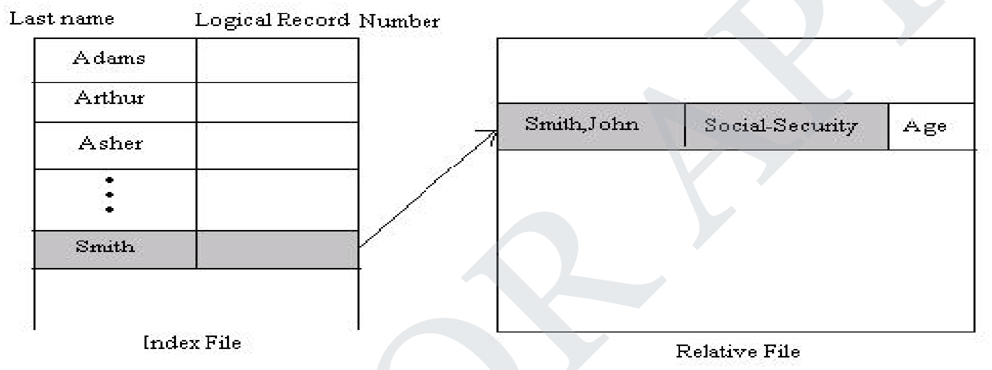
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ❖ As a simple example, on an air line – reservation system, we might store all the information about a particular flight (for example, flight 713) in the block identified by the flight number.
- ❖ Thus, the number of available seats for flight 713 is stored in block 713 of the reservation file. To store information about a larger set, such as people, we might compute a hash function on the people's names, or search a small in- memory index to determine a block to read and search.

3. Other Access methods

- ❖ Other access methods can be built on top of a direct – access method these methods generally involve the construction of an index for the file. The index like an index in the back of a book contains pointers to the various blocks in find a record in the file. We first search the index, and then use the pointer to access the file directly and the find the desired record.



- ❖ With large files, the index file itself may become too large to be kept in memory. One solution is to create an index for the index file. The primary index file would contain pointers to secondary index tiles, which would point to the actual data items.

4.9 DIRECTORY STRUCTURE

There are five directory structures. They are

1. Single-level directory
2. Two-level directory
3. Tree-Structured directory
4. Acyclic Graph directory
5. General Graph directory

1. Single – Level Directory

- ❖ The simplest directory structure is the single- level directory.
- ❖ All files are contained in the same directory.

STUCOR APP

DOWNLOADED FROM STUCOR APP

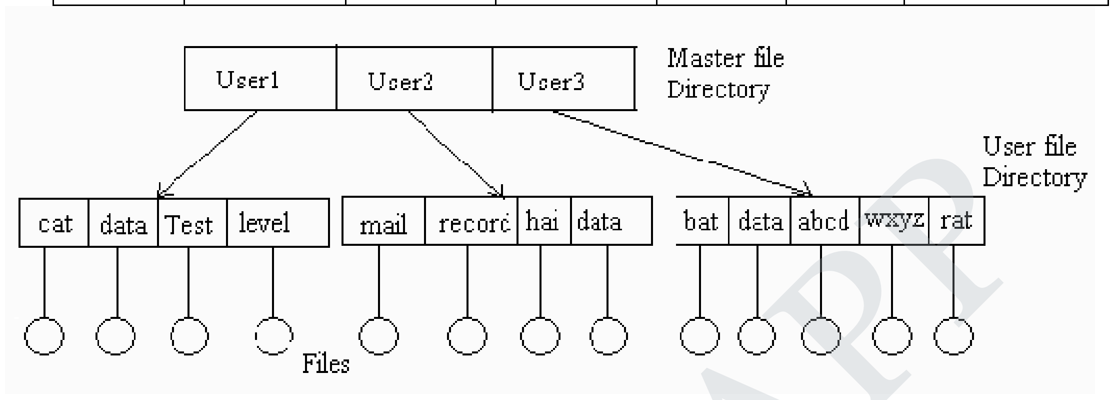
DOWNLOADED FROM STUCOR APP

Disadvantage:

❖ When the number of files increases or when the system has more than one user, since all files are in the same directory, they must have unique names.

Directory

Abc	Cde	Fgh	Jkl	Mno	Pre	test



2. Two – Level Directory

❖ In the two level directory structures, each user has her own user file directory (UFD).

❖ When a user job starts or a user logs in, the system's master file directory (MFD) is searched. The MFD is indexed by user name or account number, and each entry points to the UFD for that user.

❖ When a user refers to a particular file, only his own UFD is searched.

❖ Thus, different users may have files with the same name.

❖ Although the two – level directory structure solves the name-collision problem

Disadvantage:

❖ Users cannot create their own sub-directories.

3. Tree – Structured Directory

❖ A tree is the most common directory structure.

❖ The tree has a root directory. Every file in the system has a unique path name.

❖ A path name is the path from the root, through all the subdirectories to a specified file.

❖ A directory (or sub directory) contains a set of files or sub directories.

STUCOR APP

DOWNLOADED FROM STUCOR APP

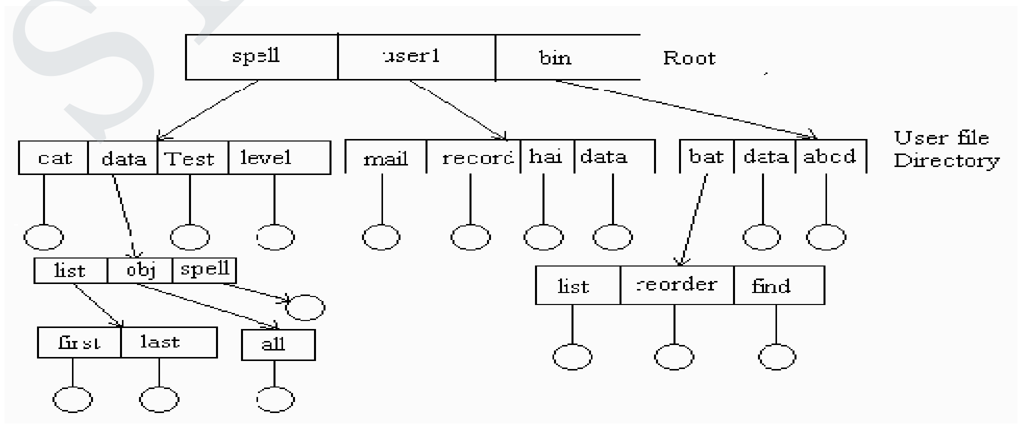
DOWNLOADED FROM STUCOR APP

- A directory is simply another file. But it is treated in a special way.
- All directories have the same internal format.
- One bit in each directory entry defines the entry as a file (0) or as a subdirectory (1).
- Special system calls are used to create and delete directories.
- Path names can be of two types: absolute path names or relative path names.
- An absolute path name begins at the root and follows a path down to the specified file, giving the directory names on the path.
- A relative path name defines a path from the current directory.

4. Acyclic Graph Directory.

- An acyclic graph is a graph with no cycles.
- To implement shared files and subdirectories this directory structure is used.
- An acyclic – graph directory structure is more flexible than is a simple tree structure, but it is also more complex. In a system where sharing is implemented by symbolic link, this situation is somewhat easier to handle. The deletion of a link does not need to affect the original file; only the link is removed.
- Another approach to deletion is to preserve the file until all references to it are deleted. To implement this approach, we must have some mechanism for determining that the last reference to the file has been deleted.

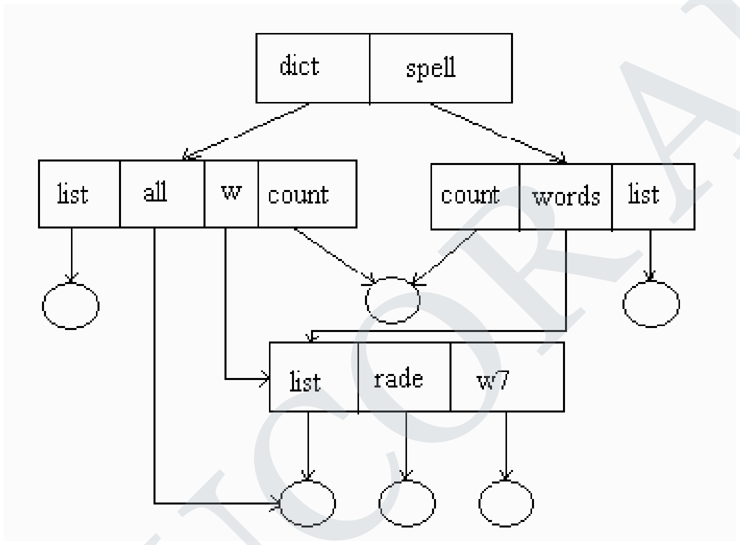
STUCOR APP



STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP



4.10 ALLOCATION METHODS

- ✓ The main problem is how to allocate space to these files so that disk space is utilized effectively and files can be accessed quickly .
- ✓ There are there major methods of allocating disk space:

  1. Contiguous Allocation
  2. Linked Allocation
  3. Indexed Allocation

1. Contiguous Allocation

- ✓ The contiguous – allocation method requires each file to occupy a set of contiguous blocks on the disk.

STUCOR APP

DOWNLOADED FROM STUCOR APP

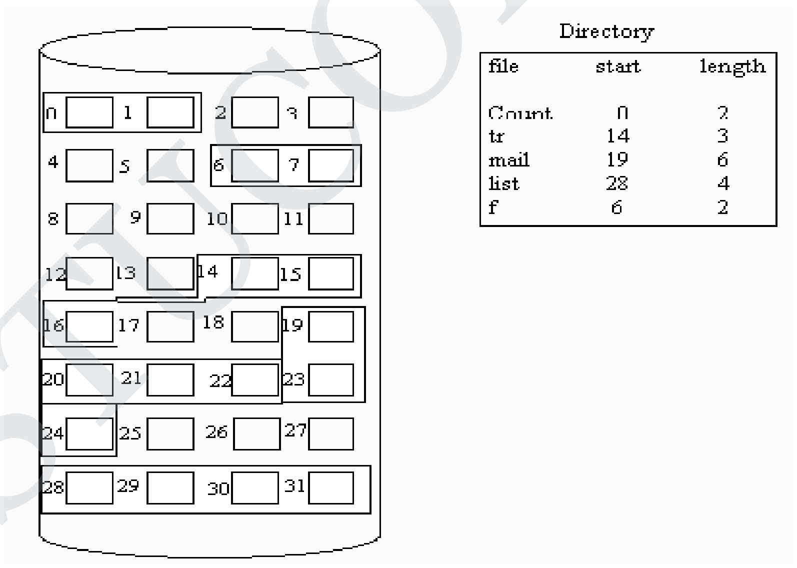
DOWNLOADED FROM STUCOR APP

- ✓ Contiguous allocation of a file is defined by the disk address and length (in block 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 for each file indicates the address of the starting block and the length of the area allocated for this file.

STUCOR APP

Disadvantages:

1. Finding space for a new file.



Directory

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

- The contiguous disk space-allocation problem suffer from the problem of external fragmentation. As file are allocated and deleted, the free disk space is broken into chunks. It becomes a problem when the largest contiguous chunk is insufficient for a request; storage is fragmented into a number of holes, no one of which is large enough to store the data.

2. Determining how much space is needed for a file.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- When the file is created, the total amount of space it will need must be found an allocated how does the creator know the size of the file to be created?
- If we allocate too little space to a file, we may find that file cannot be extended. The other possibility is to find a larger hole, copy the contents of the file to the new space, and release the previous space. This series of actions may be repeated as long as space exists, although it can be time –consuming. However, in this case, the user never needs to be informed explicitly about what is happening ; the system continues despite the problem, although more and more slowly.
- Even if the total amount of space needed for a file is known in advance pre-allocation may be inefficient.
- A file that grows slowly over a long period (months or years) must be allocated enough space for its final size, even though much of that space may be unused for a long time the file, therefore has a large amount of internal fragmentation.

To overcome these disadvantages:

- Use a modified contiguous allocation scheme, in which a contiguous chunk of space called as an extent is allocated initially and then, when that amount is not large enough another chunk of contiguous space an extent is added to the initial allocation.
- Internal fragmentation can still be a problem if the extents are too large, and external fragmentation can be a problem as extents of varying sizes are allocated and deallocated.

2. Linked Allocation

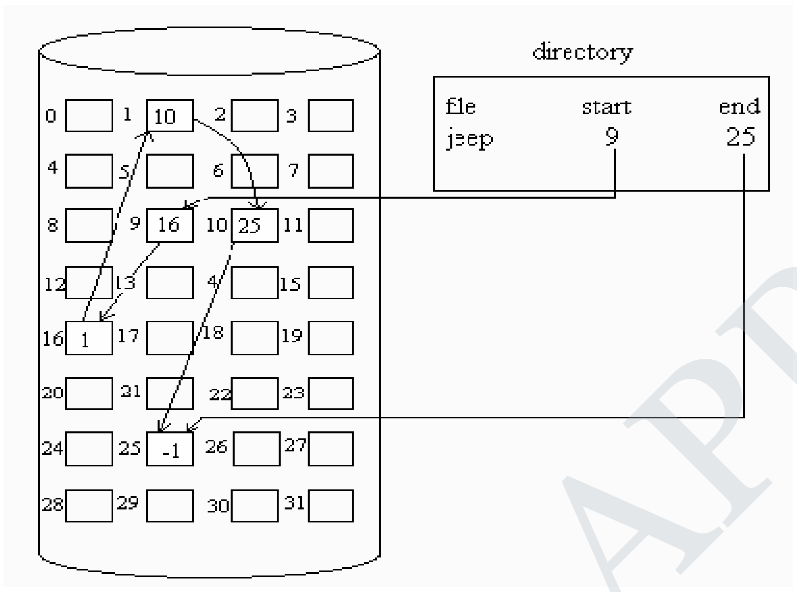
- ✓ Linked allocation solves all problems of contiguous allocation.
- ✓ With linked allocation, each file is a linked list of disk blocks, the disk blocks may be scattered any where on the disk.
- ✓ The directory contains a pointer to the first and last blocks of the file. For example, a file of five blocks might start at block 9, continue at block 16, then block 1, block 10, and finally bock 25.
- ✓ Each block contains a pointer to the next block. These pointers are not made available to the user.
- ✓ There is no external fragmentation with linked allocation, and any free block on the free space list can be used to satisfy a request.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

✓ The size of a file does not need to the declared when that file is created. A file can continue to grow as long as free blocks are available consequently, it is never necessary to compacts disk space.



Disadvantages:

1. Used effectively only for sequential access files.

- To find the ith block of a file, we must start at the beginning of that file, and follow the pointers until we get to the ith block. Each aces to a pointer requires a disk read, and sometimes a disk seek consequently, it is inefficient to support a direct- access capability for linked allocation files.

2. Space required for the pointers

- If a pointer requires 4 bytes out of a 512-byte block, then 0.78 percent of the disk is being used for pointers, rather than for information.
- Solution to this problem is to collect blocks into multiples, called clusters, and to allocate the clusters rather than blocks. For instance, the file system may define a clusters as 4 blocks, and operate on the disk in only cluster units.

3. Reliability

- Since the files are linked together by pointers scattered all over the disk hardware failure might result in picking up the wrong pointer. This error could result in linking into the free- space list or into another file. Partial solution are to use doubly linked lists or to store the file names in a relative block number in each block; however, these schemes require even more over head for each file.

