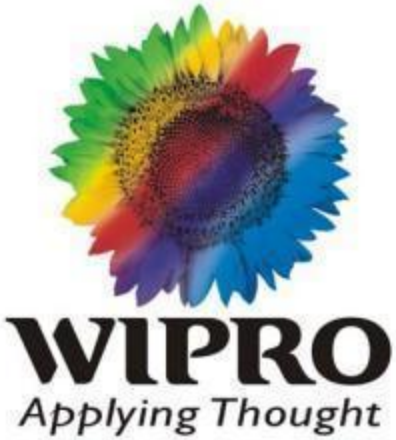


Operating Systems

Agenda

1	Introduction
2	Functions of Operating Systems
3	Process Management
4	CPU Scheduling
5	File Management
6	Memory Management
7	Memory Management (Advanced)

Introduction

Module 1

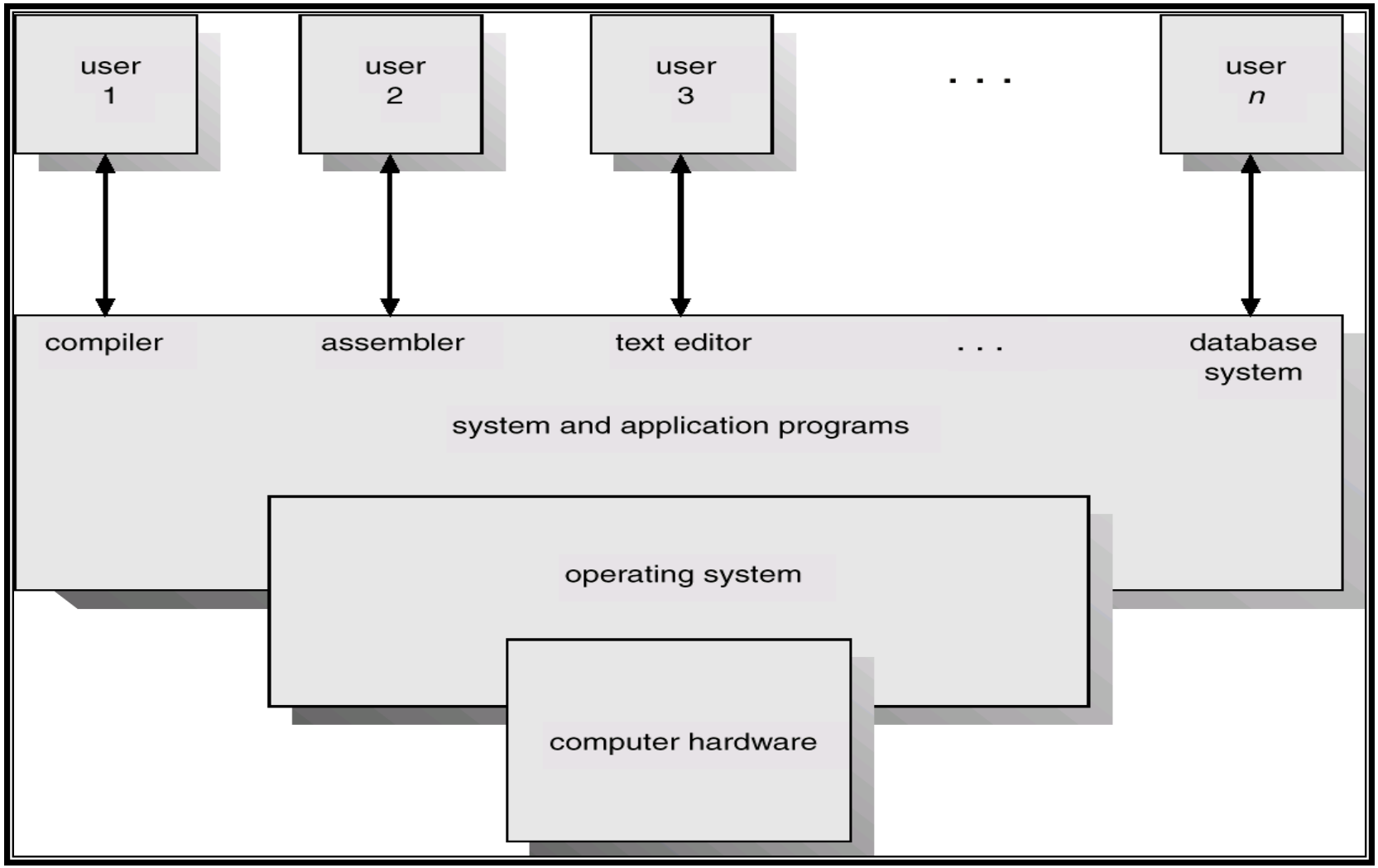
Objectives

At the end of this module, you will be able to:

- Recognize the need of Operating Systems
- Identify Computer System Components
- Recognize Functions of Operating Systems
- Explain various types of Operating System

Duration: 1 hr

Conceptual View of System Components



Computer System Components

- Hardware
  – provides basic computing resources (CPU, memory, I/O devices)

- Operating system
  – controls and manage the use of hardware among the various application programs for the various users

- Applications programs
  – define the ways in which the system resources are used to solve the computing problems of the users (video games ,compilers, database systems,, business programs)

- Users (computers, machines, people)

Goals of Operating System

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

- Operating system goals:
  - Execute user programs and assist user to solve problems
  - Make the computer system easy to use
  - Use the computer hardware in an proficient manner
  - Provides information protection
  - Gives each user a slice of the resources

Functions of Operating Systems

- Resource allocator
  – manages and allocates resources

- Control program
  – controls the execution of user programs and operations of I/O devices

- Kernel
  – the one program running at all times (all else being application programs)

Types of OS

- General Purpose Operating System (GPOS)
  - Desktop System

- Real-Time Operating System (RTOS)
  - Time sharing Operating Systems

Time-Sharing Systems –Interactive Computing

- The CPU is multiplexed among several jobs that are kept in memory and on disk.
- A job swapped in and out of memory to the disk.
- On-line communication between the user and the system is provided; when the operating system finishes the execution of one command, it seeks the next command from the user.

Desktop Systems

- Personal computers
  – computer system dedicated to a single user

- Input/output devices
  – keyboards, display screens, printers

- User convenience and responsiveness

- Can adopt technology developed for larger operating system often individuals have sole use of computer and do not need high-end CPU utilization of protection features.

- May run several different types of operating systems (Windows, Mac OS, UNIX, Linux)

Real-Time Systems

- Often used as a control device in a dedicated application
  - controlling scientific experiments
  - medical imaging systems,
  - industrial control systems
  - few display systems
- Well-defined fixed-time constraints
- Real-Time systems may be either hard or soft real-time

Real-Time Systems (Contd.).

- Hard real-time
  - Secondary storage maybe absent or limited, data stored in short term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, and not supported by general-purpose operating systems

- Soft real-time
  - Limited utility in industrial control of robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features

Handheld Systems

- Personal Digital Assistants (PDAs)
- Cellular telephones
- Issues:
  - Limited memory
  - Slow processors
  - Small display screens

Other OS types

- Mainframe
- Parallel processor
- Distributed systems
- Client Sever models

Summary

In this module, we discussed:

- Role of Operating System
- Abstract view of Operating System
- Different types of Operating System

Functions of Operating System

Module 2

Objectives

At the end of this session, you will be able to:

- Explain the functions performed by an OS
- Explain File Management in Operating Systems
- Describe the concept of Interrupts
- Explain the I/O Structure
- Identify Security and Protection in OS

Duration: 2 hrs

Operating System Functionalities

- An operating system performs large number of functions. Each function is carried out by a component of the operating system.

- The components of an operating system are:
  - Process management sub system
  - Memory management sub system
  - File management sub system
  - I/O system management sub system
  - Secondary storage management sub system
  - Network management sub system
  - Protection sub system
  - User interface sub system

File Management

- The main functionalities of File management system are:
  - The creation and deletion of files
  - The creation and deletion of directories
  - The support of primitives for manipulating files and directories
  - The mapping of files onto secondary storage
  - The back up of files on permanent storage media

Main-Memory Management

- Program to be executed, it need to be in primary memory.

- Major task of OS
  - keep track of which part of memory are currently being used and by which program
  - decide which process are loaded into memory when memory space becomes available
  - Allocate and reallocated memory space as needed

Process Management

- A process is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task

- The operating system is responsible for the following activities in connection with process management

  - Process creation and deletion
  - Process continuation and suspension
  - Provision of mechanisms for:
    - process synchronization
    - process communication

I/O Management - structure

- **After I/O starts, control returns to user program only upon I/O completion.**
  - Wait instruction idles the CPU until the occurrence of next interrupt
  - Wait loop (memory access contention)
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing
- **After I/O starts, control returns to user program without waiting for I/O completion**
  - System call – request to the operating system to allow user to wait for completion of I/O
  - Device-status table maintains a details of each I/O device such as device type, its address, and state
  - Operating system looks into I/O device table to find device status such that to modify table entry to include interrupt

I/ O Management - Operations

- Each program needs an input and generates output.
- User desires that the I/O has been performed without any details.
- User looks for convenience to run programs.
- Hides the user the details of underlying H/W for the I/O.

