COURSE CODE: CSC 322

COURSE TITLE: COMPUTER OPERATING SYSTEM I

NUMBER OF UNITS: 3 Units

COURSE DURATION: Three hours per week

COURSE DETAILS:

Course Coordinator: Adebayo Felix Adekoya B.Sc., M.Sc., Ph.D.

Email: lanlenge@yahoo.com

Office Location: **Room B321, COLNAS.**

COURSE CONTENT:

Historical development of operating systems and computer hardware, operating systems types, necessary hardware requirements and operating characteristics, batch versus time sharing, multi-dislocation, interrupts and interrupts handling, device, handlers, memory, organization visual memory and visual machine, remote job entry, pipeline processing, command languages, more about DOS/Vs/JOL in respect of maintenance of libraries and job organization

COURSE REQUIREMENTS:

This is a compulsory course for all students in the Departments of Computer Science, Electrical and Electronics Engineering, and Mathematics. However, students in the Department of Statistics do offer the course as an elective. In view of this, students are expected to participate in all the course activities and have minimum of 75% attendance to be able to write the final examination.

READING LIST:

Lister, A. M. *Fundamentals of Operating Systems*.3rd Edition. New York. Springer-Verlag, 1984. . ISBN 0-387-91251-7

1. Ritchie, C. *Operating Systems: Incorporating UNIX and MS-DOS*. 2nd Edition. London. DP Publications. 1996.
2. Hansen, P. B. *Operating Systems Principles*. New Delhi, Prentice-Hall of India private Limited. 1986. ISBN 0-87692-135-7.
3. Wolf, W. *Computers as Components: Principles of Embedded Computing System Design*. San Diego. Morgan Kaufmann Publishers and Academy Press. 2001. ISBN 1-55860-541-X
4. Tanenbaum, A. S. *Structured Computer Organization*. Pearson Prentice Hall, New Jersey. 2006. ISBN 0-13-148521-0.
5. Campbell, J. G. *Lecture Notes on Computer Systems and Operating Systems*. Department of Computing, Letterkenny Institute of Technology, Co. Donegal, Ireland. Report No: jc/04/0001/ 2004. http://homepage.ntlworld.com/jg.campbell/lyitcsys/
6. Barton Miller. *Operating Systems: Lecture Notes for CS537*. Department of Computer Sciences, University of Wisconsin, Madison. Last modified: Wed Mar 5 17:08:08 CST 2008. Copyright Â© 2001, 2002, 2008.
7. Martin C. Rinard *Operating Systems Lecture Notes*.
8. Brookshear, J. G. *Computer Science: An Overview*. Boston. Pearson Education. 2009. ISBN: 10 0-321-54428-5

LECTURE NOTES

1. History and evolution of operating systems

Computer hardware provides us with the means of processing and storing information. System software are therefore designed to suit the needs of the hardware, and facilitates the development and running of applications. The most significant item of system software is the operating systems, which is present in all computers, except for a few very specialized applications.

The role of the operating systems is to complement the hardware by providing a layer of services which manage the resources of the hardware and permit the users to drive the system.

The evolution of operating systems has been driven by technological advances and by the demands and expectation of the users. An examination of this evolutionary process would help us to understand the workings of modern systems and to better appreciate their essential

principles. It also reveals the important fact that many modern techniques are of a much greater age than generally appreciated.

1. Program Loading-Bootstrapping

   The first step toward improving and simplifying computer use was to address the problem of loading the program. The loader is built into the computer facility whereby one startup, the computer automatically read a primitive loader program.

   This basic loader is the executed and read in a larger, more extensive loader program, which could then load any use program.

2. Early Printers And Terminals

   The improvement of the output was also a concern, and this was done by using a simple character terminal to display textual results. The keyboard was used a control console enabling the operator to communicate with the system and therefore becomes the principal user interface (PUI) device for future operating systems.

3. Punched Card. (1950s-1980s)

   Though it predates this period, it has been used as an input medium in electro-mechanized calculating equipment, the concept of a job arose; a deck of cards containing a program followed by the input data for the program. Cards were also employed as output media, and hence could in effect serve as off line storage.

4. Speed And Cost

   In the 1950s, computers were very expensive is relating to their through put measures the early pre-occupation was therefore to get as much use out of them as possible and as quickly. However, as the hardware steadily improved, the execution time of programs fell hence.

   a. The set-up true (the time between jobs pent loading the next programs and data) become disproportionate to the run time of the job

   b. The I/O devices were seen to be much slower than the processor: the processor spent most of the idle waiting for card to be read or punched, or character.

5. Standard I/O Routines were written to load into the memory at start-up and retained there for use by successive jobs. This input-output control system (10cs).

6. Other New Peripheral were developed to improve system performance.

   In particular magnetic recording techniques were applied to digital signals to give us magnetic tape drives. Later magnetic drums and consequently disks appeared which offered fast direct access to stored data. These developments increased the complicating of the 10cs routines.