File Allocation Table (FAT)

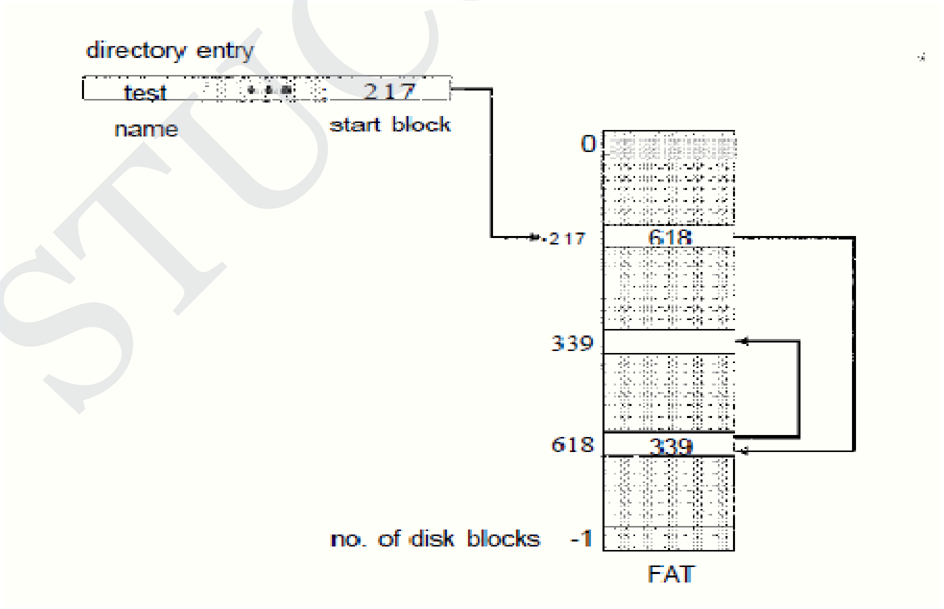
STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ❖ An important variation on the linked allocation method is the use of a file allocation table(FAT).
- ❖ This simple but efficient method of disk- space allocation is used by the MS-DOS and OS/2 operating systems.
- ❖ A section of disk at beginning of each partition is set aside to contain the table.
- ❖ The table has entry for each disk block, and is indexed by block number.
- ❖ The FAT is much as is a linked list.
- ❖ The directory entry contains the block number the first block of the file.
- ❖ The table entry indexed by that block number contains the block number of the next block in the file.
- ❖ This chain continues until the last block which has a special end – of – file value as the table entry.
- ❖ Unused blocks are indicated by a 0 table value.
- ❖ Allocating a new block file is a simple matter of finding the first 0 – valued table entry, and replacing the previous end of file value with the address of the new block.
- ❖ The 0 is replaced with the end – of – file value, an illustrative example is the FAT structure for a file consisting of disk blocks 217,618, and 339.

STUCOR APP

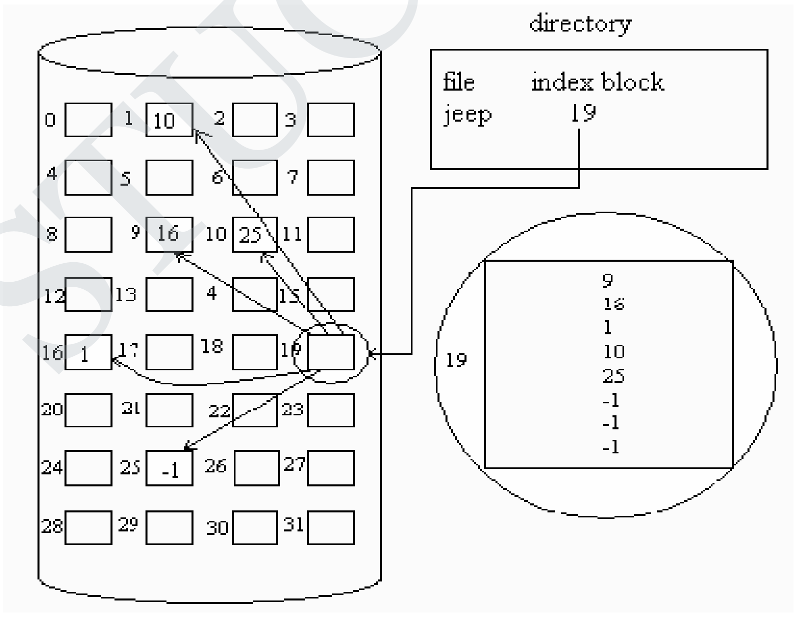


DOWNLOADED FROM STUCOR APP
STUCOR APP

DOWNLOADED FROM STUCOR APP

3. Indexed Allocation

- ✓ Linked allocation solves the external – fragmentation and size- declaration problems of contiguous allocation.
- ✓ Linked allocation cannot support efficient direct access, since the pointers to the blocks are scattered with the blocks themselves all over the disk and need to be retrieved in order.
- ✓ Indexed allocation solves this problem by bringing all the pointers together into one location: the index block.
- ✓ Each file has its own index block, which is an array of disk – block addresses.
- ✓ The ith entry in the index block points to the ith block of the file.
- ✓ The directory contains the address of the index block .
- ✓ To read the ith block, we use the pointer in the ith index – block entry to find and read the desired block this scheme is similar to the paging scheme .
- ✓ When the file is created, all pointers in the pointers in the index block are set to nil. when the ith block is first written, a block is obtained from the free space manager, and its address is put in the ith index – block entry.



- ✓ Indexed allocation supports direct access, without suffering from external fragmentation , because any free block on the disk may satisfy a request for more space.

STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Disadvantages

1.Pointer Overhead

Indexed allocation does suffer from wasted space. The pointer over head of the index block is generally greater than the pointer over head of linked allocation.

2. Size of Index block

If the index block is too small, however, it will not be able to hold enough pointers for a large file, and a mechanism will have to be available to deal with this issue:

Linked Scheme: An index block is normally one disk block. Thus, it can be read and written directly by itself. To allow for large files, we may link together several index blocks.

Multilevel index: A variant of the linked representation is to use a first level index block to point to a set of second – level index blocks.

Combined scheme:

- Another alternative, used in the UFS, is to keep the first, say, 15 pointers of the index block in the file's inode.
- The first 12 of these pointers point to direct blocks; that is for small (no more than 12 blocks) files do not need a separate index block
- The next pointer is the address of a single indirect block.
- The single indirect block is an index block, containing not data, but rather the addresses of blocks that do contain data.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- Then there is a double indirect block pointer, which contains the address of a block that contain pointers to the actual data blocks. The last pointer would contain pointers to the actual data blocks.
- The last pointer would contain the address of a triple indirect block.

4.11 FREE SPACE MANAGEMENT

- ✓ Since disk space is limited, we need to reuse the space from deleted files for new files, if possible.
- ✓ To keep track of free disk space, the system maintains a free-space list.
- ✓ The free-space list records all free disk blocks – those not allocated to some file or directory.
- ✓ To create a file, we search the free-space list for the required amount of space, and allocate that space to the new file.
- ✓ This space is then removed from the free-space list.
- ✓ When a file is deleted, its disk space is added to the free-space list.

1. Bit Vector

- ❖ The free-space list is implemented as a bit map or bit vector.
- ❖ Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.
- ❖ For example, consider a disk where block
- ❖ 2,3,4,5,8,9,10,11,12,13,17,18,25,26 and 27 are free, and the rest of the block are allocated. The free space bit map would be
- ❖ 001111001111110001100000011100000 …
- ❖ The main advantage of this approach is its relatively simplicity and efficiency in finding the first free block, or n consecutive free blocks on the disk.

2. Linked List

- ❖ Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.
- ❖ This first block contains a pointer to the next free disk block, and so on. In our example, we would keep a pointer to block 2, as the first free block. Block 2 would contain a pointer to block 3, which would point to block 4, which would point to block 5, which would point to block 8, and so on.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

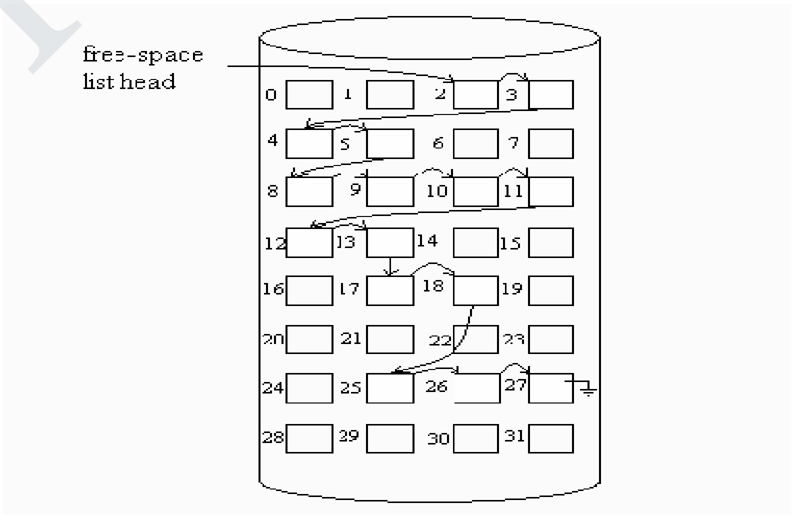
- However, this scheme is not efficient; to traverse the list, we must read each block, which requires substantial I/O time. The FAT method incorporates free-block accounting data structure. No separate method is needed.

3. Grouping

- A modification of the free-list approach is to store the addresses of n free blocks in the first free block.
- The first n-1 of these blocks are actually free.
- The last block contains the addresses of another n free blocks, and so on.
- The importance of this implementation is that the addresses of a large number of free blocks can be found quickly.

4. Counting

- We can keep the address of the first free block and the number n of free contiguous blocks that follow the first block.
- Each entry in the free-space list then consists of a disk address and a count.
- Although each entry requires more space than would a simple disk address, the



STUCOR APP

DOWNLOADED FROM STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

overall list will be shorter, as long as the count is generally greater than 1.

Recovery

Files and directories are kept both in main memory and on disk, and care must be taken to ensure that system failure does not result in loss of data or in data inconsistency.

1. Consistency Checking

- The directory information in main memory is generally more up to date than is the corresponding information on the disk, because cached directory information is not necessarily written to disk as soon as the update takes place.
- Frequently, a special program is run at reboot time to check for and correct disk inconsistencies.
- The consistency checker—a systems program such as chkdsk in MS-DOS—compares the data in the directory structure with the data blocks on disk and tries to fix any inconsistencies it finds. The allocation and free-space-management algorithms dictate what types of problems the checker can find and how successful it will be in fixing them.

2. Backup and Restore

- Magnetic disks sometimes fail, and care must be taken to ensure that the data lost in such a failure are not lost forever. To this end, system programs can be used to back up data from disk to another storage device, such as a floppy disk, magnetic tape, optical disk, or other hard disk.
- Recovery from the loss of an individual file, or of an entire disk, may then be a matter of restoring the data from backup.

A typical backup schedule may then be as follows:

- Day 1: Copy to a backup medium all files from the disk. This is called a **full backup.**
- Day 2: Copy to another medium all files changed since day 1. This is an **incremental backup.**
- Day 3: Copy to another medium all files changed since day 2.
- Day N: Copy to another medium all files changed since day N— 1. Then go back to Day 1.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

STUCOR APP

UNIT –V CASE STUDY

Linux System- Basic Concepts; System Administration-Requirements for Linux System Administrator, Setting up a LINUX Multifunction Server, Domain Name System, Setting Up Local Network Services; Virtualization- Basic Concepts, Setting Up Xen, VMware on Linux Host and Adding Guest OS.

5.1 LINUX SYSTEM

5.1.1 Basic Concepts

- ✓ Linux looks and feels much like any other UNIX system; indeed, UNIX compatibility has been a major design goal of the Linux project. However, Linux is much younger than most UNIX systems. Its development began in1991, when a Finnish university student, Linus Torvalds, began developing a small but self-contained kernel for the 80386 processor, the first true 32-bitprocessor in Intel's range of PC-compatible CPUs. of arbitrary files (but only read-only memory mapping was implemented in 1.0).
- ✓ A range of extra hardware support was included in this release. Although still restricted to the Intel PC platform, hardware support had grown to include floppy-disk and CD-ROM devices, as well as sound cards, a range of mice, and international keyboards. Floating-point emulation was provided in the kernel for 80386 users who had no 80387

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

math coprocessor. System V UNIX-style interprocess communication (IPC), including shared memory, semaphores, and message queues, was implemented.

- ✓ At this point, development started on the 1.1 kernel stream, but numerous bug-fix patches were released subsequently for 1.0. A pattern was adopted as the standard numbering convention for Linux kernels. Kernels with an odd minor-version number, such as 1.1 or 2.5, are development kernels; even numbered minor-version numbers are stable production kernels. Updates for the stable kernels are intended only as remedial versions, whereas the development kernels may include newer and relatively untested functionality.

- ✓ As we will see, this pattern remained in effect until version 3.was given a major version-number increment because of two major new capabilities: support for multiple architectures, including a 64-bit native Alpha port, and symmetric multiprocessing (SMP) support. Additionally, the memory management code was substantially improved to provide a unified cache for file-system data independent of the caching of block devices.

- ✓ As a result of this change, the kernel offered greatly increased file-system and virtual memory performance. For the first time, file-system caching was extended to networked file systems, and writable memory-mapped regions were also supported. Other major improvements included the addition of internal kernel threads, a mechanism exposing dependencies between loadable modules, support for the automatic loading of modules on demand, file-system quotas, and POSIX-compatible real-time process-scheduling classes.

- ✓ Improvements continued with the release of Linux 2.2 in 1999. A port to Ultra SPARC systems was added. Networking was enhanced with more flexible firewalling, improved routing and traffic management, and support for TCP large window and selective acknowledgement. Acorn, Apple, and NT disks could now be read, and NFS was enhanced with a new kernel-mode NFS daemon. Signal handling, interrupts, and some I/O were locked at a finer level than before to improve symmetric multiprocessor (SMP) performance.

5.1.2 The Linux System

- ✓ Linux kernel is composed entirely of code written from scratch specifically for the Linux project, much of the supporting software that makes up the Linux system is not exclusive to Linux but is common to a number of UNIX-like operating systems. In particular, Linux uses many tools developed as part of Berkeley's BSD operating system, MIT's X Window System, and the Free Software Foundation's GNU project.

- ✓ This sharing of tools has worked in both directions. The main system libraries of Linux were originated by the GNU project, but the Linux community greatly improved the libraries by addressing omissions, inefficiencies, and bugs. Other components, such as the GNU C compiler (gcc), were already of sufficiently high quality to be used directly in Linux. The network administration tools under Linux were derived from code first developed for 4.3 BSD, but more recent BSD derivatives, such as FreeBSD, have borrowed code from Linux in return. Examples of this sharing include the Intel floating-point-emulation math library and the PC sound-hardware device drivers.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- ✓ The Linux system as a whole is maintained by a loose network of developers collaborating over the Internet, with small groups or individuals having responsibility for maintaining the integrity of specific components.
- ✓ A small number of public Internet file-transfer-protocol (FTP) archive sites act as de facto standard repositories for these components. The File System Hierarchy Standard document is also maintained by the Linux community as a means of ensuring compatibility across the various system components.
- ✓ This standard specifies the overall layout of a standard Linux file system; it determines under which directory names configuration files, libraries, system binaries, and run-time data files should be stored.

5.1.3 Linux Distributions

- ✓ In theory, anybody can install a Linux system by fetching the latest revisions of the necessary system components from the FTP sites and compiling them. In Linux's early days, this is precisely what a Linux user had to do. As Linux has matured, however, various individuals and groups have attempted to make this job less painful by providing standard, precompiled sets of packages for easy installation.
- ✓ These collections, or distributions, include much more than just the basic Linux system. They typically include extra system-installation and management utilities, as well as precompiled and ready-to-install packages of many of the common UNIX tools, such as news servers, web browsers, text-processing and editing tools, and even games.
- ✓ The first distributions managed these packages by simply providing a means of unpacking all the files into the appropriate places. One of the important contributions of modern distributions, however, is advanced package management. Today's Linux distributions include a package-tracking database that allows packages to be installed, upgraded, or removed painlessly.

STUCOR APP

5.1.4 Linux Licensing

- ✓ The Linux kernel is distributed under version 2.0 of the GNU General Public License (GPL), the terms of which are set out by the Free Software Foundation. Linux is not public-domain software. Public domain implies that the authors have waived copyright rights in the software, but copyright rights in Linux code are still held by the code's various authors. Linux is free software, however, in the sense that people can copy it, modify it, use it in any manner they want, and give away (or sell) their own copies.
- ✓ The main implication of Linux's licensing terms is that nobody using Linux, or creating a derivative of Linux (a legitimate exercise), can distribute the derivative without including the source code. Software released under the GPL cannot be redistributed as a binary-only product.
- ✓ If you release software that includes any components covered by the GPL, then, under the GPL, you must make source code available alongside any binary distributions. (This restriction does not prohibit making—or even selling—binary software distributions, as long as anybody who receives binaries is also given the opportunity to get the originating source code for a reasonable distribution charge.)

