Biyani's Think Tank

Concept based notes

Operating Systems

M.Sc. (IT)

Priyamvada Pareek
MCA

Poonam Sharma
MCA, M.Sc.(IT)
Lecturer
Deptt. of Information Technology
Biyani Girls College, Jaipur

Published by :
Think Tanks
Biyani Group of Colleges

Sector-3, Vidhyadhar Nagar,
Jaipur-302 023 (Rajasthan)
Ph : 0141-2338371, 2338591-95 • Fax : 0141-2338007
E-mail : acad@biyanicolleges.org
Website :www.gurukpo.com; www.biyanicolleges.org

First Edition : 2009

Leaser Type Setted by :
Biyani College Printing Department

Preface

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the "Teach Yourself" style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

This book covers basic concepts related to the microbial understandings about diversity, structure, economic aspects, bacterial and viral reproduction etc.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director* (*Acad.*) Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this Endeavour. They played an active role in coordinating the various stages of this Endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

Author

Syllabus

M.Sc. (IT) (Sem.-I)

Operating Systems

Computer Software and Languages : Types of Software (System/Application), Programming Languages [Level (SGL/4GL/3L/2GL/1GL/OGL), Translators (Interpreters/Compilers), OGL (Registers, Switches, Cycles, Interrupt).

Operating System : Functions of Operating System, Characteristics of Operating System (Single/Multi-user, Single/Multi-tasking, Portability), Information Management (File System, Device Drivers, Terminal I/O), Process Management (Process Definition, Control, Interacting Processes, Implementation of Interacting Processes, Threads, Scheduling), Memory Management (Contiguous, Non-contiguous, Paging, Segmentation, Virtual Memory), Deadlocks, Security and Protection, Parallel Processing.

DOS : Booting Sequence, System Files and Commands, Files and Directories, Overview of MS DOS Commands, FDISK and DISK Organization.

Windows : Graphical User Interfaces, Windows 98 Installation, Scan Disk, Task Bars, Toolbars, Settings, Control Panel and all features there in, Files and Folder Management, Windows Explorer, Installing and Running Programs, Backing up Files, Configuring Keyboard and Mouse adding and removing Hardware, Setting up Printers and Fonts, Working with Sound, Graphics and Video, Configuring Windows by using the Accessibility Wizard, Configuring Modem, Connecting to PPP and SLIP Accounts, Creating Windows Peer to Peer Network, Connecting Windows 98PC to Novell Netware and Windows NT Networks, Sharing Drivers and Printers, Compressing Disks and Partitions, Tuning Windows 98 for Maximum Performance, Registering Programs and File Types, Troubleshooting Windows 98.

UNIX : Logging In and Out, Directory, Redirecting Input and Output cat Command vi, Editor, Shell Scripts, Shell and Sub-shell Variables, Meta-characters, Commands – sort, head, tail, split, cut, paste, find, tr, dd, Gripping and Seeding UUPC, UNIX and Networking, Accessing the Internet, UNIX System Administration.

□ □ □

Content

□ □ □

Chapter-1

Computer Software

Q.1. Define Computer Software and its types.

Ans.: **Computer Software** is a general term used to describe a collection of computer programs, procedures and documentation that perform some tasks on a computer system. "Software" is sometimes used in a broader context to mean anything which is not hardware but which is *used* with hardware, such as film, tapes and records.

Practical computer systems divide software systems into three major classes: system software, programming software and application software, although the distinction is arbitrary, and often blurred.

- **System Software** helps run the computer hardware and computer system. It includes operating systems, device drivers, diagnostic tools, servers, windowing systems, utilities and more. The purpose of systems software is to insulate the applications programmer hardware features, and such as accessory devices as communications, printers, readers, displays, keyboards, etc. as much as possible from the details of the particular computer complex being used, especially memory and other
- **Programming Software** usually provides tools to assist a programmer in writing computer programs, and software using different programming languages in a more convenient way. The tools include text editors, compilers, interpreters, linkers, debuggers, and so on. An Integrated development environment (IDE) merges those tools into a software bundle, and a programmer may not need to type multiple commands for compiling, interpreting, debugging, tracing, and etc., because the IDE usually has an advanced *graphical user interface,* or GUI.
- **Application Software** allows end users to accomplish one or more specific (non-computer related) tasks. Typical applications include industrial automation, business software, educational software, medical software, databases, and computer games. Businesses are probably the biggest users of application software, but almost every field of human activity now uses some form of application software.

Q.2 Define the various levels of Programming Languages.

Ans.: Levels of Programming Languages : The different levels of programming languages are defined as follows :

(i) **1GL or First-Generation language :** It is machine language or the level of instructions and data that the processor is actually given to work on

(ii) **2GL or Second-Generation Language :** It is also called "assembly" language. An assembler is required to converts the assembly language statements into machine language.

(iii) **3GL or Third-Generation Language :** It is a "high-level" programming language, such as PL/I, C, or Java. Java language. A compiler is required to converts the statements of a specific high-level programming language into machine language.

(iv) **4GL or Fourth-Generation Language :** It is designed to be closer to natural language than a 3GL language. Languages for accessing databases are often described as 4GLs.