Security Management & User management

- Login
- User name
- Password
- Environment variables
- Profile files

Security Management (Contd.).

User Management:

- All users will have unique identifier typically, user name in the system
- Different types of users are as follows
  - Regular Users
  - Administrator's
  - Guest Users

Computer System Operation

- CPU and I/O devices can run concurrently.
- Every I/O device controller will be in charge of a particular device type.
- Every device manager has a local buffer.
- CPU moves data from/to main memory to/from local buffers

Interrupts

Common Functions :

- Transfer of control to interrupt service routine through interrupt vector, control having addresses of all the service routines.
- Architecture needs to save address of the interrupted instruction.
- Disabling incoming interrupts, especially when another interrupt is being processed to prevent a lost interrupt.

Interrupt Handling

- The Operating System needs to maintain state of the CPU by storing register values and the program counter particulars.

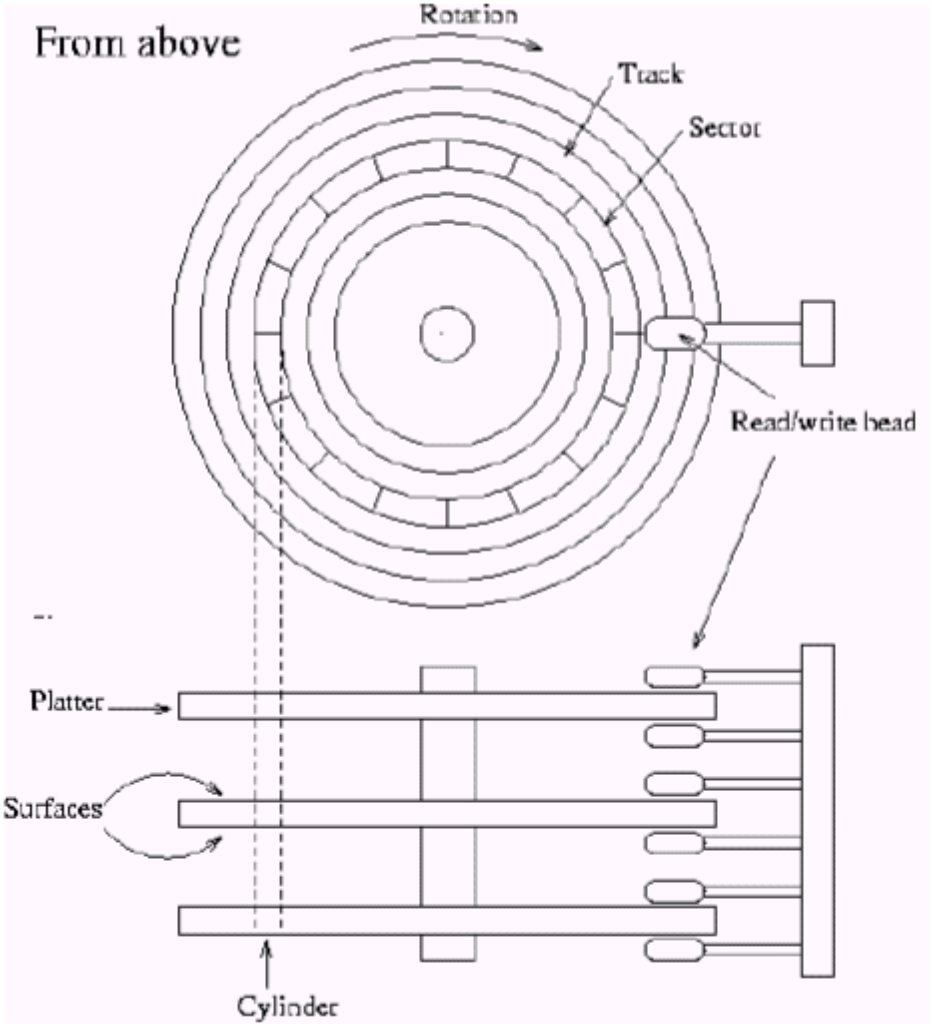
- Require to determine type of interrupt such as
  - polling
  - vectored interrupt system

- Separate segments of code will determine what action need to be taken for each type of interrupt.

Storage Structure

- Main memory
- Secondary storage
- Magnetic disks

Moving-Head Disk Mechanism



Direct Memory Access Structure

- Used for high-speed I/O devices
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only on interrupt is generated per block, rather than the one interrupt per byte

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility

- Caching – copying information into faster storage system

Caching

- High-speed memory is used to hold data accessed recently
- Requires a cache management policy
- Caching introduces one more level in storage hierarchy.
- Caching requires data that is concurrently stored in more than one level and when it need to be consistent

Memory & I/O Protection

- Every I/O instructions are usually privileged instructions
  - Need to ensure that a user program will not gain control of computer in monitor mode
- Must provide memory protection at least for interrupt vector and interrupt service routines
- To achieve memory protection added two registers, which will determine the range of legal addresses of a program may access such as
  - **Base register**
  - **Limit register**
- Memory outside the defined range is protected

35

CPU Protection

- Timer – interrupts computer system after specified period of time to ensure operating system retains the control.
- Timer is regularly used to implement time sharing
- Timer is furthermore used to calculate the current time
- Load-timer is a privileged instruction
- After some period to guarantee

36

Booting a system

- Loading the operating system into memory is called booting the system
- The word booting is used because figuratively speaking the operating system pulls itself up by its bootstraps

Summary

In this module, we discussed :

- Operating System - functions
- Operating Systems - File Management
- Interrupts
- OS - I/O Structure
- OS - Security & Protection

Process Management

Module 3

Objectives

At the end of this session, you will be able to:

- Define Process model
- Describe Process States and Transitions
- Explain Process life cycle
- Define Threads
- Discuss user level threads library

Duration: 3 hrs

Process Management

- Processes carry out tasks within the operating system
- A process can be thought of as a program in execution
- Examples for multiprocessing systems are : UNIX/LINUX

Processes

- Many resources are used by the process during its lifetime.
- Operating system need to keep track of the process itself and managing system resources along with other processes in the system fairly

Process Execution

Process mode:

In case of UNIX, the computer hardware must provide two modes of execution

- kernel mode
- user mode

However, some computers will take more than two execution modes

eg: Intel m/cs

Each process has virtual address space , references to virtual memory are translated to physical memory locations using set of address translation maps

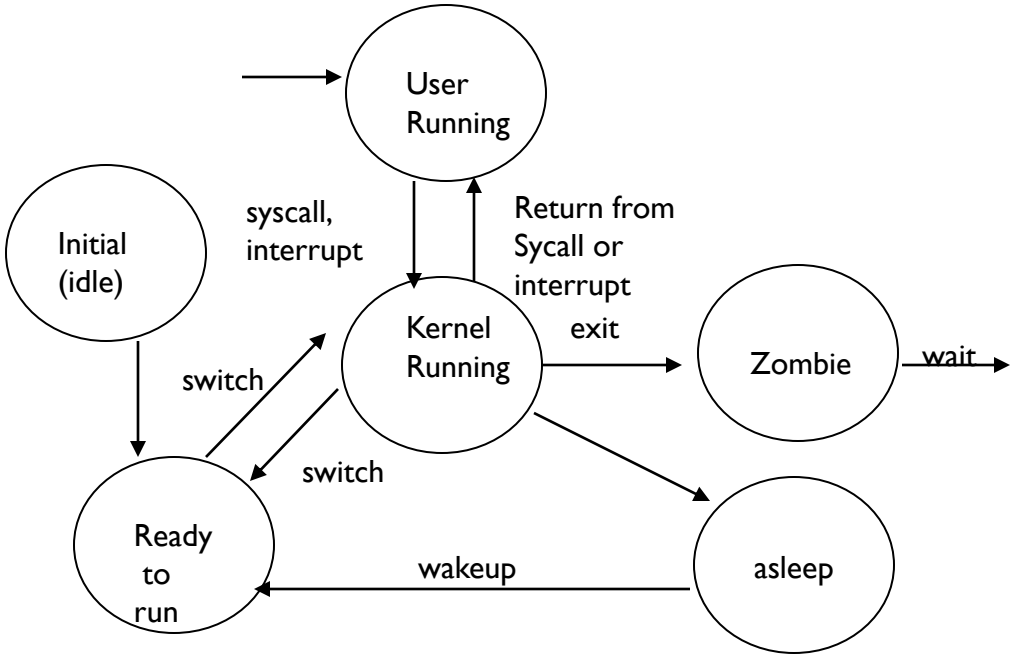
Process Life cycle

- As a process executes it changes state according to its circumstances.