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

5.2 SYSTEM ADMINISTRATION

- ✓ In its overall design, Linux resembles other traditional, nonmicrokernel UNIX implementations. It is a multiuser, preemptively multitasking system with a full set of UNIX-compatible tools. Linux's file system adheres to traditional UNIX semantics, and the standard UNIX networking model is fully implemented. The internal details of Linux's design have been influenced heavily by the history of this operating system's development.

- ✓ Although Linux runs on a wide variety of platforms, it was originally developed exclusively on PC architecture. A great deal of that early development was carried out by individual enthusiasts rather than by well-funded development or research facilities, so fromthe start Linux attempted to squeeze as much functionality as possible from limited resources. Today, Linux can run happily on a multiprocessor machine with many gigabytes of main memory and many terabytes of disk space, but it is still capable of operating usefully in under 16 MB of RAM.

5.2.1 Components of a Linux System

- ✓ The Linux system is composed of three main bodies of code, in line with most traditional UNIX implementations:

  1. Kernel. The kernel is responsible for maintaining all the important abstractions of the operating system, including such things as virtualmemory and processes.

  2. System libraries. The system libraries define a standard set of functions through which applications can interact with the kernel. These functions implement much of the operating-system functionality that does not need the full privileges of kernel code. The most important system library is the C library, known as libc. In addition to providing the standard C library, libc implements the user mode side of the Linux system call interface, as well as other critical system-level interfaces.

  3. System utilities. The system utilities are programs that perform individual, specialized management tasks. Some system utilities are invoked just once to initialize and configure some aspect of the system. Others —known as daemons in UNIX terminology—run permanently, handling such tasks as responding to incoming network connections, accepting logon requests from terminals, and updating log files.

5.2.2 Kernel Modules

- ✓ The Linux kernel has the ability to load and unload arbitrary sections of kernel code on demand. These loadable kernel modules run in privileged kernel mode and as a consequence have full access to all the hardware capabilities of the machine on which they run. In theory, there is no restriction on what a kernel module is allowed to do. Among other things, a kernel module can implement a device driver, a file system, or a networking protocol.

- ✓ Kernel modules are convenient for several reasons. Linux's source code is free, so anybody wanting to write kernel code is able to compile a modified kernel and to reboot into that new functionality. However, recompiling, relinking, and reloading the entire kernel is a cumbersome cycle to undertake when you are developing a new driver. If you use kernel modules, you do not have to make a new kernel to test a new driver—the

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

driver can be compiled on its own and loaded into the already running kernel. Of course, once a new driver is written, it can be distributed as a module so that other users can benefit from it without having to rebuild their kernels.

✓ The module support under Linux has four components:

1. The **module-management system** allows modules to be loaded into memory and to communicate with the rest of the kernel.

2. The **module loader and unloader**, which are user-mode utilities, work with the module-management system to load a module into memory.

3. The **driver-registration system** allows modules to tell the rest of the kernel that a new driver has become available.

4. A **conflict-resolution mechanism** allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver.

1. Module Management

❖ Loading a module requires more than just loading its binary contents into kernel memory. The system must also make sure that any references the correct locations in the kernel's address space. Linux deals with this reference updating by splitting the job of module loading into two separate sections: the management of sections of module code in kernel memory and the handling of symbols that modules are allowed to reference.

❖ Linux maintains an internal symbol table in the kernel. This symbol table does not contain the full set of symbols defined in the kernel during the latter's compilation; rather, a symbol must be explicitly exported. The set of exported symbols constitutes a well-defined interface by which a module can interact with the kernel.

2. Driver Registration

❖ Once a module is loaded, it remains no more than an isolated region of memory until it lets the rest of the kernel know what new functionality it provides. The kernel maintains dynamic tables of all known drivers and provides a set of routines to allow drivers to be added to or removed from these tables at any time. The kernel makes sure that it calls a module's startup routine when that module is loaded and calls the module's cleanup routine before that module is unloaded. These routines are responsible for registering the module's functionality.

❖ A module may register many types of functionality; it is not limited to only one type. For example, a device driver might want to register two separate mechanisms for accessing the device. Registration tables include, among others, the following items:

• **Device drivers.** These drivers include character devices (such as printers, terminals, and mice), block devices (including all disk drives), and network interface devices.

• **File systems.** The file system may be anything that implements Linux's virtual file system calling routines. It might implement a format for storing files on a disk, but it might equally well be a network file system, such as NFS, or a virtual

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

file system whose contents are generated on demand, such as Linux's /proc file system.

• **Network protocols.** A module may implement an entire networking protocol, such as TCP or simply a new set of packet-filtering rules for a network firewall.

• **Binary format.** This format specifies a way of recognizing, loading, and executing a new type of executable file.

3. Conflict Resolution

❖ Commercial UNIX implementations are usually sold to run on a vendor's own hardware. One advantage of a single-supplier solution is that the software vendor has a good idea about what hardware configurations are possible. PC hardware, however, comes in a vast number of configurations, with large numbers of possible drivers for devices such as network cards and video display adapters. The problem of managing the hardware configuration becomes more severe when modular device drivers are supported, since the currently active set of devices becomes dynamically variable.

❖ Linux provides a central conflict-resolution mechanism to help arbitrate access to certain hardware resources. Its aims are as follows:

• To prevent modules from clashing over access to hardware resources

• To prevent autoprobes—device-driver probes that auto-detect device configuration—from interfering with existing device drivers

• To resolve conflicts among multiple drivers trying to access the same hardware—as, for example, when both the parallel printer driver and the parallel line IP (PLIP) network driver try to talk to the parallel port.

5.3 REQUIREMENTS FOR LINUX SSYTEM ADMINISTRATOR

5.3.1 Hardware-Abstraction Layer

❖ The HAL is the layer of software that hides hardware chipset differences from upper levels of the operating system. The HAL exports a virtual hardware drivers. Only a single version of each device driver is required for each CPU architecture, no matter what support chips might be present. Device drivers map devices and access them directly, but the chipset-specific details of mapping memory, configuring I/O buses, setting up DMA, and coping with motherboard-specific facilities are all provided by the HAL interfaces.

5.3.2 Kernel

❖ The kernel layer ofWindows has four main responsibilities: thread scheduling, low-level processor synchronization, interrupt and exception handling, and switching between user mode and kernel mode. The kernel is implemented in the C language, using assembly language only where absolutely necessary to interface with the lowest level of the hardware architecture.

5.3.3 Kernel Dispatcher

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

❖ The kernel dispatcher provides the foundation for the executive and the subsystems. Most of the dispatcher is never paged out of memory, and its execution is never preempted. Its main responsibilities are thread scheduling and context switching, implementation of synchronization primitives, timer management, software interrupts (asynchronous and deferred procedure calls), and exception dispatching.

5.3.4 Threads and Scheduling

❖ Like many other modern operating systems, Windows uses processes and threads for executable code. Each process has one or more threads, and each thread has its own scheduling state, including actual priority, processor affinity, and CPU usage information.

❖ There are six possible thread states: **ready, standby, running, waiting, transition, and terminated**. Ready indicates that the thread is waiting to run. The highest-priority ready thread is moved to the standby state, which means it is the next thread to run. In a multiprocessor system, each processor keeps one thread in a standby state. A thread is running when it is executing on a processor. It runs until it is preempted by a higher-priority thread, until it terminates, until its allotted execution time (quantum) ends, or until it waits on a dispatcher object, such as an event signaling I/O completion. A thread is in the waiting state when it is waiting for a dispatcher object to be signaled. A thread is in the transition state while it waits for resources necessary for execution; for example, it may be waiting for its kernel stack to be swapped in from disk. A thread enters the terminated state when it finishes execution.

5.3.5 Implementation of Synchronization Primitives

❖ Key operating-system data structures are managed as objects using common facilities for allocation, reference counting, and security. Dispatcher objects control dispatching and synchronization in the system. Examples of these objects include the following:

• The event object is used to record an event occurrence and to synchronize this occurrence with some action. Notification events signal all waiting threads, and synchronization events signal a single waiting thread.

• The mutant provides kernel-mode or user-mode mutual exclusion associated with the notion of ownership.

• The mutex, available only in kernel mode, provides deadlock-free mutual exclusion.

• The semaphore object acts as a counter or gate to control the number of threads that access a resource.

• The thread object is the entity that is scheduled by the kernel dispatcher. It is associated with a process object, which encapsulates a virtual address space. The thread object is signaled when the thread exits, and the process object, when the process exits.

• The timer object is used to keep track of time and to signal timeouts when operations take too long and need to be interrupted or when a periodic activity needs to be scheduled.

5.4 SETTING UP A LINUX MULTIFUNCTION SERVER

Follow the steps below to avoid any complications during the hardware installation:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

1. Confirm that the printer you will use to connect to the DPR-1020 is operating correctly.

2. When you have confirmed that the printer is operating correctly, switch its power OFF.

3. Confirm that your network is operating normally.

4. Using a CAT 5 Ethernet cable, connect the DPR-1020 Ethernet Port (labelled LAN) to the network.

5. While the printer is turned OFF, connect the USB printer cable to the printer and then to the USB port on the Print Server.

6. Switch on the printer.

7. Insert the power adapter's output plug into the DC 5V power socket on the rear panel of the Print Server.

8. Connect the other end of the power adapter into a power outlet. This will supply power to the Print Server. The blue LED on the Print Server's front panel should turn on and the Print Server's self-test will proceed.

Power ON Self-Test

- When the DPR-1020 is powered ON, it automatically performs a Self-Test on each of its major components. The final result of the Self-Test is signaled by the state of the USB LED indicator following the Self-Test. Preliminary to the actual component tests, the three LED indicators are tested to confirm their operation.

- Immediately after power-up, all three of the blue LEDs should illuminate steadily for several seconds. Then the USB LED should light OFF simultaneously. Irregularity of any of the three LEDs during these LED tests may mean there is a problem with the LEDs themselves.

- The actual component tests immediately follow the LED tests. A normal (no fault) result is signaled by simultaneous flashing of the LEDs three times, followed by a quiescent state with all three LEDs dark.

- If the Self-Test routine traps any component error, then following the LED tests the Self-Test will halt and the LEDs will continuously signal the error according to the following table. In the event of any such error signal, contact your dealer for correction of the faulty unit.

Getting Started

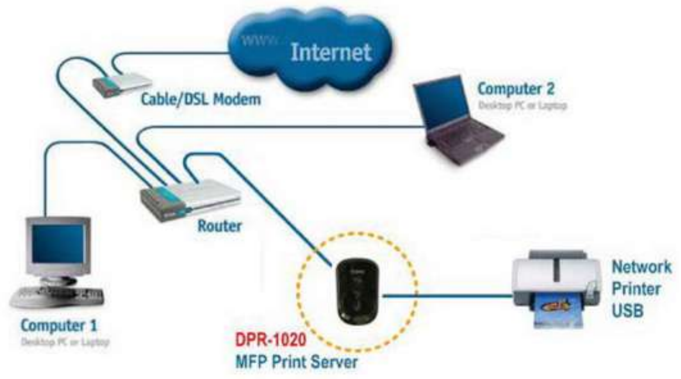
- Below is a sample network using the DPR-1020. The DPR-1020 has a built- in web configurator that allows users to easily configure the Print Server and manage multiple print queues through TCP/IP.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP



Auto-Run Installation

Insert the included installation CD into your computer's CD-ROM drive to initiate the auto-run program. If auto-run does not start, click My Computer > [CD ROM Drive Letter].

The content of the installation CD-ROM includes:

• Install PS Software – click this to install the PS Software, which contains PS-Link and PS-Wizard that can configure more settings for the MFP Server, such as:

- Change the IP address

- Support the multi-functions (Print/Scan/Copy/Fax) of a MFP printer, GDI printing, and other software from any MFP/GDI printer.

`- Easily add a printer to your computer.

• View Quick Installation Guide – click this to preview the Quick Installation Guide in PDF format for step-by-step instructions of the MFP Server Installation.

• View Manual – click this to preview the User Manual in PDF format for detailed information regarding the MFP Server.

• Install Acrobat Reader – click this to install Acrobat Reader for the viewing and printing of PDF files found in this Installation CD-ROM.

• Exit – click to close the Auto-Run program.

5.5 DOMAIN NAME SYSTEM

The domain name, or network name, is a unique name followed by a standard Internet suffixes such as .com, .org, .mil, .net, etc. You can pretty much name your LAN anything if it has a simple dial-up connection and your LAN is not a server providing some type of service to other hosts directly. In addition, our sample network is considered private since it uses IP addresses in the range of 192.168.1.x. Most importantly, the domain name of choice should not be accessible from the Internet if the above constraints are strictly enforced. Lastly, to obtain an "official" domain name you could register through InterNIC, Network Solutions or Register.com. See the Resources section later in this article for the Web sites with detailed instructions for obtaining official domain names.

Hostnames

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- Another important step in setting up a LAN is assigning a unique hostname to each computer in the LAN. A hostname is simply a unique name that can be made up and is used to identify a unique computer in the LAN. Also, the name should not contain any blank spaces or punctuation. For example, the following are valid hostnames that could be assigned to each computer in a LAN consisting of 5 hosts: hostname 1 - Morpheus; hostname 2 - Trinity; hostname 3 - Tank; hostname 4 - Oracle; and hostname 5 - Dozer. Each of these hostnames conforms to the requirement that no blank spaces or punctuation marks are present. Use short hostnames to eliminate excessive typing, and choose a name that is easy to remember.

- Every host in the LAN will have the same network address, broadcast address, subnet mask, and domain name because those addresses identify the network in its entirety. Each computer in the LAN will have a hostname and IP address that uniquely identifies that particular host. The network address is 192.168.1.0, and the broadcast address is 192.168.1.128. Therefore, each host in the LAN must have an IP address between 192.168.1.1 to 192.168.127.

IP address	Example	Same/unique
Network address	192.168.1.0	Same for all hosts
Domain name	www.yourcompanyname.com	Same for all hosts
Broadcast address	192.168.1.128	Same for all hosts
Subnet mask	255.255.255.0	Same for all hosts
Hostname	Any valid name	Unique to each host
Host addresses	192.168.1.*x*	*x* must be unique to each host

5.6 SETTING UP LOCAL NETWORK SERVICES

- ✓ Linux is increasingly popular in the computer networking/telecommunications industry. Acquiring the Linux operating system is a relatively simple and inexpensive task since virtually all of the source code can be downloaded from several different FTP or HTTP sites on the Internet. In addition, the most recent version of Red Hat Linux can be purchased from computer retail stores for between $25 and $50, depending on whether you purchase the standard or full version. The retail brand is indeed a worthwhile investment (vs. the free FTP or HTTP versions) since valuable technical support is included directly from the Red Hat Linux engineers for at least a year. This can be very helpful if, for instance, you can not resolve an installation/configuration problem after consulting the Red Hat Linux manuals.
- ✓ This article describes how to put together a Local Area Network (LAN) consisting of two or more computers using the Red Hat Linux 6.2 operating system. A *LAN* is a communications network that interconnects a variety of devices and provides a means for exchanging information among those devices. The size and scope of a LAN is usually

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

small, covering a single building or group of buildings. In a LAN, modems and phone lines are not required, and the computers should be close enough to run a network cable between them.

- ✓ For each computer that will participate in the LAN, you'll need a network interface card (NIC) to which the network cable will be attached. You will also need to assign a unique hostname and IP address to each computer in the LAN (described later in this article), but this requires a basic understanding of TCP/IP (Transmission Control Protocol/Internet Protocol).

Introduction to TCP/IP

- ✓ TCP/IP is the suite of protocols used by the Internet and most LANs throughout the world. In TCP/IP, every host (computer or other communications device) that is connected to the network has a unique IP address. An IP address is composed of four octets (numbers in the range of 0 to 255) separated by decimal points. The IP address is used to uniquely identify a host or computer on the LAN. For example, a computer with the hostname Morpheus could have an IP address of 192.168.7.127. You should avoid giving two or more computers the same IP address by using the range of IP addresses that are reserved for private, local area networks; this range of IP addresses usually begins with the octets 192.168.