(v) **5GL or Fifth-Generation Language :** It is language that uses a visual or graphical development interface to create source language that is usually compiled with a 3GL or 4GL language compiler.

(v) **0GL or Hardware Level :** This is the level at which we want to study the actual computer architecture.

Q.3. Define Interpreter and Compiler and give their differences.

Ans.: Definition of Compiler : Compiler is a program that translates a computer program written on one computer language to another computer language. A "compiler" is primarily used for programs that translate source code from a high level language to a lower level language (e.g., assembly language or machine language). A program that translates from a low level language to a higher level one is a decompiler. A compiler for a relatively simple language written by one person might be a single, monolithic, piece of software. When the source language is large and complex, and high quality output is required the design may be split into a number of relatively independent phases, or passes. Having separate phase's means development can be parceled up into small parts and given to different people. It also becomes much easier to replace a single phase by an improved one, or to insert new phases later.

Definition of Interpreter : Interpreter is a program that translates an instruction into a machine language and executes it before proceeding to the next instruction..... A high-level programming language translator that translates and

runs the program at the same time. It translates one program statement into machine language, executes it, and then proceeds to the next statement. This differs from regular executable programs that are presented to the computer as binary-coded instructions. Interpreted programs remain in the source language the programmer wrote in, which is human readable text. Interpreted programs run slower than their compiler counterparts. Whereas the compiler translates the entire program before it is run, interpreters translate a line at a time while the program is being run. However, it is very convenient to write an interpreted program, since a single line of code can be tested interactively.

Difference between Compiler and Interpreter : Interpreter translate the program line by line and compiler translate the entire program interpreter requires less memory and compiler requires more memory.

E.g. of Compiler – **Java**

E.g. of Interpreter – **PHP**

Q.4 Write the short note on Registers, Switches, Cycles and Interrupts.

Ans.: Registers : In computer architecture, a processor register is a small amount of storage available on the CPU whose contents can be accessed more quickly than storage available elsewhere. Processor registers are at the top of the memory hierarchy, and provide the fastest way for a CPU to access data.

Switches : They are used to separate bus and Register to make it possible the data transfer from registers to the bus or vice versa . They are also acts as latches: - data once deposited in a register does not get mixed up with the signals or the data in the bus.

Cycles : An Instruction cycle can be defined as the sum of an instruction fetch time and the instruction execution time.

Interrupts : Interrupt is the ability of the hardware to stop the currently running program and turn the system's attention to something else.

Q.5. Write short notes on following :

Ans. Registers : Registers are fast memory, almost always connected to circuitry that allows various arithmetic, logical, control, and other manipulations, as well as possibly setting internal flags.

Most early computers had only one data register that could be used for arithmetic and logic instructions. Often there would be additional special purpose registers set aside either for temporary fast internal storage or assigned to logic circuits to implement certain instructions. Some early computers had one

or two address registers that pointed to a memory location for memory accesses (a pair of address registers typically would act as source and destination pointers for memory operations). Computers soon had multiple data registers, address registers, and sometimes other special purpose registers. Some computers have general purpose registers that can be used for both data and address operations. Every digital computer using a von Neumann architecture has a register (called the program counter) that points to the next executable instruction. Many computers have additional control registers for implementing various control capabilities. Often some or all of the internal flags are combined into a flag or status register.

There are different types of registers. These are :

- **Accumulators :** Accumulators are registers that can be used for arithmetic, logical, shift, rotate, or other similar operations. The first computers typically only had one accumulator. Many times there were related special purpose registers that contained the source data for an accumulator. Accumulators were replaced with data registers and general purpose registers. Accumulators reappeared in the first microprocessors.
- **Address Registers :** Address registers store the addresses of specific memory locations. Often many integer and logic operations can be performed on address registers directly (to allow for computation of addresses).Sometimes the contents of address register(s) are combined with other special purpose registers to compute the actual physical address. This allows for the hardware implementation of dynamic memory pages, virtual memory, and protected memory.
- **General Purpose Registers :** General purpose registers can be used as either data or address registers.
- **Constant Registers :** Constant registers are special read-only registers that store a constant. Attempts to write to a constant register are illegal or ignored. In some RISC processors, constant registers are used to store commonly used values (such as zero, one, or negative one) — for example, a constant register containing zero can be used in register to register data moves, providing the equivalent of a clear instruction without adding one to the instruction set. Constant registers are also often used in floating point units to provide such value as pi or e with additional hidden bits for greater accuracy in computations.
- **Floating Point Registers :** Floating point registers are special registers set aside for floating point math.
- **Index Registers :** Index registers are used to provide more flexibility in addressing modes, allowing the programmer to create a memory address by combining the contents of an address register with the contents of an

index register (with displacements, increments, decrements, and other options). In some processors, there are specific index registers (or just one index register) that can only be used only for that purpose. In some processors, any data register, address register, or general register (or some combination of the three) can be used as an index register.

- **Base Registers :** Base registers or **segment registers** are used to segment memory. Effective addresses are computed by adding the contents of the base or segment register to the rest of the effective address computation. In some processors, any register can serve as a base register. In some processors, there are specific base or segment registers (one or more) that can only be used for that purpose. In some processors with multiple base or segment registers, each base or segment register is used for different kinds of memory accesses (such as a segment register for data accesses and a different segment register for program accesses).
- **Control Registers :** Control registers control some aspect of processor operation. The most universal control register is the program counter.