- Standard Unix/Linux processes have the following states:

– Ready
– Running
– Wait
– stopped
– zombie

Process Life cycle (Contd.).



Times and Timers

- The kernel keeps track of a processes creation time as well as the CPU time that it consumes during its lifetime
- Each clock tick, the kernel updates the amount of time that the current process has spent in system and in user mode
- Linux also supports process specific interval timers.
- Processes use system calls to set up timers to send indications to themselves when the timers expire.

Process Scheduling

- All processes run partially in user mode and partially in system mode
- User mode has far less privileges than system mode. Each time a process makes a system call it swaps from user mode to system mode and continues execution
- At this point the kernel is executing on behalf of the process
- The program that selects the most justifiable process to execute of all of the executable processes in the system is known as Scheduler.

Process Creation

- Parent process creates derived processes, which, in turn create other processes, forming a tree of processes
- Resource sharing
  - Parent and derived one will share all resources
  - Derived one will share subset of parent's resources
  - Parent and Derived one will share no resources
- Execution
  - Parent and derived process run concurrently
  - Parent waits until derived process terminate
- Address space
  - Derived process duplicate of parent.
  - Derived process has a program loaded into it

Process & Thread

- Process:

– In computing, a process is an instance of a computer program that is being sequentially executed by a computer system that has the ability to run several computer programs concurrently

- Thread:

– A single process may contain several executable programs(threads) that work together as a coherent whole

- The Thread that share global data and address space with other threads running in the same process

Thread

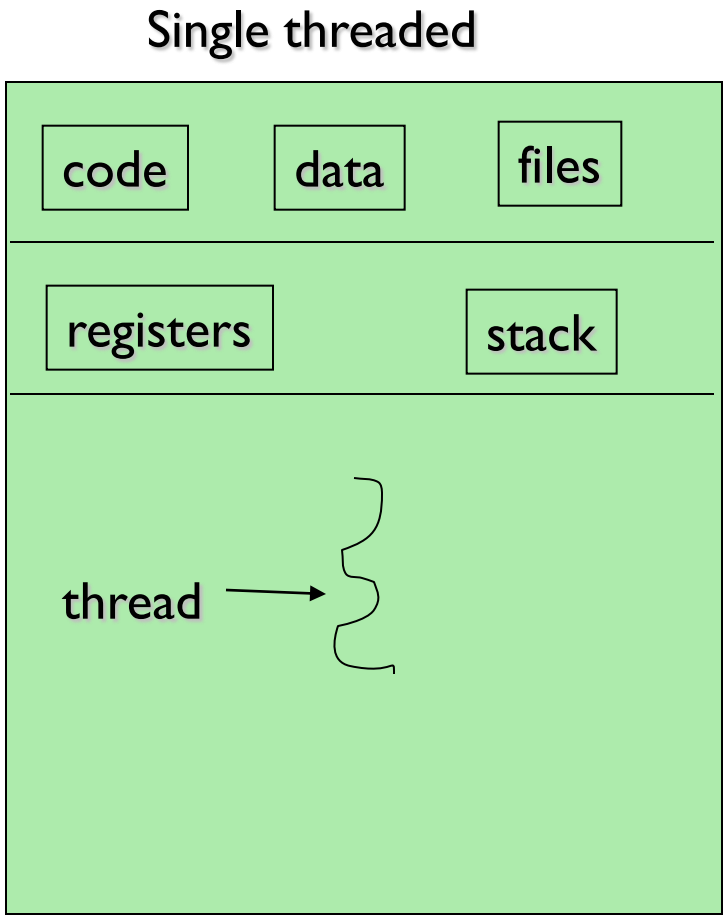
- What is a thread?

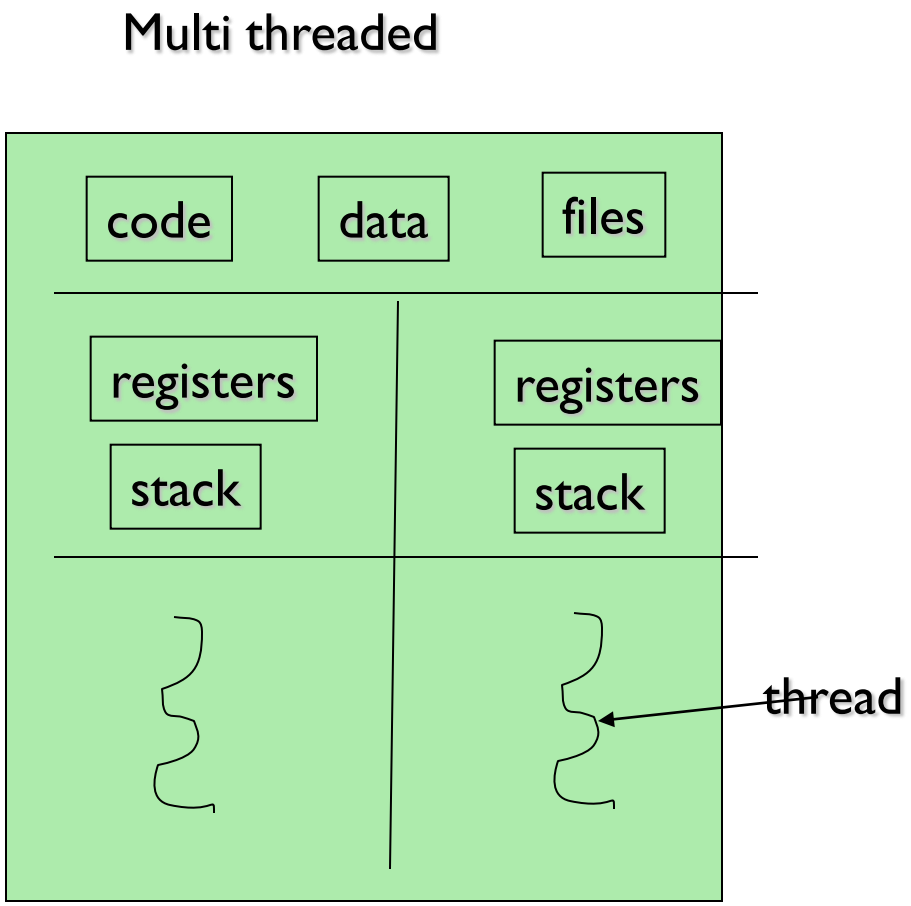
- A thread is an encapsulation of the flow of control in a program
- Putting it other way a thread is Single Flow of Execution / Control

Multiple Threads:

- Multiple Flows of Execution / Control

Single and Multithreaded Processes





Threads - Benefits

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

Types of Thread

- User Thread
- Kernel Threads
- PThreads

User Threads

- User-level threads library will manages the Threads
- Thread library contains code for instantiating and destroying threads,.
- Passing messages and data between threads, for scheduling thread execution and for saving and restoring thread contexts
- Examples:

  - POSIX Threads
  - Mach C-threads
  - Solaris threads

Kernel Threads

- Supported by the Kernel
- Thread management is done by the kernel
- If one thread in a process is blocked, kernel usually schedules another thread of the same process
- Transfer of control from one thread to another process (within) requires a mode switch to the kernel

Pthreads

- API specifies behavior of the thread library, implementation is up to development of the library
- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization
- Common in UNIX operating systems

Process Management : Hands-on (1 hour)

- Explore commands of Unix OS to illustrate process
  - Starting a process (use any command)
  - List running processes (*ps*)
  - Stop the process (*kill*)
- Explore General purpose command of Unix OS
  - Date, who, who am I, ls

Summary

In this module, we discussed:

- Process and different states of Process
- Threads and usage of Threads

CPU Scheduling

Module 4

Objectives

At the end of this session you will be able to:

- Define CPU Scheduling
- Explain Scheduling Criteria
- Explain various Scheduling Algorithms

Duration: 2 hrs

CPU Utilization

Maximum CPU utilization obtained with multiprogramming

CPU – I/O Burst Cycle – Process execution consists of a cycle of CPU execution and I/O wait

CPU burst distribution

CPU Scheduling

- Scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them

- CPU scheduling decisions may take place when a process:

  - Is Switching between running and waiting state
  - Is Switching between running and ready state
  - Is Switching between waiting and ready
  - Is Dead

- Scheduling under 1 and 4 is non-preemptive

- All other scheduling is preemptive

Scheduling Criteria - Optimization

- CPU utilization
  – keep the CPU as busy as possible

- Throughput
– # of processes that complete their execution per time unit

- Turnaround time
  – amount of time to execute a particular process
- Waiting time
– amount of time a process has been waiting in the ready queue

- Response time
– amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Scheduling Algorithm

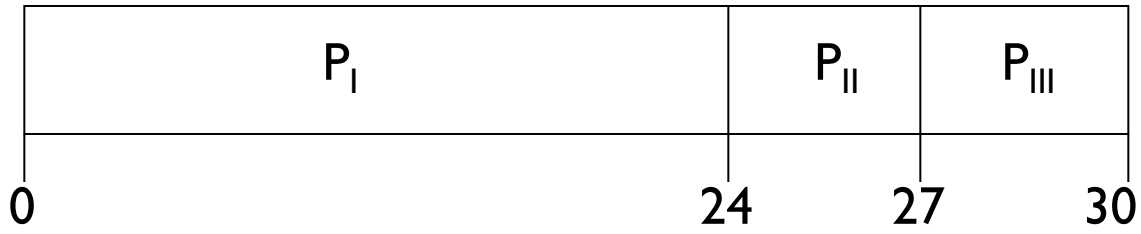
- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
- Priority Scheduling
- Round Robin (RR) Scheduling
- Real-Time Scheduling

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_I	24.0
P_{II}	3.0
P_{III}	3.0

Suppose that processes appear in the order: P_I , P_{II} , P_{III}

The Gantt Chart for the schedule is:



Waiting time for P_I = 0.0; P_{II} = 24.0; P_{III} = 27.0
Average waiting time: (0.0 + 24 .0+ 27.0)/3 = 17.0

FCFS Scheduling (Contd.).