- ✓ LAN network address The first three octets of an IP address should be the same for all computers in the LAN. For example, if a total of 128 hosts exist in a single LAN, the IP addresses could be assigned starting with 192.168.1.x, where x represents a number in the range of 1 to 128. You could create consecutive LANs within the same company in a similar manner consisting of up to another 128 computers. Of course, you are not limited to 128 computers, as there are other ranges of IP addresses that allow you to build even larger networks.

- ✓ There are different classes of networks that determine the size and total possible unique IP addresses of any given LAN. For example, a class A LAN can have over 16 million unique IP addresses. A class B LAN can have over 65,000 unique IP addresses. The size of your LAN depends on which reserved address range you use and the subnet mask (explained later in the article) associated with that range (see Table 1.).

Address range	Subnet mask	Provides	Addresses per LAN
10.0.0.0 - 10.255.255.255.255	255.0.0.0	1 class A LAN	16,777,216
172.16.0.0 - 172.31.255.255	255.255.0.0	16 class B LANs	65,536
192.168.0.0 - 192.168.255.255	25.255.255.0	256 class C LANs	256

Address ranges and LAN sizes

Network and broadcast addresses

- Another important aspect of building a LAN is that the addresses at the two extreme ends of the address range are reserved for use as the LAN's network address and broadcast address. The *network address* is used by an application to represent the overall network. The *broadcast address* is used by an application to send the same message to all other hosts in the network simultaneously.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- For example, if you use addresses in the range of 192.168.1.0 to 192.168.1.128, the first address (192.168.1.0) is reserved as the network address, and the last address (192.168.1.128) is reserved as the broadcast address. Therefore, you only assign individual computers on the LAN IP addresses in the range of 192.168.1.1 to 192.168.1.127:

Network address:	192.168.1.0
Individual hosts:	192.168.1.1 to 192.168.1.127
Broadcast address:	192.168.1.128

Subnet masks

- Each host in a LAN has a subnet mask. The *subnet mask* is an octet that uses the number 255 to represent the network address portion of the IP address and a zero to identify the host portion of the address. For example, the subnet mask 255.255.255.0 is used by each host to determine which LAN or class it belongs to. The zero at the end of the subnet mask represents a unique host within that network.

Assigning IP addresses in a LAN

There are two ways to assign IP addresses in a LAN. You can manually assign a static IP address to each computer in the LAN, or you can use a special type of server that automatically assigns a dynamic IP address to each computer as it logs into the network.

❖ **Static IP addressing**

- Static IP addressing means manually assigning a unique IP address to each computer in the LAN. The first three octets must be the same for each host, and the last digit must be a unique number for each host. In addition, a unique hostname will need to be assigned to each computer. Each host in the LAN will have the same network address (192.168.1.0), broadcast address (192.168.1.128), subnet mask (255.255.255.0), and domain name (yourcompanyname.com). It's a good idea to start by visiting each computer in the LAN and jotting down the hostname and IP address for future reference.

❖ **Dynamic IP addressing**

- Dynamic IP addressing is accomplished via a server or host called DHCP (Dynamic Host Configuration Program) that automatically assigns a unique IP address to each computer as it connects to the LAN. A similar service called BootP can also automatically assign unique IP addresses to each host in the network. The DHCP/ BootP service is a program or device that will act as a host with a unique IP address. An example of a DHCP device is a router that acts as an Ethernet hub (a communications device that allows multiple host to be connected via an Ethernet jack and a specific port) on one end and allows a connection to the Internet on the opposite end. Furthermore, the DHCP server will also assign the

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

network and broadcast addresses. You will not be required to manually assign hostnames and domain names in a dynamic IP addressing scheme.

5.7 SETTING UP Xen , VMare ON LINUX HOST AND ADDING GUEST OS

What Is VMware Player?

- ✓ VMware Player is a free desktop application that lets you run virtual machines on a Windows or Linux PC.
- ✓ VMware Player is the only product on the market that lets you run virtual machines without investing in virtualization software, making it easier than ever to take advantage of the security, flexibility, and portability of virtual machines. VMware Player lets you use host machine devices, such as CD and DVD drives, from the virtual machine.
- ✓ VMware Player provides an intuitive user interface for running preconfigured virtual machines created with VMware Workstation, ESX Server, VMware Server, and GSX Server. On Windows host machines, VMware Player also opens and runs Microsoft Virtual PC and Virtual Server virtual machines and Symantec Backup Exec System Recovery (formerly LiveState Recovery) system images. VMware Player makes VMware virtual machines accessible to colleagues, partners, customers, and clients, whether or not they have purchased VMware products. Anyone who downloads VMware Player can open and run compatible virtual machines.

What You Can Do with VMware Player

With VMware Player, you can:

- **Use and evaluate prebuilt applications**-Download and safely run prebuilt application environments in virtual machines that are available from the Virtual Appliance Marketplace at http://vam.vmware.com.The Virtual Appliance Marketplace includes virtual machines from leading software vendors, including Oracle, Red Hat, Novell, BEA, SpikeSource, IBM, and MySQL, as well as virtual machines that are preconfigured with popular open source software.
- **Transform software distribution**-Simplify software distribution by shipping preconfigured software in virtual machines. End users can experience the benefits of your products immediately, without setup hassles. VMware Player is ideal for shipping evaluation copies or beta software. You can package complex, sophisticated applications, complete with a full working environment, in a virtual machine that can be used by anyone who downloads VMware Player.
- **Collaborate with colleagues**-VMware Player makes it easy for support, development, and QA to share customer scenarios in virtual machines.

Features in VMware Player

- VMware Player is a free desktop application for running virtual machines. VMware Player does not include features found in other VMware products, such as the ability to create virtual machines.
- VMware Player provides the following features:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- You can connect, disconnect, and use configured host devices, including USB devices, in the virtualmachine.
- You can set preferences, such as how devices are displayed in VMware Player.
- You can change the amount of memory allocated to the virtual machine.
- You can drag and drop files between a Linux or Windows host and a Linux, Windows, or Solaris guest.(Linux hosts and Linux and Solaris guests must be running X Windows.) You can use this feature if theperson who created the virtual machine you are running also installed VMware Tools in it.
- You can copy and paste text between a Windows or Linux host and a Windows, Linux, or Solaris guest.
- You can use this feature if the person who created the virtual machine you are running also installed VMware Tools in it.
- You can copy and paste files between a Windows or Linux host and a Windows, Linux, or Solaris guest.
- You can use this feature if the person who created the virtual machine you are running also installed VMware Tools in it.

To install VMware Player on a Linux host

1 Log on to your Linux host with the user name you plan to use when running VMware Player.

2 In a terminal window, become root so you can perform the initial installation steps:

```
su -
```

3 Mount the VMware Player CD-ROM.

4 Change to the Linux directory on the CD.

5 To use the RPM installer, skip to Step 6. To use the tar installer, follow these steps:

a. If you have a previous tar installation, delete the VMware Player distribution directory before installing from a tar file again. The default location of this directory is:

```
/tmp/vmware-player-distrib
```

b Copy the tar archive to a temporary directory on your hard drive, for example, /tmp:

```
cp VMware-<xxxx>.tar.gz /tmp
```

VMware-<xxxx>.tar.gz is the installation file. (In the filename, <xxxx-xxxx> is a series of numbers representing the version and build numbers.)

c Change to the directory to which you copied the file:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

```
cd /tmp
```

d Unpack the archive:

```
tar zxpf VMware-<xxxx>.tar.gz
```

e Change to the installation directory:

```
cd vmware-player-distrib
```

f Run the installation program:

```
./vmware-install.pl
```

g Press return to accept the default values at the prompts.

h Press return (Yes) when prompted to run vmware-config.pl.

i Skip to Step 7.

Adding Guest OS

To install the OS from an ISO image in a virtual machine:

1. Save the ISO image file in any location accessible to your host. For example:

    Windows: C:\Temp or %TEMP%

    Linux: /tmp or /usr/tmp

Note: For best performance, place this image on the host computer's hard drive. However, to make the ISO image accessible to multiple users, you can also place the ISO image on a network share drive (Windows) or exported filesystem (Linux). If your OS install spans multiple discs, you need to use an ISO image of each disc and place them all of them in a location accessible to the host.

2. Create a new virtual machine. Go to File > New > Virtual Machine.

3. Select Typical to accept Workstation's recommendations for various settings (such as processors, RAM, and disk controller type). Select Custom if you want to select these options yourself.

4. On the Guest Operating System Installation screen, when prompted where to install from, select Installer disc image file (iso).

5. Click Browse, and navigate to the location where you saved the ISO image file.

6. Click next, and proceed through the new virtual machine wizard.

7. Before you click Finish, to create the virtual machine, deselect Power on this virtual machine after creation.

8. Edit the virtual machine settings so that its virtual CD/DVD device is configured to use the ISO image rather than the physical CD/DVD drive:

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

a.Select the tab for the virtual machine you just created.

b.Click Edit virtual machine settings.

c.On the Hardware tab, select the CD/DVD drive.

d.On the right side:

i.Select Connect at power on.

ii.Use ISO image file.

iii.Click Browse and navigate to where you saved the ISO image file.

e.Click OK.

9. Power on the virtual machine.

When you are finished installing the guest OS, you can edit the virtual machine settings so that it is once more set to use the host computer's physical drive. You do not need to leave the drive set to connect at power on.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

A. SUMMARY

- An operating system is software that manages the computer hardware, as well as providing an environment for application programs to run. Perhaps the most visible aspect of an operating system is the interface to the computer system it provides to the human user.
- For a computer to do its job of executing programs, the programs must be in main memory. Main memory is the only large storage area that the processor can access directly. It is an array of bytes, ranging in size from millions to billions. Each byte in memory has its own address.
- The main memory is usually a volatile storage device that loses its contents when power is turned off or lost. Most computer systems provide secondary storage as an extension of main memory.
- Secondary storage provides a form of nonvolatile storage that is capable of holding large quantities of data permanently. The most common secondary-storage device is a magnetic disk, which provides storage of both programs and data.
- The wide variety of storage systems in a computer system can be organized in a hierarchy according to speed and cost. The higher levels are expensive, but they are fast. As we move down the hierarchy, the cost per bit generally decreases, whereas the access time generally increases.
- There are several different strategies for designing a computer system. Single-processor systems have only one processor, while multiprocessor systems contain two or more processors that share physical memory and peripheral devices.
- The most common multiprocessor design is symmetric multiprocessing (or SMP), where all processors are considered peers and run independently of one another. Clustered systems are a specialized form of multiprocessor systems and consist of multiple computer systems connected by a local-area network.
- To best utilize the CPU, modern operating systems employ multiprogramming, which allows several jobs to be in memory at the same time, thus ensuring that the CPU always has a job to execute. Time-sharing systems are an extension of multiprogramming wherein CPU scheduling algorithms rapidly switch between jobs, thus providing the illusion that each job is running concurrently.
- The operating system must ensure correct operation of the computer system. To prevent user programs from interfering with the proper operation of the system, the hardware has

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

two modes: user mode and kernel mode. Various instructions (such as I/O instructions and halt instructions) are privileged and can be executed only in kernel mode.

❖ The memory in which the operating system resides must also be protected from modification by the user. A timer prevents infinite loops. These facilities (dual mode, privileged instructions, memory protection, and timer interrupt) are basic building blocks used by operating systems to achieve correct operation.

❖ A process (or job) is the fundamental unit of work in an operating system. Process management includes creating and deleting processes and providing mechanisms for processes to communicate and synchronize with each other.

❖ Operating systems must also be concerned with protecting and securing the operating system and users. Protection measures control the access of processes or users to the resources made available by the computer system. Security measures are responsible for defending a computer system from external or internal attacks.

❖ Several data structures that are fundamental to computer science are widely used in operating systems, including lists, stacks, queues, trees, hash functions, maps, and bitmaps.

❖ The free software movement has created thousands of open-source projects, including operating systems. Because of these projects, students are able to use source code as a learning tool. They can modify programs and test them, help find and fix bugs, and otherwise explore mature, full-featured operating systems, compilers, tools, user interfaces, and other types of programs.

❖ Operating systems provide a number of services. At the lowest level, system calls allow a running program to make requests from the operating system directly. At a higher level, the command interpreter or shell provides a mechanism for a user to issue a request without writing a program.

❖ Commands may come from files during batch-mode execution or directly from a terminal or desktop GUI when in an interactive or time-shared mode. System programs are provided to satisfy many common user requests.

❖ The types of requests vary according to level. The system-call level must provide the basic functions, such as process control and file and device manipulation. Higher-level requests, satisfied by the command interpreter or system programs, are translated into a sequence of system calls.

❖ System services can be classified into several categories: program control, status requests, and I/O requests. Program errors can be considered implicit requests for service.

❖ The design of a new operating system is a major task. It is important that the goals of the system be well defined before the design begins. The type of system desired is the foundation for choices among various algorithms and strategies that will be needed.

❖ Throughout the entire design cycle, we must be careful to separate policy decisions from implementation details (mechanisms). This separation allows maximum flexibility if policy decisions are to be changed later.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- A process is a program in execution. As a process executes, it changes state. The state of a process is defined by that process's current activity. Each process may be in one of the following states: **new, ready, running, waiting, or terminated.**
- Each process is represented in the operating system by its own **process control block (PCB).**
- A process, when it is not executing, is placed in some waiting queue. There are two major classes of queues in an operating system: **I/O request queues and the ready queue.**
- The ready queue contains all the processes that are ready to execute and are waiting for the CPU. Each process is represented by a PCB.
- The operating system must select processes from various scheduling queues. Long-term (job) scheduling is the selection of processes that will be allowed to contend for the CPU.
- Normally, **long-term scheduling** is heavily influenced by resource-allocation considerations, especially memory management.
- **Short-term (CPU) scheduling** is the selection of one process from the ready queue.
- A thread is a flow of control within a process. A multithreaded process contains several different flows of control within the same address space. The benefits of multithreading include increased responsiveness to the user, resource sharing within the process, economy, and scalability factors, such as more efficient use of multiple processing cores.
- User-level threads are threads that are visible to the programmer and are unknown to the kernel. The operating-system kernel supports and manages kernel-level threads. In general, user-level threads are faster to create and manage than are kernel threads, because no intervention from the kernel is required.
- Three different types of models relate user and kernel threads. The **many to- one model** maps many user threads to a single kernel thread.
- The **one-to-one model** maps each user thread to a corresponding kernel thread.
- The **many-to many** model multiplexes many user threads to a smaller or equal number of kernel threads.
- Most modern operating systems provide kernel support for threads. These include **Windows, Mac OS X, Linux, and Solaris.**
- Thread libraries provide the application programmer with an API for creating and managing threads.
- Three primary thread libraries are in common use: **POSIX Pthreads,Windows threads, and Java threads.**
- In addition to explicitly creating threads using the API provided by a library, we can use implicit threading, in which the creation and management of threading is transferred to compilers and run-time libraries.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- Strategies for implicit threading include **thread pools, OpenMP, and Grand Central Dispatch.**
- Multithreaded programs introduce many challenges for programmers, including the semantics of the **fork() and exec() system calls**.
- Other issues include **signal handling, thread cancellation, thread-local storage, and scheduler activations.**
- Given a collection of cooperating sequential processes that share data, mutual exclusion must be provided to ensure that a critical section of code is used by only one process or thread at a time.
- Typically, computer hardware provides several operations that ensure mutual exclusion. However, such hardware based solutions are too complicated for most developers to use.
- **Mutex locks and semaphores** overcome this obstacle. Both tools can be used to solve various synchronization problems and can be implemented efficiently, especially if hardware support for atomic operations is available.
- Various synchronization problems (such as the bounded-buffer problem, he readers–writers problem, and the dining-philosophers problem) are important mainly because they are examples of a large class of concurrency-control problems. These problems are used to test nearly every newly proposed synchronization scheme.
- The operating system must provide the means to guard against timing errors, and several language constructs have been proposed to deal with these problems. Monitors provide a synchronization mechanism for sharing abstract data types.
- A condition variable provides a method by which a monitor function can block its execution until it is signaled to continue.
- Operating systems also provide support for synchronization. For example, Windows, Linux, and Solaris provide mechanisms such as **semaphores, mutex locks, spinlocks, and condition variables** to control access to shared data.
- The Pthreads API provides support for mutex locks and semaphores, as well as condition variables.
- Several alternative approaches focus on synchronization for multicore systems. One approach uses transactional memory, which may address synchronization issues using either software or hardware techniques.
- Another approach uses the compiler extensions offered by **OpenMP**. Finally, functional programming languages address synchronization issues by disallowing mutability.
- CPU scheduling is the task of selecting a waiting process from the ready queue and allocating the CPU to it. The CPU is allocated to the selected process by the dispatcher.
- **First-come, first-served (FCFS) scheduling** is the simplest scheduling algorithm, but it can cause short processes to wait for very long processes.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- **Shortest job- first (SJF) scheduling** is provably optimal, providing the shortest average waiting time. Implementing SJF scheduling is difficult, however, because predicting the length of the next CPU burst is difficult. The SJF algorithm is a special case of the general priority scheduling algorithm, which simply allocates the CPU to the highest-priority process. Both priority and SJF scheduling may suffer from starvation. Aging is a technique to prevent **starvation**.