Program Counter : Almost every digital computer ever made uses a **program counter**. The program counter points to the memory location that stores the next executable instruction. Branching is implemented by making changes to the program counter. Some processor designs allow software to directly change the program counter, but usually software only indirectly changes the program counter (for example, a JUMP instruction will insert the operand into the program counter). An assembler has a **location counter**, which is an internal pointer to the address (first byte) of the next location in storage (for instructions, data areas, constants, etc.) while the source code is being converted into object code.

Processor Flag : Processor flags store information about specific processor functions. The processor flags are usually kept in a **flag register** or a general **status register**. This can include **result flags** that record the results of certain kinds of testing, information about data that is moved, certain kinds of information about the results of compations or transformations, and information about some processor states. Closely related and often stored in the same processor word or status register (although often in a privileged portion) are **control flags** that control processor actions or processor states or the actions of certain instructions.

Stack Pointer : Stack pointers are used to implement a processor stack in memory. In many processors, address registers can be used as generic data stack pointers and queue pointers. A specific stack pointer or address register may be hardwired for certain instructions. The most common use is to store return addresses, processor state information, and temporary variables for subroutines.

Subroutine Return Pointer : One RISC processors include a special **subroutine return pointer** rather than using a stack in memory. The return address for subroutine calls is stored in this register rather than in memory. More than one level of subroutine calls requires storing and saving the contents of this register to and from memory

□ □ □

Chapter-2

Operating System

Q.1. What is Operating System? Give its examples.

Ans.: An **operating system** (commonly abbreviated *OS* and *O/S*) is the software component of a computer system that is responsible for the management and coordination of activities and the sharing of the resources of the computer. The operating system acts as a host for application programs that are run on the machine. As a host, one of the purposes of an operating system is to handle the details of the operation of the hardware. This relieves application programs from having to manage these details and makes it easier to write applications. Almost all computers, including hand-held computers, desktop computers, supercomputers, and even modern video game consoles, use an operating system of some type. Some of the oldest models may however use an embedded OS, that may be contained on a compact disk or other storage device.

Microsoft Windows, Mac OS X, Linux and Solaris are examples of operating system.

Q. 2. Write short note on services provide by operating systems.

Ans. Following are the five services provided by operating systems to the convenience of the users :

(1) Program Execution : The purpose of a computer systems is to allow the user to execute programs. So the operating systems provides an environment where the user can conveniently run programs. The user does not have to worry about the memory allocation or multitasking or anything. These things are taken care of by the operating systems.

Running a program involves the allocating and deallocating memory, CPU scheduling in case of multiprocess. These functions cannot be given to the user-level programs. So user-level programs cannot help the user to run programs independently without the help from operating systems.

(2) **I/O Operations :** Each program requires an input and produces output. This involves the use of I/O. The operating systems hides the user the details of underlying hardware for the I/O. All the user sees is that the I/O has been performed without any details. So the operating systems by providing I/O makes it convenient for the users to run programs.

For efficiently and protection users cannot control I/O so this service cannot be provided by user-level programs.

(3) **File System Manipulation :** The output of a program may need to be written into new files or input taken from some files. The operating systems provides this service. The user does not have to worry about secondary storage management. User gives a command for reading or writing to a file and sees his her task accomplished. Thus operating systems makes it easier for user programs to accomplished their task.

This service involves secondary storage management. The speed of I/O that depends on secondary storage management is critical to the speed of many programs and hence I think it is best relegated to the operating systems to manage it than giving individual users the control of it. It is not difficult for the user-level programs to provide these services but for above mentioned reasons it is best if this service s left with operating system.

(4) **Communications :** There are instances where processes need to communicate with each other to exchange information. It may be between processes running on the same computer or running on the different computers. By providing this service the operating system relieves the user of the worry of passing messages between processes. In case where the messages need to be passed to processes on the other computers through a network it can be done by the user programs. The user program may be customized to the specifics of the hardware through which the message transits and provides the service interface to the operating system.

(5) **Error Detection :** An error is one part of the system may cause malfunctioning of the complete system. To avoid such a situation the operating system constantly monitors the system for detecting the errors. This relieves the user of the worry of errors propagating to various part of the system and causing malfunctioning.

This service cannot allowed to be handled by user programs because it involves monitoring and in cases altering area of memory or deallocation of memory for a faulty process. Or may be relinquishing the CPU of a process that goes into an infinite loop. These tasks are too critical to be handed over to the user programs. A user program if given these privileges can interfere with the correct (normal) operation of the operating systems.

Q.3. What are the functions of Operating System?

Ans.: Functions of Operating System : Main functions of Operating systems are :

(1) Processor Management

(2) Memory Management

(3) Device I/O Management

(4) File Management

(5) Disk Management

Q.4. Explain characteristics of Operating System.

Ans.: Multi-user : Multi-user is a term that defines an operating system or application software that allows concurrent access by multiple users of a computer. Time-sharing systems are multi-user systems. Most batch processing systems for mainframe computers may also be considered "multi-user", to avoid leaving the CPU idle while it waits for I/O operations to complete. However, the term "multitasking" is more common in this context.