Suppose that processes appear in the order P_{II} , P_{III} , P_I

The Gantt chart for the schedule is:

Waiting time for P_I = 6.0; P_{II} = 0.0; P_{III} = 3.0
Average waiting time: (6.0 + 0.0 + 3.0)/3 = 3.0

Much better than previous case

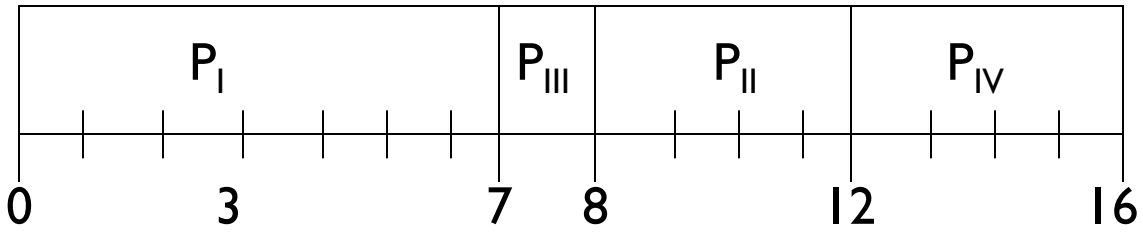
Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- Two schemes:
  - Non-preemptive
  - preemptive
- SJF is optimal

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_I	0.0	7
P_{II}	2.0	4
P_{III}	4.0	1
P_{IV}	5.0	4

SJF (non-preemptive)

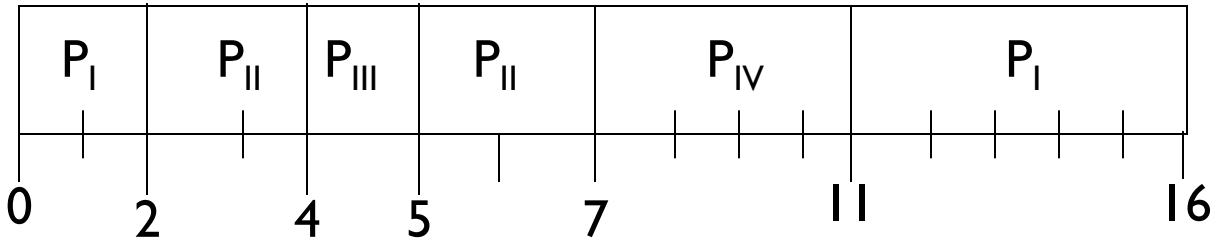


Average waiting time = (0 + 6 + 3 + 7)/4

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_I	0.0	7.0
P_{II}	2.0	4.0
P_{III}	4.0	1.0
P_{IV}	5.0	4.0

SJF (pre-emptive)



Average waiting time = (9 .0+ 1.0 + 0.0 +2.0)/4 =3

Priority Scheduling

- Each process associates with priority.
- The maximum priority process will be allocated by the CPU.
  - Pre-emptive
  - Non-pre-emptive

- SJF is a unique of priority scheduling with priorities associated according to the no. of CPU burst necessary by the processes.

- Problem =Starvation – low priority processes will not get a chance to execute
- Solution =Aging – as time progresses increase the priority of the process

Round Robin (RR) Scheduling

- On the principle that CPU must be shared between each ready process.
- CPU time is divided into time-quantum or time-slice.
- The ready processes are queued up in a circular queue.
- Context switching takes place whenever either the running process exhausts its allocated time-quantum or completes its execution.

Summary

In this module, we discussed:

- CPU Scheduling
- Scheduling Criteria
- Range of Scheduling Algorithms

File Management

Module 5

Objectives

At the end of this session, you will be able to:

- Identify file types, characterstics of file
- Explain various operations performed on file
- Explain Directory Structure, characteristics of directory and operations
- Identify the features of file sharing system – protection

Duration: 4 hrs

File System

- Provides the mechanism for online storage and access to both data and programs

- User may have expectations from file system are:
  - suitable and quick access to files
  - Consistent storage of files
  - Controls file sharing

File Types

- Data
- Program
- Documents

File Name & Extension

File type	Usual extensions	function
Executable	exe, com, bin or none	Read to run machine-language program
Object	obj, o	Compiled, machine language, not linked
Source code	c,cc,java,pas,asm	Source code in various language
Text	txt, doc	Textual data, documents
Library	Lib, a, so, dll, mov,rm	Libraries of routines for programmers
Archive	Arc,zip,tar	Related files grouped into one file.
Multimedia	Mpeg,mov,rm	Binary file containing audio or A/V information

File Properties

- Name
- Type
- Location
- Size
- Protection
- Time, date, and user identification
- Information about files are kept in the directory structure, which is maintained on the disk

File Operations

- Major file operations are:
- Read operation
- Write operation
- Execute
- Coping
- Renaming
- Delete
- Open(Fx)

  – search the directory structure on disk for entry Fx, and loads the contents in the memory.
- Close (Fx)

  – moves the contents of entry Fx in memory to directory structure on to the disk.

Open Files

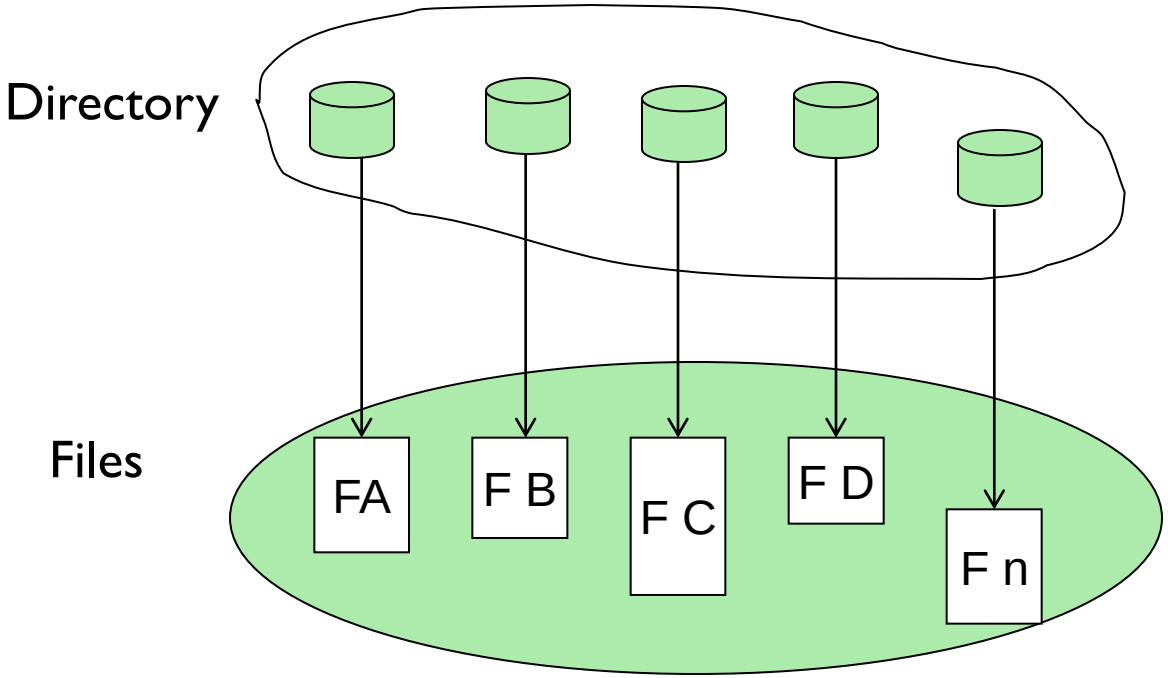
- Data that are needed to be managed while opening files:
  - File pointer
  - File-open count:
  - Disk location of the file
  - Access rights

Directory Organization - Goals

- Efficiency - locating a file quickly
- Naming - convenient to users
  - Two users can have the same name for dissimilar files
  - The same file can have several dissimilar names
- Grouping
  - Logical grouping of files by properties (eg: all Pascal programs, all games…)

Directory Structure

- A collection of nodes containing information about all files



Both files and directory structure reside on disk

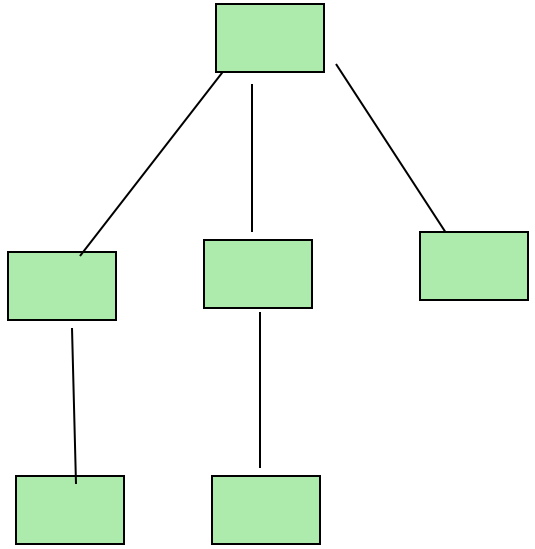
Backups of these two structures are kept on tapes

Directory Structure (Contd.).