- **Round-robin (RR) scheduling** is more appropriate for a time-shared (interactive) system. RR scheduling allocates the CPU to the first process in the ready queue for q time units, where q is the time quantum. After q time units, if the process has not relinquished the CPU, it is preempted, and the process is put at the tail of the ready queue.

- The major problem is the selection of the time quantum. If the quantum is too large, RR scheduling degenerates to FCFS scheduling. If the quantum is too small, scheduling overhead in the form of context-switch time becomes excessive.

- The FCFS algorithm is nonpreemptive; the RR algorithm is preemptive. The SJF and priority algorithms may be either preemptive or nonpreemptive.

- **Multilevel queue algorithms** allow different algorithms to be used for different classes of processes. The most common model includes a foreground interactive queue that uses RR scheduling and a background batch queue that uses FCFS scheduling.

- **Multilevel feedback queues** allow processes to move from one queue to another.

- A dead locked state occurs when two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes. There are three principal methods for dealing with deadlocks:

    - Use some protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlocked state.

    - Allow the system to enter a deadlocked state, detect it, and then recover.

    - Ignore the problem altogether and pretend that deadlocks never occur in the system.

- The third solution is the one used by most operating systems, including Linux and Windows.

- A deadlock can occur only if four necessary conditions hold simultaneously in the system: **mutual exclusion, hold and wait, no preemption, and circular wait.** To prevent deadlocks, we can ensure that at least one of the necessary conditions never holds.

- **Hardware support -** A simple base register or a base–limit register pair is sufficient for the single- and multiple-partition schemes, whereas paging and segmentation need mapping tables to define the address map.

- **Performance** - As the memory-management algorithm becomes more complex, the time required to map a logical address to a physical address increases. For the simple systems, we need only compare or add to the logical address—operations that are fast. Paging and

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

segmentation can be as fast if the mapping table is implemented in fast registers. If the table is in memory, however, user memory accesses can be degraded substantially. A TLB can reduce the performance degradation to an acceptable level.

- **Fragmentation -** A multiprogrammed system will generally perform more efficiently if it has a higher level of multiprogramming. For a given set of processes, we can increase the multiprogramming level only by packing more processes into memory.
- **Relocation -** One solution to the external-fragmentation problem is compaction. Compaction involves shifting a program in memory in such a way that the program does not notice the change. This consideration requires that logical addresses be relocated dynamically, at execution time. If addresses are relocated only at load time, we cannot compact storage.
- **Swapping** - Swapping can be added to any algorithm. At intervals determined by the operating system, usually dictated by CPU-scheduling policies, processes are copied from main memory to a backing store and later are copied back to main memory.
- **Virtual memory** is commonly implemented by demand paging. Pure demand paging never brings in a page until that page is referenced.
- The first reference causes a page fault to the operating system. The operating-system kernel consults an internal table to determine where the page is located on the backing store. It then finds a free frame and reads the page in from the backing store.
- The **page table** is updated to reflect this change, and the instruction that caused the page fault is restarted. This approach allows a process to run even though its entire memory image is not in main memory at once. As long as the page-fault rate is reasonably low, performance is acceptable.
- Disk-scheduling algorithms can improve the effective bandwidth, the average response time, and the variance in response time.
- Algorithms such as **SSTF, SCAN, C-SCAN, LOOK, and C-LOOK** are designed to make such improvements through strategies for disk-queue ordering. Performance of disk-scheduling algorithms can vary greatly on magnetic disks. In contrast, because solid-state disks have no moving parts, performance varies little among algorithms, and quite often a simple FCFS strategy is used.
- The operating system manages the disk blocks. First, a disk must be low level- formatted to create the sectors on the raw hardware—new disks usually come preformatted.
- A **file** is an abstract data type defined and implemented by the operating system. It is a sequence of logical records. A logical record may be a byte, a line (of fixed or variable length), or a more complex data item. The operating system may specifically support various record types or may leave that support to the application program.
- The major task for the operating system is to map the logical file concept onto physical storage devices such as magnetic disk or tape. Since the physical record size of the device may not be the same as the logical record size, it may be necessary to order logical records into physical records. Again, this task may be supported by the operating system or left for the application program.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

- Each device in a file system keeps a volume table of contents or a device directory listing the location of the files on the device. In addition, it is useful to create directories to allow files to be organized.
- A **single-level directory** in a multiuser system causes naming problems, since each file must have a unique name.
- A **two-level directory** solves this problem by creating a separate directory for each user's files. The directory lists the files by name and includes the file's location on the disk, length, type, owner, time of creation, time of last use, and so on.
- Disks are segmented into one or more volumes, each containing a file system or left "raw."
- File systems may be mounted into the system's naming structures to make them available. The naming scheme varies by operating system. Once mounted, the files within the volume are available for use. File systems may be unmounted to disable access or for maintenance.
- The file system resides permanently on secondary storage, which is designed to hold a large amount of data permanently. The most common secondary-storage medium is the disk.
- Physical disks may be segmented into partitions to control media use and to allow multiple, possibly varying, file systems on a single spindle.
- These file systems are mounted onto a logical file system architecture to make them available for use. File systems are often implemented in a layered or modular structure. The lower levels deal with the physical properties of storage, devices. Upper levels deal with symbolic file names and logical properties of files. Intermediate levels map the logical file concepts into physical device properties.
- Any file-system type can have different structures and algorithms. A VFS layer allows the upper layers to deal with each file-system type uniformly. Even remote file systems can be integrated into the system's directory structure and acted on by standard system calls via the VFS interface.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

B. QUESTION BANK
UNIT – I

Part- A

1. What is an Operating System?

An operating system is a program that manages the computer hardware. It also provides a basis for application programs and act as an intermediary between a user of a computer and the computer hardware. It controls and coordinates the use of the hardware among the various application programs for the various users.

2. Why is the Operating System viewed as a resource allocator & control program?

A computer system has many resources – hardware & software that may be required to solve a problem, like CPU time, memory space, file-storage space, I/O devices & so on. The OS acts as a manager for these resources so it is viewed as a resource allocator. The OS is viewed as a control program because it manages the execution of user programs to prevent errors & improper use of the computer.

3. What is the Kernel?

A more common definition is that the OS is the one program running at all times on the computer, usually called the kernel, with all else being application programs.

4. What are Batch Systems?

Batch systems are quite appropriate for executing large jobs that need little interaction. The user can submit jobs and return later for the results. It is not necessary to wait while the job is processed. Operators batched together jobs with similar needs and ran them through the computer as a group.

5. What is the advantage of Multiprogramming?

Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute. Several jobs are placed in the main memory and the processor is switched from job to job as needed to keep several jobs advancing while keeping the peripheral devices in use. Multiprogramming is the first instance where the Operating system must make decisions for the users. Therefore they are fairly sophisticated.

6. What is an Interactive Computer System?

Interactive computer system provides direct communication between the user and the system. The user gives instructions to the operating system or to a program directly, using a keyboard or mouse, and waits for immediate results.

7.What do you mean by Time-Sharing Systems?

Time-sharing or multitasking is a logical extension of multiprogramming. It allows many

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

users to share the computer simultaneously. 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.

8.What are Multiprocessor Systems & give their advantages?

Multiprocessor systems also known as parallel systems or tightly coupled systems are systems that have more than one processor in close communication, sharing the computer bus, the clock and sometimes memory & peripheral devices. Their main advantages are,

- Increased throughput
- Economy of scale
- Increased reliability

9.What are the different types of Multiprocessing?

Symmetric multiprocessing (SMP): In SMP each processor runs an identical copy of the OS & these copies communicate with one another as needed. All processors are peers.

Examples are Windows NT, Solaris, Digital UNIX, and OS/2 & Linux.

***Asymmetric multiprocessing*:** Each processor is assigned a specific task. A master processor controls the system; the other processors look to the master for instructions or predefined tasks. It defines a master-slave relationship.

Example: SunOS Version 4.

10. What is Graceful Degradation?

In multiprocessor systems, failure of one processor will not halt the system, but only slow it down. If there is ten processors & if any one fails then the remaining nine processors pick up the work of the failed processor. This ability to continue providing service is proportional to the surviving hardware is called graceful degradation.

11. What is Dual- Mode Operation?

The dual mode operation provides us with the means for protecting the operating system from wrong users and wrong users from one another. User mode and monitor mode are the two modes. Monitor mode is also called supervisor mode, system mode or privileged mode. Mode bit is attached to the hardware of the computer in order to indicate the current mode. Mode bit is '0' for monitor mode and '1' for user mode.

12. What are Privileged Instructions?

Some of the machine instructions that may cause harm to a system are designated as privileged instructions. The hardware allows the privileged instructions to be executed only in

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

monitor mode.

13. How can a user program disrupt the normal operations of a system?

A user program may disrupt the normal operation of a system by,

- ❖ Issuing illegal I/O operations
- ❖ By accessing memory locations within the OS itself
- ❖ Refusing to relinquish the CPU

14. How is the protection for memory provided?

The protection against illegal memory access is done by using two registers. The base register and the limit register. The base register holds the smallest legal physical address; the limit register contains the size of the range. The base and limit registers can be loaded only by the OS using special privileged instructions

15. What are the various OS Components?

The various system components are,

Process management

- □ Main-memory management
- □ File management

I/O-system management

- □ Secondary-storage management
- □ Networking
- □ Protection system
- □ Command-interpreter system

16. What is a Process?

A process is a program in execution. It is the unit of work in a modern operating system. A process is an active entity with a program counter specifying the next instructions to execute and a set of associated resources. It also includes the process stack, containing temporary data and a data section containing global variables.

17. What is a Process State and mention the various States of a Process?

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process.

Each process may be in one of the following states:

- □ New
- □ Running
- □ Waiting
- □ Ready
- □ Terminated

18. What is Process Control Block (PCB)?

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Each process is represented in the operating system by a process control block also called a task control block. It contains many pieces of information associated with a specific process. It simply acts as a repository for any information that may vary from process to process. It contains the following information:

- Process state
- Program counter
- CPU registers
- CPU-scheduling information
- Memory-management information
- Accounting information
- I/O status information

19. What is the use of Job Queues, Ready Queues & Device Queues?

As a process enters a system, they are put into a job queue. This queue consists of all jobs in the system. The processes that are residing in main memory and are ready & waiting to execute are kept on a list called ready queue. The list of processes waiting for a particular I/O device is kept in the device queue.

20. What is meant by Context Switch?

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is known as context switch. The context of a process is represented in the PCB of a process.

21. What is Spooling?

Spooling means **Simultaneous Peripheral Operations On Line**. It is a high-speed device like a disk is interposed between a running program and a low –speed device involved with the program in input/output. It disassociates a running program from the slow operation of devices like printers.

22. What are System Calls?

System calls provide the interface between a process and the Operating system. System Calls are also called as Monitor call or Operating-system function call. When a system call is executed, it is treated as by the hardware as software interrupt. Control passes through the interrupt vector to a service routine in the operating system, and the mode bit is set to monitor mode.

23. List the services provided by an Operating System?

- Program execution
- I/O Operation
- File-System manipulation
- Communications
- Error detection

24. What are the two types of Real Time Systems?

- Hard real time system
- Soft real time system

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

25. What is the difference between Hard Real Time System and Soft Real Time System?

A hard real time system guarantees that critical tasks complete on time. In a soft real time system, a critical real-time task gets priority over the other tasks, and retains that priority until it completes. Soft real time systems have more limited utility than do hard real-time systems.

26. Write the difference between Multiprogramming and Non - Multiprogramming?

The operating system picks and begins to execute one of the jobs in the memory. Eventually, the job may have to wait for some task, such as a tape to be mounted, or an I/O operation to complete. In a non-multiprogrammed system, the CPU would sit idle. In a multiprogramming system, the operating system simply switches to and executes another job.

When that job needs to wait, the CPU is switched to another job, and so on. Eventually, the first job finishes waiting and gets the CPU back. As long as there is always some job to execute, the CPU will never be idle.

27. What are the design goals of an Operating System?

The requirements can be divided into two basic groups: User goals and System goals.Users desire that the system should be convenient and easy to use, easy to learn, reliable, safe and fast. The Operating system should be easy to design, implement, and maintain. Also it should be flexible, reliable, error free and efficient. These are some of the requirements, which are vague and have no general solution.

28. What are the five major categories of System Calls?

- Process Control
- File-management
- Device-management
- Information maintenance
- Communications

29. What is the use of Fork and Execve System Calls?

Fork is a System calls by which a new process is created. Execve is also a System call, which is used after a fork by one of the two processes to replace the process memory space with a new program.

30. Define Elapsed CPU time and Maximum CPU time?

Elapsed CPU Time: Total CPU time used by a process to date.

Maximum CPU Time: Maximum amount of CPU time a process may use.

PART – B (16 MARKS)

1. Explain the various types of computer systems.

2. Explain how protection is provided for the hardware resources by the operating system.

3. What are the system components of an operating system and explain them?

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

4. What are the various process scheduling concepts?

5. Explain about inter process communication.

6. List five services provided by an operating system. Explain how each provides convenience to the users. Explain also in which cases it would be impossible for user level programs to provide these services.

7. Explain the System Structure of Operating System.

8. Explain the concept of Virtual Machine with neat sketch.

9. Explain Client-Server communication with an example.

10. Explain the various threading issues.

UNIT – II

Part – A

1. **What is a Thread?**

A thread otherwise called a lightweight process (LWP) is a basic unit of CPU utilization, it comprises of a thread id, a program counter, a register set and a stack. It shares with other threads belonging to the same process its code section, data section, and operating system resources such as open files and signals.

2. **What are the benefits of Multithreaded Programming?**

The benefits of multithreaded programming can be broken down into four major categories:

- ❖ Responsiveness
- ❖ Resource sharing
- ❖ Economy
- ❖ Utilization of multiprocessor architectures

3. Define Thread Cancellation & Target Thread.

The thread cancellation is the task of terminating a thread before it has completed. A thread that is to be cancelled is often referred to as the target thread. For example, if multiple threads are concurrently searching through a database and one thread returns the result, the remaining threads might be cancelled.

4. **What are the different ways in which a Thread can be cancelled?**

Cancellation of a target thread may occur in two different scenarios:

Asynchronous cancellation: One thread immediately terminates the target thread is called asynchronous cancellation.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Deferred cancellation: The target thread can periodically check if it should terminate, allowing the target thread an opportunity to terminate itself in an orderly fashion.

5. Define CPU Scheduling.

CPU scheduling is the process of switching the CPU among various processes. CPU scheduling is the basis of multiprogrammed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.

6. What is Preemptive and Non - Preemptive scheduling?

Under non - preemptive scheduling once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or switching to the waiting state.

Preemptive scheduling can preempt a process which is utilizing the CPU in between its execution and give the CPU to another process.

