

## Unit 1: Introduction

Anjib Man Mulepati

www.geocities.com/anjibcs 1 manjib@mail.com.np

## Agenda

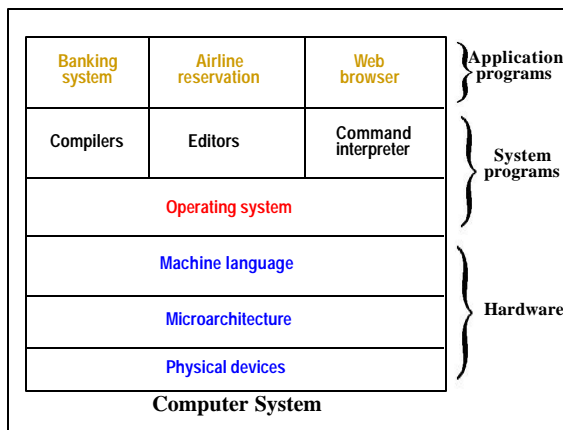
- Introduction to OS
- History of OS
- Variety of OS
- OS Concepts
- OS Structure

www.geocities.com/anjibcs 2 manjib@mail.com.np

## Computer System

- A modern computer system is a complex system.
- Without its software, a computer is basically a useless lump of metal.
- Computer software can be roughly divided into two kind
  - System programs and
  - Application programs
- The most fundamental of all the system programs is the operating system

www.geocities.com/anjibcs 3 manjib@mail.com.np



## Computer System

- **Physical Devices:** IC chips, cards, wires, power supply, cathode ray tubes(CRT) etc.
- **Microarchitecture:** Physical devices grouped together to form functional units
- **Machine language:** Set of instructions like move data, do arithmetic & compare values.
- **Operating System:** the portion of the system programs that runs in kernel (supervisor) mode; generally protected from user tampering by the hardware
- **Compiler, editor, command interpreter:** on top of OS; runs in user mode; user can make own compiler/editor
- **Application:** on top of system programs; programs by the users to solve their own particular problems

www.geocities.com/anjibcs 5 manjib@mail.com.np

## Floppy Disk I/O

- handles floppy disk controller activities through 16 commands specified between 1 and 9 bytes into a device register
- commands - read/write data, move disk arm, format tracks, initialize, sense, reset controller etc
- most basic command READ/WRITE require 13 parameters packed into 9 bytes
- parameters - address of the disk block to be read, number of sectors per track, inter-sector gap spacing, what to do with deleted-data-address-mark - quite esoteric!!
- operation complete results 23 status and error fields packed into 7 bytes
- motor ON/OFF status, delay caused; why not all time ON?

www.geocities.com/anjibcs 6 manjib@mail.com.np

### OS as an Extended Machine

---

- **High-level abstraction for programmer**
  - disk contains a collection of named files
  - each file can be opened for READ/WRITE
  - after READ/WRITE complete close that file
  - no any other detail to deal
- the abstraction hides the truth about the hardware from the programmer and presents a nice, simple view of named files that can be read and written
- in addition to disk hardware - unpleasant businesses concerning interrupt, timer, memory management etc.

**OS function is to present the user with the equivalent of an extended machine or virtual machine that is easier to program than the underlying hardware**

www.geocities.com/anjibcs 7 manjib@mail.com.np

### OS as a Resource Manager

---

- ~~What happens if THREE programs try to print their output on the same printer at the same time?~~
- What happens if TWO network users try to update a shared document at same time?
- Resource management include sharing resources in
  - time and
  - space

**OS primary function is to keep track of who is using which resource, to grant resource requests, to account for usage, and to mediate conflicting requests from different programs and users**

www.geocities.com/anjibcs 8 manjib@mail.com.np

### Operating System

---

- **Top-down view:** extended machine through convenient interface
- **Bottom-up view:** resource manager for all functioning pieces of complex system
- Some of the functions are:
  - User interface implementation
  - share hardware among users
  - allow users to share data
  - prevent users from interfering with one another
  - scheduling resources among users
  - facilitate input/output
  - facilitate parallel operations
  - organize data for secure and rapid access
  - handle network communications

www.geocities.com/anjibcs 9 manjib@mail.com.np

### History

Generation	Component	OS Types
1 <sup>st</sup> 1945 - 1955	Vacuum Tubes and Plugboards	User Driven
2 <sup>nd</sup> 1955 - 1965	Transistors	Batch Systems
3 <sup>rd</sup> 1964 - 1980	IC	Multiprogramming
4 <sup>th</sup> 1980 - Present	PC	Client Server / Distributed

www.geocities.com/anjibcs 10 manjib@mail.com.np

### Evolution

---

- **1940's and 50's**

**Situation:** One user at a time, one program at a time and programmer operates the machine.

**Problem:** lot more programmers wanting to use machine
- **1950's and 60's**

**Situation:** Operator hired to run computer. Programmers have to submit jobs, receive results later. Operator schedules jobs. No random access medium yet. Sequential tapes of jobs are prepared by operator for the CPU to run through once. The operator performs functions of an OS.

**Problem:** random (disk) access required.

**Development:** random disk Access

www.geocities.com/anjibcs 11 manjib@mail.com.np

### Evolution

---

- **Situation:** Pool of jobs now on disk allowing the OS to implement the human operator's algorithms in deciding sequence of jobs to run. Scheduling of jobs now totally automated (true OS).
- **Problem:** input output (i/o) device speed is much slower than CPU speed, so CPU still often idle while i/o going on.
- **Development:**
  - Some *parallelisation* of i/o and computation. Device controller is a piece of hardware that can work in parallel with the CPU.
  - *Spooling* (Simultaneous Peripheral Operations On Line) Print device controller is still copying data from disk to printer while CPU has already begun working on next job. Next job can begin while previous is still printing out.

www.geocities.com/anjibcs 12 manjib@mail.com.np

### Evolution

---

- **Problem:**
  - wait times still too long.
  - long jobs delay everything else
- **Development: Multi-programming**
- **Situation:** Multiple jobs in memory (and running) at the same time. Each memory space protected from each other. OS picks one, executes it for a while, stops (e.g. when program waits for input, or else randomly), picks other to run. CPU busy, but still batch model, not interactive.

www.geocities.com/anjibcs 13 manjib@mail.com.np

### Evolution

---

- **1970's and 80's**
- **Development:** Interactive **time-sharing** on mainframes
- **Situation:** Multi-programming where the program may be waiting on a USER. OS will in the meantime service another program, which may be interacting with another user. Result: Multiple users share the CPU. If the time-slicing is quick enough, they all feel as if they have their own dedicated machine. User interaction at run-time allows a whole world of programs that were never possible before. In practice shared systems can get overloaded and slow down.

www.geocities.com/anjibcs 14 manjib@mail.com.np

### Evolution

---

- **Problem:**
  - increasing number of users who want to interact with programs while they are running.
  - humans' time is expensive - don't want to wait.
- **Development:** Faster and cheaper computers

www.geocities.com/anjibcs 15 manjib@mail.com.np

### Evolution

---

- **1980's and 90's**
- **Situation:** Stand alone PC becomes more common. To some extent a return to no traffic problems. Networks are still considered unimportant. LANs beginning to be used to coordinate file sharing.
- **Problem:** network becomes important, standalone machine now seen as too isolated.
- **Development :** Internet, Web, 1993
- **Situation:** Information shared on remote web servers. Shared remote file systems and email systems. Return to some of the traffic problems.

www.geocities.com/anjibcs 16 manjib@mail.com.np

### Evolution

---

- **Current Requirements:**
  - faster web access
  - easier method of upgrades and compatibility

www.geocities.com/anjibcs 17 manjib@mail.com.np

### Variety of OS

---

- Mainframe OS
- Server OS
- Multiprocessor OS
- PC OS
- Real-Time OS
- Embedded OS
- Smart Card OS

www.geocities.com/anjibcs 18 manjib@mail.com.np

## Mainframe OS

- Mainframe distinguish themselves from PC in terms of their I/O capacity.
- Mainframes are used for Web servers or business server.
- They typically offer three kinds of services:
  - Batch: routine jobs processes without any interactive user present.e.g. daily reporting
  - Transaction: atomic actions e.g check processing
  - Timesharing: allows multiple remote users to run jobs at once.
- Example: OS/360, OS390

www.geocities.com/anjibcs

19

manjib@mail.com.np

## Server OS

- Run on servers, which are either very large PC,workstations or even mainframes.
- Serve multiple users at once over a network and allow the users to share hardware and software resources.
- Provide print service, file service or web service.
- Example: Windows 2000 Server, UNIX

www.geocities.com/anjibcs

20

manjib@mail.com.np

## Multiprocessor OS

- Multiple CPUs are connected into a single system to achieve high computing power.
- Depending on how they are connected and what is shared, these systems are called parallel computer or multiprocessors.
- Special OS with special features for communication and connectivity

www.geocities.com/anjibcs

21

manjib@mail.com.np

## PC OS

- Most widely known OS.
- Aim to provide good service to a single user.
- Example: Windows 98, Windows 2000 Professional

www.geocities.com/anjibcs

22

manjib@mail.com.np

## Real-Time OS

- These systems are characterized by having time as a key parameter.
- Actions are depend upon time
- If the action absolutely must occur at a certain moment, we have a *hard real-time system*.
- Another kind of real-time system is a soft real-time system, in which missing an occasional deadline is acceptable.

www.geocities.com/anjibcs

23

manjib@mail.com.np

## Embedded OS

- Used in palmtop or embedded systems.
- They are special due to their size,memory and power.
- Example: PalmOS, Windows CE(Consumer Electronics)

www.geocities.com/anjibcs

24

manjib@mail.com.np

### Smart Card OS

- The smallest OS run on smart cards, which are credit-card sized devices containing a CPU chip.
- They have very severe processing power and memory constraints.
- Can be divide into two families:
  - General purpose like data storage
  - Dedicated like electronic payment
- Some of important functions are:
  - Management of card security and the cryptographic algorithm procedures.
  - Management of various phases of the card's life cycles.

www.geocities.com/anjibcs
25
manjib@mail.com.np

### OS Basic Concepts

---

- Processes
- Deadlocks
- Memory Management
- Input/Output
- Files
- Security
- The Shell

www.geocities.com/anjibcs
26
manjib@mail.com.np

### Process

---

- A key concept in all OS
- Define as a program in execution.
- Each process has **address space**.
- Address space is a list of memory location which the process can read and write.
- Address space contains
  - executable program
  - program's data
  - stack
  - set of registers like Program Counter(PC), Stack Pointer(SP), Hardware registers and
  - all other information needed to run the program.

www.geocities.com/anjibcs
27
manjib@mail.com.np

### Process Table

---

- In timesharing system, periodically the OS decides to stop running one process and start running another.
- While switching from one process to another all information about the current process must be explicitly saved somewhere during the suspension.
- Process table of OS is such a place where all the information about each process, other than the contents of its own address space is stored.

www.geocities.com/anjibcs
28
manjib@mail.com.np

### Process Management

---


- Process management deal with the creation and termination of processes.
- Process can create one or more other processes known as child processes.
- Thus we can have process tree structure.
- Also processes need to communicate and synchronize their activities. This communication is called **interprocess communication**.
- Other process system calls are available to request more memory, release unused memory.

www.geocities.com/anjibcs
29
manjib@mail.com.np

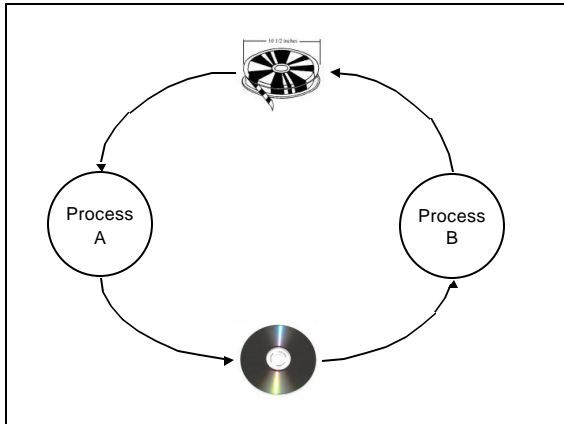
### Deadlocks

---

- When two or more processes are waiting for a particular event that will not occur, then it is called deadlock state.
- Real-world example is traffic deadlock.
- In computer resource allocation problem.



www.geocities.com/anjibcs
30
manjib@mail.com.np



## Memory Management

---

- In simple OS, only one program at a time is in a memory.
- More sophisticated OS allow multiple program to be in memory at the same time.
- Objective is to allow multiple programs to be in memory without interfering each other.
- Also managing the address space of the processes is equally important.
- What happen if a process has more address space than main memory and the process wants to use it all?
- Memory management is also concern with virtual ~~memory management~~.

www.geocities.com/anjibcs 32 manjib@mail.com.np

## Input/Output

---

- All computers have physical devices for acquiring input and producing output.
- Every operating system has an I/O subsystem for managing the I/O devices
- Some of the I/O software is device independent, that is, applies to many or all I/O devices equally well.
- Other parts of it, such as device drivers, are specific to particular I/O devices.

www.geocities.com/anjibcs 33 manjib@mail.com.np

## Files

---

- ~~Another key concept supported by virtually all~~ operating systems is the file system.
- OS hide the complex work of disk and I/O devices.
- To provide a place to keep files, most operating systems have the concept of a **directory** as a way of grouping files together.
- The top of the directory hierarchy is the root directory and process current directory is called working directory.
- When file is to be read or write, it must be opened, at which time the permission are checked.
  - If the access is permitted, the system returns a small integer called a file descriptor to use in subsequent operations.
  - If the access is prohibited, an error code is returned.

www.geocities.com/anjibcs 34 manjib@mail.com.np

## File Tree Vs. Process Tree.

---

- File system is organized as tress so do process but there are some differences such as
  - Process tree are usually not very deep (more than three levels is unusual), whereas file tree are deep.
  - Process tree have short life span, whereas the directory may exist for years.
  - Typically, only a parent process may control or even access a child process, but mechanism nearly always exist to allow files and directories to be read by a wider group than just the owner.

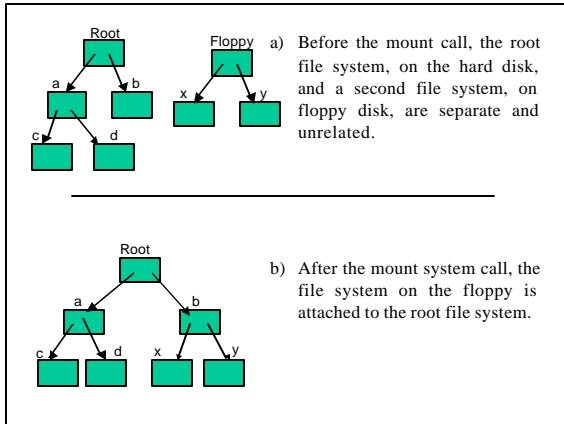
www.geocities.com/anjibcs 35 manjib@mail.com.np

## Mounted File System

---

- Used in UNIX to provide an elegant way to deal with removable media.
- UNIX allows the file system on a floppy disk to be attached to the main tree.(see fig)

www.geocities.com/anjibcs 36 manjib@mail.com.np



### Special File

- Another important concept is the special file, which are provided in order to make I/O devices look like files.
- Advantage is that they can be read and write using the same system calls as are used for reading and writing files.
- Two kinds of special files exist:
  - block special files and
  - character special files
- Block special files are used to model devices that consist of a collection of randomly addressable files, such as disks.
- Character special files are used to model printers, modems and other devices that accepts or output a character stream.

www.geocities.com/anjibcs
38
manjib@mail.com.np

### Pipe

---

- A pipe is a sort of pseudofile that can be used to connect two processes.
- Output of one process is accepted by another process as input.

www.geocities.com/anjibcs
39
manjib@mail.com.np

### Security

---

- One of the most important function of OS is to provide security to user data.
- UNIX files are protected by assigning 9-bit binary protection code.
- For example, the protection code `rwrx-x-x` means that the owner can read, write or execute the file, other group members can read or execute but not write, and everybody else can execute but not read and write the file.
- For directory `x` indicates search permission.
- Other protection issue are: protecting the system from unwanted intruders, both human and nonhuman (e.g virus).

www.geocities.com/anjibcs
40
manjib@mail.com.np

### The Shell

---

- In Command line interface shell act as interface between a user and the operating system.
- Some example in shell
  - \$ `date` → create a child process and run the date program
  - \$ `date > file` → redirected the output to a file
  - \$ `sort < file1 > file2` → invoke the sort program with input taken from file1 and output sent to file2
  - \$ `cat file1 file2 file3 | sort > dev/lp` → invoke the cat program to concatenate three files and send the output to sort to arrange all the lines in alphabetical order. the output of sort is redirected to the file `/dev/lp`, typically printer.

If we puts an `&` after a command, the shell does not wait for it to complete. instead it just gives a prompt immediately.

www.geocities.com/anjibcs
41
manjib@mail.com.np

### Assignment I

---

1. What are the two main functions of an OS?
2. Define terms multiprogramming, spooling,
3. List some difference between PC and mainframe OS.
4. Why is the process table needed in a timesharing system?
5. What is the essential difference between a block special file and a character special file?

www.geocities.com/anjibcs
42
manjib@mail.com.np

### OS Structure

---

- Some of the OS designs are:
  - Monolithic Systems
  - Layered Systems
  - Kernel System
  - Virtual Machines
  - Exokernels
  - Client-server systems

www.geocities.com/anjibcs 43 manjib@mail.com.np

### Monolithic Systems

---

- The structure is that there in no structure.
- The OS is written as a collection of procedures, each of which can call any of the other ones whenever it needs to.
- Each procedure has a well-defined parameters and results.
- Each procedure can call any other one, if the latter provides some useful computation that the former needs.
- All the procedures are binds in a single object file using system linker.
- No information hiding

www.geocities.com/anjibcs 44 manjib@mail.com.np

### Improvement

---

- A main program that invokes the requested service procedure.
- A set of service procedures that carry out the system calls.
- A set of utility procedures that help the service procedures.

www.geocities.com/anjibcs 45 manjib@mail.com.np

### Layered Systems

---

- OS is broken up into a number of layers, each built on top of lower layers.
- The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface.
- For example, a simple batch, THE system has following layers:

Layer	Function
5	The operator
4	User programs
3	I/O Management
2	Operator-process communication
1	Memory Management
0	Processor allocation and multiprogramming

www.geocities.com/anjibcs 46 manjib@mail.com.np

### Pros and Cons

---

- Layered approach provide **modularity**.
  - Layer by layer debugging
  - Layer can simple call operation on lower layer
- But difficult is in defining layers.
- Also they tend to be less efficient since in sequence of execution, parameters may be modified and each layer adds overhead to the system call.

www.geocities.com/anjibcs 47 manjib@mail.com.np

### Kernel System

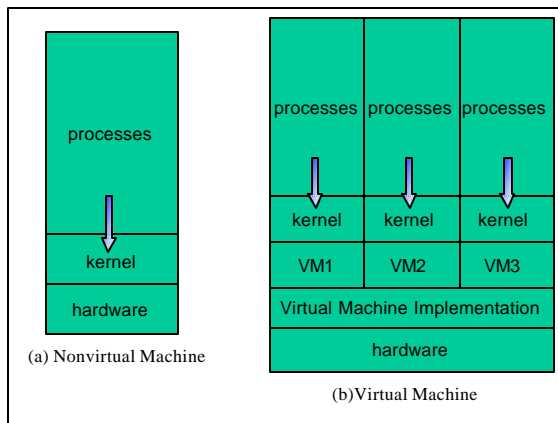
---

- This system consists of two separable parts: the kernel and the system programs.
- The kernel is further separated into a series of interfaces and device drivers, which are expanded over the time.
- This make difficult to enhance, as changes in one section could adversely affect other areas.
- Later **microkernel** approach is developed.
- This method structures the operating systems by removing all nonessential components from the kernel. But what is nonessential components?
- The benefits of the microkernel approach include the ease of extending the OS since change is not in kernel.

www.geocities.com/anjibcs 48 manjib@mail.com.np

## Virtual Machine

- Virtual machine approach does not provide any additional functionality, but rather provides an interface that is identical to the underlying bare hardware.
- Each process is provided with a virtual copy of the underlying computer.
- The physical computer shares resources to create the virtual machines



## Exokernels

- Give each user a clone of actual computer, but with a subset of the resources.
- A program called **Exokernels** allocate resources to virtual machines and then check attempt to use them to make sure no machine is trying to use somebody else's resources.
- in other design remapping of resource is required but here no such remapping is needed.