An example is a Unix server where multiple remote users have access (such as via Secure Shell) to the Unix shell prompt at the same time. Another example uses multiple X Window sessions spread across multiple terminals powered by a single machine - this is an example of the use of thin client.

Management systems are implicitly designed to be used by multiple users, typically one system administrator or more and an end-user community.

Multitasking : Multitasking is a method by which multiple tasks, also known as processes, share common processing resources such as a CPU. In the case of a computer with a single CPU, only one task is said to be *running* at any point in time, meaning that the CPU is actively executing instructions for that task. Multitasking solves the problem by scheduling which task may be the one running at any given time, and when another waiting task gets a turn. The act of reassigning a CPU from one task to another one is called a context switch. When context switches occur frequently enough the illusion of parallelism is achieved. Even on computers with more than one CPU (called multiprocessor machines),

multitasking allows many more tasks to be run than there are CPUs.

Multiprograming : Multiprogramming is a rudimentary form of parallel processing in which several programs are run at the same time on a uniprocessor. Since there is only one processor , there can be no true simultaneous execution of different programs. Instead, the operating system executes part of one program, then part of another, and so on. To the user it appears that all programs are executing at the same time.

If the machine has the capability of causing an interrupt after a specified time interval, then the operating system will execute each program for a given length of time, regain control, and then execute another program for a given length of time, and so on. In the absence of this mechanism, the operating system has no choice but to begin to execute a program with the expectation, but not the certainty, that the program will eventually return control to the operating system.

Time Sharing : Time-sharing refers to sharing a computing resource among many users by multitasking.

Because early mainframes and minicomputers were extremely expensive, it was rarely possible to allow a single user exclusive access to the machine for interactive use. But because computers in interactive use often spend much of their time idly waiting for user input, it was suggested that multiple users could share a machine by allocating one user's idle time to service other users. Similarly, small slices of time spent waiting for disk, tape, or network input could be granted to other users.

Portability : Portability is a characteristic attributed to a computer program if it can be used in an operating systems other than the one in which it was created without requiring major rework. *Porting* is the task of doing any work necessary to make the computer program run in the new environment. In general, programs that adhere to *standard* program interfaces such as the X/Open Unix 95 standard C language interface are portable. Ideally, such a program needs only to be compiled for the operating system to which it is being ported.

Portability has usually meant some work when moving an application program to another operating system. Recently, the Java programming language and runtime environment has made it possible to have programs that run on any operating system that supports the Java standard (from Sun Microsystems) without any porting work. Java applets in the form of precompiled bytecode can be sent from a server program in one operating system to a client program (your

□ □ □

Chapter-3

Information Management

Q.1. What is File System. Explain different file attributes.

Ans.: In computing, a **file system** (often also written as **filesystem**) is a method for storing and organizing computer files and the data they contain to make it easy to find and access them. File systems may use a data storage device such as a hard disk or CD-ROM and involve maintaining the physical location of the files, they might provide access to data on a file server by acting as clients for a network protocol (e.g., NFS, SMB, or 9P clients), or they may be virtual and exist only as an access method for virtual data (e.g., procfs).

Different File Attributes : A file's attributes vary from one operating system to another but typically consist of these :

- **Name :** The symbolic file name is the only information kept in human readable form.
- **Identifier :** This unique tag, usually a number , identifies the file within the file system; it is non-human-readable name for the file.
- **Type :** This information is needed for systems that support different types of files.
- **Location :** This information is a pointer to a device and to the location of the file on that device.
- **Size :** The current size of the file and possibly the maximum allowed size are included in this attribute.
- **Protection :** Access-control information determines who can do reading, writing, executing and so on.
- **Time ,Date, and user identification :** This information may be kept for creation, last modification, and last use.

Q.2. What is Device Driver? Explain it's purpose.

Ans.: A **device driver**, or **software driver**, is a computer program allowing higher-level computer programs to interact with a device.

A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware is connected. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke

routines in the original calling program. Drivers are hardware-dependent and operating-system-specific. They usually provide the interrupt handling required for any necessary asynchronous time-dependent hardware interface.

Purpose : A device driver simplifies programming by acting as a translator between a device and the applications or operating systems that use it. The higher-level code can be written independently of whatever specific hardware device it may control. Every version of a device, such as a printer, requires its own specialized commands. In contrast, most applications access devices(such as sending a file to a printer)by using high-level, generic commands, such as PRINTLN. The driver accepts these generic statements and converts them into the low-level commands required by the device.

Q.3. What is the use of Device Controller?

Ans.: Device Controller : Each device uses a device controller to connect devices to computer data and address bus. The controller provides a set of physical components that CPU instruction can manipulate to perform I/O operations. Generally Operating system is deals with the controller and not the device

Q.4 Explain various Disk Scheduling Algorithms in brief.

OR

Explain various I/O Optimization Techniques.

Ans.: I/O Optimization Techniques : The various disk scheduling algorithm in order to minimize total service time and wait time are as follows :

(1) **First-In-First-Served :** The simplest form of scheduling is first-in-first-out (FIFS) scheduling, which processes items from the queue in sequential order.