7. What is a Dispatcher?

The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves:

- Switching context
- Switching to user mode
- Jumping to the proper location into the user program to restart that program.

8. What is Dispatch Latency?

The time taken by the dispatcher to stop one process and start another running is known as dispatch latency.

9. What are the various scheduling criteria for CPU Scheduling?

The various scheduling criteria are,

- CPU utilization
- Throughput
- Turnaround time
- Waiting time
- Response time

10. Define Throughput?

Throughput in CPU scheduling is the number of processes that are completed per unit time. For long processes, this rate may be one process per hour; for short transactions, throughput might be 10 processes per second.

11. What is Turnaround Time?

Turnaround time is the interval from the time of submission to the time of completion of

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

a process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

12. Define Race Condition.

When several process access and manipulate same data concurrently, then the outcome of the execution depends on particular order in which the access takes place is called race condition. To avoid race condition, only one process at a time can manipulate the shared variable.

13. What is Critical Section problem?

Consider a system consists of ‘n‘processes. Each process has segment of code called a critical section, in which the process may be changing common variables, updating a table, writing a file. When one process is executing in its critical section, no other process can allowed executing in its critical section.

14. What are the requirements that a solution to the Critical Section Problem must satisfy?

The three requirements are,

- Mutual exclusion
- Progress
- Bounded waiting

15. Define Entry Section and Exit Section.

The critical section problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of the code implementing this request is the entry section. The critical section is followed by an exit section. The remaining code is the remainder section.

16. Give two hardware instructions and their definitions which can be used for implementing Mutual Exclusion.

Test And Set

```
boolean TestAndSet (boolean &target)
{
        boolean rv = target;
        target = true;
        return rv;
}
```

Swap

```
void Swap (boolean &a, boolean &b)
{
        boolean temp = a;
        a = b;
        b = temp;
```

17. What is a Semaphore?

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

A semaphore 'S' is a synchronization tool which is an integer value that, apart from initialization, is accessed only through two standard atomic operations; wait and signal.

Semaphores can be used to deal with the n-process critical section problem. It can be also used to solve various synchronization problems.

The classic definition of 'wait'

```
wait (S)
{
        while (S<=0)
        S--;
}
```

The classic definition of 'signal'

```
signal (S)
{
        S++;
}
```

18. Define Busy Waiting and Spinlock.

When a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the entry code. This is called as busy waiting and this type of semaphore is also called a spinlock, because the process while waiting for the lock.

20. How can we say the First Come First Served scheduling algorithm is Non Preemptive?

Once the CPU has been allocated to the process, that process keeps the CPU until it releases, either by terminating or by requesting I/O. So we can say the First Come First Served scheduling algorithm is non preemptive.

21. What is Waiting Time in CPU scheduling?

Waiting time is the sum of periods spent waiting in the ready queue. CPU scheduling algorithm affects only the amount of time that a process spends waiting in the ready queue.

22. What is Response Time in CPU scheduling?

Response time is the measure of the time from the submission of a request until the first response is produced. Response time is amount of time it takes to start responding, but not the time that it takes to output that response.

23. Differentiate Long Term Scheduler and Short Term Scheduler

The long-term scheduler or job scheduler selects processes from the job pool and loads them into memory for execution.

The short-term scheduler or CPU scheduler selects from among the process that are ready to execute, and allocates the CPU to one of them.

24. Write some classical problems of Synchronization?

- The Bounded-Buffer Problem

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

 The Readers-Writers Problem

 The Dining Philosophers Problem

25. When the error will occur when we use the Semaphore?

 When the process interchanges the order in which the wait and signal operations on

the semaphore mutex.

 When a process replaces a signal (mutex) with wait (mutex).

 When a process omits the wait (mutex), or the signal (mutex), or both.

26. What is Mutual Exclusion?

A way of making sure that if one process is using a shared modifiable data, the other processes will be excluded from doing the same thing. Each process executing the shared data variables excludes all others from doing so simultaneously. This is called mutual exclusion.

27. Define the term Critical Regions?

Critical regions are small and infrequent so that system through put is largely unaffected by their existence. Critical region is a control structure for implementing mutual exclusion over a shared variable.

28. What are the drawbacks of Monitors?

 Monitor concept is its lack of implementation most commonly used programming languages.
 There is the possibility of deadlocks in the case of nested monitor's calls.

29. What are the two levels in Threads?

Thread is implemented in two ways.

 User level and Kernel level

30. What is a Gantt Chart?

A two dimensional chart that plots the activity of a unit on the Y-axis and the time on the X-axis. The chart quickly represents how the activities of the units are serialized.

31. Define Deadlock.

A process requests resources; if the resources are not available at that time, the process enters a wait state. Waiting processes may never again change state, because the resources they have requested are held by other waiting processes. This situation is called a deadlock.

32. What is the sequence in which resources may be utilized?

Under normal mode of operation, a process may utilize a resource in the

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

following sequence:

- Request: If the request cannot be granted immediately, then the requesting process must wait until it can acquire the resource.
- Use: The process can operate on the resource.
- Release: The process releases the resource.

33. What are conditions under which a deadlock situation may arise?

A deadlock situation can arise if the following four conditions hold simultaneously in a system:

- Mutual exclusion
- Hold and wait
- No pre-emption
- Circular wait

34. What is a Resource-Allocation Graph?

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 the set consisting of all active processes in the system and R the set consisting of all resource types in the system.

35. Define Request Edge and Assignment Edge.

A directed edge from process Pi to resource type Rj is denoted by Pi Rj; it signifies that process Pi requested an instance of resource type Rj and is currently waiting for that resource. A directed edge from resource type Rj to process Pi is denoted by RjàPi, it signifies that an instance of resource type has been allocated to a process Pi. A directed edge PiàRj is called a request edge. A directed edge RjàPi is called an assignment edge.

36. What are the methods for Handling Deadlocks?

The deadlock problem can be dealt with in one of the three ways:

- Use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlock state.
- Allow the system to enter the deadlock state, detect it and then recover.
- Ignore the problem all together, and pretend that deadlocks never occur in the system.

37. Define Deadlock Prevention.

Deadlock prevention is a set of methods for ensure that at least any one of the four necessary conditions like mutual exclusion, hold and wait, no pre-emption and circular wait cannot hold. By ensuring that that at least one of these conditions cannot hold, the occurrence of a deadlock can be prevented.

38. Define Deadlock Avoidance.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

An alternative method for avoiding deadlocks is to require additional information about how resources are to be requested. Each request requires the system consider the resources currently available, the resources currently allocated to each process, and the future requests and releases of each process, to decide whether the could be satisfied or must wait to avoid a possible future deadlock.

39. What are a Safe State and an Unsafe State?

A state is safe if the system can allocate resources to each process in some order and still avoid a deadlock. A system is in safe state only if there exists a safe sequence. A sequence of processes <P1,P2,….Pn> is a safe sequence for the current allocation state if, for each Pi, the resource that Pi can still request can be satisfied by the current available resource plus the resource held by all the Pj, with j<i. if no such sequence exists, then the system state is said to be unsafe.

40. What is Banker's Algorithm?

Banker's algorithm is a deadlock avoidance algorithm that is applicable to a resource-allocation system with multiple instances of each resource type. The two algorithms used for its implementation are:

***Safety algorithm**:* The algorithm for finding out whether or not a system is in a safe state.

***Resource-request algorithm**:* if the resulting resource-allocation is safe, the transaction is completed and process Pi is allocated its resources. If the new state is unsafe Pi must wait and the old resource-allocation state is restored.

41. Define Logical Address and Physical Address.

address generated by the CPU is referred as logical address. 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.

42. What are Logical Address Space and Physical Address Space?

The set of all logical addresses generated by a program is called a logical address space; the set of all physical addresses corresponding to these logical addresses is a physical address space.

43. What is the main function of the Memory-Management Unit?

The runtime mapping from virtual to physical addresses is done by a hardware device called a memory management unit (MMU).

44. What are the methods for dealing the Deadlock Problem?

- Use a protocol to ensure that the system will never enter a deadlock state.
- Allow the system to enter the deadlock state and then recover.
- Ignore the problem all together, and pretend that deadlocks never occur in the system.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

45. Differentiate Deadlock and Starvation.

A set of processes is in deadlock state when every process in the set is waiting for an event that can be caused only by the other process in the set. Starvation or indefinite blocking is a situation where processes wait indefinitely within the semaphore.

PART - B

1. Write about the various CPU scheduling algorithms.

2. What is critical section problem and explain two process solutions and multiple process solutions?

3. Explain what semaphores are, their usage, implementation given to avoid busy waiting and binary semaphores.

4. Explain about critical regions and monitors

5. Explain the various classic problems of synchronization

6. Write note on TSL and SWAP instruction.

7. Give a detailed description about deadlocks and its characterization

8. Explain about the methods used to prevent deadlocks

9. Explain the Banker's algorithm for deadlock avoidance.

10. Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

	Process	Burst Time	Priority
1.	P1	10	3
2.	P2	1	1
3.	P3	2	3
4.	P4	1	4
5.	P5	5	2

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

a. Draw four Gantt charts illustrating the execution of these processes using FCFS,SJF,A non preemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1) scheduling. (4)

b. What is the turnaround time of each process for each of the scheduling algorithms (4)

c. What is the waiting time of each process for each of the scheduling algorithms (4)

d. Which of the schedules in part a results in the minimal average waiting time (over all processes)?

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

UNIT - III

Part – A

1. Define Dynamic Loading

To obtain better memory-space utilization dynamic loading is used. With dynamic loading, a routine is not loaded until it is called. All routines are kept on disk in a relocatable load format. The main program is loaded into memory and executed. If the routine needs another routine, the calling routine checks whether the routine has been loaded. If not, the relocatable linking loader is called to load the desired program into memory.

2. Define Dynamic Linking.

Dynamic linking is similar to dynamic loading, rather that loading being postponed until execution time, linking is postponed. This feature is usually used with system libraries, such as language subroutine libraries. A stub is included in the image for each library-routine reference. The stub is a small piece of code that indicates how to locate the appropriate memory-resident library routine, or how to load the library if the routine is not already present.

3. What are Overlays?

To enable a process to be larger than the amount of memory allocated to it, overlays are used. The idea of overlays is to keep in memory only those instructions and data that are needed at a given time. When other instructions are needed, they are loaded into space occupied previously by instructions that are no longer needed.

4. Define Swapping.

A process needs to be in memory to be executed. However a process can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution. This process is called swapping.

5. What do you mean by Best Fit?

Best fit allocates the smallest hole that is big enough. The entire list has to be searched, unless it is sorted by size. This strategy produces the smallest leftover hole.

6. What do you mean by First Fit?

First fit allocates the first hole that is big enough. Searching can either start at the beginning of the set of holes or where the previous first-fit search ended. Searching can be stopped as soon as a free hole that is big enough is found.

7. How is memory protected in a paged environment?

Protection bits that are associated with each frame accomplish memory protection in a paged environment. The protection bits can be checked to verify that no writes are being made to a read-only page.

8. What is External Fragmentation?

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

External fragmentation exists when enough total memory space exists to satisfy a request, but it is not contiguous; storage is fragmented into a large number of small holes.

9. What is Internal Fragmentation?

When the allocated memory may be slightly larger than the requested memory, the difference between these two numbers is internal fragmentation.

10. What do you mean by Compaction?

Compaction is a solution to external fragmentation. The memory contents are shuffled to place all free memory together in one large block. It is possible only if relocation is dynamic, and is done at execution time.

11. What are Pages and Frames?

Paging is a memory management scheme that permits the physical-address space of a process to be non-contiguous. In the case of paging, physical memory is broken into fixed-sized blocks called frames and logical memory is broken into blocks of the same size called pages.

12. What is the use of Valid-Invalid Bits in Paging?

When the bit is set to valid, this value indicates that the associated page is in the process's logical address space, and is thus a legal page. If the bit is said to invalid, this value indicates that the page is not in the process's logical address space. Using the valid-invalid bit traps illegal addresses.

13. What is the basic method of Segmentation?

Segmentation is a memory management scheme that supports the user view of memory. A logical address space is a collection of segments. The logical address consists of segment number and offset. If the offset is legal, it is added to the segment base to produce the address in physical memory of the desired byte.

14. A Program containing relocatable code was created, assuming it would be loaded at address 0. In its code, the program refers to the following addresses: 50,78,150,152,154. If the program is loaded into memory starting at location 250, how do those addresses have to be adjusted?

All addresses need to be adjusted upward by 250.So the adjusted addresses would be 300, 328, 400, 402, and 404.

15. What is Virtual Memory?

Virtual memory is a technique that allows the execution of processes that may not be completely in memory. It is the separation of user logical memory from physical memory. This separation provides an extremely large virtual memory, when only a smaller physical memory is available.

16. What is Demand Paging?

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Virtual memory is commonly implemented by demand paging. In demand paging, the pager brings only those necessary pages into memory instead of swapping in a whole process. Thus it avoids reading into memory pages that will not be used anyway, decreasing the swap time and the amount of physical memory needed.

17. Define Lazy Swapper.

Rather than swapping the entire process into main memory, a lazy swapper is used. A lazy swapper never swaps a page into memory unless that page will be needed.

18. What is a Pure Demand Paging?

When starting execution of a process with no pages in memory, the operating system sets the instruction pointer to the first instruction of the process, which is on a non-memory resident page, the process immediately faults for the page. After this page is brought into memory, the process continues to execute, faulting as necessary until every page that it needs is in memory. At that point, it can execute with no more faults. This schema is pure demand paging.

19. Define Effective Access Time.

Let p be the probability of a page fault (0£p£1). The value of p is expected to be close to 0; that is, there will be only a few page faults. The effective access time is,

Effective access time = (1-p) * ma + p * page fault time.

ma : memory-access time

20. Define Secondary Memory.

This memory holds those pages that are not present in main memory. The secondary memory is usually a high speed disk. It is known as the swap device, and the section of the disk used for this purpose is known as swap space.

21. What is the basic approach of Page Replacement?

If no frame is free is available, find one that is not currently being used and free it. A frame can be freed by writing its contents to swap space, and changing the page table to indicate that the page is no longer in memory. Now the freed frame can be used to hold the page for which the process faulted.

22. What is the various Page Replacement Algorithms used for Page Replacement?

- FIFO page replacement
- Optimal page replacement
- LRU page replacement
- LRU approximation page replacement
- Counting based page replacement
- Page buffering algorithm.

23. What are the major problems to implement Demand Paging?

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

The two major problems to implement demand paging is developing,

- Frame allocation algorithm
- Page replacement algorithm

24. What is a Reference String?

An algorithm is evaluated by running it on a particular string of memory references and computing the number of page faults. The string of memory reference is called a reference string.

PART – B

1. Explain Dynamic Storage-Allocation Problem

2. Explain about Fragmentation

3. Explain the concept of Paging

4. Explain the types of Page Table Structure

5. Explain about Segmentation in detail.

UNIT – IV

Part – A

1 . What is a File?

A file is a named collection of related information that is recorded on secondary storage. A file contains either programs or data. A file has certain "structure" based on its type.

- File attributes: Name, identifier, type, size, location, protection, time, date
- File operations: creation, reading, writing, repositioning, deleting, truncating, appending, renaming
- File types: executable, object, library, source code etc.

2. List the various File Attributes.

A file has certain other attributes, which vary from one operating system to another, but typically consist of these: Name, identifier, type, location, size, protection, time, date and user identification.

3. What are the various File Operations?

The basic file operations are,

- Creating a file
- Writing a file
- Reading a file
- Repositioning within a file
- Deleting a file
- Truncating a file

STUCOR APP

DOWNLOADED FROM STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

4. What is the information associated with an Open File?

Several pieces of information are associated with an open file which may be:

- File pointer
- File open count
- Disk location of the file
- Access rights

5. What are the different Accessing Methods of a File?

The different types of accessing a file are:

- Sequential access: Information in the file is accessed sequentially
- Direct access: Information in the file can be accessed without any particular order.
- Other access methods: Creating index for the file, indexed sequential access method (ISAM) etc.