## Client Server Model

- All the kernel does is handle the communication between clients and servers.
- Advantage
  - Bug in server will not bring whole system down
  - It is adaptable to use in distributed systems.

## Questions

1. Classify different types of OS based on evolution.
2. What are the prime goals of any OS? Are they contradicting? Explain.
3. The advantage of multitasking is self-evident in interactive multi-user systems, since it allows users to share in time, machine resources, under the illusion of having the machine all for themselves. Yet multitasking can significantly increase the productivity of single-user or non-interactive systems as well. If Yes, how? Justify your answer.
4. Write short notes on:
  - a. Goals of OS
  - b. Microprogram Control

THE END

---

## Unit 2: Process

Anjib Man Mulepati

www.geocities.com/anjibcs55manjib@mail.com.np

## Agenda

---

- Process
- Threads
- Interprocess Communication
- Process Scheduling

www.geocities.com/anjibcs56manjib@mail.com.np

## What is Process?

---

- A program is an inanimate entity.
- Only when a processor "**breathes life**" into it, it become the "**active**" entity.
- And we call it process
- So, Process is a program in execution

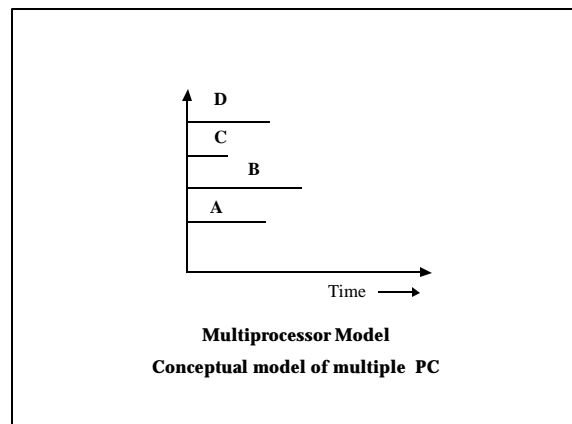
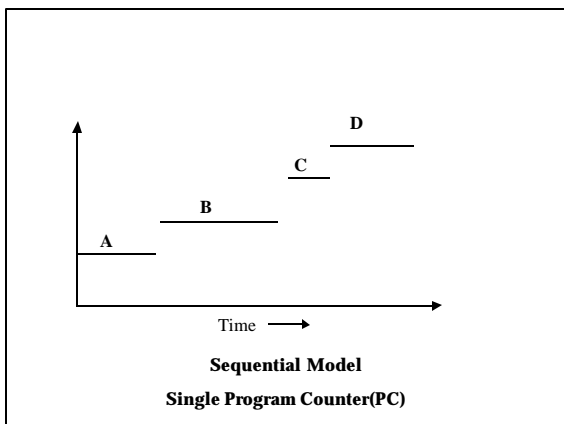
www.geocities.com/anjibcs57manjib@mail.com.np

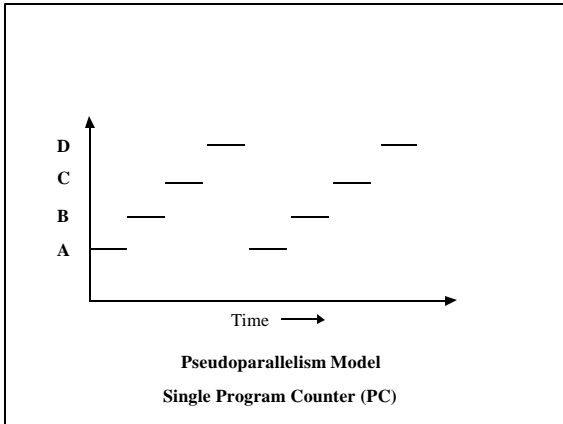
## Process Model

---

- Sequential Model
- Multiprocessor Model
- Pseudoparallelism Model

www.geocities.com/anjibcs58manjib@mail.com.np





### Program Process Analogy

---

**Scenario 1 : A computer scientist is baking a birthday cake for his daughter**

- Computer scientist
- Birthday cake recipe
- ingredients
- activities :
  - reading the recipe
  - fetching the ingredients
  - baking the cake
- CPU
- program
- input data
- processes

www.geocities.com/anjibcs 62 manjib@mail.com.np

### Program Process Analogy

---

**Scenario 2 : Scientist's son comes running in crying, saying he has been stung by a bee**

<ul style="list-style-type: none"> <li>• Scientist records where he was in the recipe</li> <li>• reach first aid book and materials</li> <li>• follow first aid action (high priority job)</li> <li>• on completion of aid, cake baking starts again from where it was left</li> </ul>	<ul style="list-style-type: none"> <li>• the state of the running process saved</li> <li>• another processor fetched</li> <li>• processor switched for new process</li> <li>• completion of high priority job &amp; return back to last one</li> </ul>
--	--

www.geocities.com/anjibcs 63 manjib@mail.com.np

### Key Idea

---

- A process is an activity of some kind.
- It has
  - a program
  - input
  - output and
  - a state
- A single processor may be shared among several processes, with some scheduling algorithm.

www.geocities.com/anjibcs 64 manjib@mail.com.np

### Process States

---

- A process goes through a series of discrete process states.
- Various events can cause a process to change states.
- Different states are:
  - **Running**: if it currently has the CPU
  - **Ready**: if it could use a CPU if one were available
  - **Blocked** if it is waiting for some event to happen before it can be processed.

www.geocities.com/anjibcs 65 manjib@mail.com.np

### How OS manage process?

---

- Only one process may be running at a time, but several processes may be ready, and several may be blocked.
- We can have
  - **Ready list** of ready processes maintained in priority order
  - **Blocked list** of blocked processes normally unordered. (Why and Is that always true?)

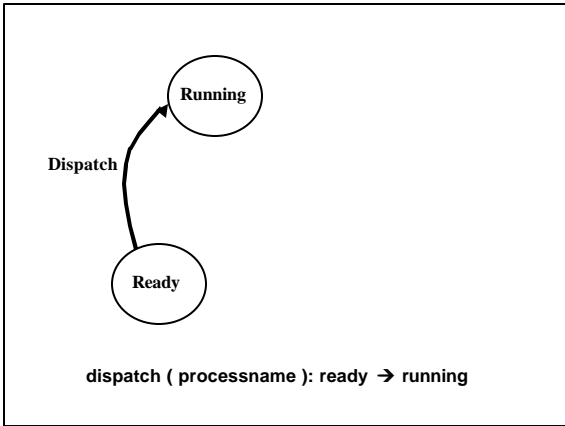
www.geocities.com/anjibcs 66 manjib@mail.com.np

### Process State Transitions

---

- Job is admitted to the system
- Corresponding process is created.
- Process is inserted at the back of the ready list.
- Process gradually moves upward towards the head of ready list as the processes before it completes their turn of using CPU
- If the process is at top and the CPU becomes available, the particular process gets CPU; state changes from ready to running
- The **dispatcher** assigns the CPU for right process(top of the ready list) through the signal **dispatch()**

www.geocities.com/anjibcs 67 manjib@mail.com.np

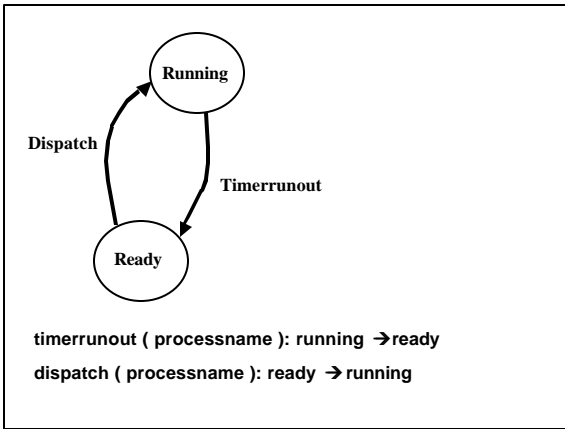


### Process State Transitions (cont..)

---

- To prevent monopolizing the system (either accidentally or maliciously) by any process, OS set a hardware interval timer to allow this user to run for specific time interval or quantum.
- If a running process does not release the CPU before quantum (time interval) expiration, interrupt generation from OS to regain system CPU control
- Timer generates the **timerrunout()** signal
- The current running process is made to stop using CPU and sent to ready list
- Top of the ready list process acquires CPU with the change of state from ready to running

www.geocities.com/anjibcs 69 manjib@mail.com.np

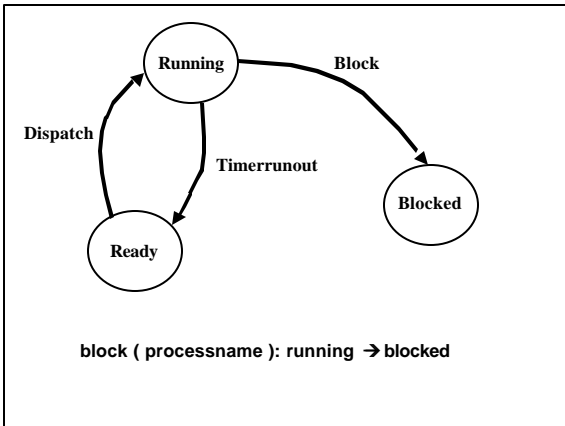


### Process State Transitions (cont..)

---

- If a running process initiates an input/output operation before quantum expirations, running process voluntarily release the CPU
- The **block()** signal is generated
- The process is sent to the blocked list
- Another top of the ready process acquires the CPU changes to running

www.geocities.com/anjibcs 71 manjib@mail.com.np

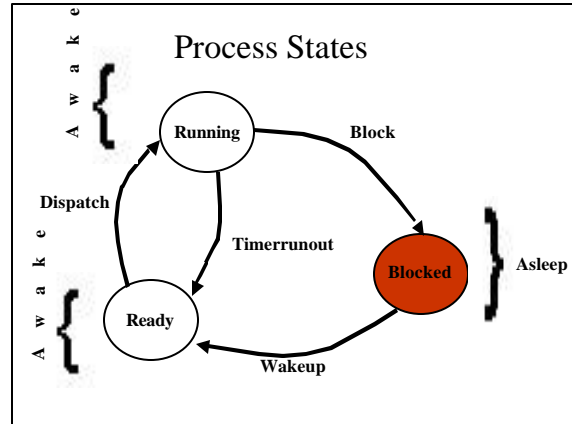


### Process State Transitions (cont..)

- When I/O operation (or some other event the process is waiting for) completes, the process makes the transition from the blocked state to the ready state.
- The **wakeup()** signal is generated.

**wakeup(processname): blocked → ready**

www.geocities.com/anjibcs 73 manjib@mail.com.np



### Can you name the only state transition signal initiated by the user process itself?

**block**

www.geocities.com/anjibcs 75 manjib@mail.com.np

### PCB

- Process Control Block (PCB) is a data structure containing certain important information about the process including
  - the current state of the process
  - unique identification of the process
  - a pointer to the parent process
  - pointers to the child processes
  - the process's priority
  - pointers to locate the process's memory
  - pointers to allocated resources
  - a register save area
  - the processor it is running on ( in multiprocessor system)

www.geocities.com/anjibcs 76 manjib@mail.com.np

### Quiz I

- Is process a synonym of program? If not how different they are?
- What do you understand by pseudo-parallelism? What makes pseudo-parallelism more interesting to OS designers in comparison to full-parallelism?
- Draw complete process states transition.

www.geocities.com/anjibcs 77 manjib@mail.com.np

### Operations on Process

- create a process
- destroy a process
- suspend a process
- resume a process
- change a process's priority
- block a process
- wakeup a process
- dispatch a process
- enable a process for communication (IPC - interprocess communication)

www.geocities.com/anjibcs 78 manjib@mail.com.np

### Process Creation

---

- Creating a process involves many operations including:
  - name the process
  - insert it in the system's known processes list (process table)
  - determine the process's initial priority
  - create the process control block(PCB)
  - allocate the process's initial resources

www.geocities.com/anjibcs 79 manjib@mail.com.np

### Process Creation Events

---

- There are four principal events that cause processes to be created:
  - System initialization
  - Execution of process creation system call by a running process
  - A user request to create a new process
  - Initiation of a batch job.

www.geocities.com/anjibcs 80 manjib@mail.com.np

### System Initialization

---

- When an OS is booted, typically several processes are created.
- Some of these are foreground, others are background.
- Background process is also known as daemons.
- E.g mail service, server request

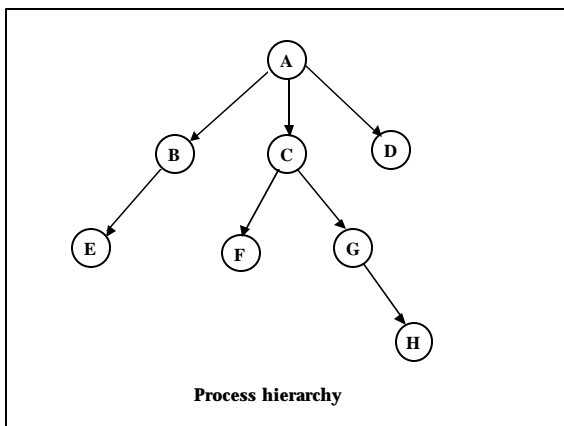
www.geocities.com/anjibcs 81 manjib@mail.com.np

### Child Process

---

- A process may create a new process.
- If it does, the creating process is called the parent process and the created process is called the child process.
- They are created to help parent do its job.
- Such creation yields a hierarchical process structure.

www.geocities.com/anjibcs 82 manjib@mail.com.np



### Others creation

---

- User can start a process by typing a command or (double) clicking an icon.
- And in batch system when the OS decides that it has the resources to run another job, it create a new process and runs the next job from the input queue in it.

www.geocities.com/anjibcs 84 manjib@mail.com.np

### Destroying Process

- Destroying a process involves:
  - erase it from the system
  - resources are returned to the system
  - it is purged from any system lists or tables.
  - PCB is erased
- What if process has child?

### Process Termination Condition

- Normal Exit (voluntary)
  - After work finish
- Error Exit (voluntary)
  - executing illegal instruction
  - referencing nonexistent memory
  - divide by zero
- Fatal Error (involuntary)
  - call nonexistent file
- Killed by another process (involuntary)

### Suspend and Resume

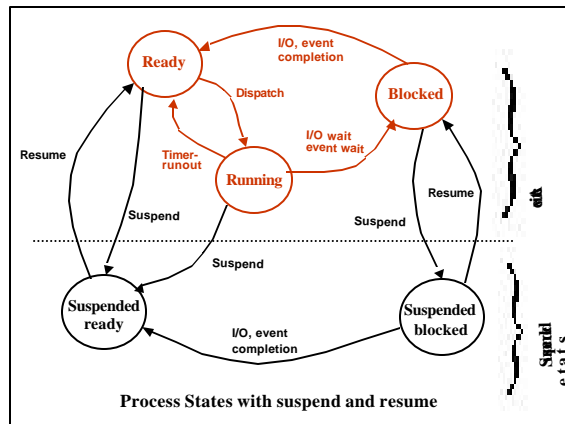
- A suspension may be initiated either by a process or by another process.
- In uniprocessor system a running process may suspend itself.
- On a multiprocessor system, a running process may be suspended by another process running at that moment on a different processor.
- Suspension normally last only brief periods of time.

### Why Suspend and Resume ?

- These operations are important for several reasons:
  - If a system is functioning poorly and may fail, then current processes may be suspended to be resumed after the problem is corrected.
  - A user suspicious about the partial results of process may suspend it (rather than aborting it) until the user can certain whether or not the process is functioning correctly.
  - In response to short-term fluctuations in a system load, some processes may be suspended and resumed later when the load settles back to normal levels

### Suspend and Resume operations

- A running process may be suspended by `suspend(processname): running → suspendedready`
- A ready process may be suspended by `suspend(processname): ready → suspendedready`
- A suspendedready process may be made ready by `resume(processname): suspendedready → ready`
- A blocked process may be suspended by `suspend(processname): blocked → suspendedblocked`
- A suspendedblocked process may be resumed by `resume(processname): suspendedblocked → blocked`



### An Issue

---

- Arguments
  - Why suspending a blocked process?
  - Why not wait for blocked process to complete the I/O completion or event completion occur and process become ready and send to suspendedready state?
- Justification
  - completion may never come or it may be delayed indefinitely.
  - So designer has two choice either performing the suspension of the blocked process or setting up a mechanism such that when the completion occurs, the suspension will be made from the ready state.
  - Because the suspension is normally a high-priority activity, it should be performed immediately.

www.geocities.com/anjibcs 91 manjib@mail.com.np

### completion()

---

- When the completion finally occurs, the suspendedblocked process makes the transition:  
**completion (processname): suspendedblocked → suspendedready**

www.geocities.com/anjibcs 92 manjib@mail.com.np

### Priority Changing

---

- Changing the priority of a process normally involves nothing more than modifying the priority value in the process' control block.

www.geocities.com/anjibcs 93 manjib@mail.com.np

### Quiz II

---

- What are condition for process creation?
- What are the condition for process termination?
- Why process need to be suspended?
- Why there is need for suspended block state?

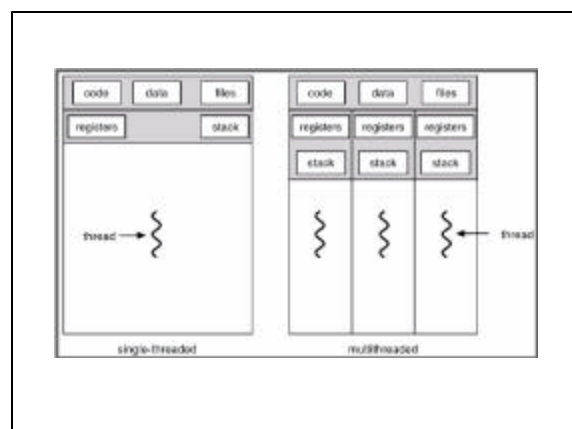
www.geocities.com/anjibcs 94 manjib@mail.com.np

### Threads

---

- A thread is a basic unit of CPU utilization.
- It also known as lightweight process (LWP).
- It has its own
  - ID
  - Program counter
  - Register
  - Stack
  - State
- But share
  - code section
  - data section
  - open files
- A traditional process has a single thread of control.

www.geocities.com/anjibcs 95 manjib@mail.com.np



### Why multiple thread?

---

- If the process has a multiple thread , it can do more than one task at a time.
- For example in word processor reading keystroke and spelling checking can be done.
- Also thread is suitable for running an application that required to perform several similar tasks.
- For example Web server have to serve many clients.

www.geocities.com/anjibcs
97
manjib@mail.com.np

### Benefits

---

- **Responsiveness**
  - If one "task" takes too long, other "tasks" can still proceed
- **Resource sharing**
  - Grammar checker can check the buffer as it is being typed
- **Economy**
  - Process creation is expensive (spell checker)
- **Utilization of multiprocessor architectures**
  - If we had four processors (say), the word processor can fully leverage them

www.geocities.com/anjibcs
98
manjib@mail.com.np

### Thread Types

---

- **User level threads**
  - Implemented as a library
  - Fast to create and manage
  - But cannot have blocking system calls
- **Kernel level threads**
  - Slower to create and manage
  - Blocking system calls are no problem
  - Most OS's support these threads

www.geocities.com/anjibcs
99
manjib@mail.com.np

### Threading Models

---

- One to One model
- Many to One model
- Many to Many model

www.geocities.com/anjibcs
100
manjib@mail.com.np

### One to One Model

---

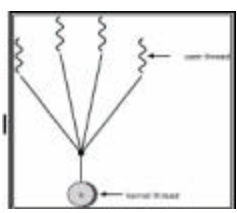
- Map each user thread to one kernel thread
- It provide concurrency by allowing another thread to run when a thread makes a blocking system call
- Also allows multiple threads to run in parallel on multiprocessors.
- Main drawback is that creating a user thread requires creating the corresponding kernel thread.
- Burden problem may arise so limitation impose.

www.geocities.com/anjibcs
101
manjib@mail.com.np

### Many to One Model

---

- Map many user threads to a single kernel thread
- Cannot exploit multiprocessors
- But entire process will block if a thread makes a blocking system call.