- File Name
- File Type
- Address or Location
- Current Length
- Maximum Length
- Date created
- Date last updated (may be to dump)
- Date last accessed (may be to perform archival)
- Owner ID and
- Protection information

Directory Structure Example

- Tree-Structured Directories :
- Efficient searching
- Current directory (working directory)
- Grouping Capability
- Easy maintenance



Directory - Operations

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file
- Traverse the file system

Access Methods

- Linear Access Method
- Direct Access Method (n = relative block number)

File Sharing

- In multi-user (systems) environment file sharing would be desirable
- Sharing may be done through a **protection** scheme
- In distributed systems, files may be required to share across the network
- Network File System is a universal distributed file-sharing method

File Sharing – Multiple Users

- **User IDs** identify users, allowing permissions and protections to be per-user

- **Group IDs** allow users to be in groups, permitting group access rights

File Sharing – Remote File Systems

- Uses networking to allow file system access between systems
  - Manually via programs like FTP
  - Automatically, easily using distributed file systems
  - Semi automatically via the world wide web

- Client-server model allows clients to mount remote file systems from servers
  - Server can serve multiple clients
  - Client and user-on-client identification is insecure or complicated
  - NFS is standard UNIX client-server file sharing protocol
  - CIFS is standard Windows protocol
  - Remote calls are translated by standard operating system file calls

- Distributed Information Systems (distributed naming services) such as Lightweight Directory Access Protocol (LDAP), Domain Name System (DNS), Network Information Service (NIS) implement unified access to information needed for remote computing

File Sharing – Failure Modes

- Remote file systems add new failure modes, due to network failure, server failure
- Recovery from failure can involve state information about status of each remote request
- Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security

Protection

- File owner or creator should be able to control:
  - what can be done
  - who to do (by whom)

- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

File System - Unix

- Organized in tree structure
- File tree can be arbitrarily deep
- File name length must not be more than 256 chars
- Single path name length must not be more than 1023 chars

Example (Contd.).

- Creating a File System
- Mounting File System
  - File tree is composed of File System
  - *mount*
    - *mount /dev/hda4 /usr*
  - *umount*
    - Detaching will fail if the file system is busy

Example – (Contd.).

- Types of Files

- Regular Files
  - binary
    - GIF, JPEG, Executable etc
  - text
    - scripts, program source code, documentation
  - Supports sequential access and random access
- Device File
  - Allows programs to interact with hardware
  - Kernel unit handles device management
- Directory
  - Can contain any type of files
  - Think about "." and ".."??

Example – (Contd.).

- File Permissions:-

- The Setuid and Setgid bits
  - Setuid with octal value 4000
  - Setgid with octal value 2000

File System: Hands-on (2 hours)

- Using Unix commands explore file related operations
  - Creating / opening a file (*cat*)
  - Copying file (*cp*)
  - Removing (*rm*)

Summary

In this module, we discussed:

- Different operations of File System
- Different techniques of storing the data
- Usage of File permissions

Memory Management

Module 6

Objectives

At the end this, session you will be able to:

- Identify need of Memory Management
- Explain operations and Techniques of Memory Management unit
- Explain Swapping Technique
- Describe Contiguous Allocation
- Explain Paging Technique

Duration: 2 hrs

Memory Management - why?

- Program need to brought into memory and to be found within a process for it to be run

- Input queue

  – collection of processes on the disk waiting to be brought into memory to run the program

- User programs go through several steps before being run

Memory Management Why?(Contd.).

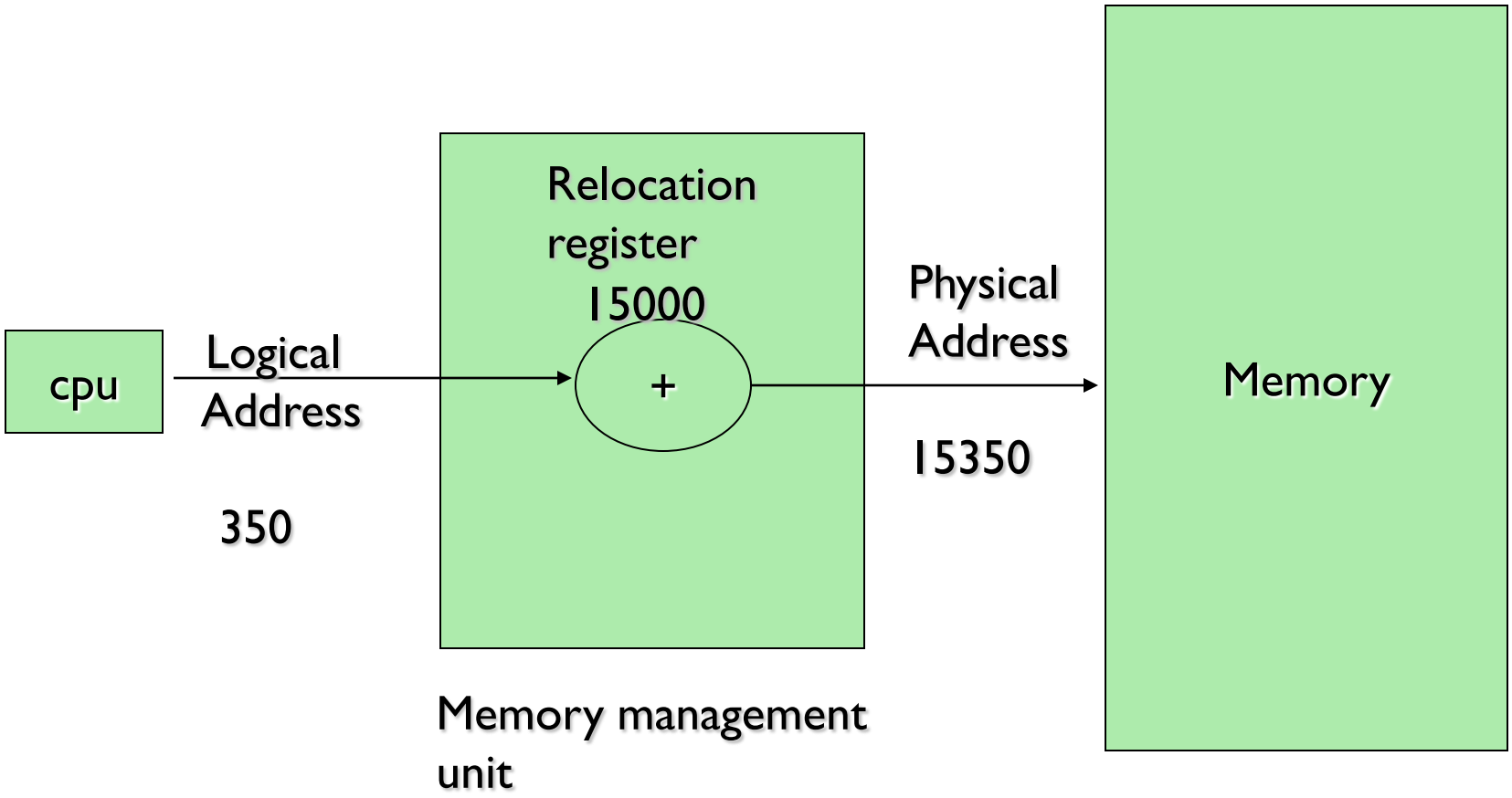
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory

- The user program deals with logical addresses; it does not see the real physical addresses.

Memory Management - How?

- Logical versus Physical Address
- Logical address space that is bound to a separate physical address space is central to right memory management
- Logical address
  – address generated by CPU; also known as virtual address
- Physical address
  – address seen by the memory unit
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme

Memory Management Operations



Memory Management operations (Contd.).