6. What is Directory?

The device directory or simply known as directory records information-such as name, location, size, and type for all files on that particular partition. The directory can be viewed as a symbol table that translates file names into their directory entries.

7. What are the operations that can be performed on a Directory?

The operations that can be performed on a directory are,

- Search for a file
- Create a file
- Delete a file
- Rename a file
- List directory
- Traverse the file system

8. What are the most common schemes for defining the Logical Structure of aDirectory?

The most common schemes for defining the logical structure of a directory

- Single-Level Directory
- Two-level Directory
- Tree-Structured Directories
- Acyclic-Graph Directories
- General Graph Directory

9. Define UFD and MFD.

In the two-level directory structure, each user has own user file directory (UFD). Each UFD has a similar structure, but lists only the files of a single user. When a job starts the system's master file directory (MFD) is searched. The MFD is indexed by the user name or account number, and each entry points to the UFD for that user.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

10. What is a Path Name?

A pathname is the path from the root through all subdirectories to a specified file. In a two-level directory structure a user name and a file name define a path name.

11. What is Access Control List (ACL)?

The most general scheme to implement identity-dependent access is to associate with each file and directory an access control unit.

12. Define Equal Allocation.

The way to split *'m'* frames among *'n'* processes is to give everyone an equal share, *m/n* frames. For instance, if there are 93 frames and 5 processes, each process will get 18 frames. The leftover 3 frames could be used as a free-frame buffer pool. This scheme is called equal allocation.

13. What is the cause of Thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

Thrashing is caused by under allocation of the minimum number of pages required by a process, forcing it to continuously page fault. The system can detect thrashing by evaluating the level of CPU utilization as compared to the level of multiprogramming. It can be eliminated by reducing the level of multiprogramming.

14. If the average page faults service time of 25 ms and a memory access time of 100ns.Calculate the effective access time.

Effective access time = (1-p)*ma + p*page fault time
= (1-p)*100+p*25000000
= 100-100p+25000000*p
= 100 + 24999900p

15. What is Belady's Anomaly?

For some page replacement algorithms, the page fault rate may increase as the number of allocated frames increases.

16. What are the different types of Access?

Different types of operations may be controlled in access type. These are,
- Read
- Write
- Execute
- Append
- Delete
- List

17. What are the types of Path Names?

Path names can be of two types.
- ***Absolute path name:*** Begins at the root and follows a path down to the

STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

specified file, giving the directory names on the path.
 Relative path name: Defines a path from the current directory.

18. What is meant by Locality of Reference?

The locality model states that, as a process executes, it moves from locality to locality. Locality is of two types.

 Spatial locality
 Temporal locality.

19. What are the various layers of a File System?

` The file system is composed of many different levels. Each level in the design uses the feature of the lower levels to create new features for use by higher levels.

Application programs
 Logical file system
 File-organization module
 Basic file system
 I/O control
 Devices

20. What are the Structures used in File-System Implementation?

Several on-disk and in-memory structures are used to implement a file system

 On-disk structure include
 Boot control block
 Partition block
 Directory structure used to organize the files in File control block (FCB)
 In-memory structure include
 In-memory partition table
 In-memory directory structure
 System-wide open file table
 Per-process open table

21. What are the Functions of Virtual File System (VFS)?

It has two functions,

 It separates file-system-generic operations from their implementation defining a clean VFS interface. It allows transparent access to different types of file systems mounted locally.

 VFS is based on a file representation structure, called a vnode. It contains a numerical value for a network-wide unique file .The kernel maintains one vnode structure for each active file or directory.

22. Define Seek Time and Latency Time.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

The time taken by the head to move to the appropriate cylinder or track is called seek time. Once the head is at right track, it must wait until the desired block rotates under the read-write head. This delay is latency time.

23. What are the Allocation Methods of a Disk Space?

Three major methods of allocating disk space which are widely in use are

- Contiguous allocation
- Linked allocation
- Indexed allocation

24. What are the advantages of Contiguous Allocation?

The advantages are,

- Supports direct access
- Supports sequential access
- Number of disk seeks is minimal.

25. What are the drawbacks of Contiguous Allocation of Disk Space?

The disadvantages are,

- Suffers from external fragmentation
- Suffers from internal fragmentation
- Difficulty in finding space for a new file
- File cannot be extended
- Size of the file is to be declared in advance

26. What are the advantages of Linked Allocation?

The advantages are,

- No external fragmentation
- Size of the file does not need to be declared

27. What are the disadvantages of Linked Allocation?

The disadvantages are,

- Used only for sequential access of files.
- Direct access is not supported
- Memory space required for the pointers.
- Reliability is compromised if the pointers are lost or damaged

28. What are the advantages of Indexed Allocation?

The advantages are,

- No external-fragmentation problem
- Solves the size-declaration problems
- Supports direct access

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

29. How can the index blocks be implemented in the Indexed Allocation Scheme?

The index block can be implemented as follows,

- Linked scheme
- Multilevel scheme
- Combined scheme

30. Define Rotational Latency and Disk Bandwidth.

Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head. The disk bandwidth is the total number of bytes transferred, divided by the time between the first request for service and the completion of the last transfer.

31. How free-space is managed using Bit Vector Implementation?

The free-space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

32. Define Buffering.

A buffer is a memory area that stores data while they are transferred between two devices or between a device and an application. Buffering is done for three reasons,

- To cope with a speed mismatch between the producer and consumer of a data stream
- To adapt between devices that have different data-transfer sizes
- To support copy semantics for application I/O.

STUCOR APP

PART - B

1. Explain the File System Structure in detail

2. Discuss the File System Organization and File System Mounting.

3. Explain about File Sharing.

4. Explain about the File System Implementation.

5. Explain about various Allocation Methods.

6. Write note on (i) Log structured file system

(ii) Efficiency and Usage of disk space

(iii) File system mounting

UNIT – V

PART – A

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

1. Define Caching.

A cache is a region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original. Caching and buffering are distinct functions, but sometimes a region of memory can be used for both purposes.

2. Define Spooling.

A spool is a buffer that holds output for a device, such as printer, that cannot accept interleaved data streams. When an application finishes printing, the spooling system queues the corresponding spool file for output to the printer. The spooling system copies the queued spool files to the printer one at a time.

3. What are the various Disk-Scheduling Algorithms?

The various disk-scheduling algorithms are,

- First Come First Served Scheduling
- Shortest Seek Time First Scheduling
- SCAN Scheduling
- C-SCAN Scheduling
- LOOK scheduling

4. What is Low-Level Formatting?

Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called low-level formatting or physical formatting. Low-level formatting fills the disk with a special data structure for each sector. The data structure for a sector consists of a header, a data area, and a trailer.

5. What is the use of Boot Block?

For a computer to start running when powered up or rebooted it needs to have an initial program to run. This bootstrap program tends to be simple. It finds the operating system on the disk loads that kernel into memory and jumps to an initial address to begin the operating system execution. The full bootstrap program is stored in a partition called the boot blocks, at fixed location on the disk. A disk that has boot partition is called boot disk or system disk.

6. What is Sector Sparing?

Low-level formatting also sets aside spare sectors not visible to the operating system. The controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as sector sparing or forwarding.

7. What are the techniques used for performing I/O.

- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access (DMA).

8. Give an example of an application in which data in a file should be accessed in the

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

following order:

***Sequentially* -** Print the content of the file.
Randomly - Print the content of record *i*. This record can be found using hashing or index techniques

9. What problems could occur if a system allowed a file system to be mounted simultaneously at more than one location?

There would be multiple paths to the same file, which could confuse users or encourage mistakes. (Deleting a file with one path deletes the file in all the other paths.)

10. Why must the bit map for file allocation be kept on mass storage rather than in main memory?

In case of system crash (memory failure), the free-space list would not be lost as it would be if the bit map had been stored in main memory.

11. What criteria should be used in deciding which strategy is best utilized for a particular file?

- **Contiguous -** File is usually accessed sequentially, if file is relatively small.
- **Linked** - File is usually accessed sequentially, if the file is large.
- **Indexed** - File is usually accessed randomly, if file is large.

12. What is meant by RAID?

"RAID" is now used as an umbrella term for computer data storage schemes that can divide and replicate data among multiple hard disk drives. The different schemes architectures are named by the word RAID followed by a number, as in RAID 0, RAID 1, etc. RAID's various designs involve two key design goals: increase data reliability and/or increase output performance. When multiple physical disks are set up to use RAID technology, they are said to be *in* a *RAID* array.

13. What is meant by Stable Storage?

Stable storage is a classification of computer data storage technology that guarantees atomicity for any given write operation and allows software to be written that is robust against some hardware and power failures. To be considered atomic, upon reading back a just written-to portion of the disk, the storage subsystem must return either the write data or the data that was on that portion of the disk before the write operation.

14. What is meant by Tertiary Storage?

Tertiary storage or **tertiary memory** provides a third level of storage. Typically it involves a robotic mechanism which will *mount* (insert) and *dismount* removable mass storage media into a storage device according to the system's demands; this data is often copied to secondary storage before use.

15. Write a note on Descriptor?

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

UNIX processes use *descriptors* to reference I/O streams. Descriptors are small unsigned integers obtained from the *open* and *socket* system calls.. A *read* or *write* system call can be applied to a descriptor to transfer data.

The *close* system call can be used to deallocate any descriptor. Descriptors represent underlying objects supported by the kernel, and are created by system calls specific to the type of object. In 4.4BSD, three kinds of objects can be represented by descriptors: files, pipes, and sockets.

16. Write short notes on Pipes?

A *pipe* is a linear array of bytes, as is a file, but it is used solely as an I/O stream, and it is unidirectional. It also has no name, and thus cannot be opened with *open*.Instead, it is created by the *pipe* system call, which returns two descriptors, one of which accepts input that is sent to the other descriptor reliably, without duplication, and in order. The system also supports a named pipe or FIFO. A FIFO has properties identical to a pipe, except that it appears in the file system; thus, it can be opened using the *open* system call. Two processes that wish to communicate each open the FIFO: One opens it for reading, the other for writing.

PART – B

1. Explain the allocation methods for disk space?

2. What are the various methods for free space management?

3. Write about the kernel I/O subsystem.

4. Explain the various disk scheduling techniques
   - FCFS
   - SSTF
   - SCAN
   - C-SCAN
   - C-LOOK

5. Write notes about disk management and swap-space management.

6. Explain in detail the allocation and freeing the file storage space.

7. Explain the backup and recovery of files.

8. Discuss with diagrams the following three disk scheduling: FCFS, SSTF, C-SCAN.

9. Compare and contrast the FREE SPACE and SWAP SPACE management.

10. Explain the disk scheduling algorithms

11. Describe the most common schemes for defining the logical structure of a Directory.

12. Explain the life cycle of an I/O request with flowchart.

13. Discuss about the UNIX file system in detail.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

14. Discuss briefly about Memory Management in UNIX.

15. Explain the process management under LINUX OS.

16. In what ways the directory is implemented?

17. Explain linked allocation in detail.

18. Write the indexed allocation with its performance.

19. Explain the I/O hardware.

20. Explain in detail about Raid
    - RAID 1
    - RAID 2
    - RAID 3
    - RAID 4
    - RAID 5

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Answer ALL questions

PART A — (10 × 2 = 20 Marks)

1. List down the functions of operating systems.
2. What do you mean by multiprogramming?
3. What do you mean by short term scheduler?
4. What is a deadlock?
5. What is address binding?
6. What do you mean by page fault?
7. What is a file? List some operations on it.
8. What are the various file accessing methods?
9. What is rotational latency?
10. What is a swap space?

PART B — (5 × 16 = 80 Marks)

11. (a) (i) Explain how hardware protection can be achieved and discuss in detail the dual mode of operations. (Marks 8)

(ii) Explain in detail any two operating system structures. (Marks 8)

Or

(b) What is meant by a process? Explain states of process with neat sketch and discuss the process state transition with a neat diagram. (Marks 16)

12. (a) What is a critical section? Give examples. What are the minimum requirements that should be satisfied by a solution to critical section problem? Write Peterson Algorithm for 2-process synchronization to critical section problem and discuss briefly. (Marks 16)

Or

(b) Allocation Maximum Available

a b c d a b c d a b c d

P0 0 0 1 2 0 0 1 2 1 5 2 0

P1 1 0 0 0 1 7 5 0

P2 1 3 5 4 2 3 5 6

P3 0 6 3 2 0 6 5 2

P4 0 0 1 4 0 6 5 6

Answer the following

(i) What is the content of the need matrix?

(ii) Is the system in a safe state?

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

(iii) If the request for process P1 arrives for (0,4,2,0) can it be granted immediately. (16)

13. (a) Given memory partition of 100 KB, 500 KB, 200 KB and 600 KB (inorder). Show with neat sketch how would each of the first-fit, best-fit and worst fit algorithms place processes of 412 KB, 317 KB, 112 KB and 326 KB (in order). Which algorithm is most efficient in memory allocation? (Marks 16)

Or

(b) Explain the concept of demand paging. How can demand paging be implemented with virtual memory? (Marks 16)
14. (a) (i) Explain various file allocation methods in detail. (Marks 8)
(ii) What are the possible structures for directory? Discuss them in detail. (Marks 8)

Or

(b) Explain in detail the free space management with neat diagram. (16)
15. (a) Explain in detail various disk scheduling algorithms with suitable example. (16)

Or

(b) Write short notes on the following :
(i) I/O Hardware (Marks 8)

(ii) RAID structure. (Marks 8)

STUCOR APP

Answer ALL questions

PART A — (10 × 2 = 20 marks)

1. What are the five major categories of system calls?

2. What is the function of system programs? Write the name of the categories

in which the system programs can be divided.

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

3. Which are the criteria used for CPU scheduling?

4. Write the three ways to deal the deadlock problem.

5. Define TLB.

6. How do you limit the effects of thrashing?

7. Write the attributes of a file.

8. What is the content of a typical file control block?

9. Write the three basic functions which are provided by the hardware clocks and timers.

10. What is storage-area network?

PART B — (5 × 16 = 80 marks)

11. (a) (i) Briefly illustrate how a server communicates to a client with a java-based sockets program. (12)

(ii) Briefly demonstrate how Remote Method Invocation process works. (4)

Or

(b) (i) Write about the three common types of threading implementation. (6)

(ii) Discuss the threading issues which are considered with multithreaded programs. (10)

12. (a) Explain briefly any four CPU scheduling algorithms with examples. (16)

Or

(b) (i) Write short notes on readers-writers problem and the dining philosophers problem. (8)

(ii) Explain the two solutions of recovery from deadlock. (8)

13. (a) Explain the most commonly used techniques for structuring the page table. (16)

Or

(b) Explain FIFO, Optimal and LRU page replacement algorithms. (16)

14. (a) Explain the two-level directory and three-structured directory. (16)

Or

(b) Give short notes on (16)

(i) Linux file system

(ii) windows XP file system

15. (a) Briefly describe the services which are provided by the kernel's I/O subsystem. (16)