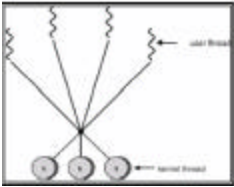


www.geocities.com/anjibcs
102
manjib@mail.com.np

### Many to Many Model

---

- Map  $m$  user threads to  $n \times m$  kernel threads
- Can create as many user thread as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor.



The diagram illustrates the Many to Many Model. It shows three circles at the bottom labeled 'Kernel Thread'. From each kernel thread, multiple lines (representing threads) extend upwards to several wavy lines at the top labeled 'User Thread'. This indicates that multiple user threads are mapped to multiple kernel threads, which can then run in parallel on a multiprocessor.

www.geocities.com/anjibcs 103 manjib@mail.com.np

### Threading Issues

---

- Cancellation
- Signal Handling
- Pooled Threads
- Thread-Specific Data

www.geocities.com/anjibcs 104 manjib@mail.com.np

### Cancellation

---

- 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**".
- Two types of cancellation are:
  - Asynchronous cancellation: One thread immediately terminates the target thread.
  - Deferred cancellation: The target thread can periodically check if it should terminate.
- In asynchronous cancellation, problem may occur where resources have been allocated to a cancelled thread but not in the case of deferred cancellation.

www.geocities.com/anjibcs 105 manjib@mail.com.np

### Signal Handling

---

- A signal is used to notify a process about particular event.
- Basic steps
  - A signal is generated by the occurrence of a particular events.
  - A generated signal is delivered to a process.
  - One delivered, the signal must be handled.
- Signal can be
  - Synchronous
  - Asynchronous

www.geocities.com/anjibcs 106 manjib@mail.com.np

### Synchronous and Asynchronous Signals

---

- Synchronous signals are delivered to the same process that performed the operation causing the signal. e.g illegal operation
- When a signal is generated by an event external to a running process, that process receives the signal asynchronously e.g Ctrl+C
- Typically an asynchronous signal is sent to another process.

www.geocities.com/anjibcs 107 manjib@mail.com.np

### Handler

---

- Every signal may be handled by one of the possible handlers:
  - A default signal handler run by kernel
  - A user-defined signal handler
- Default signal handling may be overridden by a user-defined signal handler.

www.geocities.com/anjibcs 108 manjib@mail.com.np

## Handling Signal

- Handling signals in a single-threaded programs is straightforward; signals are always delivered to a process.
- However, delivering signals is more complicated in multithreaded programs, as process may have several threads.
- Various options are:
  - Deliver the signal to the thread to which the signal applies.
  - Deliver the signal to every thread in the process.
  - Deliver the signal to certain thread in the process.
  - Assign a specific thread to receive all signals for the process.

www.geocities.com/anjibcs

109

manjib@mail.com.np

## Thread Pools

- Like in a multithreading a web server how many threads to create?
- Unlimited threads could exhaust system resources.
- So the solution will be to use thread pools.
- The general idea behind a thread pool is
  - to create a number of threads at process startup and place them into a pool.
  - allocate threads to request if available
  - else wait for threads to be free

www.geocities.com/anjibcs

110

manjib@mail.com.np

## Benefit

- It is usually faster to service a request with an existing thread than waiting to create a thread.
- It limits the number of threads that exist at any one point.

www.geocities.com/anjibcs

111

manjib@mail.com.np

## How to set pool size?

- Heuristic calculation based on the resources and the expected number of concurrent clients.  
OR
- Dynamically adjust according to usage patterns.

www.geocities.com/anjibcs

112

manjib@mail.com.np

## Thread-Specific Data

- Although threads share the data of the process, it may be required to keep a separate copy of data for a particular thread, such data is called thread-specific data.

www.geocities.com/anjibcs

113

manjib@mail.com.np

## Quiz III

- How are threads related to a process?
- List out the benefits of multiple threading.
- How are threads modeled?
- Discuss different threading issues.

www.geocities.com/anjibcs

114

manjib@mail.com.np

### IPC

---

- Processes need to communicate with each other in a well-structured way without using interrupts.
- IPC Issues
  - Information Exchange
  - Avoid engagement with each other
  - Proper Sequencing
- Equally applied to threads (don't be confuse)

www.geocities.com/anjibcs 115 manjib@mail.com.np

### Scenario I

---

www.geocities.com/anjibcs 116 manjib@mail.com.np

### Scenario II

**Local Variable**  
*Next\_free\_slot*

Process A

Process B

**Two Shared Variables:**  
*Out* → Points to the next file to be printed  
*In* → next free slot in the directory

- A reads in and stores 6 in local variable next\_free\_slot
- Quantum Expire
- B reads in and stores 6 in local variable next\_free\_slot
- B Update in to be an 7 and goes off
- A runs again, find next\_free\_slot 6
- Write file name in 6, erasing B's file
- A Update in to be 7.

Printer daemon running through spooler directory management

www.geocities.com/anjibcs 118 manjib@mail.com.np

### Race Condition

---

- Thus the situation where two or more processes are sharing data and the final result depends on who runs precisely when, are called **race condition**.

*Believe me such debugging such is boring*

www.geocities.com/anjibcs 118 manjib@mail.com.np

### How to avoid race condition?

---

- We can have **mutual exclusion** using **critical region (or critical section)**.
- When one process increments the shared variable, all other processes desiring to do so should be kept waiting; when that process has finished accessing the shared variable, one of the processes waiting to do so should be allowed to proceed.
- That part of the program where the shared memory is accessed is called **critical section/ critical region**.

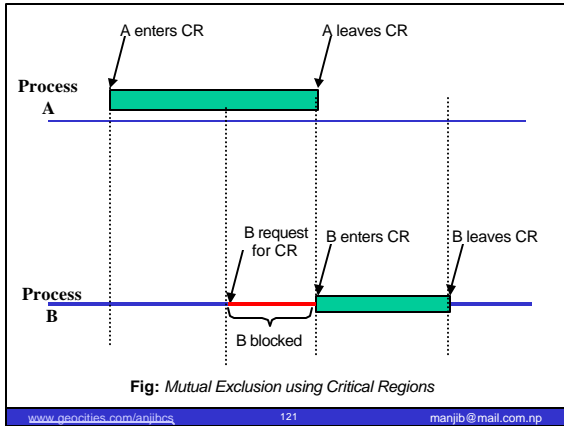
www.geocities.com/anjibcs 119 manjib@mail.com.np

### Conditions to avoid race

---

- No two processes may be simultaneously inside their critical regions
- No assumptions may be made about the speeds or the number of CPUs
- No process running outside its critical region may block other processes
- No process should have to wait forever to enter its critical region.

www.geocities.com/anjibcs 120 manjib@mail.com.np



## ME with Busy Waiting

- Disabling Interrupts
- Lock Variables
- Strict Alternation
- Peterson's Algorithm
- Hardware Solution: TSL Instruction

## Disabling Interrupts

Each process disable all interrupts just after entering its critical region and re-enable them just before leaving it. No clock interrupt; no CPU switching to another process until process turn on the interrupt.

Problem:

- Interrupt turning off power to user process
- The chance of never turned on – is a disaster
- Not applicable in multiprocessor system

## Lock Variables

- Use a single shared (lock) variable, initially set to 0.
- A process wants to enter CR checks the lock
- If the lock 0, process sets it to 1, enter CR, sets 0 again at the end
- If the lock 1, process keeps waiting until it becomes 0

Problems:

Spooler directory like problem; one process reads lock as 0 but before setting it to 1 another process is scheduled and enters the CR; when last process runs again, it will also set 1 and can have two processes at CR

## Strict Alternation

```

while(TRUE) {
    while (turn !=0); /* wait */
    critical_region();
    turn = 1;
    noncritical_region();
}
while(TRUE) {
    while (turn !=1); /* wait */
    critical_region();
    turn = 0;
    noncritical_region();
}
    
```

Problem:

What happens if process 0 just finished CR work and again need to enter CR and process 1 is still busy at nonCR work?

Thus violates no. 3 condition

Avoided all race conditions but Not allowed to spool two in a row.

## Refinement

```

#define TRUE 1
#define FALSE 0
#define i 0
#define j 1
int interested[2];

while(TRUE) {
    interested[i] = TRUE;
    while (interested[j]); /* wait */
    critical_region();
    interested[i] = FALSE;
    noncritical_region();
}

while(TRUE) {
    interested[j] = TRUE;
    while (interested[i]); /* wait */
    critical_region();
    interested[j] = FALSE;
    noncritical_region();
}
    
```

Problem is Indefinite Postponed

### Peterson's Algorithm

```

#define FALSE 0
#define TRUE 1
#define N 2 /* number of processes */
int turn; /* whose turn is it ? */
int interested[N]; /* all values initially 0 (FALSE) */
void enter_region(int process) { /* process is 0 or 1 */
    int other; /* other process number */
    other = 1 - process /* opposite of process */
    interested[process] = TRUE; /* show that interested to
enter CR */
    turn = process; /* flag set */
    while (turn == process && interested[other] == TRUE); /* null
stmt */
}
    
```

### TSL Instruction

```

enter_region:
    TSL REGISTER,LOCK /* copy lock to register and set lock to 1
    CMP REGISTER,#0 /* was lock zero?
    JNE enter_region /* if it was non-zero, lock was set, so loop
    RET /* return to caller; CR entered

Leave_region:
    MOVE LOCK,#0 /* store a 0 in lock
    RET /* return to caller
    
```

- ### How Works?
- Hardware instruction Test and Set Lock (TSL)
  - Reads the contents of the memory word into a register and then stores a nonzero value at that memory address
  - The operation reading the word and storing into it are guaranteed to be indivisible – no other processor can access the memory word until the instruction is finished
  - CPU lock the memory bus to prohibit other CPUs from accessing memory until it is done
  - When lock variable is 0, any process may set it to 1 using the TSL instruction and then read write shared memory; when it is done the process sets lock back to 0

### Busy Waiting Alternate?

- ❑ If a process wants to enter its CR
  - ❑ checks to see if the entry is allowed
  - ❑ If not, process just sits in a tight loop waiting until it is

Waste of CPU time for NOTHING!

Possibility of  
SLEEP and WAKEUP pair  
instead of WAITING

### Producer-Consumer Problem

```

#define N 100 /* number of slots in the buffer */
int count = 0; /* number of items in the buffer */
void producer (void){
    while(TRUE){
        /* repeat forever
        produce_item(); /* generate next item
        if(count== N) sleep(); /*if buffer is full go to sleep
        enter_item(); /*put item in buffer
        count=count+1; /*increment counts of items
        if(count==N) wakeup(consumer);
    }
}
void consumer (void){
    while(TRUE){
        /* repeat forever */
        if(count==0) sleep(); /*if buffer is empty, go to sleep
        remove_item(); /*take item out of buffer */
        count = count -1; /*increment counts of items in buffer
        if (count == N-1) wakeup(producer);
        consume_item();
    }
}
    
```

- ### Why not perfect?
- What happens if
    - the buffer empty and consumer just read count
    - scheduler stops consumer and starts producer
    - producer enters an item in buffer increment count and also calls wakeup for consumer
    - consumer not yet asleep, wakeup signal lost
    - consumer on next run has the count value 0 from last read so go to sleep
    - producer keeps on producing and fill the buffer and go to sleep too

## Semaphores

- E.W. Dijkstra (1965) suggested using an integer variable – **Semaphore** – to count the number of wakeups.
- It could have the value 0, indicating no wakeups were saved, or some positive value if one or more wakeups were pending
- Operations Down & Up (P & V in Dutch)
  - Down: checks if value greater than 0;
    - Yes - decrement (i.e. uses one stored wakeup) & continue
    - No – process is put to sleep without completing down
  - Up : increment the value; if process(es) sleeping, unable to complete early down operation, now allowed to complete down operation of one at a random; this one Up operation could not wakeup all; rather no. of sleeping processes is now 1 less

www.geocities.com/anlibcs 133 manjib@mail.com.np

```

#define N 100 // number of slots in the buffer */
typedef int semaphore; //semaphores are special kind of int */
semaphore mutex =1; // control access to CR */
semaphore empty =N; // counts empty buffer slots */
semaphore full =0; // counts full buffer slots */

void producer (void) {
    int item;
    while(TRUE) { // TRUE is the constant 1 */
        produce_item(&item); // generate next item */
        down(&empty); // decrement empty count */
        down(&mutex); // enter CR */
        enter_item(item); // put new item in buffer */
        up(&mutex); // leave CR */
        up(&full); // increment count of full slots */
    }
}

void consumer (void) {
    int item;
    while(TRUE) { // infinite loop */
        down(&full); // decrement full count */
        down(&mutex); // enter CR */
        remove_item(&item); // take item from buffer */
        up(&mutex); // leave CR */
        up(&empty); // increment count of empty slots */
        consume_item(item); // do something with the item */
    }
}
    
```

## Monitors

- Collection of procedures, variables and data structure.
- Processes may call the procedures in a monitors whenever they want to, but they can't directly access the monitor's internal data structures from procedures declared outside the monitors.  
(Remember something???)

```

monitor example
integer i;
condition c1, c2;
procedure producer(x);
--
end;
procedure consumer(x);
--
end;
end monitor;
                
```

www.geocities.com/anlibcs 135 manjib@mail.com.np

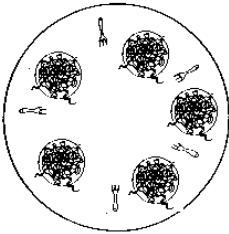
## Message Passing

- Widely practiced interprocess communication (IPC)
- System calls rather than language constructs
  - send(destination, &message)*
  - receive(source, &message)*
- But some design issues
  - Acknowledgement → Make sure message receive.
  - Duplication → Make sure no duplicate message send
  - Authentication → Send unambiguous message

www.geocities.com/anlibcs 136 manjib@mail.com.np

### IPC Problem – Dining Philosophers

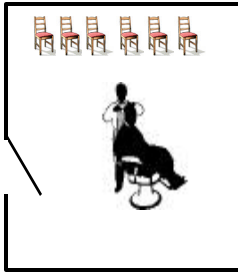
- Five philosophers seated around a circular table
- Each philosopher has a plate of spaghetti
- One fork between two plates
- Spaghetti is so slippery, each philosopher needs 2 forks to eat that
- Life of philosopher – alternate periods of *eating* & *thinking*
- Hungry and waiting states
- *No one should be starved*



www.geocities.com/anlibcs 137 manjib@mail.com.np

### IPC Problem – The Sleeping Barber

- One barber
- One barber chair
- Several waiting chairs
- If no customer – barber sits down in barber chair & falls asleep
- When a customer arrive, he has to wake up barber
- Additional customers either sit down on empty chair or leave the shop if chairs full



www.geocities.com/anlibcs 138 manjib@mail.com.np

## Assignment II

---

- Define the term mutual exclusion, critical region.
- Analysis following methods for achieving mutual exclusion with their problems
  - Disabling Interrupts
  - Lock Variable
  - Strict Alternation
- Explain the following solutions for achieving mutual exclusion:
  - Peterson's Solution
  - TSL Instruction

www.geocities.com/anjibcs
139
manjib@mail.com.np

## Scheduling

---

- In multiprogrammed system, multiple processes can be competing for the CPU at the same time.
- OS must select the processes in turn.
- This is handle by the part of OS known as Scheduler using different scheduling algorithms.

www.geocities.com/anjibcs
140
manjib@mail.com.np

## When to Schedule?

---

- When a new process is created.
- When a process exits.
- When a process blocks.
- When an I/O interrupts occurs.

www.geocities.com/anjibcs
141
manjib@mail.com.np

## Scheduling Objectives

---

- **Be Fair**
  - Comparable processes should get comparable service
- **Policy Enforcement**
  - Apply the policies
- **Balance Resource Use**
  - Keeping all parts of the system busy when possible.
- **Meet Deadlines**
  - avoid losing data

www.geocities.com/anjibcs
142
manjib@mail.com.np

## Scheduling Objective (cont..)

---

- **Maximize Throughput**
  - maximize jobs per hour
- **Minimize Turnaround Time**
  - minimize time between submission and termination
- **Maximize CPU Utilization**
  - Keeping CPU busy all the time
- **Quick Response Time**
  - Response to requested quickly
- **Maintain Proportionality**
  - Meet user's expectations

www.geocities.com/anjibcs
143
manjib@mail.com.np

## Preemptive Vs. Nonpreemptive

---

<ul style="list-style-type: none"> <li>• A scheduling discipline is preemptive if the CPU can be taken away.</li> <li>• Preemptive scheduling is useful in systems in which high-priority processes require rapid attention.</li> <li>• Context switching involves overhead.</li> </ul>	<ul style="list-style-type: none"> <li>• A scheduling discipline is nonpreemptive if, once a process has been given the CPU, the CPU cannot be taken away from that process.</li> <li>• Treatment of all processes is fairer.</li> <li>• Shot jobs are made to wait by longer jobs.</li> </ul>
---	--

www.geocities.com/anjibcs
144
manjib@mail.com.np

### First-Come First-Served

---

- Nonpreemptive
- Process dispatch according to arrival time to queue
- It is fair
- Unimportant jobs may make important jobs wait.

Ready List

C

B

A

→

CPU

→

Completion

www.geocities.com/anjibcs
146
manjib@mail.com.np

Process	Burst Time	Waiting Time
P <sub>1</sub>	24	0
P <sub>2</sub>	3	24
P <sub>3</sub>	3	27

$$\text{Avg. Waiting Time} = \frac{0 + 24 + 27}{3} = 17 \text{ milli sec onds}$$

What will be average waiting time if the processes arrive in the order P<sub>2</sub>, P<sub>3</sub>, P<sub>1</sub>?

3 milliseconds

www.geocities.com/anjibcs
147
manjib@mail.com.np

### Shortest Job First (SJF)

---

- Nonpreemptive
- Process with the smallest estimated run-time-to-completion is run next.
- Reduces average waiting time over FIFO. (see previous example)
- Favor short jobs
- Requires precise knowledge of how long a process will run. (Not usually available)

www.geocities.com/anjibcs
147
manjib@mail.com.np

### Shortest Remaining Time (SRT)

---

- Preemptive counterpart of SJF.
- Scheduler always choose the process whose remaining run time is the shortest.
- New arrival job also may get chance.
- Higher overhead than SJF:
  - should maintain elapsed service time of the running job
  - must handle occasional preemptions
- Longer jobs have an even longer mean waiting time.

www.geocities.com/anjibcs
148
manjib@mail.com.np

### Round Robin (RR)

---

- Preemptive counterpart of FCFS.
- Designed especially for time-sharing systems.
- Each process is assigned a time interval, called its quantum.

Ready List

A

C

B

A

→

CPU

→

Completion

Preemption

www.geocities.com/anjibcs
149
manjib@mail.com.np

### Quantum Size Issue

---

- Should quantum be large or small ?
- Should it be fixed or variable?