- Better memory-space utilization; routine not used is never loaded.
- Especially useful when, Large amounts of code are needed to handle infrequently occurring cases.
- No special support required from the operating system as implemented through program design

Dynamic Linking

- Dynamic linking is particularly useful for libraries.
- Linking postponed until execution time.
- Small piece of code, stub, used to locate the appropriate memory-resident library routine.
- Stub replaces itself with the address of the routine, and executes the routine.
- OS needs to check if routine is in processes' memory address.

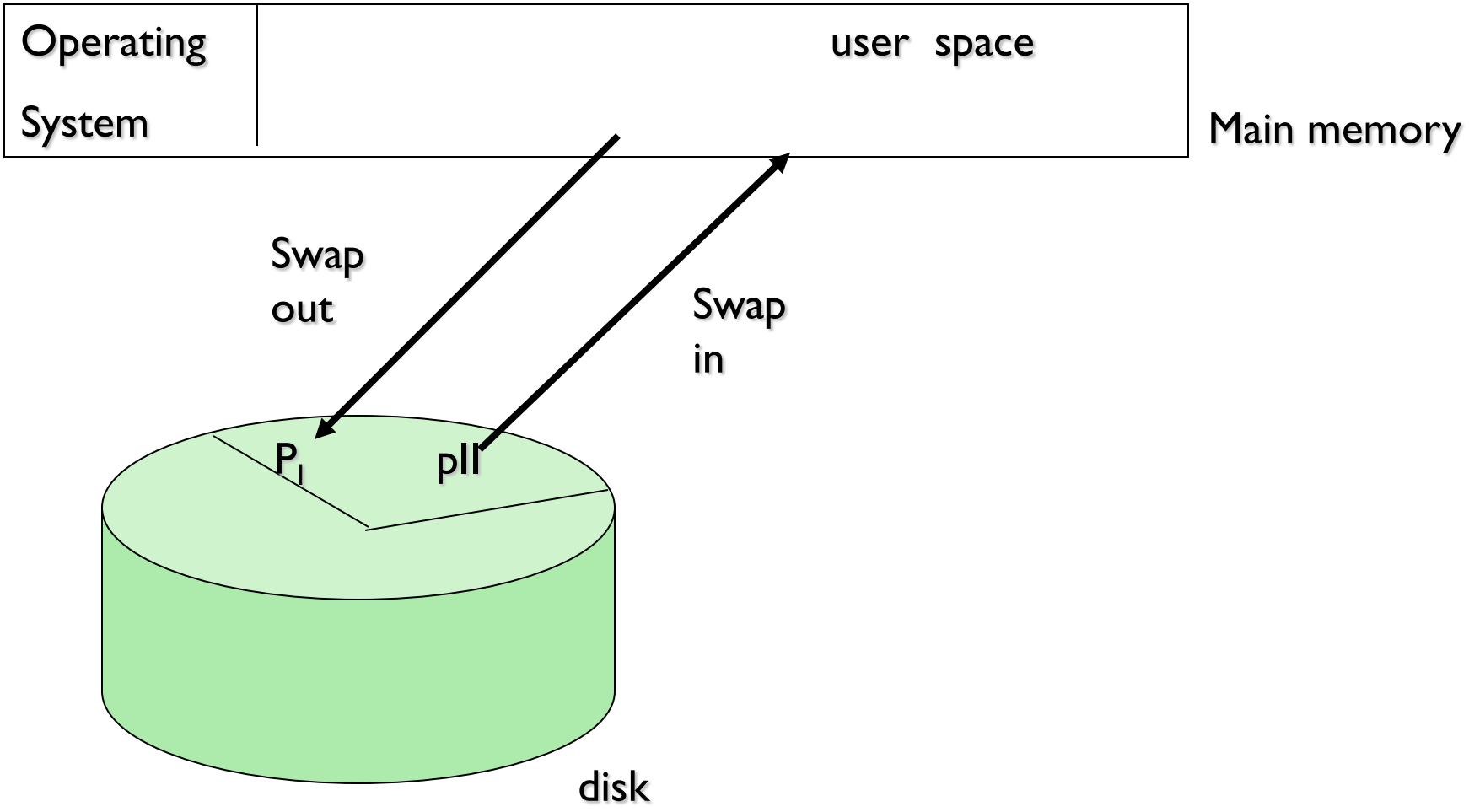
Overlays

- Maintains in memory only those statements and data required at any given time.
- Required when process is more than amount of memory allocated to it.

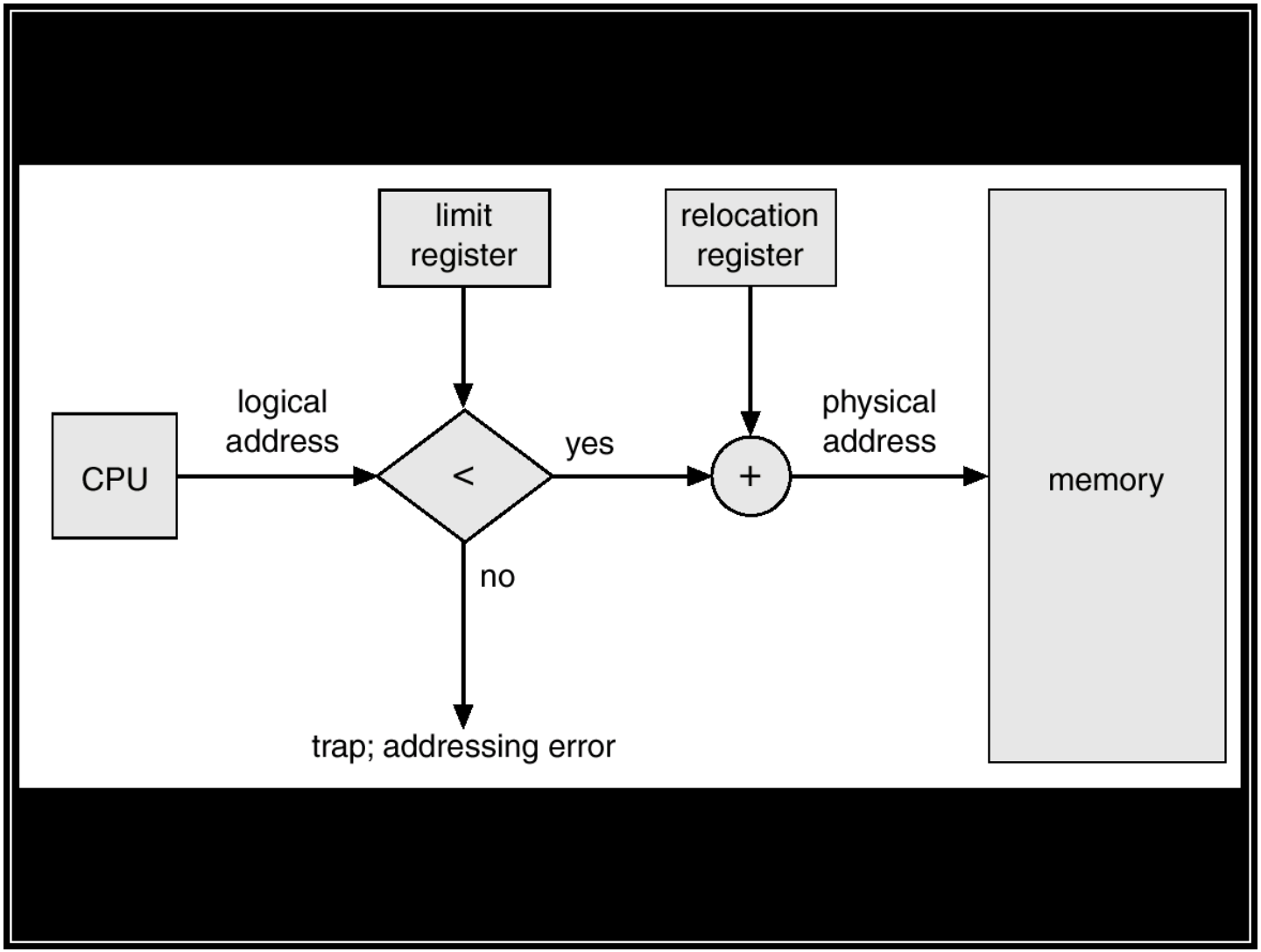
Swapping

- back up storage – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped
- Modified versions of swapping are found on many systems,.

Swapping - Schematic View



Relocation and Limit Registers - Hardware Support



Contiguous Allocation

- Main memory usually into two partitions:

– Resident operating system, usually held in low memory with interrupt vector.
– User processes then held in high memory Single- partition allocation
– Relocation-register scheme used to protect user processes from each other, and from changing operating-system code and data.
– Relocation register contains value of smallest physical address; limit register contains range of logical addresses – each logical address must be less than the limit register

Contiguous Allocation (Contd.).