(2) **Shortest Seek Time First :** The SSTF policy is to select the disk I/O request that requires the least movement of the disk arm from its current position.

(3) **Scan :** In this scheme, the read/write keeps moving from one end to another end and when it reaches to the other end, the direction of head movement is reserved and servicing continues.

(4) **C-Scan :** The C-SCAN policy restricts scanning to one direction only. Thus, when the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again.

Q.5 Define the strategies of Contiguous, Linked and Indexed Allocation in File System.

Ans.: (i) **Contiguous Allocation :** The contiguous allocation method requires each file to occupy a set of contiguous blocks on the disk. Disk addresses define a linear ordering on the disk. The directory entry for each file indicates the address of the starting block and the length of the area allocate for this file. Accessing a file that has been allocated contiguously is easy.

(ii) **Linked Allocation :** With link allocation, each file is a linked list disk blocks; the disk blocks may be scattered anywhere on the disk and any free block on the free-space list can be used to satisfy a request ,there is no need to declare the size of a file when that file is created. A file can continue to grow as long as there are free blocks.

(iii) **Indexed Allocation :** The problem with Linked allocation is that, the pointers to the blocks are scattered with the blocks themselves all over the disk and need to be retrieved in order, while in Indexed allocation all the pointers bring together into one location: the index block. These types of allocation supports direct access.

Q.6 What is the use of Directories?

Ans.: **Directories :** The Directories are treated as files which keep track of all other files. The directory contain information about the files such as location and owner of the file etc. the directory is itself a file, owned by the operating system and accessible by various file management routines.

Q.7 Explain the various types of Directory Systems.

Ans.: **Type of Directories :** Directories can be organized in following ways :

(1) **Single-Level Directory :** In a single-level directory system, all the files are placed in one directory. This is very common on single-user OS's. Even with a single-user, as the number of files increases, it becomes difficult to remember the names of all the files in order to create only files with unique names.

(2) **Two-Level Directory :** In the two-level directory system, the system maintains a master block that has one entry for each user. This master block contains the addresses of the directory of the users.

(3) **Tree-Structured Directory :** In the tree-structured directory, the directory themselves are files. This leads to the possibility of having sub-directories that can contain files and sub-subdirectories.

(4) **Acyclic-Graph Directory :** The acyclic directory structure is an extension of the tree-structured directory structure. In the acyclic structure a

directory or file under directory can be owned by several users. Thus an acyclic graph structure allows directories to have shared subdirectories and files.

(5) **General Graph Directory :** One problem with using an acyclic graph structure is ensuring that there is no cycle. However to avoid traversing shared sectors of an acyclic graph twice, we can allow cycles to exist. If cycles are allowed to exist in the directory, we generally, want to avoid searching any component twice, for reason of correctness and performance.

□ □ □

Chapter-4

Process Management

Q.1. What is Process? Define different Process states.

Ans.: The term "process" was first used by the designers of the MULTICS in 1960's. Since then, the term process, used somewhat interchangeably with 'task' or 'job'. The process has been given many definitions for instance

- A program in Execution.
- An asynchronous activity.
- The 'animated sprit' of a procedure in execution.
- The entity to which processors are assigned.
- The 'dispatch able' unit.

and many more definitions have given. As we can see from above that there is no universally agreed upon definition, but the definition "*Program in Execution*" seem to be most frequently used. And this is a concept are will use in the present study of operating systems.

Now that we agreed upon the definition of process, the question is what is the relation between process and program. It is same beast with different name or when this beast is sleeping (not executing) it is called program and when it is executing becomes process. Well, to be very precise. Process is not the same as program. In the following discussion we point out some of the difference between process and program. As we have mentioned earlier.

Process is not the same as program. A process is more than a program code. A process is an 'active' entity as oppose to program which consider to be a 'passive' entity. As we all know that a program is an algorithm expressed in some suitable

notation, (e.g., programming language). Being a passive, a program is only a part of process.

A process is the unit of work in a system.

The process state consist of everything necessary to resume the process execution if it is somehow put aside temporarily. The process state consists of at least following :

- Code for the program.
- Program's static data.
- Program's dynamic data.
- Program's procedure call stack.
- Contents of general purpose registers.
- Contents of program counter (PC)
- Contents of program status word (PSW).
- Operating Systems resource in use.