Quantum size is critical to the effective use operation of a computer system

www.geocities.com/anjibcs
150
manjib@mail.com.np

➤ *If quantum is very large, each process is given as much time as it needs for completion; RR degenerate to FIFO*

➤ *If quantum is very small, system busy at just switching from one process to another process, the overhead of context switching causes the system efficiency degrading*

### Highest Response Ratio Next (HRN)

- Nonpreemptive scheduling
- Removes the excessive bias against longer jobs and the excessive favoritism toward short new jobs
- Priority of each job is a function not only of the job's service time but also of amount of time the job has been waiting for service

$$\text{Priority} = \frac{\text{time waiting} + \text{service time}}{\text{service time}}$$

**Denominator service time favors for shorter jobs**

**Numerator waiting time favors longer jobs**

www.genitias.com/anjibcs 152 manjib@mail.com.np

### Priority Scheduling

- Each process is assigned a priority.
- Process with the highest priority is allowed to run.
- **High-priority process can run indefinitely.**
- Priorities can be assigned dynamically. Dynamic priority decreasing mechanism can be applied.
- Assign maximum quantum to process and on next highest priority process is given a chance to run.

www.genitias.com/anjibcs 153 manjib@mail.com.np

Process	Burst Time	Priority
P <sub>1</sub>	3	3
P <sub>2</sub>	1	1
P <sub>3</sub>	2	4
P <sub>4</sub>	1	5
P <sub>5</sub>	5	2

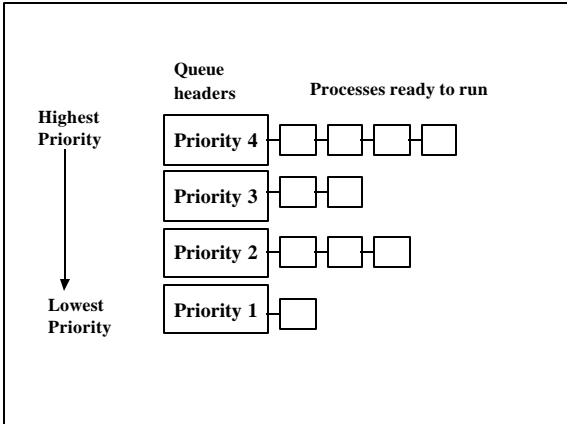
  

Avg. Waiting Time =  $\frac{6+0+9+11+1}{5} = 5.4 \text{ milliseconds}$

### Multilevel Queue Scheduling

- Group of priority class is defined.
- On each class RR scheduling is applied.
- Lower priority class is processed only after all process of upper priority class is processed.
- If process on upper class is inserted while running lower priority class, it is preempted and run upper class process.

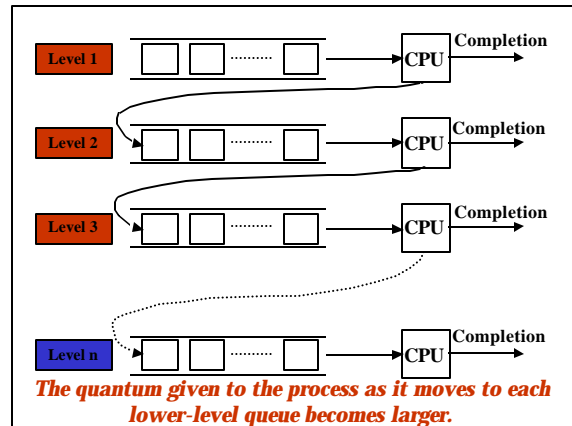
www.genitias.com/anjibcs 155 manjib@mail.com.np



### Multilevel Feedback Queue

- A new process enters the queue at the top level.
- It moves through that queue FIFO until it gets the CPU.
- If job complete it leave.
- Else it is placed at the back of the next lower level queue.
- The process is next serviced when it reaches the head of that queue if the first queue is empty.
- This shifting continue until it reach the bottom level queue, where RR scheduling is applied.

www.geocities.com/anjibcs 157 manjib@mail.com.np



### Guaranteed Scheduling

$$\text{Ratio} = \frac{\text{Actual CPU Time Consumed}}{\text{CPU Time Entitled}}$$

where  $\text{CPU Time Entitled} = \text{Time Since Creation} / n$

**Process with lowest priority is run.**

www.geocities.com/anjibcs 159 manjib@mail.com.np

### Lottery Scheduling

- Each processes is given lottery tickets.
- At the time of selection, scheduler choose lottery ticket at random.
- Winning chance is proportion to the number of tickets it holds.
- Cooperating processes may exchange tickets if they wish.
- Differ from priority scheduling in the sense it give clear sense of priority.

www.geocities.com/anjibcs 160 manjib@mail.com.np

### Fair-Share Scheduling

- What happen if two user A and B having 9 and 1 process to be run in round robin or equal priority fashion?
- User A will get 90% of the CPU and B will get 10% of it.
- So we can divide CPU to individual users also.
- In above case if each user is given 50% of CPU then job order will be  
A1 B1 A2 B1 A3 B1 A4 B1 A5 B1 A6 B1 A7 B1 A8 B1 A9 B1 .....

www.geocities.com/anjibcs 161 manjib@mail.com.np

### THE END

www.geocities.com/anjibcs 162 manjib@mail.com.np

## Unit 3: Deadlocks

Anjib Man Mulepati

www.geocities.com/anjibcs163manjib@mail.com.np


## Agenda

- Deadlock
- Resources Concepts
- Deadlock Condition
- Deadlock Modeling
- Deadlock Prevention
- Deadlock Avoidance
- Deadlock Detection
- Deadlock Recovery

www.geocities.com/anjibcs164manjib@mail.com.np

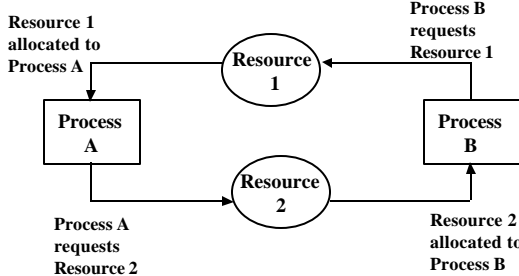
### Scenario I: A Traffic Deadlock

- **No. of automobiles trying to cross**
- **Traffic completely stopped**
- **Not possible without backing some**
- **Normal state only with much annoyance, effort & loss of time**




www.geocities.com/anjibcs165manjib@mail.com.np

### Scenario II: Resource Deadlock



www.geocities.com/anjibcs166manjib@mail.com.np

### Scenario III: Spooling Systems



- In spooling system, jobs are temporarily routed to a much faster alternate device. (e.g routed to HDD before printer)
- In some spooling the complete output from a program must be available before actual printing can begin; then several partially completed jobs become deadlocked in case the available space is already full before any job completes

Solutions : Saturation threshold  
Dynamic allocation

www.geocities.com/anjibcs167manjib@mail.com.np

### Scenario IV: One Process

# Can you think ?

www.geocities.com/anjibcs168manjib@mail.com.np

## What is resources?

- An operating system is primarily a resource manager.
- Resource can be either hardware device (e.g. a tape drive) or piece of information (e.g. record in a database)

www.geocities.com/anjibcs

169

manjib@mail.com.np

## Preemptable Vs. Nonpreemptable

- Preemptible resource is one that can be taken away from the process owning it with no ill effect, e.g. memory
- Non-preemptible resource is one that can not be taken away from its current owner without causing the computation to fail, e.g. printer

www.geocities.com/anjibcs

170

manjib@mail.com.np

## Shared Vs. Dedicated

- Some resources may be shared among several processes e.g. Main memory, files
- While others are dedicated to single processes e.g. disk drives

*When we call particular resources shared, we must be careful to state whether they may be used by several processes simultaneously, or whether they may be used by several processes, but only by one at a time.*

www.geocities.com/anjibcs

171

manjib@mail.com.np

## Resource Utilization Steps

- **Request:** If the resource is not available when it is requested, the requesting process is forced to wait.
- **Use:** The process can operate on the resource.
- **Release:** The process release the resource.

www.geocities.com/anjibcs

172

manjib@mail.com.np

## Definition

*A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.*

www.geocities.com/anjibcs

173

manjib@mail.com.np

## Deadlock Condition

- **Mutual exclusion condition** : Processes claim exclusive control of the resources they require
- **Hold and wait condition** : Processes hold resources already allocated to them while waiting for additional resources
- **No preemption condition** : Resources cannot be removed from the processes holding them until the resources are used to completion
- **Circular wait condition** : A circular chain of processes exists in which each process holds one or more resources that are requested by the next process in the chain

www.geocities.com/anjibcs

174

manjib@mail.com.np

### Deadlock Modelling

- Deadlock can be described more precisely in terms of directed graph called a **system resource allocation graph**.

www.geocities.com/anjibcs 175 manjib@mail.com.np

### Resource Allocation Graph

- This graph  $G=(V,E)$  consists of
  - a set of vertices  $V$  and
  - a set of edges  $E$
- The set of vertices  $V$  is partitioned into
  - a set of active processes  $P=\{P_1, P_2, \dots, P_n\}$
  - a set of all resource  $R=\{R_1, R_2, \dots, R_m\}$
- A directed edge from process  $P_i$  to resource  $R_j$  is denoted by  $P_i \rightarrow R_j$ ; it signifies that process  $P_i$  requested an instance of resource type  $R_j$ .
- A directed edge from resource  $R_j$  to process  $P_i$  is denoted by  $R_j \rightarrow P_i$ ; it signifies that an instance of resource type  $R_j$  has been allocated to process  $P_i$ .

www.geocities.com/anjibcs 176 manjib@mail.com.np

### Remember

- If the graph contains cycle then deadlock may exist.
- If each resource type has exactly one instance, then a cycle implies that a deadlock has occurred.
- If each resource type has several instance, then a cycle does not necessarily imply that a deadlock has occurred.

www.geocities.com/anjibcs 177 manjib@mail.com.np

Which one have deadlock?

www.geocities.com/anjibcs 178 manjib@mail.com.np

**Exercise:** Imagine we have three processes **A, B and C** and three resources **R, S and T**. Draw resource allocation graph and detect whether there will be deadlock or not for following set of command.

(a)	(b)	(c)
A request R	A request R	A request R
A request S	B request S	C request T
A release R	C request T	A request S
A release S	A request S	C request R
B request S	B request T	A release R
B request T	C request R	A release S
B release S		
B release T		
C request T		
C request R		
C release T		
C release T		

www.geocities.com/anjibcs 180 manjib@mail.com.np

### Quiz

- Give an example of a deadlock taken from politics.
- What are four necessary conditions for deadlock?

www.geocities.com/anjibcs 180 manjib@mail.com.np

### Deadlock Handling

---

- Deadlock handling strategies :
  - Just ignore the problem altogether
  - Prevention, by structurally negating one of the four required conditions
  - Dynamic avoidance by careful resources allocation
  - Detection and recovery

www.geocities.com/anjibcs 181 manjib@mail.com.np

### The Ostrich Algorithm

---

- Pretend there is no problem at all.
- Reaction:
  - Mathematician find it totally unacceptable and say that it must be prevent at all costs.
  - Engineers ask how frequent it happen?

www.geocities.com/anjibcs 182 manjib@mail.com.np

### Deadlock Prevention

---

- Concern is to deny any of the four necessary condition.
- Prevention is a clean solution
- But it can often result in poor resource utilization.

*Nevertheless, deadlock prevention methods are widely practiced.*

www.geocities.com/anjibcs 183 manjib@mail.com.np

### Deny exclusive use of the resource

---

- Sharable resource do not require mutually exclusive access, and thus cannot be involved in a deadlock e.g. read only files.
- Whereas on non-shareable resources mutual-exclusion condition must hold.
- But the question “ **Won't it create chaos?**”
- Solution: Spooling
- Some suggest we don't have to break this condition since we want to allow *dedicated resources*.

www.geocities.com/anjibcs 184 manjib@mail.com.np

### Denying the “wait for” Condition

---

- Resource grant on an “**all or none**” basis.
  - if all resources needed for processing are available then granted and allowed to process
  - if complete set of resources is not available, the process must wait until the set available
- While the process waits, it does not hold any resources.

www.geocities.com/anjibcs 185 manjib@mail.com.np

### Problems

---

- Serious waste of resources.
  - since many of the resources may be allocated but unused for a long period.
- Starvation is possible.
  - A process that needs several popular resources may have to wait indefinitely, because at least one of the resources that it needs is always allocated to some other process.

www.geocities.com/anjibcs 186 manjib@mail.com.np

### Alternative Approach

---

- Divide a program into several program steps that run relatively independently of one another.
- Then resource allocation can be controlled by step rather than for the entire process.
- Problem:
  - Involves a greater overhead in the design of application system, as well as in their execution.

www.geocities.com/anjibcs
187
manjib@mail.com.np

### Denying the “No-Preemption”

---

- If a process is holding some resources and requests another resource that cannot be immediately allocated to it, then process must release its held resources and, if necessary, request them again together with the additional resources.
- Problem:
  - When a process release resources the process may lose all its work to that point.
  - Possibility of starvation

www.geocities.com/anjibcs
188
manjib@mail.com.np

### Denying the “Circular Wait”

---

- All resources are uniquely numbered, and process must request resources in linear ascending order
- With this rule, the resource allocation graph can never have cycles.

www.geocities.com/anjibcs
189
manjib@mail.com.np

Deadlock will occur if *A request j and B request i*

Assume  $i \neq j$

If  $i > j$ , then A is not allowed to request j because that is lower than what it already has.

If  $i < j$ , then B is not allowed to request i because that is lower than what it already has.

Thus deadlock is impossible.

### Problem

---

- Same sequence may not be valid to another process.
- When new resource is added re-sequencing is needed again.
- Doesn't not provide elegant way of developing application,.

www.geocities.com/anjibcs
191
manjib@mail.com.np

### Summary

---

Condition	Approach
Mutual Exclusion	Spool everything
Hold and Wait	Request all resources initially
No preemption	Take resources away
Circular Wait	Order resources numerically

www.geocities.com/anjibcs
192
manjib@mail.com.np

### Deadlock Avoidance

---

- Avoidance does not pre-condition the system to remove all possibility of deadlock.
- Allow the possibility of deadlock to loom, but whenever a deadlock is approached, it is carefully sidestepped.
- Require additional information about how resources are to be requested.
- The most famous deadlock-avoidance algorithm is *Dijkstra's Banker's Algorithm*.

www.geocities.com/anjibos
193
manjib@mail.com.np

### Banker's Algorithm

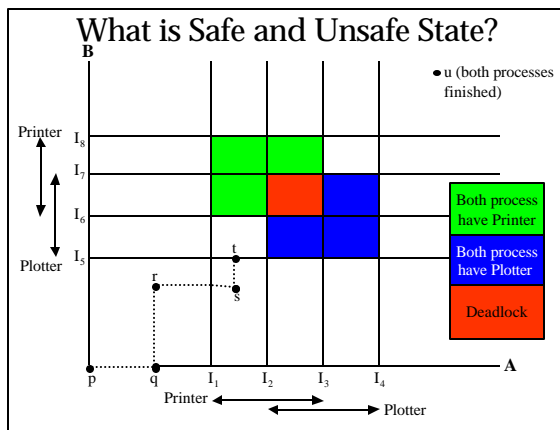
- OS shares a fixed number of resources,  $t$  among a fixed number of users,  $u$ .
- Each user specifies in advance the maximum number of resources (s)he will need during the execution of the job on the system
- OS will accept a user's request if that user's maximum need for resources does not exceed  $t$ .
- A user may obtain / release resource one by one
- Sometime wait may be needed but finite wait.
- The current number of resources allocated to a user will never exceed that user's stated maximum need
- If the OS is able to satisfy a user's maximum need for resources, then the user guarantees that the resources will be used and released to the OS within a finite time

### Banker's Algorithm

---

*Dijkstra's Banker's Algorithm says to allocate resources to users only when the allocation result in **safe states** rather than in **unsafe states***

www.geocities.com/anjibos
195
manjib@mail.com.np



### State I ( Safe )

	Current Loan	Maximum Need
User1	1	4
User2	4	6
User3	5	8
Available	2	

### State II ( Unsafe )

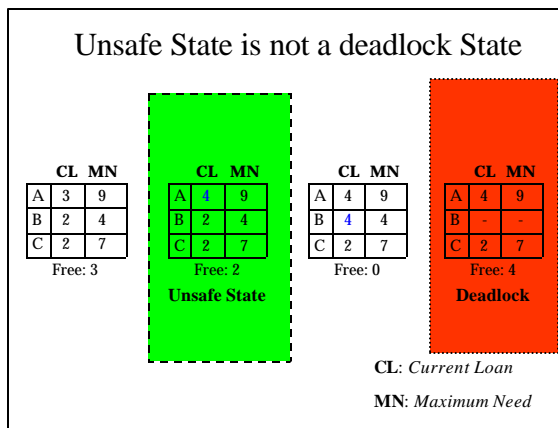
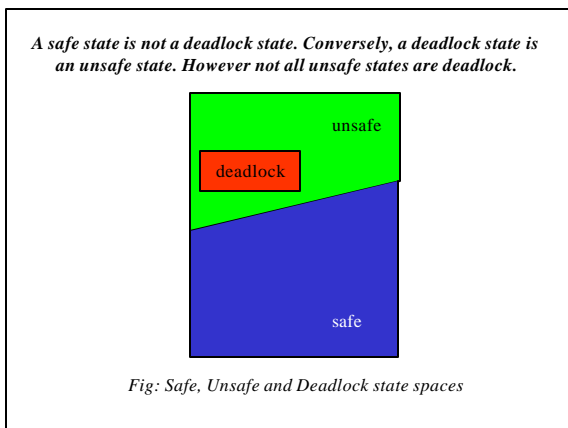
	Current Loan	Maximum Need
User1	1	4
User2	4	6
User3	6	8
Available	1	

### State III ( Transition to unsafe state)

	Current Loan	Maximum Need
User1	1	4
User2	4	6
User3	6	8
<b>Available</b>	<b>1</b>	

### Safe and Unsafe State

- The current state of the system is called *safe* if it is possible for the OS to allow all current users to complete their jobs within a finite time. If not then the current system state is called *unsafe*.



### Exercise

CL	MN
A	1 6
B	1 7
C	2 4
D	4 7

Free: 2

- Is this state safe? Justify your answer.
- What would happen if request from B for one more unit were granted? Will the state be safe?
- In same state if D asks for one more unit, does this lead to a safe state or an unsafe one? What if the request came from C instead of D?