- Multiple-partition allocation

– Hole – block of available memory; holes of various size are scattered throughout memory
– When a process arrives, it is allocated memory from a hole large enough to accommodate it
– Operating system maintains information about:
– a) allocated partitions b) free partitions (hole)

Dynamic Storage-Allocation Problem

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

- **How to satisfy a request of size *n* from a list of free holes**
- **First-fit and best-fit better than worst-fit in terms of speed and storage utilization**

Summary

In this module, we discussed

- Dynamic Loading
- Dynamic Linking and swapping

Memory Management (Advanced)

Module 7

Objectives

At the end of this session, you will be able to:

- Define fragmentation
- Explain technique used to implement fragmentation
- List various Inter process communications methods
- Explain Pipe technique to implement IPC
- Define Semaphore

Duration: 2 hrs

Fragmentation

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

- Internal Fragmentation – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

- Decrease external fragmentation by compaction

– Mix up memory contents to place all free memory together in one large block
– Compaction is possible only if relocation is dynamic, and is done at execution time
– I/O problem
• Latch job in memory while it is involved in I/O
• Do I/O only into Operating System buffers

Paging

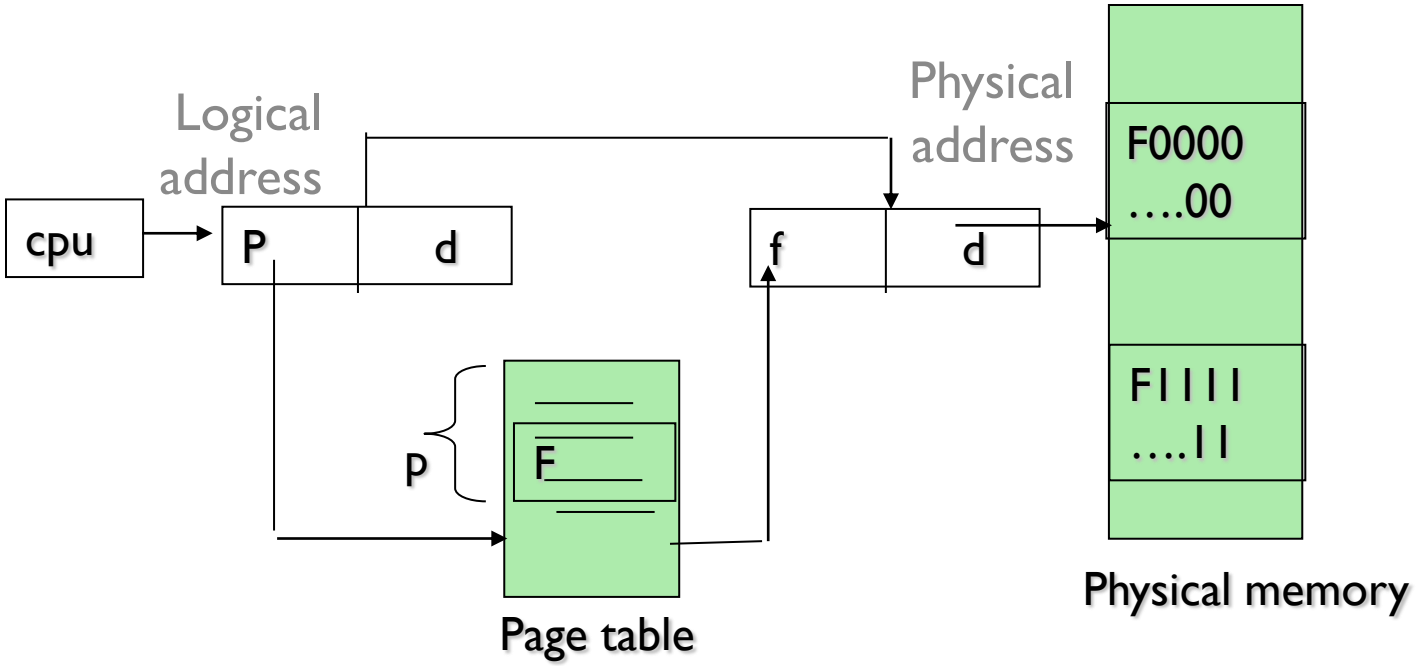
- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
- Divide physical memory into fixed-sized blocks called frames
- Divide logical memory into blocks of similar size called pages
- Follow-up of all free frames
- To run a program of size n pages, need to find n free frames and load program

Paging - Address Translation Scheme

Address generated by CPU is bifurcated into:

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

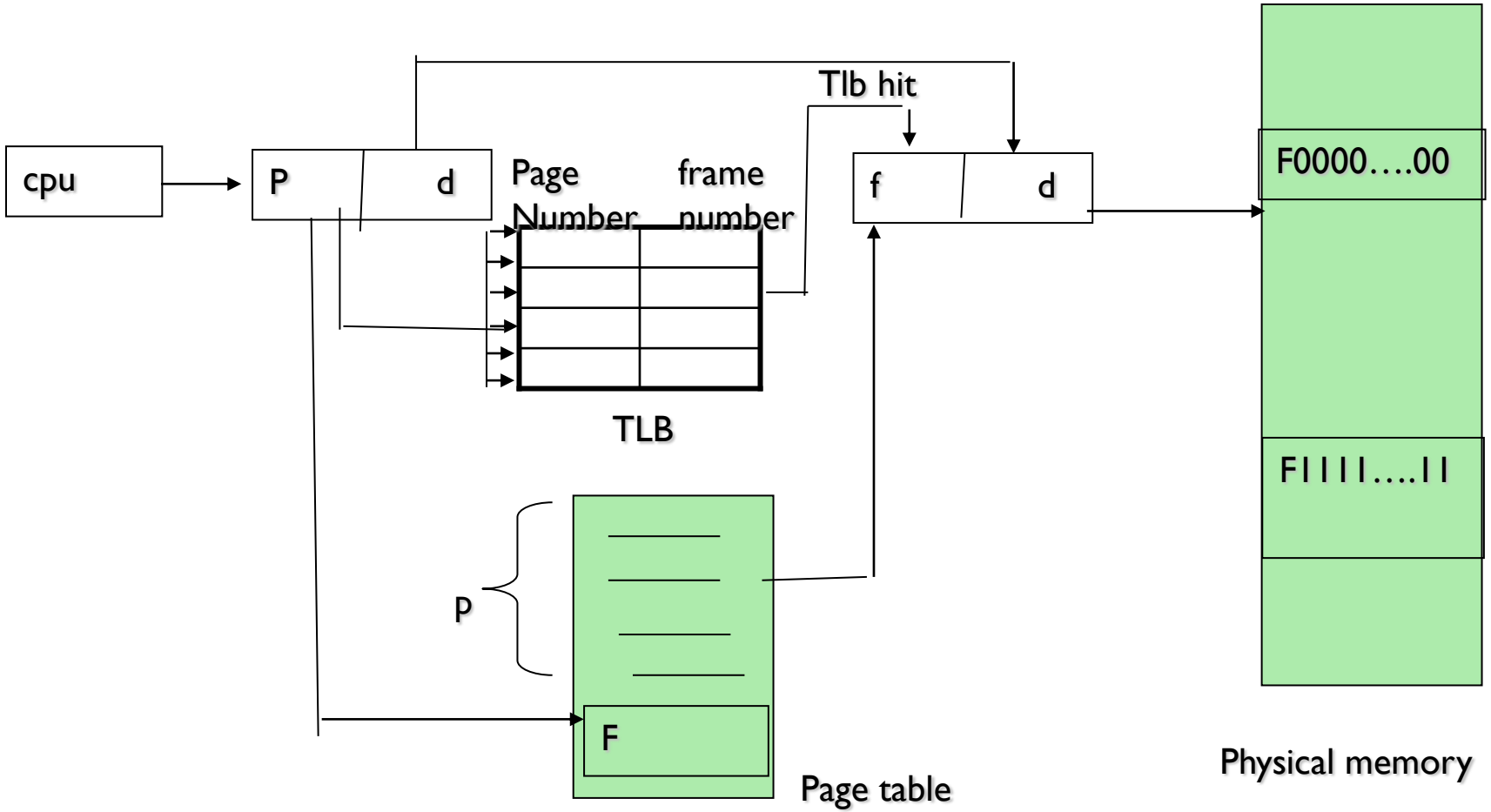
Page offset (d) – combined with base address to define the physical memory address that is sent to the memory unit



Implementation of Page Table

- Page table is kept in main memory
- Page-table base register (PTBR) points to the page table
- Page-table length register (PRLR) indicates size of the page table
- In this scheme every data/instruction access requires two memory accesses. One for the page table and one for the data/instruction
- The two memory access problem can be solved by the use of a special fast-lookup hardware cache called associative memory or translation look-aside buffers (TLBs)

Paging Hardware With TLB



Inter Process Communication

- A way for processes to converse and to synchronize their actions
- Message system – processes communicate with each other without resorting to shared variables.
- IPC facility presents two operations:
  - send(message) – receive(message)
- If P and Q wish to converse , they need to:
  - establish a communication link between them
  - swap messages through send/receive
- Implementation of communication link
  - physical (e.g., shared memory, hardware bus)
  - logical (e.g., logical properties)