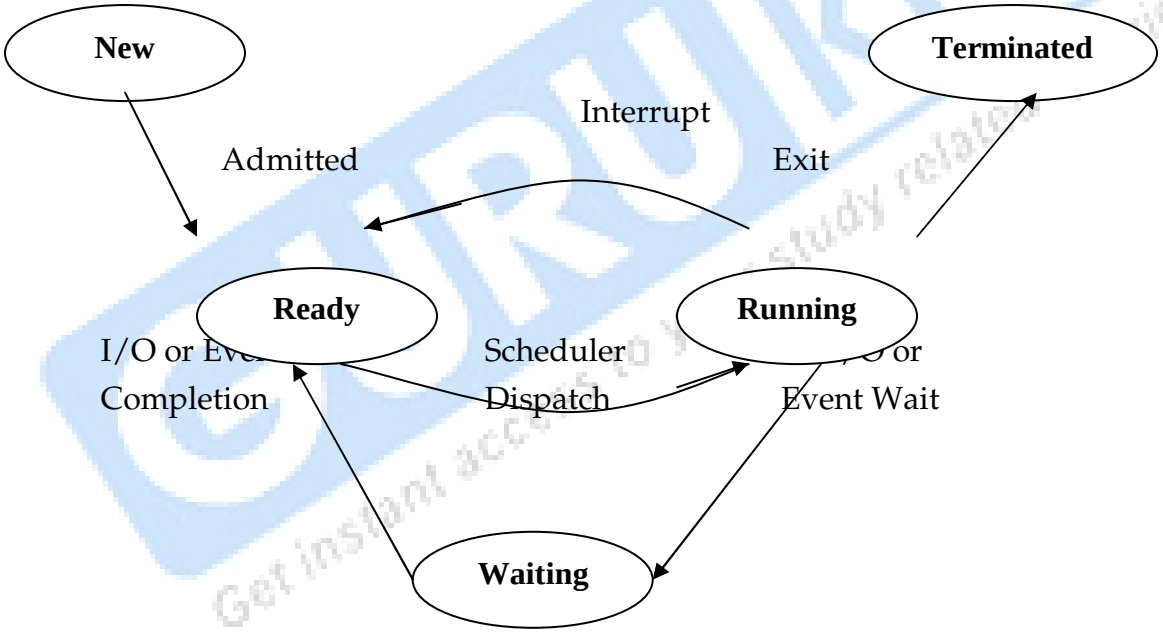


Diagram of Process State

A process goes through a series of discrete process states :

- **New State :** The process being created.
- **Running State :** A process is said to be running if it has the CPU, that is, process actually using the CPU at that particular instant.
- **Blocked or Waiting State :** A process is said to be blocked if it is waiting for some event to happen such that as an I/O completion before it can

proceed. Note that a process is unable to run until some external event happens.

- **Ready State :** A process is said to be ready if it use a CPU if one were available. A ready state process is runable but temporarily stopped running to let another process run.
- **Terminated State :** The process has finished execution.

Q.3 Explain the Process Control Block (PCB) with diagram.

Ans.: Process Control Block (PCB) : To control the various processes the Operating System maintains a table called the process table. The process table contains one entry per process. These entries are referred to as "Process Control Block". It contains many piece of information related with a specific process including the:- Process Number, Process State, Program Counter, CPU Registers, CPU Scheduling Information, Memory management Information, Accounting Information, and I/O Status Information.

Q.4 What do you mean by Interacting Processes?

Ans.: Interacting Processes : In single CPU systems, by the multiprogramming several processes may permit to be executing concurrently. The processes of n application interact with one another to co-ordinate their activities or to share some data. These processes are called interacting processes.

Q.5. What are Threads?

Ans.: *A thread is a single sequence stream within in a process*. Because threads have some of the properties of processes, they are sometimes called *lightweight processes*. In a process, threads allow multiple executions of streams. In many respect, threads are popular way to improve application through parallelism. The CPU switches rapidly back and forth among the threads giving illusion that the threads are running in parallel.

Q.6. What is the similarities and difference between a Process and Thread?

Ans.: Similarities :

- Like processes threads share CPU and only one thread active (running) at a time.
- Like processes, threads within processes, threads within processes execute sequentially.
- Like processes, thread can create children.
- And like process, if one thread is blocked, another thread can run.

Differences :

- Unlike processes, threads are not independent of one another.
- Unlike processes, all threads can access every address in the task.
- Unlike processes, threads are design to assist one other. Note that processes might or might not assist one another because processes may originate from different users.

Q.7. What is Process Scheduling? Define its objectives.

Ans.: The assignment of physical processors to processes allows processors to accomplish work. The problem of determining when processors should be assigned and to which processes is called processor scheduling or CPU scheduling.

When more than one process is run able, the operating system must decide which one first. The part of the operating system concerned with this decision is called the scheduler, and algorithm it uses is called the scheduling algorithm.

Goals of Scheduling (Objectives) : In this section we try to answer following question: What the scheduler try to achieve?

Many objectives must be considered in the design of a scheduling discipline. In particular, a scheduler should consider fairness, efficiency, response time, turnaround time, throughput, etc., Some of these goals depends on the system one is using for example batch system, interactive system or real-time system, etc. but there are also some goals that are desirable in all systems.

General Goals :

Fairness : Fairness is important under all circumstances. A scheduler makes sure that each process gets its fair share of the CPU and no process can suffer indefinite postponement. Note that giving equivalent or equal time is not fair. Think of *safety control* and *payroll* at a nuclear plant.

Policy Enforcement : The scheduler has to make sure that system's policy is enforced. For example, if the local policy is safety then the *safety control processes* must be able to run whenever they want to, even if it means delay in *payroll processes*.

Efficiency : Scheduler should keep the system (or in particular CPU) busy cent percent of the time when possible. If the CPU and all the Input/Output devices can be kept running all the time, more work gets done per second than if some components are idle.

Response Time : A scheduler should minimize the response time for interactive user.

Turnaround : A scheduler should minimize the time batch users must wait for an output.

Throughput : A scheduler should maximize the number of jobs processed per unit time.