### Multiple Resources

- The banker's algorithm can be generalized to handle multiple resources.
- We need to define following data structure:
  - Current Allocation Matrix  $C=C[i][j]$
  - Request Matrix  $R=R[i][j]$
  - Existing resource vector  $E=(E_1, E_2, \dots, E_m)$
  - Possessed resource vector  $P=(P_1, P_2, \dots, P_m)$
  - Available resource vector  $A=(A_1, A_2, \dots, A_m)$

### Example

Process	A B C	A B C	Total Available
P <sub>0</sub>	0 1 0	7 4 3	<b>E = (10,5,7)</b>
P <sub>1</sub>	2 0 0	1 2 2	<b>Currently Assigned</b> <b>P = (7,2,5)</b>
P <sub>2</sub>	3 0 2	6 0 0	<b>Available</b> <b>A = (3,3,2)</b>
P <sub>3</sub>	2 1 1	0 1 1	
P <sub>4</sub>	0 0 2	4 3 1	
	<b>C</b> <b>Resource</b> <b>Assigned</b>	<b>R</b> <b>Resource</b> <b>Needed</b>	<b>Is system safe?</b>

www.geocities.com/anjibcs
205
manjib@mail.com.np

### Safety Checking Algorithm

1. Look for a row, R, whose unmet resource needs are all smaller than or equal to A. If no such row exists, the system will eventually deadlock since no process can run to completion.
2. Assume the process of the row chosen requests all the resources it need (which is guaranteed to be possible) and finishes. Mark that process as terminated and add all its resources to the A vector.
3. Repeat steps 1 and 2 until either all processes are marked terminated, in which case the initial state was safe, or until a deadlock occurs, in which case it was not.

www.geocities.com/anjibcs
206
manjib@mail.com.np

### Problem

Suppose P<sub>1</sub> request Q<sub>1</sub>=(1,0,2).  
 Can request granted?

www.geocities.com/anjibcs
207
manjib@mail.com.np

### Resource-Request Algorithm

1. If  $Q_i \leq R_i$ , go to step 2. Otherwise raise an error condition, since the process has exceeded its maximum claim.
2. If  $Q_i \leq A_i$ , go to step 3. Otherwise, P<sub>i</sub> must wait, since resources are not available.
3. Assume the system allocate the requested resources to process, compute new state as:
  1.  $A_i = A_i - Q_i$
  2.  $C_i = C_i + Q_i$
  3.  $R_i = R_i - Q_i$
 If the resulting resource-allocation state is safe, P<sub>i</sub> is allocated its resources. However, if the new state is unsafe, then P<sub>i</sub> must wait and the old state is restored.

www.geocities.com/anjibcs
208
manjib@mail.com.np

### Do it

1. What would happen if P<sub>4</sub> request Q<sub>4</sub>=(3,3,0)?
2. What would happen if P<sub>0</sub> request P<sub>0</sub>=(0,2,0)?

www.geocities.com/anjibcs
209
manjib@mail.com.np

### Weakness

- Algorithm requires resources and users be fixed which may not be possible all time.
- Algorithm requires that all request and repay within a finite time but not suitable for real time systems where prompt action is needed.
- Algorithm requires that users states their maximum needs in advance, which is impractical.

www.geocities.com/anjibcs
210
manjib@mail.com.np

### Deadlock Detection and Recovery

---

- Does not attempt to prevent deadlock from occur.
- Rather lets them occur, tries to detect, and then takes some action to recover.

www.geocities.com/anjibcs 211 manjib@mail.com.np

### One resource of each type

---

- Use a resource graph
- If this graph contains one or more cycles, a deadlock exists.
- Any process that is part of a cycle is deadlocked.
- If no cycle exists, the system is not deadlocked.

www.geocities.com/anjibcs 212 manjib@mail.com.np

1. A holds R and wants S  
 2. B holds nothing and wants T  
 3. C holds nothing and wants S  
 4. D holds U and wants S and T  
 5. E holds T and wants V  
 6. F holds W and wants S  
 7. G holds V and wants U

**Is this system deadlocked, and if so, which processes are involved?**

www.geocities.com/anjibcs 215 manjib@mail.com.np

### How to find cycle?

For each node, N in the graph do

1. Set L as empty list, and designate all the arcs as unmarked.
2. Add current node to L and check if the node appears twice
  1. If yes, graph contain cycle and exit
  2. Else go to 3
3. From the given node, check if there are any unmarked outgoing arcs
  1. If yes goto 4
  2. Else goto 5
4. Pick an unmarked outgoing arc at random and mark it. Then follow it to the new node continue from 2
5. This is dead end. Remove it and go back to previous node. Make this current node and goto step 3. If this node is the initial node, the graph do not contain any cycles and the algorithm terminates.

www.geocities.com/anjibcs 216 manjib@mail.com.np

### Multiple Resource of Each Type

---

- Use Safety Checking Algorithm of Banker's Algorithm
- See slide no: 44

www.geocities.com/anjibcs 215 manjib@mail.com.np

### When to detect?

---

- Every time resource request is made
  - early detection is possible
  - but much waste of CPU time
- Check every k minutes
- When CPU utilization has dropped below some threshold.

www.geocities.com/anjibcs 216 manjib@mail.com.np

## Recovery Methods

---

- Recovery through preemption
- Recovery through rollback
- Recovery through killing processes

***None of them are especially attractive***

www.geocities.com/anjibcs
217
manjib@mail.com.np

## Recovery through Preemption

---

- Temporarily taking resource away from the current owner and give it to another process.
- Problem
  - How to decide from whom to take?
  - Frequently difficult and impossible

www.geocities.com/anjibcs
218
manjib@mail.com.np

## Recovery through Rollback

---

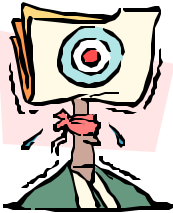
- Roll back the process to some safe state, and restart it from that state.
- Keep record of safe state with checkpoints.

www.geocities.com/anjibcs
219
manjib@mail.com.np

## Recovery through Killing

---

- Kill a process in the cycle or not in cycle.
- Problem:
  - Which process to select?



www.geocities.com/anjibcs
220
manjib@mail.com.np

## Two-Phase Locking

---

- First Phase: process tries to lock all the records it needs, one at a time.
- Second Phase: If first phase is success, real work is done and release the locks.
- Problem:
  - Not applicable in general

www.geocities.com/anjibcs
221
manjib@mail.com.np

## Starvation

---

- Variation of deadlock
- Policy implemented may lead to some process never getting service even though they are not deadlocked.

***Which algorithm will solve this problem?***

www.geocities.com/anjibcs
222
manjib@mail.com.np

**Unit 4: Memory Management**

Anjib Man Mulepati

www.geocities.com/anjibcs 223 manjib@mail.com.np

**Agenda**

- Basic Memory Management
- Swapping
- Virtual Memory
- Page Replacement Algorithms
- Paging
- Segmentation
- Segmentation with Paging

www.geocities.com/anjibcs 224 manjib@mail.com.np

**Why Memory Management?**

- Memory is an important resource
- Software seems to growing even faster than memory.
- Tread towards multimedia is growing

**Parkinson's Law**

***Programs expand to fill the memory available to hold them***

www.geocities.com/anjibcs 225 manjib@mail.com.np

**What programmer wants?**

- Infinitely large memory
- Infinitely fast memory
- Nonvolatile
- Inexpensive

www.geocities.com/anjibcs 226 manjib@mail.com.np

**Is this possible?**

www.geocities.com/anjibcs 227 manjib@mail.com.np

**Memory Manager**

- It job of OS to coordinate how these memories are used.
- And the part of OS responsible for this is known as **Memory Manager**.
- Its jobs is
  - to keep track of memory part used or unused
  - to allocate memory to processes when they need it and deallocate when they are done
  - to manage swapping between main memory and disk.

www.geocities.com/anjibcs 228 manjib@mail.com.np

### Monoprogramming Model

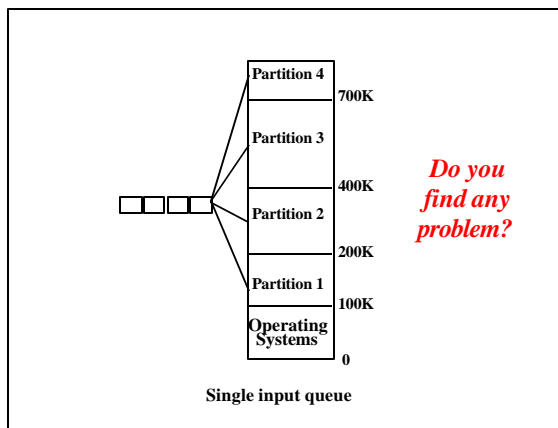
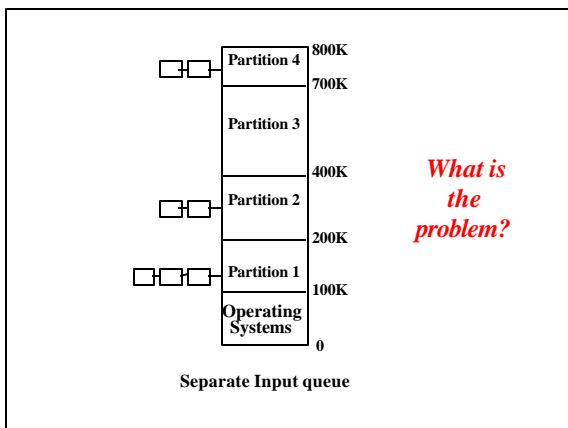
(a) (b) (c)

- Running one program at a time.
- Sharing memory between that program and the operating system.

### Multiprogramming

- Multiple process running at once.
- One process waiting for I/O to finish, another one can use the CPU.
- CPU utilization can be improved
- Variation of multiprogramming are:
  - Fixed Partitions with separate input queues
  - Fixed Partitions with single input queues
  - Variable Partition

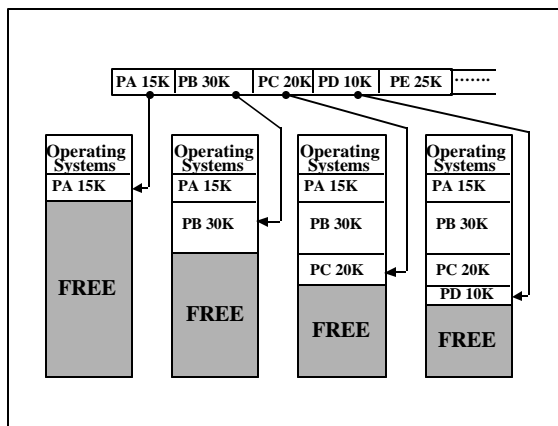
www.geocities.com/anjibcs 230 manjib@mail.com.np



### Variable Partition

- Problem in fixed partition
  - Memory are waste when job are allocated to bigger enough partition.
- Jobs are given as much storage as they required

www.geocities.com/anjibcs 233 manjib@mail.com.np



### Variable Partition

- Problem
  - Holes
- Solution
  - Allocate holes to new jobs, But again unallocated space may remain.

www.geocities.com/anjibcs 235 manjib@mail.com.np

### Coalescing Holes

- Merge two or more adjacent holes.

www.geocities.com/anjibcs 236 manjib@mail.com.np

### Compaction

- Problem with Coalescing
  - Even holes are merged they may be distributed.
- Solution is compaction
- Moving all occupied area of storage to one end.
- Drawbacks
  - Consume system resources
  - System is halted
  - Readdressing must be done

www.geocities.com/anjibcs 237 manjib@mail.com.np

### Swapping

- Brining process to main memory when needed and temporarily take out from main memory to a backing store.
- Example: Quantum expire in RR
- If user process is of size 1MB and the backing store has transfer rate of 5MB/sec. Then 1MB transfer will take 200ms. If avg. latency is 8ms. Then total swapping in and out takes  $2 \times 208\text{ms} = 416\text{ms}$ . So we must set quantum size larger than .0416sec

www.geocities.com/anjibcs 239 manjib@mail.com.np

### Swapping

- When priority-based scheduling algorithm is used, upon arrival of higher-priority process low-priority process are swap out this is known as **roll out** and later after finishing higher job, lower-priority process is swapped back and this is known as **roll in**.

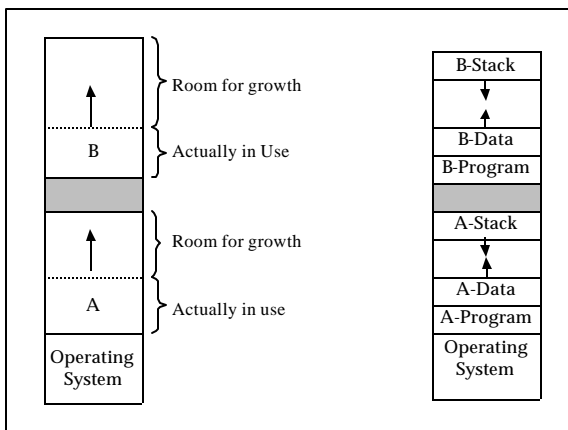
www.geocities.com/anjibcs 240 manjib@mail.com.np

### What job to swap?

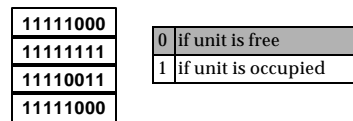
- Idle Jobs (Why?)
- In the case of process waiting for I/O, problem can arise.

### How much to allocate?

- What if process grows in runtime?
- If a hole or free space is adjacent no problem
- IF not so, then process will have to wait or be killed.
- Solution
  - Allocate little extra memory
  - If process have two growing segment, leave extra space in-between them



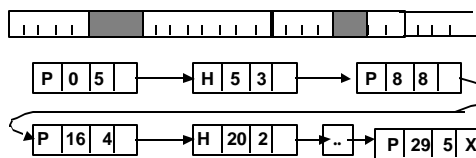
### Memory Management with Bitmaps



### What must be size of allocation unit?

- Smaller the allocation unit, the larger the bitmap
  - Larger the allocation unit, smaller the bitmap.
- $\text{Bitmap Size} \propto 1 / \text{Allocation Unit}$
- Problem
    - to bring a k unit process into memory, the memory manager have to search the bitmap to find a run of k consecutive 0 bits in the map

### Memory Management with Linked Lists



- Each entry in the list specifies
  - Hole(H) / Process(P)
  - the address at which it starts
  - the length and
  - the pointer to next entry

### Sorted Segment Why?

---

Before X terminates

A	X	B
---	---	---

A	X	
---	---	--

	X	B
--	---	---

	X	
--	---	--

After X terminates

A		B
---	--	---

A		
---	--	--

		B
--	--	---

--	--	--

Better if double-linked list

www.geocities.com/anjibcs
247
manjib@mail.com.np

### Storage Placement Strategies

---

- Storage placement strategies are used to determine where in the main memory to place incoming programs and data.
- Some of these are
  - First Fit
  - Next Fit
  - Best Fit
  - Worst Fit
  - Quick Fit

www.geocities.com/anjibcs
248
manjib@mail.com.np

### First Fit

---

- An incoming job is placed in the main memory in the first available hole large enough to hold it.
- Placement decision is Fast

www.geocities.com/anjibcs
249
manjib@mail.com.np

### Next Fit

---

- A variation of first fit
- Begins each search for an available hole at the point where the previous search ended.
- Next fit gives slightly worse performance than first fit.

www.geocities.com/anjibcs
250
manjib@mail.com.np

### Think

---

- What happens in first-fit if the search reaches the end of storage and discovers it is not large enough?
- What happens in next-fit if the search reaches the end of storage and discovers it is not large enough?
- So what data structure should be implemented?

www.geocities.com/anjibcs
251
manjib@mail.com.np

### Best Fit

---

- An incoming job is placed in the hole in main storage in which it fits most tightly and leaves the smallest amount of unused space.
- Best fit is slower than first fit (Why?)
- Results in more wasted memory (Why?)

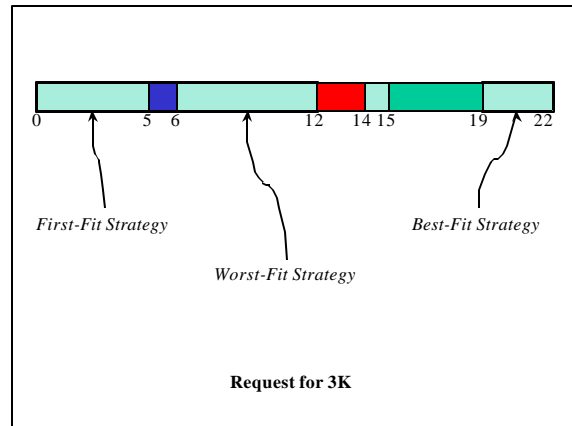
www.geocities.com/anjibcs
252
manjib@mail.com.np

### Worst Fit

---

- Place a program in main storage in the largest possible hole (Are we mad?)

www.geocities.com/anjibcs 253 manjib@mail.com.np



### Analysis

---

- Both first fit and best fit are better than worst fit in term of time and holes generate
- Neither first fit nor best fit is clearly better in terms of storage, but first fit is generally faster.

www.geocities.com/anjibcs 255 manjib@mail.com.np

### Improvement

---

- For improving above strategies we can maintain separate list for process and hole.
- Keeping holes in sort from smallest to largest benefit the best fit strategy
- Also both first fit and best fit are equally fast.
- Data structure can also be modified.

www.geocities.com/anjibcs 256 manjib@mail.com.np

### Quick Fit

---

- Maintain separate list for some of the more common sizes requested

→	4KB Holes
→	8KB Holes
→	12KB Holes
→	16KB Holes
→	24KB Holes
→	28KB Holes
→	32KB Holes
→	Odd-sized Holes

www.geocities.com/anjibcs 257 manjib@mail.com.np

### Relocation

What happen if memory reference at absolute address 10?

Partition 4	700K
Partition 3	
Partition 2	400K
Partition 1	200K
Partition 1	100K
Operating Systems	0

### Protection

---

- Preventing any process to access memory location of another process.
- IBM uses following scheme:
  - Memory is divided into blocks of 2KB
  - Each block is assign 4-bit key
  - Process are assigned to block by checking the block.

www.geocities.com/anjibcs 259 manjib@mail.com.np

### Alternative Method

---

www.geocities.com/anjibcs 260 manjib@mail.com.np

### Virtual Memory

---

- How to run the programs that are too big to fit in the available memory
- Solution: **Overlays**
- But overlays is tedious job.
- Another solution is virtual memory

“Enabling larger process to operate with small physical memory by keeping currently required part in main memory, and the rest on the disk.”

www.geocities.com/anjibcs 261 manjib@mail.com.np

### Logical Vs. Physical Address

---

- **Logical address**
  - An address generated by the CPU
- **Physical address**
  - An address seen by the memory unit

www.geocities.com/anjibcs 262 manjib@mail.com.np

### Memory Management Unit

---

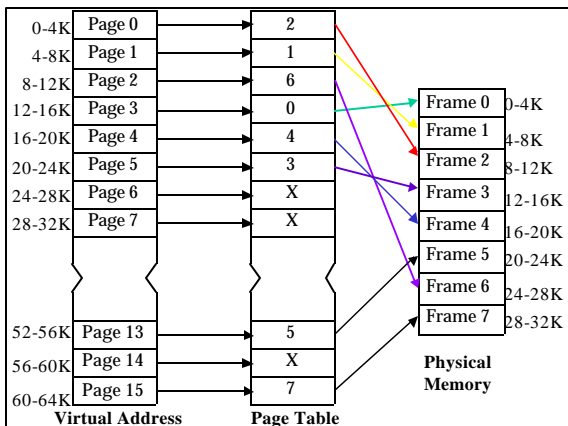
www.geocities.com/anjibcs 263 manjib@mail.com.np

### Paging

---

- The virtual address space is divided up into units called **pages**.
- The corresponding units in the physical memory are called **page frames**.
- The pages and page frames are always the same size.
- Typically sizes from 512bytes to 64KB have been used in real systems.