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Or

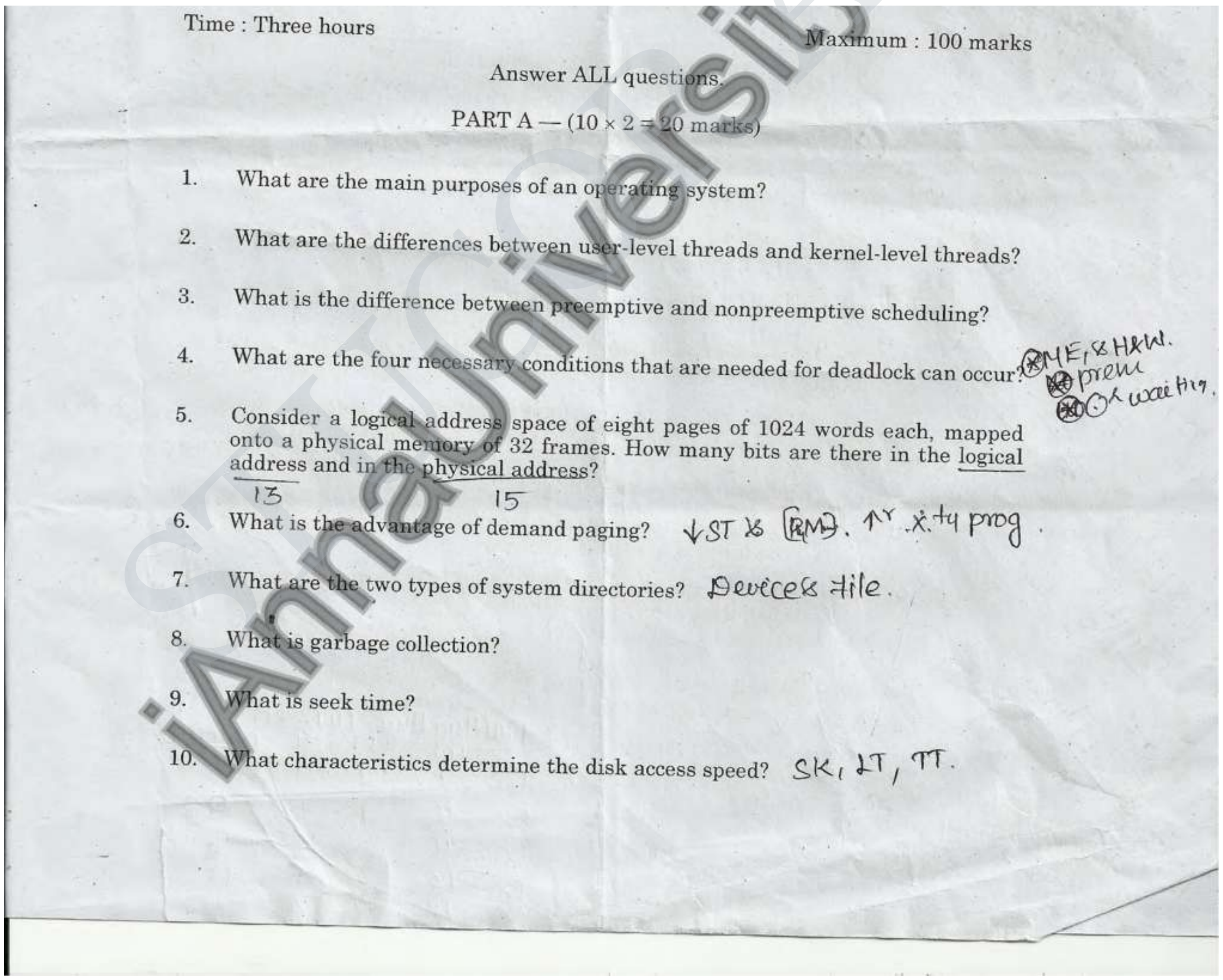
(b) Describe the different forms of disk scheduling. (16)

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP



Time : Three hours

Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What are the main purposes of an operating system?

2. What are the differences between user-level threads and kernel-level threads?

3. What is the difference between preemptive and nonpreemptive scheduling?

4. What are the four necessary conditions that are needed for deadlock can occur? ⊛ ME, & H&W. ⊛ prem ⊛ C^r waiting.

5. Consider a logical address space of eight pages of 1024 words each, mapped onto a physical memory of 32 frames. How many bits are there in the logical address and in the physical address? 13 15

6. What is the advantage of demand paging? ↓ST & (RM). ↑^r x^ty prog.

7. What are the two types of system directories? Devices file.

8. What is garbage collection?

9. What is seek time?

10. What characteristics determine the disk access speed? SK, LT, TT.

STUCOR APP
iAnnaUniversity
STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

PART B — (5 × 16 = 80 marks)

11. (a) (i) Define the essential properties of the following types of operating systems:

(1) Batch

(2) Time sharing

(3) Real time

(4) Distributed (8)

(ii) List five services provided by an operating system. Explain how each provides convenience to the users. Explain also in which cases it would be impossible for user – level programs to provide these services. Program execution, I/O operations, Commun. & Error detection. (8)

Or

(b) (i) What two advantages do threads have over multiple processes? What major disadvantages do they have? Suggest one application that would benefit from the use of threads. (8)

(ii) Explain the various issues associated with the thread in detail. (8)

1. Forks Exec. sys. calls
2. Thread Cancellation
3. Sgnl. Handling.
4. Thread pool

12. (a) (i) What is a Gantt chart? Explain how it is used. (4)

(ii) Consider the following set of processes, with the length of the CPU – burst time given in milliseconds:

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

The processes are arrived in the order P1, P2, P3, P4, P5, all at time 0.

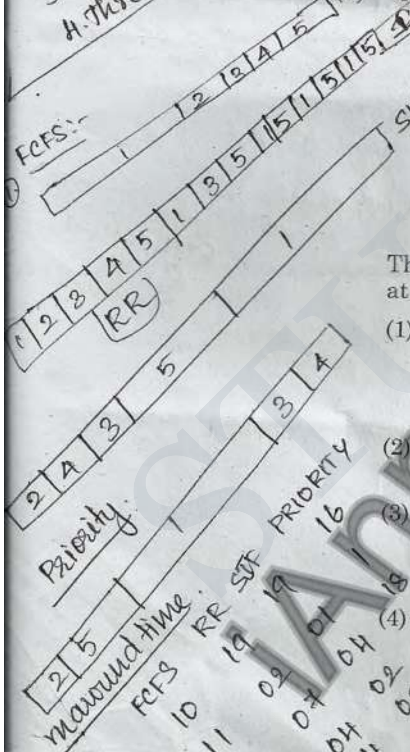
(1) Draw four Gantt charts illustrating the execution of these processes using FCFS, SJF, a nonpreemptive priority (a smaller priority number implies a higher priority), and RR (quantum = 1) scheduling. (3 M)

(2) What is the turnaround time of each process for each of the scheduling algorithms in part a? (3M)

(3) What is the waiting time of each process for each of the scheduling algorithms in Part a? (3M)

(4) Which of the schedules in part a results in the minimal average waiting time (over All processes)? (12)

SJF (3)



Turnaround time

	FCFS	RR	SJF	PRIORITY
P1	10	19	19	16
P2	11	02	01	1
P3	13	07	04	18
P4	14	04	02	19
P5	19	14	09	6

Or

waiting time (TOT - BT)

	FCFS	RR	SJF	PRIORITY
P1	0	9	9	6
P2	10	1	0	0
P3	11	5	2	16
P4	13	3	1	18
P5	14	9	4	1

2 10266

iAnnaUniversity.com

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

(b) (i) What do you mean by busy waiting? What other kinds of waiting are there? Can busy waiting be avoided altogether? Explain your answer. (8)

(ii) Consider the following snapshot of a system:

	Allocation	Max	Available		
	ABCD	ABCD	ABCD	NEED	Available
P0	0012	0012	1520	0000	1100
P1	1000	1750		0330	
P2	1354	2356		1002	
P3	0632	0652		0020	
P4	0014	0656		0642	

<P0, P2, P1, P3, P4> satisfy.

Answer the following questions based on the banker's algorithm:

(1) Define safety algorithm.

(2) What is the content of the matrix Need?

(3) Is the system in a safe state?

(4) If a request from process P1 arrives for (0,4,2,0), can the request be granted immediately? (8)

13. (a) (i) Why are segmentation and paging sometimes combined into one scheme? (4)

(ii) Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, and seven frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.

(1) LRU replacement 20, 18, 15, 10, 8, 7, 7

(2) FIFO replacement 20, 18, 16, 14, 10, 10, 7

(3) Optimal replacement 20, 15, 11, 8, 7, 7, 7. (12)

Or

(b) (i) Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

(1) 0, 430 (2) 1, 10

(3) 2, 500 (4) 3, 400 (8)

(ii) Discuss briefly about memory management in LINUX. (8)

iAnnaUniversity.com STUCOR

STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

14. (a) (i) Explain the various attributes of a file. (4)

(ii) Consider a file currently consisting of 100 blocks. Assume that the file control block (and the index block, in the case of indexed allocation) is already in memory. Calculate how many disk I/O operations are required for contiguous, linked, and indexed (single-level) allocation strategies, if, for one block, the following conditions hold. In the contiguous allocation case, assume that there is no room to grow in the beginning, but there is room to grow in the end. Assume that the block information to be added is stored in memory.

(1) The block is added at the beginning.

(2) The block is added in the middle.

(3) The block is added at the end.

(4) The block is removed from the beginning.

(5) The block is removed from the middle.

(6) The block is removed from the end. (12)

Or

(b) (i) Explain the various schemes used for defining the logical structure of a directory. (8)

(ii) Describe the approaches used in free space management. (8)

15. (a) (i) Consider the following I/O scenarios on a single-user PC.

(1) A mouse used with a graphical user interface

(2) A tape drive on a multitasking operating system (assume no device preallocation is available)

(3) A disk drive containing user files

(4) A graphics card with direct bus connection, accessible through memory-mapped I/O

For each of these I/O scenarios, would you design the operating system to use buffering, spooling, caching, or a combination? Would you use polled I/O, or interruptdriven I/O?

Give reasons for your choices. (8)

(ii) How do you choose a optimal technique among the various disk scheduling techniques? Explain. (8)

Or

(b) (i) Describe the various disk scheduling techniques. (8)

(ii) Describe the various levels of RAID. (8)

STUCOR APP
DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What does the CPU do when there are no user programs to run?

2. What is the principal advantage of the multiprogramming?

3. Define Mutual Exclusion.

4. What is Semaphore?

5. What is page frame?

6. What is internal fragmentation?

7. A direct or sequential access has a fixed file-size S-byte record. At what logical location, the first byte of record N will start?

8. Give an example of a situation where variable-size records would be useful.

9. Writable CD-ROM media are available in both 650 MB and 700 MB versions. What is the principle disadvantage, other than cost, of the 700 MB version?

10. Which disk scheduling algorithm would be best to optimize the performance of a RAM disk?

PART B — (5 × 16 = 80 marks)

11. (a) (i) Explain the important services of an operating system. (8)

(ii) Discuss in detail the concept of virtual machines, with neat sketch. (8)

Or

(b) Write detailed notes on process control and file manipulation. (16)

12. (a) Explain in detail about any two CPU scheduling algorithms with suitable examples. (16)

Or

(b) (i) What is a deadlock? What are the necessary conditions for a deadlock to occur? (6)

(ii) How can a system recover from deadlock? (10)

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

13. (a) Explain about contiguous memory allocation with neat diagram. (16)

Or

(b) What do you mean by paging? Discuss in detail about structure of page tables with appropriate examples. (16)

14. (a) Write a detailed note on various file access methods with neat sketch.(16)

Or

(b) Discuss the different file allocation methods with suitable example. (16)

15. (a) Describe the important concepts of application I/O interface. (16)

Or

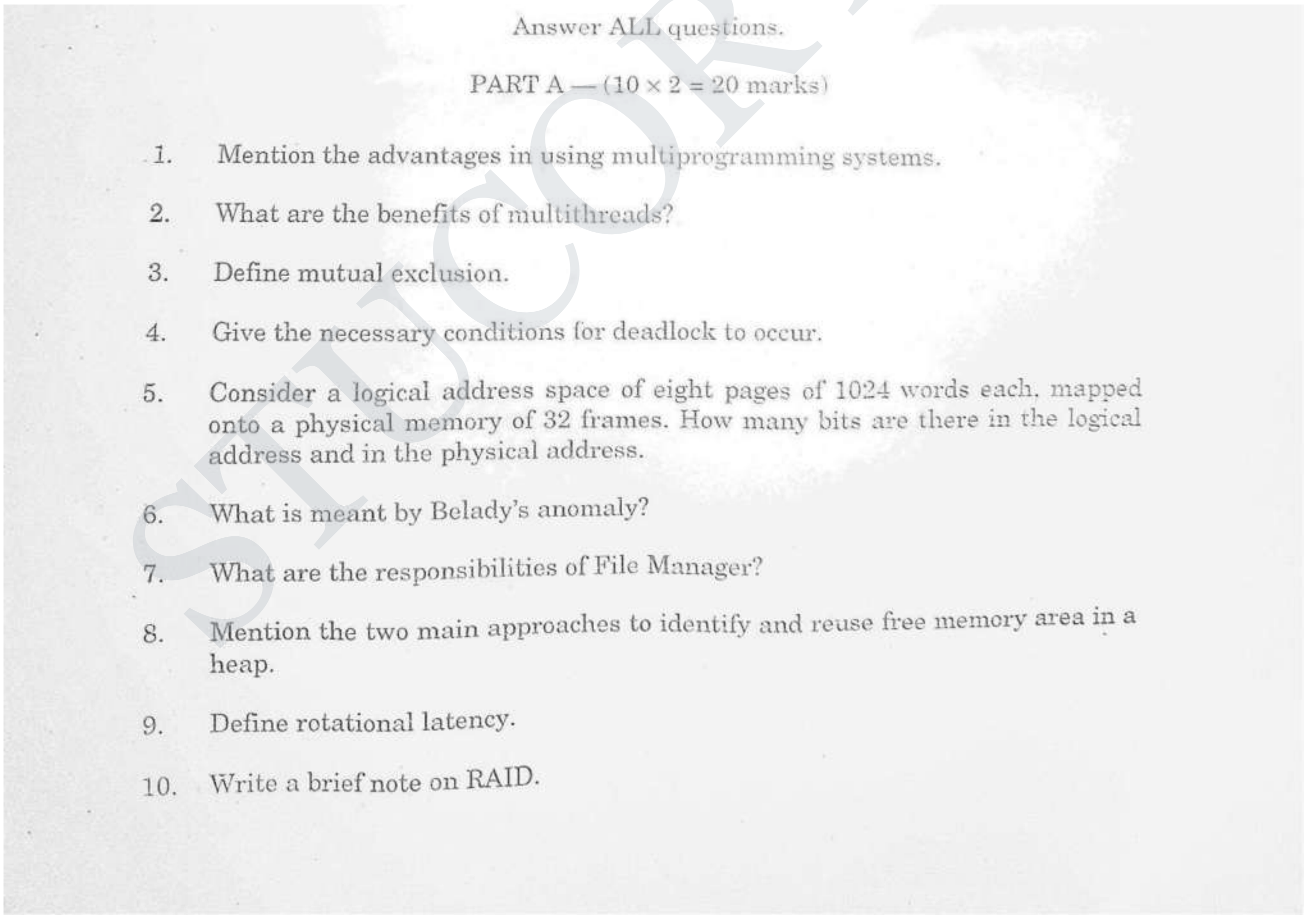
(b) Explain any two disk scheduling algorithms with suitable example. (16)

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP



Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Mention the advantages in using multiprogramming systems.

2. What are the benefits of multithreads?

3. Define mutual exclusion.

4. Give the necessary conditions for deadlock to occur.

5. Consider a logical address space of eight pages of 1024 words each, mapped onto a physical memory of 32 frames. How many bits are there in the logical address and in the physical address.

6. What is meant by Belady's anomaly?

7. What are the responsibilities of File Manager?

8. Mention the two main approaches to identify and reuse free memory area in a heap.

9. Define rotational latency.

10. Write a brief note on RAID.

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

PART B — (5 × 16 = 80 marks)

11. (a) (i) Discuss multiprocessor systems in detail. (8)

(ii) Explain the purpose and importance of system calls in detail with examples. (8)

Or

(b) Discuss how communication is done in client server systems using remote procedure calls and remote method invocation. (16)

12. (a) Discuss the different techniques used for evaluating CPU scheduling algorithms in detail. (16)

Or

(b) (i) What is meant by critical section problem? Propose a solution based on bakery algorithm. (8)

(ii) Consider the following snapshot of a system :

P0 – P4 are 5 processes present and A, B, C, D are the resources. The maximum need of a process and the allocated resources details are given in the table.

Answer the following based on banker's algorithm. Each subdivision carries two marks.

(1) What is the content of NEED matrix? (2)

(2) Is the system in a safe state? (3)

(3) If a request from process P0 arrives for (0, 2, 0) can the request be granted immediately. (3)

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

STUCOR APP

DOWNLOADED FROM STUCOR APP

DOWNLOADED FROM STUCOR APP

STUCOR APP

STUCOR APP

DOWNLOADED FROM STUCOR APP