A little thought will show that some of these goals are contradictory. It can be shown that any scheduling algorithm that favors some class of jobs hurts another class of jobs. The amount of CPU time available is finite, after all.

Q.8. What are the different types of Scheduling Queues?

Ans.: The **Scheduling Queues** in the systems are :

(1) **Job Queue :** As processes enters in the system they are put into a job queue.

(2) **Ready Queue :** The processes that are residing in the main memory and are ready and waiting to execute are kept in the ready queue.

(3) **Device Queue :** This queue contain a list processes waiting for I/O devices.

Q.9. Write about the different types of Schedulers.

Ans.: Types of Schedulers : There are basically three types of schedulers :

(1) **Long Term Scheduler :** This scheduler determines which job will be submitted for immediate processing. It selects from the job pool and loads them into memory.

(2) **Short Term Scheduler :** It is also called a CPU scheduler. It allocates processes belonging to ready queue to CPU for immediate processing.

(3) **Medium Term Scheduler :** It is also called as memory scheduler. During the execution processes are swapped- out and swapped -in by the Medium term scheduler.

Q.10. Differentiate between Non-preemptive and Preemptive Scheduling Mechanism.

Ans.: Non-preemptive Scheduling Mechanism : A Non-Preemptive scheduling algorithm selects a process to run and just let it run, until it blocks or until it voluntarily release the CPU.

Preemptive Scheduling Mechanism : In this category, suspension or preemption is allowed based on priority.

Q.11. Explain various Non- preemptive Scheduling Mechanisms.

Ans.: Non-preemptive Scheduling Mechanisms : Non- preemptive scheduling mechanisms are :

(1) **First Come First Served (FCFS) Scheduling :** With this scheme, the process that requests the CPU first is allocated the CPU first.

(2) **Shortest-Job First (SJF) Scheduling :** In this scheme, job requiring the minimal CPU time is selected first for CPU allocation.

(3) **Priority Scheduling :** In this scheme a priority is associated with each process and the CPU is allocated to the process with the highest priority.

(4) **Deadline Scheduling :** With this scheduling algorithm the process with earliest deadline is given the highest priority for execution.

Q.12. Explain various Preemptive Scheduling Mechanisms.

Ans.: Preemptive Scheduling Mechanisms : Preemptive scheduling mechanisms are :

(1) **Round-Robin Scheduling :** In This algorithm a small time slice is assigned to each process. The CPU scheduler goes around the ready queue, and allocating the CPU to each process for a time interval of one time quantum.

(2) **Two Queue Scheduling :** In this approach, the processes are classified into two different groups. One queue is allocated to CPU bound processes and other is allocated to I/O bound processes.

(3) **Multilevel Queue Scheduling :** A multilevel queue scheduling algorithm partition the ready queue in to separate queues and each queues has its own scheduling algorithms.

□ □ □

Chapter-5

Memory Management

Q.1. What is Memory Management? Explain its goals.

Ans.: **Memory Management** is the act of managing computer memory. In its simpler forms, this involves providing ways to allocate portions of memory to programs at their request, and freeing it for reuse when no longer needed. The management of main memory is critical to the computer system.

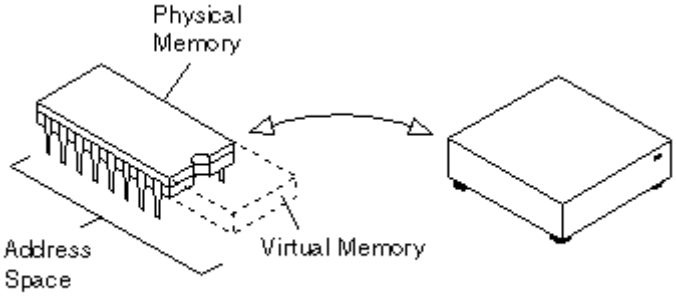
Goals of Memory Management : Historically, a number of different memory management techniques have been used, and improved upon, in the operating system. The principal goals of the operating system's memory management are :

- To provide memory space to enable several processes to be executed at the same time
- To provide a satisfactory level of performance for the system users
- To protect each program's resources
- To share (if desired) memory space between processes
- To make the addressing of memory space as transparent as possible for the programmer.

Q.2. What is Virtual Memory?

Ans.: An imaginary memory area supported by some operating systems (for example, Windows but not DOS) in conjunction with the hardware. You can think of virtual memory as an alternate set of memory addresses. Programs use these *virtual addresses* rather than real addresses to store instructions and data. When the program is actually executed, the virtual addresses are converted into real memory addresses.

The purpose of virtual memory is to enlarge the *address space*, the set of addresses a program can utilize. For example, virtual memory might



contain twice as many addresses as main memory. A program using all of virtual memory, therefore, would not be able to fit in main memory all at once. Nevertheless, the computer could execute such a program by copying into main memory those portions of the program needed at any given point during execution.

To facilitate copying virtual memory into real memory, the operating system divides virtual memory into *pages*, each of which contains a fixed number of addresses. Each page is stored on a disk until it is needed. When the page is needed, the operating system copies it from disk to main memory, translating the virtual addresses into real addresses.

The process of translating virtual addresses into real addresses is called *mapping.* The copying of virtual pages from disk to main memory is known as *paging* or *swapping*.