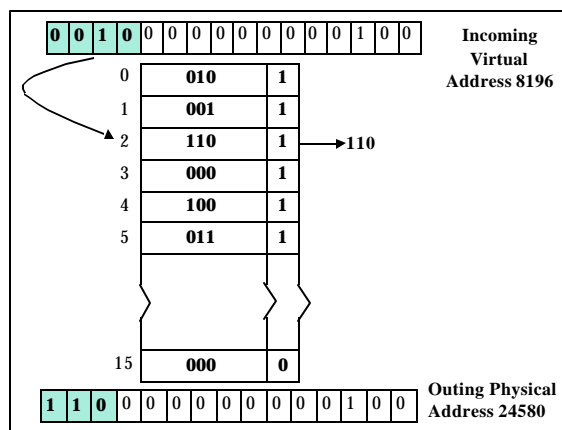
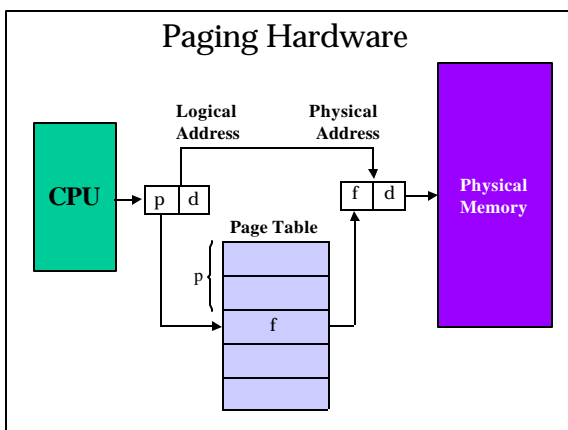
www.geocities.com/anjibcs 264 manjib@mail.com.np



### Observe

- Which address does MMU generate for call addresses
  - 0 → 8192
  - 8196 → 24580
  - 20500 → 12308
  - 32780 → Page Fault

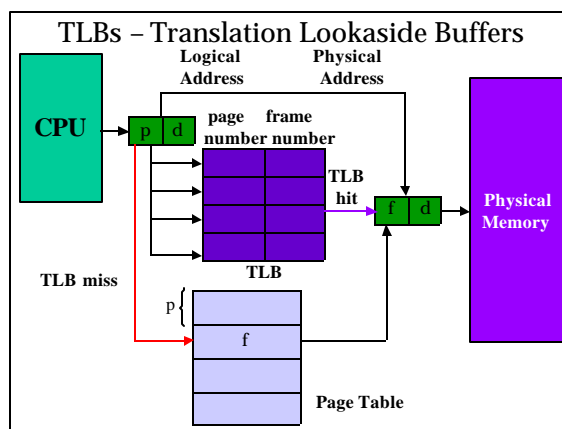
www.geocities.com/anjibcs 266 manjib@mail.com.np

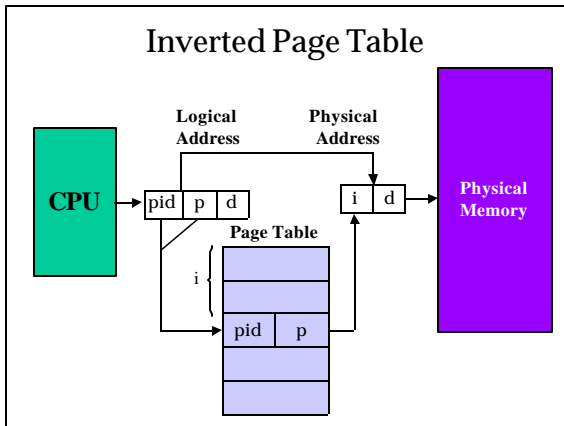


### Some Issues

- The page table can be extremely large.
- The mapping must be fast.

www.geocities.com/anjibcs 269 manjib@mail.com.np





- ### Page Replacement Strategies
1. The Principle of Optimality
  2. Random page replacement
  3. First-In-First-Out (FIFO)
  4. Least-Recently-Used (LRU)
  5. Not-Used-Recently (NRU)
  6. Second chance
  7. Clock
  8. Working set
  9. WSClock
- www.geocities.com/anjibcs 272 manjib@mail.com.np

- ### The Principle of Optimality
- Replace the page that will not be used again for the furthest time into the future.
  - Problem: Can we predict the future?
- www.geocities.com/anjibcs 273 manjib@mail.com.np

- ### Random Page Replacement
- Choose page randomly for replacement
  - Advantage:
    - No discrimination among pages
    - Low overhead
    - Quick decision
  - Problem:
    - Next page to be referenced can be selected.
- www.geocities.com/anjibcs 274 manjib@mail.com.np

- ### First-In-First-Out (FIFO)
- Time-stamp each page as it enters.
  - When a page needs to be replaced, we choose the one that has been in storage the longest.
  - Advantage:
    - Chance for all
    - Easy to implement with queue data structure
  - Disadvantage
    - Likely to replace heavily used pages.
- www.geocities.com/anjibcs 275 manjib@mail.com.np

### Do you agree?

**“The more page frames allocated to a process, the fewer page faults the process would experience.”**

www.geocities.com/anjibcs 276 manjib@mail.com.np

Page Reference	Fault / No Fault	FIFO page replacement with three pages available			Total Fault = 9
Initially		-	-	-	
A	Fault	A	-	-	
B	Fault	B	A	-	
C	Fault	C	B	A	
D	Fault	D	C	B	
A	Fault	A	D	C	
B	Fault	B	A	D	
E	Fault	E	B	A	
A	No Fault	E	B	A	
B	No Fault	E	B	A	
C	Fault	C	E	B	
D	Fault	D	C	E	
E	No Fault	D	C	E	

Page Reference	Fault / No Fault	FIFO page replacement with three pages available				Total Fault = 10
Initially		-	-	-	-	
A	Fault	A	-	-	-	
B	Fault	B	A	-	-	
C	Fault	C	B	A	-	
D	Fault	D	C	B	A	
A	No Fault	D	C	B	A	
B	No Fault	D	C	B	A	
E	Fault	E	D	C	B	
A	Fault	A	E	D	C	
B	Fault	B	A	E	D	
C	Fault	C	B	A	E	
D	Fault	D	C	B	A	
E	Fault	E	D	C	B	

### FIFO or Belady's Anomaly

---

- Under FIFO page replacement, certain page reference patterns actually cause more page faults when the number of page frames allocated to a process is increased.

**"Operating Systems are complex entities that sometimes defy intuition"**

www.geocities.com/anjibcs 279 manjib@mail.com.np

### Least-Recently-Used (LRU)

---

- Principle "Page refer long ago wont be refer again in near future"
- Selects that page for replacement that has not been used for the longest time.
- Can be implemented with
  - Time Stamp
  - Matrix
  - A List Structure

www.geocities.com/anjibcs 280 manjib@mail.com.np

### Time Stamp Implementation

---

- Each time page is reference time of reference is stamped
- When page fault occur, page with least time is selected for replacement.

www.geocities.com/anjibcs 281 manjib@mail.com.np

### Matrix Implementation

---

- For a system with n page frames, we maintain a n X n matrix initially set to 0
- Whenever page k is referenced, all bits of row k is set to 1, then set all the bits of column k to 0.
- Row whose binary value is lowest is the least recently used.

www.geocities.com/anjibcs 282 manjib@mail.com.np

### List Implementation

---

- Containing one entry for each occupied page frame
- Each time a page frame is referenced, the entry for that page is placed at the head of the list.
- Older entries migrate toward the tail of the list.
- When a page must be replaced, the entry at the tail of the list is selected, the corresponding page frame is freed, the incoming page is placed in that page frame, and the entry for that page frame is placed at the head of the list because that page is now the one that has been most recently used.

www.geocities.com/anjibcs 283 manjib@mail.com.np

### Not-Used-Recently (NUR)

---

- **Principle: “Pages not used recently are not likely to be used in the near future and they may be replaced with incoming pages”**
- **Uses two hardware bits per page**
  - **Reference bit**
    - 0 if the page has not been referenced
    - 1 if the page has been referenced
  - **Modified bit**
    - 0 if the page has not been modified
    - 1 if the page has been modified

www.geocities.com/anjibcs 284 manjib@mail.com.np

### Working

- Initially, the **Reference** and **Modified bits** of all pages are set to 0
- As a reference to a particular page occurs, the **Reference bit** of that page is set to 1.
- Whenever a page is modified, its **Modified bit** is set to 1.
- When a page is to be replaced, replace in following order
 

<b>Group 1</b>	unreferenced	unmodified
<b>Group 2</b>	unreferenced	modified
<b>Group 3</b>	referenced	unmodified
<b>Group 4</b>	referenced	modified

Is Group 2 possible?

www.geocities.com/anjibcs 285 manjib@mail.com.np

### NUR (contd..)

---

- What if all the reference bits are set on?
- Periodically set all the reference bits to 0 to get a fresh start.

www.geocities.com/anjibcs 286 manjib@mail.com.np

### Second Chance

---

- Variation of FIFO.
- The clear weakness of the FIFO strategy is that it may choose to replace a heavily used page that has been in memory for a long time.
- Examines the reference bit of the oldest page; if this bit is off, the page is immediately selected for replacement.
- If the reference bit is on, it is set off and the page is moved to the tail of the FIFO list and treated essentially as a new arrival; this page gradually moves to the head of the list from which it will be selected for replacement only if its reference bit is still off.

www.geocities.com/anjibcs 287 manjib@mail.com.np

### Second Chance (contd...)

---

- This gives the page a second chance to remain in primary storage if indeed its referenced bit is turned on before the page reaches the head of the list.
- Active pages will repeatedly have their reference bits set on, move to the head of the list, have their reference bit bits set off, and move to the tail of the list, thus remaining in primary storage.

www.geocities.com/anjibcs 288 manjib@mail.com.np

### Clock Page Replacement

---

- The clock variation of the second chance algorithm arranges the pages in a circular list instead of a linear list.
- A list pointer moves around the circular list much as the hand of a clock rotates.
- When a page's referenced bit is turned off, the pointer is moved to the next element of the list.

www.geocities.com/anjibcs 289 manjib@mail.com.np

### Not Frequently Used (NFU)

---

- Software implementation of LRU
- Each page has counter which is initially set to zero
- At each clock interrupt, R bit, which is 0 or 1, is added to the counter.
- On the page fault page with the lowest counter is chosen for replacement.
- Problem:
  - Never forgets anything.

www.geocities.com/anjibcs 290 manjib@mail.com.np

### Aging

---

- Modification of NFU
- Counters are each shifted right 1 bit before the R bit is added in.
- The R bit is added to the leftmost, rather than the rightmost bit.
- Usually 8 bit counter is enough

www.geocities.com/anjibcs 291 manjib@mail.com.np

R bit	1 0 1 0 1 1	1 1 0 0 1 0	1 1 0 1 0 1	1 0 0 0 1 0	0 1 1 0 0 0
Page 0	10000000	11000000	11100000	11110000	01111000
Page 1	00000000	10000000	11000000	01100000	10110000
Page 2	10000000	01000000	00100000	00100000	10001000
Page 3	00000000	00000000	10000000	01000000	00100000
Page 4	10000000	11000000	01100000	10110000	01011000
Page 5	10000000	01000000	10100000	01010000	00101000

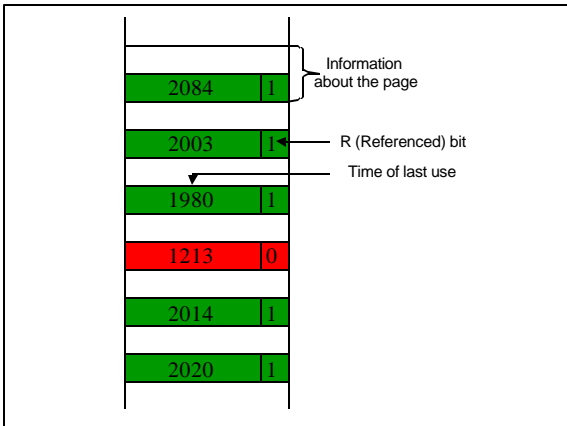
Which page to select when page fault occurs at this point?

### Working Set Page Replacement

---

- P. J. Denning define a working set as a collection of pages a process is actively referencing.
- For a program to run efficiently, its working set of pages must be maintained in primary storage.
- Otherwise excessive page fault called thrashing might occur.
- Therefore we must try to keep track of each process's working set and make sure that it is in memory before letting the process run.

www.geocities.com/anjibcs 293 manjib@mail.com.np



### Working

- We set age of the pages in the working set as  $\tau$ .
- Each page has at least two information: the approximate time the page was last used and the R bit.
- When page fault occur, each entry is processed, the R bit is examined
- If R is 1, set time of last use to current time, indicating that the page was in use at this moment
- If R is 0, the page may be a candidate for removal. So check for age, if its age is greater than  $\tau$  remove it. Update remaining entries.
- If R is 0 but the age is less than or equal to  $\tau$ , the page is still in the working set.
- If all pages have above situation, page with greatest age is selected.
- If all page have R=1, one of them is selected randomly.

### WSClock Page Replacement

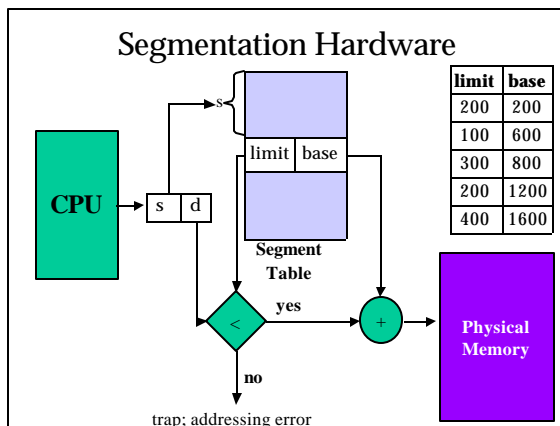
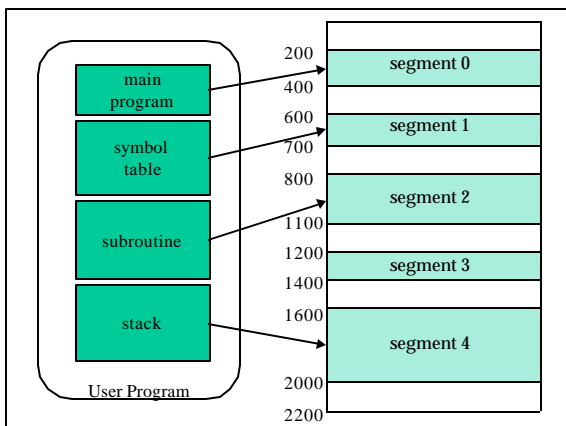
The diagram shows a circular list of page frames. Each frame is a rectangle divided into two parts: the left part contains a page number and the right part contains an R bit. The frames are arranged in a circle, and an arrow indicates the sequence of frames to be examined. The frames shown are: 2084 | 1, 1620 | 0, 2032 | 1, 2020 | 1, 2014 | 1, 1213 | 0, 1980 | 1, and 2003 | 1.

### Implementation

- At each fault the page pointed to by the hand is examined first
- If R=1, then R is not ideal candidate. Set R=0, move to next page
- If R=0, compute age, if age is greater than  $\tau$  then it can be replaced.
- If page is modified, page is not selected

### Segmentation

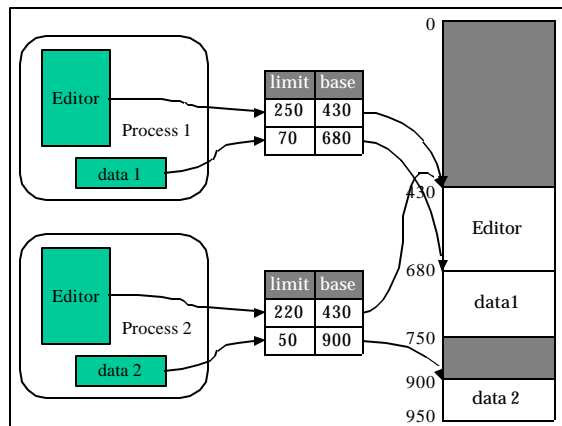
- In variable partition multiprogramming, we place running programs in one block of contiguous locations of real storage.
- In real storage **segmentation** systems, this restriction is removed and program (and its data ) are allowed to occupy many separate blocks of real storage.
- Segments are numbered and are referred to by a segment number.



### Advantages

- Each segment can be given certain access rights. For example, instruction segment can be defined as read only.
- A segment corresponding to a dynamic data structure may grow and shrink as the data structure itself grow and shrinks.
- For a data structure like array, segment corresponding to the array is as large as the array so reference to illegal index is prevented.
- Sharing of code or data

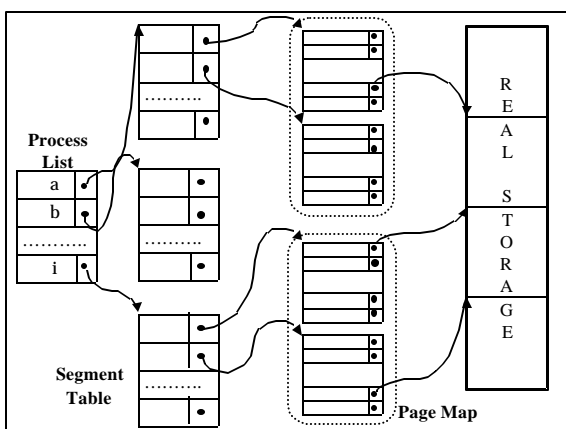
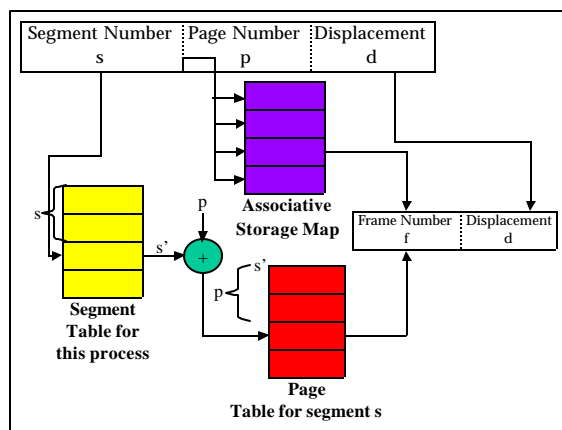
www.geocities.com/anjibcs 301 manjib@mail.com.np



### Segmentation with Paging

- Segments are usually multiples of pages in size.
- All the pages of a segment need not be in main memory at once.
- Virtual storage pages that are contiguous in virtual storage need not be contiguous in real storage.
- Addressing is three dimensional with a logical address,  $v$ , being an ordered triple  $v=(s,p,d)$  where
  - $s$  is the segment number
  - $p$  is the page number within the segment and
  - $d$  is the displacement within a page.

Segment Number	Page Number	Displacement
$s$	$p$	$d$



## THE END

www.geocities.com/anjibcs 306 manjib@mail.com.np

# INPUT/OUTPUT

Anjib Man Mulepati

www.gencities.com/anjibcs307manjib@mail.com.np

## Agenda

- Principles of I/O Hardware
- Principles of I/O Software
- I/O Software Layers
- Disks
- Clocks
- Terminals

www.gencities.com/anjibcs308manjib@mail.com.np

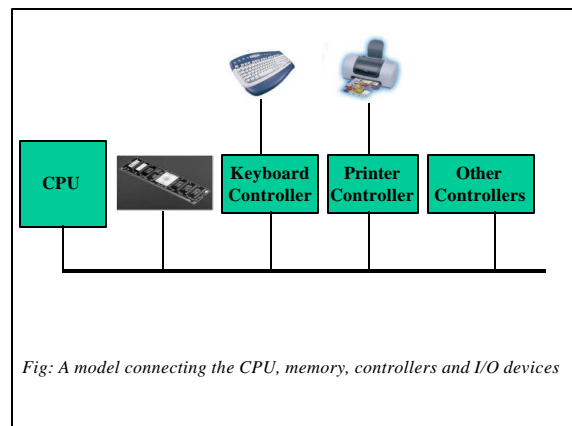
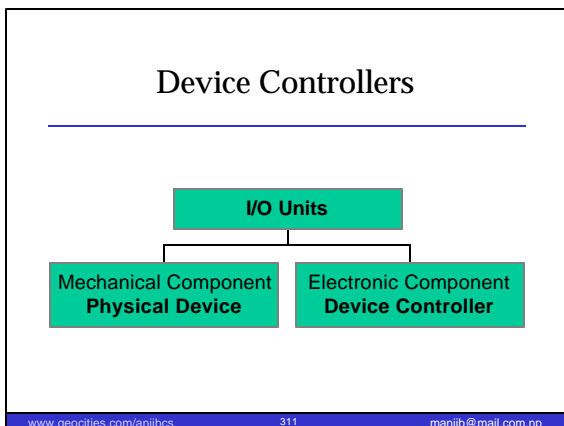
## Objective

- How the OS manages I/O?
- OS should
  - issue commands to handle the I/O devices
  - catch the interrupts
  - handle the errors
  - provide the interface (device independent) to handle them

www.gencities.com/anjibcs309manjib@mail.com.np

### I/O Devices

Aspect	Variation	Example
Data Transfer Mode	Character Block	Printer Disk
Access Method	Sequential Random	Modem CD-ROM
Sharing	Dedicated Sharable	Tape Keyboard
Transfer Schedule	Synchronous Asynchronous	Tape Keyboard
I/O Direction	Read Only Write Only Read and Write	CD-ROM Graphics Controller Disk



### Why Device Controller?

---

- All the complex part is handle by the controller.
- Example:
  - Disk controller convert the serial bit stream into a block of bytes and perform necessary error correction.
  - Monitor controller reads bytes containing the character to be displayed from memory and generates the signals used to modulate the CRT beam to cause it to write on the screen.

www.geocities.com/anjibcs 313 manjib@mail.com.np

### Parts of Controller

---

- Controller has
  - **Registers** used for communicating with the CPU
  - **Data buffer** for OS to read and write
- Using register OS can
  - deliver data to device
  - accept data
  - learn the device state

www.geocities.com/anjibcs 314 manjib@mail.com.np

### How CPU communicates?

---

www.geocities.com/anjibcs 315 manjib@mail.com.np

### Different communication methods

---

- Two address
  - Each control register is assigned an I/O port number
  - CPU can read and write in control register register as  
`IN REG, PORT`  
`OUT PORT, REG`
- Memory-Mapped I/O
  - Each control register is assigned a unique memory address to which no memory is assigned.
- Hybrid
  - Memory-mapped I/O data buffer and separate I/O ports for the control registers.