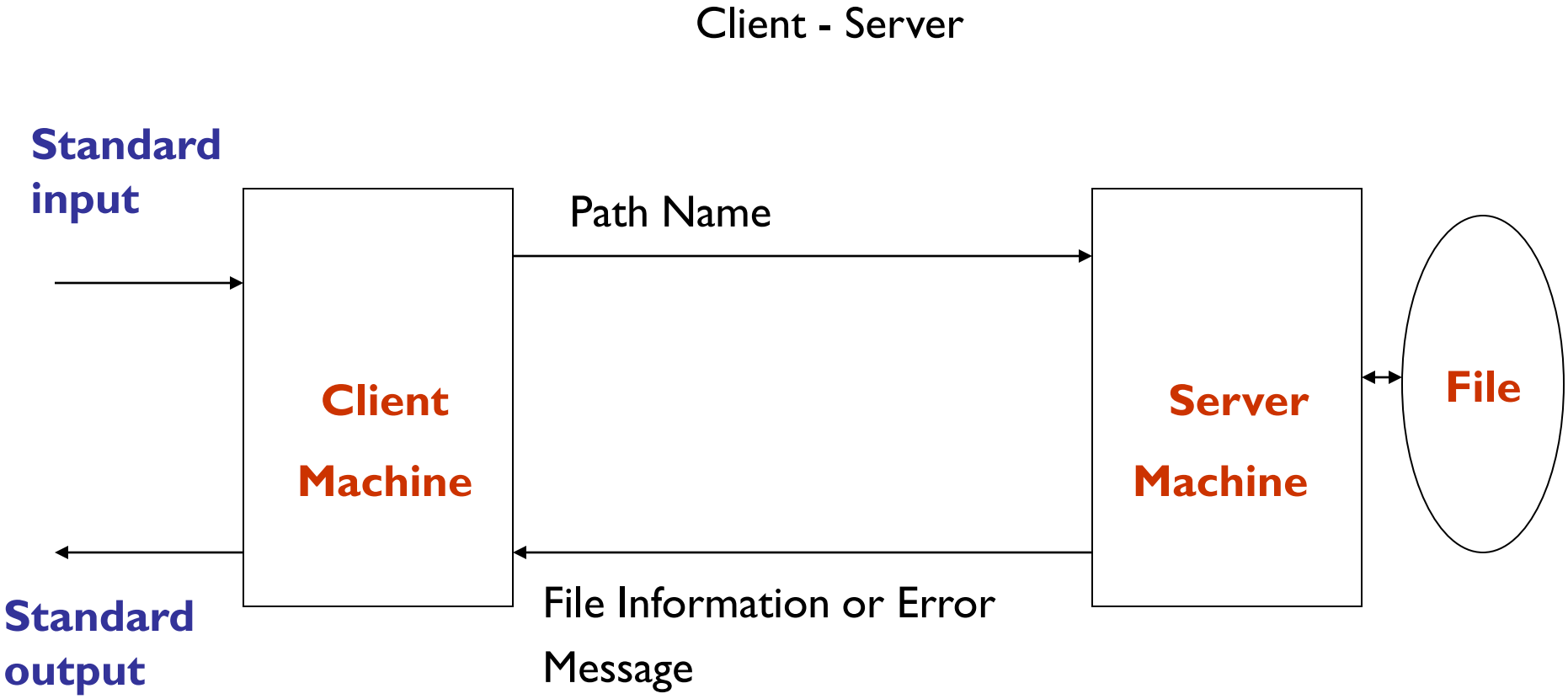
Inter Process Communications - Methods

- Pipe
- FIFO
- Message queues
- Shared memory
- Semaphores

PIPE Definition

- A pipe is a set of two file descriptors. A pipe allows two associated processes to communicate by sending some data from one process to another process
- The processes must co-operate and assume the role of a reader or a writer with respect to a specific pipe
- Data is passed in order
- Data doesn't get lost in middle
-
- Zero buffering capacity

PIPE : Example



FIFO - Introduction

- Pipes were the widely used form of Inter Process Communication.

- Problem with pipes - Used only between processes that have a common ancestor (ie., a parent-child relationship). however, fixed with the introduction of named pipes or FIFOs

FIFO Basic concepts

- Named pipe works similar to a regular pipe, but have certain differences
- Named pipes exist as a device special file in the file system
- Processes of different ancestry can share data through a named pipe
- When all I/O is done by sharing processes, the named pipe remains in the file system for later use

What is Shared Memory ?

- Shared memory, as the name implies, allows two or more processes, which have the appropriate permissions, to read and/or write to the same area of memory
- The distinguishing features of shared memory as an IPC mechanism are speed, flexibility and ease of use
- Shared memory is commonly implemented when editors or word processors are used in multi user environment
- It can also be used in the multi –player gaming environment

Why go for Shared Memory

- Shared memory is a much faster method of communication than either semaphores or message queues
- Data need not be copied to a kernel buffer and back again. Accessing shared memory takes as much time as a normal memory access
- Using shared memory is quite easy. After a shared memory segment is set up, it is manipulated exactly like any other memory area

Shared Memory (Contd.).

- Shared memory is also much more flexible than other System V IPC mechanisms. Its use is not limited to sending limited-length messages (message queues) or monitoring a counter (semaphores)

- It can be used to share entire data structures between several processes, thus eliminating the overhead of splitting the information into message-sized packets and reassembling it later

Shared Memory (Contd.).

- The steps involved are:
  - Creating shared memory
  - Connecting to the memory & obtaining a pointer to the memory
  - Reading/Writing & changing access mode to the memory
  - shedding from memory
  - delete shared segment

Semaphore

- Concurrent access to shared data may result in data inconsistency
- Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes
- Shared-memory solution to bounded-buffer problem allows at most n – 1 items in buffer at the same time. A solution, where all N buffers are used is not simple
  - Suppose that we modify the producer-consumer code by adding a variable counter, initialized to 0 and incremented each time a new item is added to the buffer

Semaphore (Contd.).

A semaphore is only a synchronization tool

It does not allow process to exchange any data, but control simultaneous access to some shared objects such as :

- shared memory segment
- message queue
- file

Semaphore are integer-valued objects that support two atomic operations – P() (Proberen-wait) and V() (Verhogen-signal)

Summary

In this module, we discussed:

- Fragmentation and Memory protection
- Various inter process communication methods

References

1. Silberschatz,Abraham. Baer,Peter,Galvin and Gagne,Greg(1999):Operating System Principles 7th Edition. Retrieved on April 11. 2011, from http://www.csc.gatech.edu/%7Ecopeland/305500/pdf/ch02_computer_struct_notes.pdf

2. Rusling A David(1996-1999) File translated from TEX by TTH, version 1.0. Retrieved on April 11. 2011, from http://www.tldp.org/LDP/tlk/kernel/processes.html

3. Silberschatz,Galvin and Gagne (2009):Operating Systems Concepts-8th Edition. Retrieved on April 11. 2011, from http://www.cse.sc.edu/~rose/311/ppt/ch08.ppt

4. Silberschatz,Galvin and Gagne(2002) Operating System Concepts-9.1. Retrieved on April 11. 2011, from http://www.massey.ac.nz/~mjjohnso/notes/59305/mod8.html

5. Siberschatz and Galvin (2011) Memory Management . Retrieved on April 11. 2011, from http://www.cse.buffalo.edu/%7Ebina/cse421/fall2002/memoryOct15.ppt

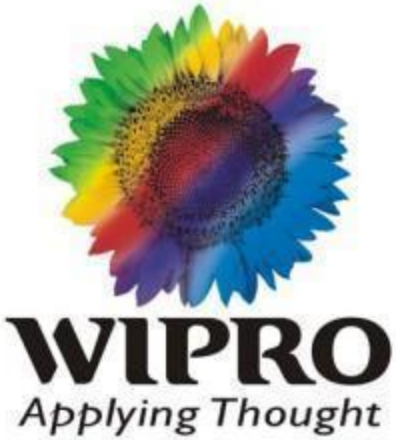
References

6. University of Rochester(2010) Basic Memory Management. Retrieved on April 11. 2011, from http://www.cs.rochester.edu/~sandhya/csc256/lectures/lecture09-11-memory.pdf

7. Unix Process , Retrieved on April 11. 2011, from http://www.cse.buffalo.edu/~bina/cse321/fall2009/UnixProcessesOct15.ppt

8. Brinton,Stephen:Gordon College, Memory Management. Retrieved on April 11. 2011, http://www.cs.gordon.edu/courses/cs312/lectures/mem_management.ppt

9. Watson,Dan (2008) Department of computer Science. Retrieved on April 11. 2011, from http://digital.cs.usu.edu/~watson/cs3100/slides/Processes.ppt



Thank You