Q.3. What is Paging?

Ans.: When a program is selected for execution, the system brings it into virtual storage, divides it into pages of four kilobytes, transfers the pages into central storage for execution. To the programmer, the entire program appears to occupy contiguous space in storage at all times. Actually, not all pages of a program are necessarily in central storage, and the pages that are in central storage do not necessarily occupy contiguous space.

The pieces of a program executing in virtual storage must be moved between real and auxiliary storage. To allow this, z/OS® manages storage in units, or **blocks**, of four kilobytes. The following blocks are defined :

- A block of central storage is a **frame**.
- A block of virtual storage is a **page**.
- A block of auxiliary storage is a **slot**.

A page, a frame, and a slot are all the same size: Four kilobytes. An active virtual storage page resides in a central storage frame. A virtual storage page that becomes inactive resides in an auxiliary storage slot (in a paging data set). Figure 1 shows the relationship of pages, frames, and slots.

In Figure 1, z/OS is performing paging for a program running in virtual storage. The lettered boxes represent parts of the program. In this simplified view, program parts A, E, F, and H are active and running in central storage frames, while parts B, C, D, and G are inactive and have been moved to auxiliary storage slots. All of the program parts, however, reside in virtual storage and have virtual storage addresses.

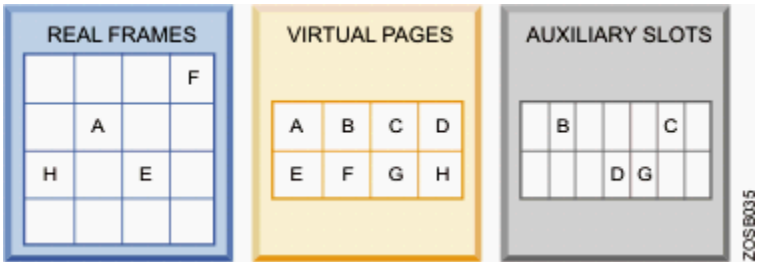


Figure : Frames, Pages, and Slots

Q.4. What are Preemptable and Nonpreemptable Resources.

Ans.: Resources come in two flavors: preemptable and nonpreemptable. A preemptable resource is one that can be taken away from the process with no ill effects. Memory is an example of a preemptable resource. On the other hand, a nonpreemptable resource is one that cannot be taken away from process (without causing ill effect). For example, *CD* resources are not preemptable at an arbitrary moment.

Reallocating resources can resolve deadlocks that involve preemptable resources. Deadlocks that involve nonpreemptable resources are difficult to deal with.

Q.5 What is difference between Logical and Physical Addresses?

Ans.: **Logical versus Physical Address Space :** An address generated by the CPU is commonly referred to as a logical address, whereas an address seen by the memory unit is commonly referred to as a physical address. The set of all logical addresses generated by a program is referred to as a logical address space; the set of all physical addresses corresponding to these logical addresses is referred to as a physical address space. The user program deals with logical addresses. The memory-mapping hardware converts logical addresses into physical addressed.

Q.6 What is the difference between Contiguous and Non-contiguous Allocation Techniques.

Ans.: **Contiguous and Non-contiguous Allocation Techniques :** Contiguous memory management scheme expect the program to be loaded in the contiguous memory locations. While non-contiguous system removes this restriction. They allow the program to be divided into different chunks and loaded at the different portion of memory.

Q.7 Explain the difference between Internal and External Fragmentation.

Ans.: External and Internal Fragmentation : As processes are loaded and removed from memory, the free memory space is broken into little pieces. External fragmentation exists when enough to the memory space exists to satisfy a request, but it is not contiguous; storage is fragmented into a large number of small holes.

While internal fragmentation means memory that is internal to partition, but is not being used.

Q.8 What do you mean by Swapping?

Ans.: Swapping : In case of Round robin CPU-scheduling or priority-based scheduling it is required to a process, that it can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution. This technique is called swapping. A process is swapped out will be swapped back into the same memory space that it occupies previously. Swapping requires a backing store. The backing store is commonly a fast disk.

Q.9 Explain the Segmentation technique in brief.

Ans.: Segmentation : Segmentation is a memory management scheme. It assumes that logical address is a collection of segment. Each segment has a name and a length. The address specifies both the segment number and the offset thus the logical address consists of two parts :

<Segment-number, offset>

Q.10 What do you understand by Demand Paging?

Ans.: Demand Paging : A demand paging is similar to a paging system with swapping. When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory. When a process is to be swapped in, the pager guesses which pages will be used before the process is swapped out again Instead of swapping in a whole process; the pager brings only those necessary pages into memory. Thus, it avoids reading into memory pages that will not be used in anyway, decreasing the swap time and the amount of physical memory needed.

Q.11 What are the various methods of Page Replacement?

Ans.: Page Replacement Algorithm : There are many different page replacement algorithms. Some are :

(1) **FIFO Algorithm :** The simplest page-replacement algorithm is a FIFO algorithm. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen.

(2) **Optimal Algorithm :** An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. It is simply replace the page that will not be used for the longest period of time.

(3) **LRU Algorithm :** The LRU replaces the page that has not been used for the longest period of time. LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses that page that has not been used for the longest period of time.

□ □ □