### How do these scheme work?

---

- When the CPU wants to read a word, either from the memory or from an I/O port, it puts the address it needs on the address bus and then pass READ signal.
- If it is memory space, the memory responds to the request. If it is I/O space, the I/O device responds to the request.

www.geocities.com/anjibcs 317 manjib@mail.com.np

### Memory-Mapped I/O

---

<p><b>Advantages</b></p> <ul style="list-style-type: none"> <li>• Device driver can be written without assembly code.</li> <li>• No special protection mechanism is needed to keep user processes from performing I/O</li> </ul>	<p><b>Disadvantage</b></p> <ul style="list-style-type: none"> <li>• Due to use of high-bandwidth bus between CPU and Memory, I/O devices have no way of seeing memory addresses as they go by on the memory bus, so they have no way of responding.</li> </ul>
--	--

www.geocities.com/anjibcs 318 manjib@mail.com.np

### Direct Memory Access (DMA)

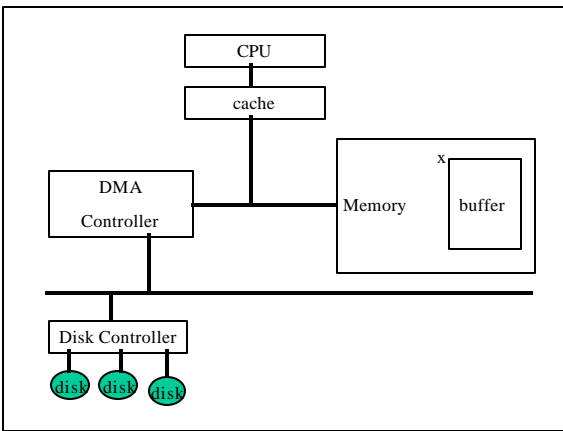
- CPU's time is wasted for each request of data to I/O controller
- Thus use DMA scheme
- DMA can be integrated into each device controller but single DMA controller can be used for regulating transfers to multiple devices.
- DMA controller has access to the system bus independent of the CPU.

www.geocities.com/anjibcs 319 manjib@mail.com.np

### How DMA works?

1. Device drive is told to transfer disk data to buffer at address x
2. Device driver tells disk controller to transfer C bytes from disk to buffer at address x.
3. Disk controller initiates DMA transfer
4. Disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C=0
6. When C=0, DMA interrupts CPU to signal transfer completion.

www.geocities.com/anjibcs 320 manjib@mail.com.np



### Interrupt

- allow a unit to gain the immediate attention of another unit
- the first unit may report the status change to the second unit
- the interrupt causes the interrupted unit's current state to be saved before the interrupt is processed
- interrupt is processed
- the state of the interrupted unit is restored back.

www.geocities.com/anjibcs 322 manjib@mail.com.np

### Polling

- allow a unit to check the status of another independently functioning unit
- the first unit checks whether the second is in a certain status
- if it is not, then the first unit proceeds with what it was doing
- later again first unit has to keep checking the second unit until the purpose of the checking was fulfilled

**Polling can be a high overhead operation**

www.geocities.com/anjibcs 323 manjib@mail.com.np

### Example

**Scenario** : a cook is cooking something in modern microwave oven equipped kitchen

**Case A** : The cook may set a timer to expire after appropriate number of minutes; the buzzer sounding after this interval is an example of *INTERRUPT*

**Case B** : The cook may regularly peek through the oven's glass door and watch as the roast cooks under Cook and Watch; this kind of regular monitoring is an example of *POLLING*

www.geocities.com/anjibcs 324 manjib@mail.com.np

### Goals of the I/O Software

---

- **Device Independence**
  - A program that read a file as input should be able to read a file on a floppy, on a hard disk or on a CD-ROM, without having to modify the program for each different device.
- **Uniform Naming**
  - The name of the device should simply be a string or an integer and not depend on the device.
- **Error Handling**
  - Error should be handled as close to the hardware as possible.
- **Synchronous Vs Asynchronous Transfers**
- **Buffering**
- **Sharable Vs Dedicated**

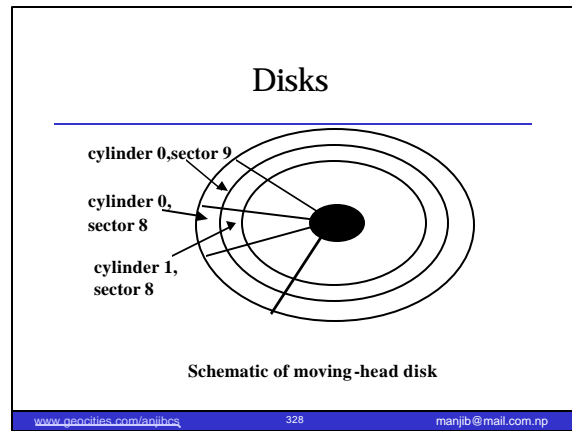
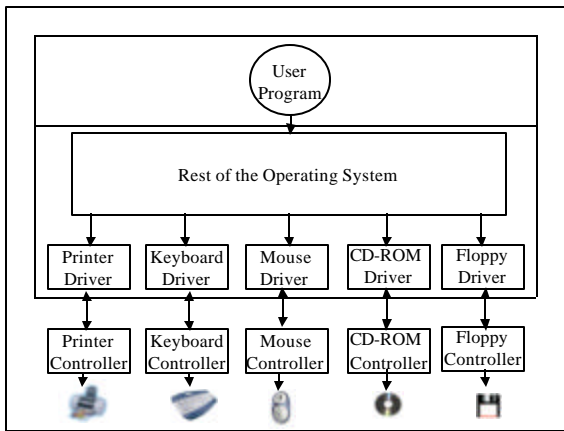
www.geocities.com/anjibcs 325 manjib@mail.com.np

### I/O Software Layer

---

Layer	Functions
User Processes	Make I/O call
Device-Independent Software	Naming, Buffering, Error Reporting, Allocation, Blocking
Device Drivers	Set up device registers; check status
Interrupt Handlers	Wake up driver when I/O completed
Hardware	Perform I/O operation

www.geocities.com/anjibcs 326 manjib@mail.com.np

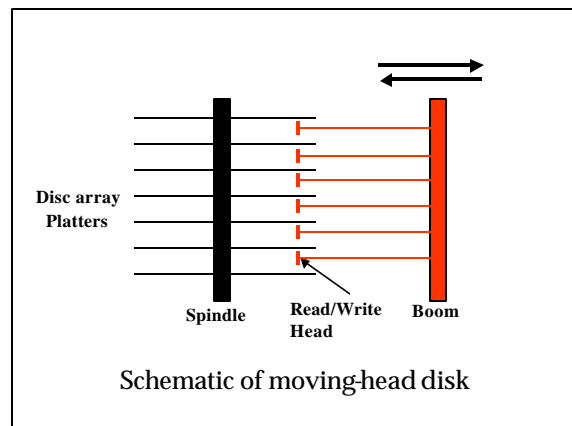


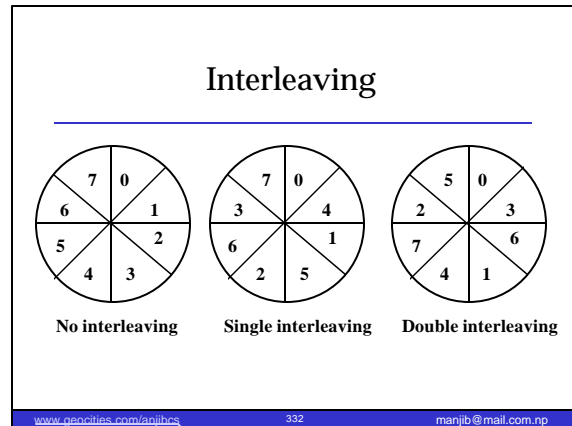
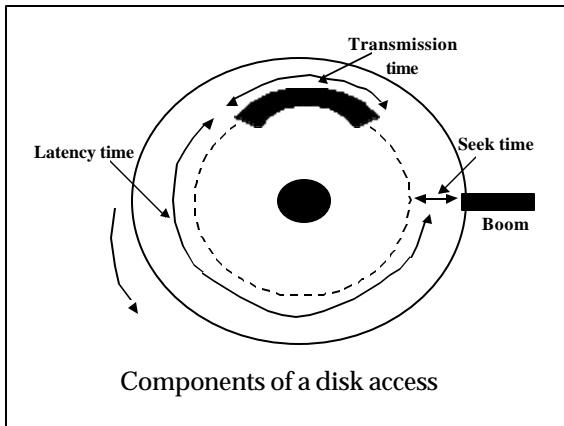
### Why Disks?

---

- Storage capacity available is much larger
- Price per bit is much lower
- Information is not lost when the power off

www.geocities.com/anjibcs 329 manjib@mail.com.np





### Cylinder Skew

- Done to improve performance.
- Allow the disk to read multiple tracks in one continuous operation without losing data.

### Disk Arm Scheduling

- First-Come, First-Served (FCFS)
- Shortest Seek First (SSF)
- Scan/Elevator Algorithm

### FCFS

- If the disk driver accepts requests one at a time and carries them out in the same order, the algorithm is First-Come First-Served (FCFS)
- FCFS is a fair method of allocating service
- When the request rate (load) becomes heavy, FCFS can result in very long waiting times

### SSF

- The next cylinder seek should be the least displacement one among the waiting requests (shortest seek first)
- With a heavily loaded disk, the arm will tend to stay in the middle of the disk most of the time, so requests at either extreme will have to wait.

## SCAN

---

- Same like SSF except that it chooses the request that results in the shortest seek distance in a preferred direction
- It does not change direction until it reaches the outermost cylinder or there are no further requests pending in the preferred direction
- Normally mean seek time of SCAN algorithm is worse than SSF algorithm
- The nicety of this algorithm is, the upper bound on the total motion is fixed, twice the number of cylinders

www.geocities.com/anjibcs 337 manjib@mail.com.np

Request: 11, 1, 36, 16, 34, 9, 12

Current	Next	Move
11	1	10
1	36	35
36	16	20
16	34	18
34	9	25
9	12	3
<b>Total</b>		111

Current	Next	Move
11	12	1
12	9	3
9	16	7
16	1	15
1	34	33
34	36	2
<b>Total</b>		61

www.geocities.com/anjibcs 340 manjib@mail.com.np

### SCAN

Current	Next	Disp/ Dir
11	12	1 / out
12	16	4 / out
16	34	18 / out
34	36	2 / out
36	9	27 / in
9	1	8 / in
<b>Total</b>		60

www.geocities.com/anjibcs 341 manjib@mail.com.np

## Disk Error

---

- **Programming Error** : driver tells the controller to seek to a nonexistent cylinder/sector, use a nonexistent head, or transfer to or from nonexistent memory
- **Transient Checksum Error** : caused by specks of dust in the air that get between the head and the disk surface
- **Permanent Checksum Error** : caused by the physical damaged disk block (in some cases the substitution is made for damaged block at only controller level but driver kept unaware)

www.geocities.com/anjibcs 340 manjib@mail.com.np

## Disk Error

---

- **Seek Error** : caused by mechanical problem in the moving arm; the difference between the initial pulse number to seek at particular cylinder and the actual arm position (written at the time of format) after movement complete
- **Controller Error** : somewhere problem within controller itself at components (software, variables, buffers, etc.) level

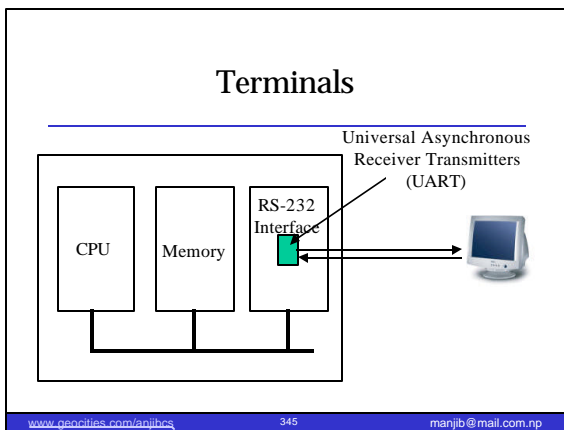
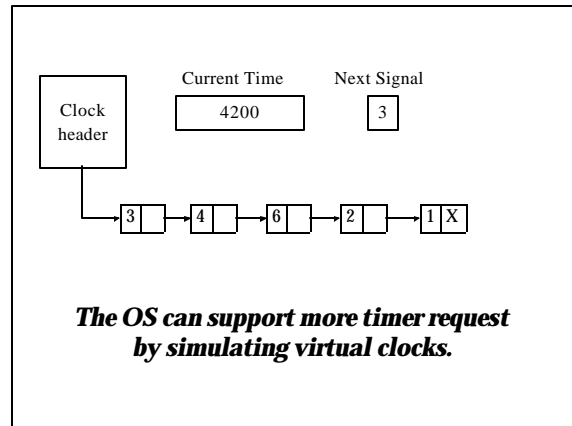
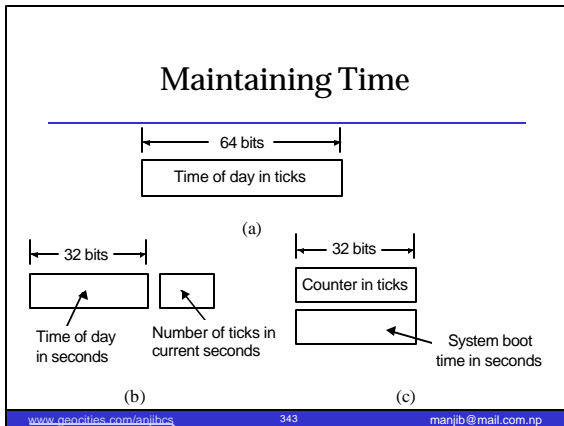
www.geocities.com/anjibcs 341 manjib@mail.com.np

## Clocks

---

- Give the current time
- Give the elapsed time
- Set a timer to trigger operation X at time T

www.geocities.com/anjibcs 342 manjib@mail.com.np



- ### Questions
1. Explain and differentiate the two categories of I/O devices, block and character.
  2. In IBM-PC, how does I/O controller function with dedicated address, allocation and interrupt request (IRQ) assignment mechanism? Explain.
  3. What is a device controller? What is a device driver? How does it work with device driver? Explain with video display (CRT) example.
  4. What is interleaving? In some cases we need to have double or triple interleaving? What forces us to have this higher degree interleaving?
  5. Which algorithm, Shortest-Seek-first (SSF) or Scan (Elevator) is better? Justify your choice.
- www.geocities.com/anjibcs 346 manjib@mail.com.np

- ### Questions
6. Differentiate between the two different ways of I/O handling, polling and interrupt.
  7. Suppose that a disk drive has 4000 cylinders, numbered 0 to 3999. The drive is currently serving a request, in FIFO order, is 916, 1509, 82, 1011, 1774, 130, 507, 250, 2681, 56. Which algorithm for disk-arm scheduling among FCFS, SSF and SCAN is better for this set of requests? Justify with calculation.
  8. What are the basic functions of the device independent software? Explain briefly.
- www.geocities.com/anjibcs 347 manjib@mail.com.np

## THE END

www.geocities.com/anjibcs 348 manjib@mail.com.np

## Unit 6: File Systems

Anjib Man Mulepati

www.geocities.com/anjibcs 349 manjib@mail.com.np

## Agenda

- Files
- Directories
- File System Implementation
- Protection Mechanisms

www.geocities.com/anjibcs 350 manjib@mail.com.np

## File System

- It must be possible to store a very large amount of information
- The information must survive the termination of the process using it
- Multiple processes must be able to access the information concurrently

**System that provides the persistent storage of information arranged in some units within the secondary memory is termed as *File systems***

www.geocities.com/anjibcs 351 manjib@mail.com.np

## File Naming

- Files are abstract mechanism; shield the user from the details of how and where the information is stored, and how the disks actually work
- Similar like object naming
- When a process create a file, it gives the file name. When the process terminates, the file continues to exists and can be accessed by other processes using its name.

www.geocities.com/anjibcs 352 manjib@mail.com.np

## Naming Rules

- DOS
  - Not case sensitive
  - File names are 1 to 8 characters, plus an optional extension of 1 to 3 characters.
- UNIX
  - Case sensitive
  - No restriction in name and extension
  - Can have two or more extension

www.geocities.com/anjibcs 353 manjib@mail.com.np

## Some File extensions

Extension	Meaning
file.txt	General text file
file.doc	Microsoft Word file
file.xls	Microsoft Excel file
file.mdb	Microsoft Access file
file.jpg	Still picture encoded with the JPEG standards
file.html	Hypertext Markup Language Document
file.mp3	Music encoded in MPEG layer 3 audio format
file.zip	Compressed file
file.c	C source program

www.geocities.com/anjibcs 354 manjib@mail.com.np

### Why file extension?

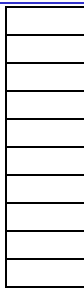
- In some systems, file extensions are just conventions and are not enforced by OS, like in UNIX
- Whereas in windows every extension have meaning.

### File Structures

- Byte Sequence
- Record Sequence
- Tree

### Byte Sequence

- Consists of unstructured sequence of bytes
- OS does not know or care what is in the file
- Any meaning must be imposed by user-level programs
- Provides maximum flexibility; user can put anything they want and name them anyway that is convenient
- Both DOS and UNIX use this.

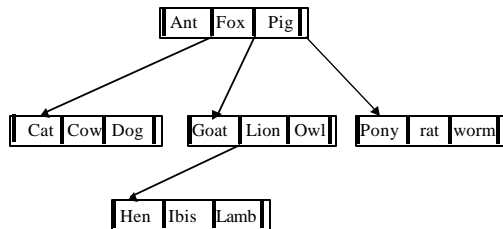


### Record Sequence

- In this model, a file is a sequence of fixed-length records, each with some internal structure
- In record sequence file, each read operation returns one record and the write operation overwrites or appends one record
- In the past many OS supported 80-character (from 80-column punch card) and 132-character (from 132-column line printer) record structure; CP/M uses 128-character record
- Variable length line devices have problem with this type of file structure



### Tree



### Tree

- Consists of a tree of records; not necessary to be same length records
- Each record contains a key field in a fixed position
- The tree is sorted on the key field, to allow rapid searching for a particular key
- Basic operation here not to get the next record rather a record with a particular key
- New record addition within file is not user decision rather OS decision
- Widely used on the large mainframe computers used in commercial data processing

## File Types

---

- Regular files : the ones that contain the user information; any structure byte sequence, record sequence or tree all are of regular file type; generally either ASCII file or Binary file
- Directories : system files for maintaining the structure of the file system
- Character special files: related to input output and used to model serial I/O devices such as terminals, printers, networks, etc.
- Block special files: used to model disks

www.geocities.com/anjibcs 361 manjib@mail.com.np

## Regular Files

---

- ASCII Files
- Binary Files

www.geocities.com/anjibcs 362 manjib@mail.com.np

## ASCII Files

---

- Consists of lines of text
- Each line is terminated either by carriage return character or by the line feed character or both
- Line lengths may be different
- They can be displayed and printed as is and can be edited with an ordinary text editor
- Easy to connect the output of one program to the input of another program provided both of them uses ASCII files

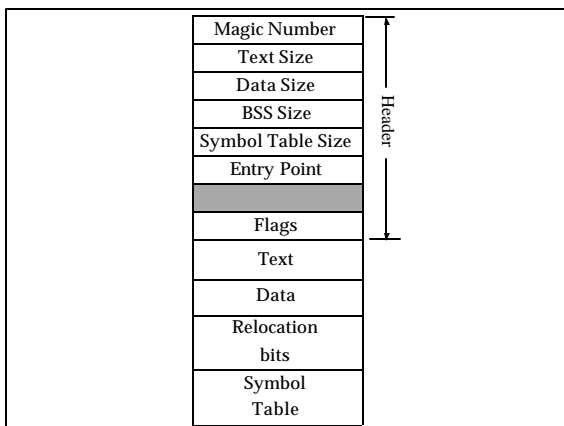
www.geocities.com/anjibcs 363 manjib@mail.com.np

## Binary Files

---

- Consists of sequence of bytes only
- Listing of the file if fed to the printer gives the incomprehensible listing full of random junk
- OS will only execute a file if it has the proper format
- Usually they have some internal structure
- A UNIX version binary file has five sections : header, text, data, relocation bits, and symbol table

www.geocities.com/anjibcs 364 manjib@mail.com.np



## File Access

---

- Two kind of file access:
  - Sequential Access
  - Random Access
- In some older mainframe OS, files are classified as either sequential or random access when they are created
- Modern OS do not make this distinction and all their files are automatically random access

www.geocities.com/anjibcs 366 manjib@mail.com.np

### Sequential Access

- In sequential, a process could read all the bytes or records in a file in order, starting at the beginning, but could not skip around and read them out of order
- Magnetic tapes used sequential access

Sequential Access	Direct Access
RESET	cp=0;
READ NEXT	READ cp; cp=cp+1;
WRITE NEXT	WRITE cp; cp=cp+1;

### Random Access

- The file type in which records can be read in any order are called random access files
- disks uses random access
- Random access files are essential for many applications, for example, data base systems (flight reservation)
- Two operations for random access are READ and SEEK; every READ operation gives the position in the file to start reading at; another special SEEK operation sets the current position at the intended point of location

### File Attributes

- In addition to name and data, all other information about file is termed as file attribute
- The list of attributes varies considerably from system to system
- The protection related attributes are Protection, Password, Creator & Owner
- The flags are bits or short fields that control or enable some specific property
- Hidden files, for example, do not appear in listing of all the files

### File Attributes

- Archive flag is a bit that keeps track of whether the file has been backed up; the backup program clears it off, and OS sets it whenever a file is changed; so backup program can find out which files need backup
- The temporary flag allows a file to be marked for automatic deletion when the process that created it terminates
- The record length, key position, and key length fields are only present in files whose records can be looked up using a key

Attributes	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of person who created the file
Owner	Current owner
Read-only flag	0 for R/W, 1 for read only
Hidden flag	0 for normal, 1 for do not display in list
System flag	0 for normal, 1 for system file
Archive flag	0 has been backed up, 1 for backup need
ASCII/Binary	0 for ASCII, 1 for Binary file
Random Access	0 for sequential only, 1 random access

Attributes	Meaning
Temporary flag	0 for normal, 1 for delete on process exit
Lock flag	0 for locked, nonzero for locked
Record Length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time, when file was created
Last access	Date & time, when file was last accessed
Last change	Date & time, when file was last changed
Current size	Number of bytes in the file
Maximum size	Maximum size file may grow to

## File Operations

- Create : file create with no data
- Delete : deletion of file to free up the space
- Open : before using the file, the process need to open a file for processing
- Close : when no more access of a file is required and to free up the internal table space
- Read : data read from file starting from the current position
- Write : data write to file (current position)

www.geocities.com/anjibcs 373 manjib@mail.com.np

## File Attributes

- Append : restricted form of write, write only at the end of the file
- Seek : For random access files, the method that provide the reposition of the pointer to current position to a specific place in the file
- Get Attribute : Like make in UNIX, checks the modification of all source and object files so that minimum no. of compilation makes update
- Set Attribute: protection mode change *chmod*
- Rename : Rename of the file

www.geocities.com/anjibcs 374 manjib@mail.com.np

## Memory-Mapped Files

- Memory mapped files are files that are mapped into the virtual memory space of the processes accessing them.
- Paging technique is used
- The major advantage of this is that their contents can be accessed through standards programming constructs.

www.geocities.com/anjibcs 375 manjib@mail.com.np

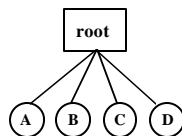
## Directories

- Also called folders
- To keep track of files
- Are themselves files
- Can be
  - Single Level
  - Two Level
  - Hierarchical

www.geocities.com/anjibcs 376 manjib@mail.com.np

## Single Level Directory Systems

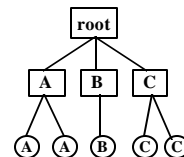
- One directory containing all the files.
- Such directory is known as **root directory**.
- Software design is simple
- Simply easy to locate files
- No duplicate file can exists.



www.geocities.com/anjibcs 377 manjib@mail.com.np

## Two-level Directory Systems

- One directory per users.
- User login and work in his directory so OS knows in which directory to work.



www.geocities.com/anjibcs 378 manjib@mail.com.np

### Hierarchical Directory Systems

---

- Each user can have as many directories as are needed so that files can be grouped together in natural ways.
- Nearly all modern file system are organized in this manner.

www.geocities.com/anjibcs
379
manjib@mail.com.np

### Path Names

---

- **Absolute Path Name**
  - Path from the root directory to the file
- **Relative Path Name**
  - A user can designate one directory as the current working directory, in which case all path names not beginning at the root directory are taken relative to the working directory.
- Most OS that support a hierarchical directory system have two special entries
  - .(dot) → current directory
  - ..(dotdot) → parent directory

www.geocities.com/anjibcs
380
manjib@mail.com.np

### Directory Operations

- **Create:** A directory is created.
- **Opendir:** Directories can be read.
- **Readdir:** Returns the next entry in an open directory
- **Rename:** Rename the directory
- **Link:** Allows a file to appear in more than one directory
- **Unlink:** A directory entry is removed
- **Closedir:** Closed to free up internal table space
- **Delete:** A directory is deleted.

www.geocities.com/anjibcs
381
manjib@mail.com.np

### File System Implementation

---

- How files and directories are stored?
- How disk space is managed?
- How to make everything work efficiently and reliably?

www.geocities.com/anjibcs
382
manjib@mail.com.np

### File System Layout

---

www.geocities.com/anjibcs
383
manjib@mail.com.np

- **Master Boot Record (MBR)**
  - Used to boot the computer
- **Partition Table**
  - Starting and ending addresses of each partition
- **Boot Block**
  - First block of the partition used to load the OS contained in that partition.
- **Super Block**
  - Contains all the key parameters about the file system and is read into memory when the computer is booted or the file system is first touched.
- **Free Block Information**
  - Information about free blocks in the file system
- **I-node**
  - an array of data structures, one per file, telling all about the file.
- **Root Directory**
  - which contains the top of the file system tree
- **Files and Directories**
  - All other files and directories

### File Implementation

- The key issue is keeping track of which disk blocks go with which file
- Some of the common methods are :
  - Contiguous allocation
  - Linked List allocation
  - Linked List Allocation using an Index
  - I-nodes

### Contiguous Allocation

- Store each file as a contiguous run of disk blocks.
- Advantages
  - Simple to implement
  - Read performance is excellent
- Drawbacks
  - In time, the disk becomes fragmented.
  - Not feasible unless maximum file size known at the time of file creation

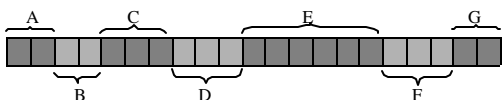


Fig: Contiguous allocation of disk space for seven files

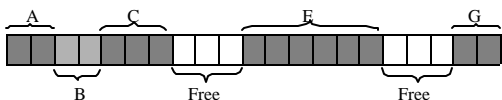


Fig: The state of the disk after files D and F have been removed

### Linked List Allocation

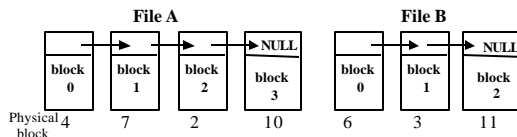
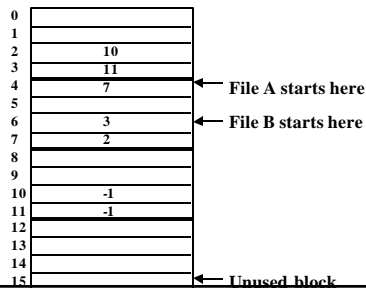


Fig: Storing a files as a linked list of disk blocks.

- No disk fragmentation; every disk block can be used.
- Random access is slow
- As the amount of data storage is not in 2<sup>n</sup> system efficiency is not optimal.

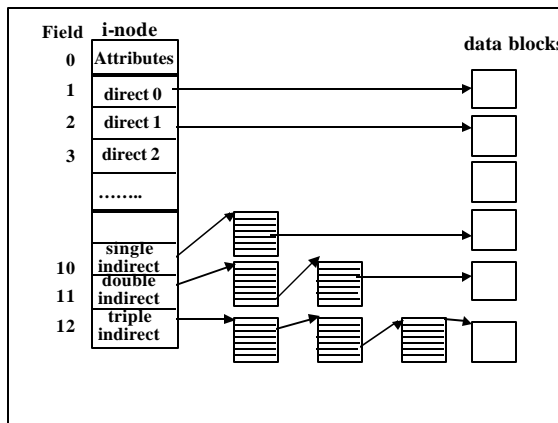
### Linked List Allocation Using a Table in Memory



- The pointer word from each disk block is kept in a table as an index in memory
- With this scheme, entire block is available for data ; random access is much easier
- The chain must still be followed but the reference is not at disk level rather at the memory level
- MS-DOS uses this technique
- Memory space occupancy is disadvantage; In 500 MB Disk with 1K blocks size table itself takes 2MB

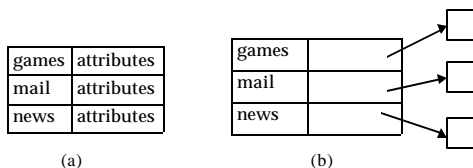
### I-nodes

- Each file is associate with a data structure called an i-node.
- The I-node contains pointers to the file's disk blocks.
- Also various file attributes such as
  - User Identification
  - Group Identification
  - File Size
  - Time of Creation
  - Time of Last Use
  - Time of Last Modification etc.



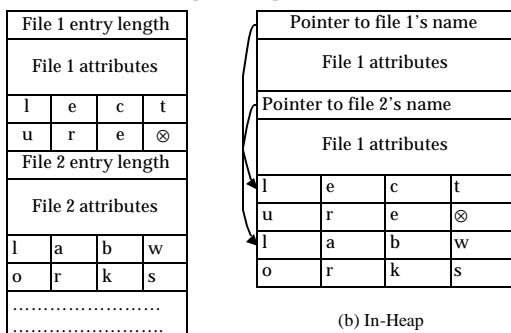
### Implementing Directories

- The directory entry provides the information needed to find the disk blocks.
  - Disk address of first block and length of block (for contiguous allocation)
  - First block number (for linked list allocation) or
  - I-node number



(a) A simple directory containing fixed-size entries with the disk addresses and attributes in the directory entry.  
 (b) A directory in which each entry just refers to an i-node.

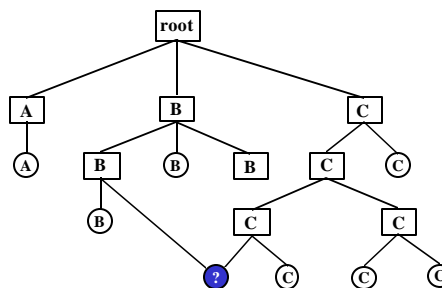
### Handling Long File Name



(a) In-Line

(b) In-Heap

### Shared Files

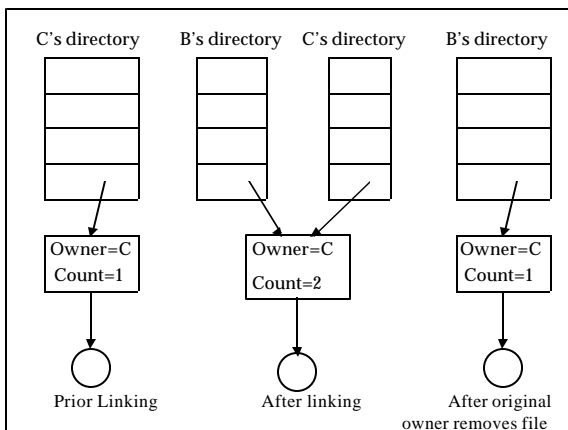


### Sharing Problems

- If disk blocks are listed in directories, changes made by one will not be visible to another.
- Solution 1: Using I-nodes
- Solution 2: Using symbolic links which just creates a shortcut to the original file.

### Potential Problems with Solutions

- In first method, when B links to a shared file, the I-node record has C as file owner. The link count in the I-node is increased.
- If C removes file and clear the I-node, B will have a directory entry pointing to an invalid I-node
- If C's entry is removed but leave the I-node intact with setting the count to 1, B will be using a file under C's file quota
- Using symbolic links this problem does not raise but symbolic links require extra overhead.



### Disk Space Management

- What should be block size?
- How to track free blocks?
- How to manage disk quotas?

### Block Size

- Larger the block size higher the data transfer rate and lesser is the disk utilization.
- See Figure 6-20 from text book page no: 412

### Free Blocks

- Using Bitmap
  - Each blocks are represented by one bit. free blocks are represented by 1s in the map, allocated blocks by 0s(or vice versa).
- Using Linked
  - Link together all the free disk blocks, the first block contains a pointer to the next free disk block and so on.

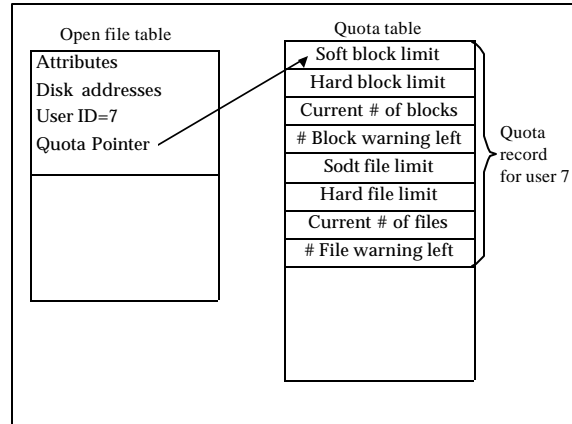


## Disk Quotas

---

- In multiuser OS system administrator can assign each user a maximum number of file and blocks user can have.
- The OS ensures that the user does not exceed this quota.
- The limit may be soft limit (which just issues a warning that the soft limit has been crossed)
- Hard limit cannot be crossed.

www.geocities.com/anjibcs 403 manjib@mail.com.np



## Backups

---

- Why?
  - Recover from disaster
  - Recover from stupidity
- What to backup?
  - Should the entire file system be backed up?
  - Should the data be compressed before backing up?
- Points to consider
  - Security
  - Space
  - Time

www.geocities.com/anjibcs 405 manjib@mail.com.np

## Incremental Dumps

---

- Make a complete dump backup periodically, say weekly or monthly
- Make a daily dump of only those files that have been modified since the last full dump.
- This minimize the backup time but it makes recovery process more complicated.

www.geocities.com/anjibcs 406 manjib@mail.com.np

## Dumping Strategies

---

- Physical Dump
  - Starts at block 0 of the disk, writes all the disk blocks onto the tape in order.
- Logical Dump
  - Starts at one or more specified directories and recursively dumps all files and directories found there that have changed since some given base date.

www.geocities.com/anjibcs 407 manjib@mail.com.np

## How to improve file system performance?

---

- Caching
  - Check read request in cache memory, if it is the read request can be satisfied without disk access. If not, it is first read into the cache, and then copied to wherever it is needed.
- Block Read Ahead
  - When the file system is asked to produce block k in a file, it does that, but when it is finished, it makes a sneaky check in the cache to see if the block k+1 is already there. If it is not, it schedules a read for block k+1 in the hope that when it is needed, it will have already arrived in the cache.
- Reducing Disk Arm Motion
  - Putting blocks that are likely to be accesses in sequence close to each other.

### Protection

---

- Operating system consists of a collection of objects, hardware or software
- Each object has a unique name and can be accessed through a well-defined set of operations.
- Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so.

www.genctiles.com/anjibcs 409 manjib@mail.com.np

### Domain Structure

---

- Access-right =  $\langle \text{object-name, rights-set} \rangle$  where *rights-set* is a subset of all valid operations that can be performed on the object.
- Domain = set of access-rights

www.genctiles.com/anjibcs 410 manjib@mail.com.np

### Access Matrix

---

- View protection as a matrix (*access matrix*)
- Rows represent domains
- Columns represent objects
- $Access(i, j)$  is the set of operations that a process executing in Domain<sub>i</sub> can invoke on Object<sub>j</sub>
- If a process in Domain  $D_i$  tries to do “op” on object  $O_j$ , then “op” must be in the access matrix.

www.genctiles.com/anjibcs 411 manjib@mail.com.np

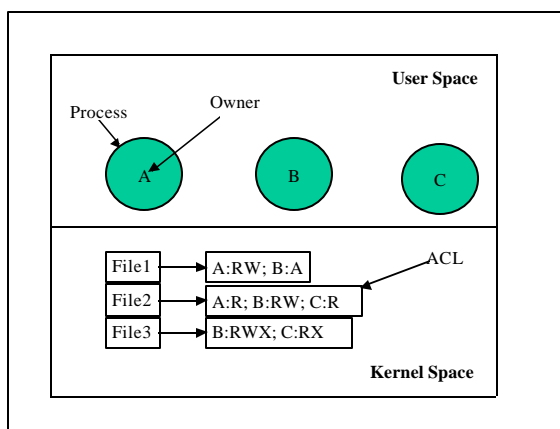
object	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	printer
D <sub>1</sub>	read		read	
D <sub>2</sub>				print
D <sub>3</sub>		read	execute	
D <sub>4</sub>	read write		read write	

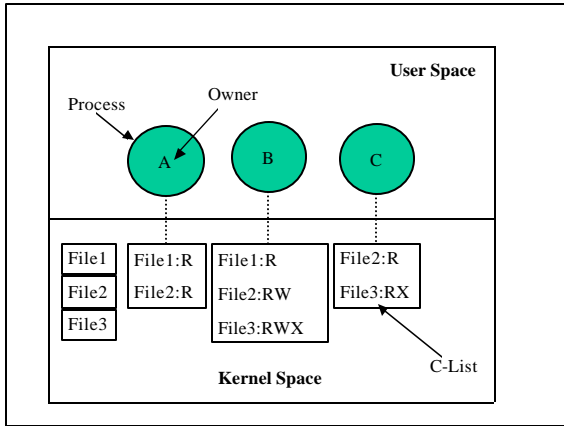
### Implementation of Access Matrix

---

- Each column = Access-control list for one object  
Defines who can perform what operation.  
File1 → A:RW; B:A  
File2 → A:R; B:RW; C:R  
File3 → B:RWX; C:RX
- Each Row = Capability List (like a key)  
Fore each domain, what operations allowed on what objects.  
User A → File1:R; File2:R  
User B → File1:R; File2:RW; File3:RWX  
User C → File2:R; File3:RX

www.genctiles.com/anjibcs 413 manjib@mail.com.np





### Access Matrix With Domains as Objects

domain \ object	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	laser printer	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>
D <sub>1</sub>	read		read			switch		
D <sub>2</sub>				print			switch	switch
D <sub>3</sub>		read	execute					
D <sub>4</sub>	read write		read write		switch			

www.geocities.com/anjibcs 416 manjib@mail.com.np

- ### The Security Problem
- Security must consider external environment of the system, and protect it from:
    - unauthorized access.
    - malicious modification or destruction
    - accidental introduction of inconsistency.
  - Easier to protect against accidental than malicious misuse.
- www.geocities.com/anjibcs 417 manjib@mail.com.np

- ### Authentication
- User identity most often established through *passwords*, can be considered a special case of either keys or capabilities.
  - Passwords must be kept secret.
    - Frequent change of passwords.
    - Use of "non-guessable" passwords.
    - Log all invalid access attempts.
  - Passwords may also either be encrypted or allowed to be used only once.
- www.geocities.com/anjibcs 418 manjib@mail.com.np

- ### System Threats
- Worms - use spawn mechanism; standalone program
  - Internet worm
    - Exploited UNIX networking features (remote access) and bugs in *finger* and *send mail* programs.
    - Grappling hook program uploaded main worm program.
  - Viruses - fragment of code embedded in a legitimate program.
    - Mainly effect microcomputer systems.
    - Downloading viral programs from public bulletin boards or exchanging floppy disks containing an infection.
    - *Safe computing*.
  - Denial of Service
    - Overload the targeted computer preventing it from doing any useful work.
- www.geocities.com/anjibcs 419 manjib@mail.com.np

### THE END

www.geocities.com/anjibcs 420 manjib@mail.com.np