7. New System Software

Writing programs in binary machine code was cumbersome and therefore new means of writing were developed.

Thus the assembly language was developed in which case programmers could write programs without concerning themselves with the detailed format of the construction words or the physical location of the instructions. The assembler had to work in collaboration with the IOCS; specific IOCS operations could be involved by the programme by use of a CALL instruction C was translated into a jump to a specific entry pt in the IOCS code.

Other programming languages (i.e. COBOL, FORTRAN) soon followed and accelerated the rate of assemblers and compilers which also complicated the structure of jobs being presented to the computer in order to execute a program; the source version had to be submitted and assembled / compiled before the program could be executed.

Thus a job consists of the following sequence.

- Load assembler (or computer)
- Road in program assembly language (or other Programming languages)
- Assemble program to another area of memory
- Execute program
- Supply input data cards required by executing program.

New hardware system

Around 1960, a revolutionary new computer (ATLAs) was designed by a team from Manchester University and the Ferrant company. The Atlas was the first computer designed with the requirements of an O/S in mind. It introduced many novel features which include interrupts, and a virtual memory system. The interrupt mechanism made immediate impression in computer and O/S design and consequently made the job of managing several programs and peripheral devices simultaneously much easier.

In 1964 1BM produced the system 360 series of computer which evolved into system 370 and then the 808X machine in use today. These range of computers provided a wide range of computing facilities within a compatible series of machines, supported by the manufacturer through many revisions and enhancements.

New Computing Technologies/Paradigms

Arising from the foregone discussions, new technologies were developed amongst which are

- Strangle stream batch processing
- Simultaneous peripheral operations on-line (spooling)
- Real time.

Modern trends include work stations, distributed systems, Graphical user interface (GUI), web-based systems etc.

ON-LINE: The user continuously directly interacts with the system entering appropriate commands and responding to system request, suitable for document preparation, program development and spreadsheet.

CPU Bound: A job using high proportion of CPU time.

I/O Bound: A job using proportionately high I/O transfer time.

Batch Processing refers to situations where jobs submitted in batches to the computer. These jobs were entered in batched started by an operator from a single control console and run in succession without operator intervention.

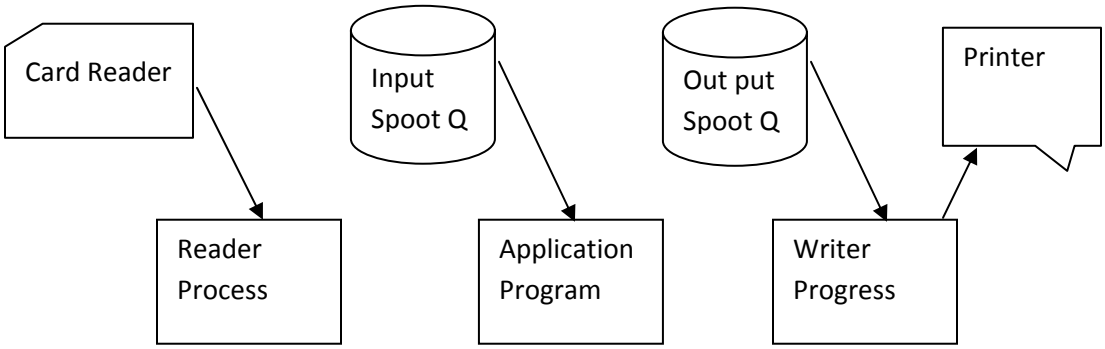
Multi Programming: implies that several users are accessing the system simultaneously and working on quite independent jobs. This can also be refers to as TIME-SHARING.

Here a series of jobs was loaded into the memory at the same time, if sufficient memory space were available. While one job was held up waiting for an I/O transfer, another job was started. A job is selected (from those ready to run) on the basis of an operator assigned priority number. It was observed that jobs differed in the balance of processes to I/O activity could be mixed.

Such jobs are run in two modes, foreground and background. The foreground job had the higher priority and was allowed to run if it was able. The background job could only run if the foreground was awaiting an I/O transfer or was other wise inactive. Background running is inherently more suited to jobs requiring little processor time but with relatively to as spooling.

SPOOLING

This term stands for simultaneous peripheral operations On-line. This technique absorbs surplus processor time by performing I/O transfer for other jobs. I/O data were routed via disk files, so disk systems which are much faster.



Spooling System

Access to slow peripherals is restricted to the one program i.e the spool print program. This facilities the sharing of devices such as printers between several running programs avoiding the need for the programs to complete for possession of the printer.

It should be noted that spooling operations run when no other jobs are available and hence use processor time which would otherwise be much faster disk system overall the through put of the system is improved.

Real-Time Systems

A real-time system is one which responds sufficiently fast that it can influence the environment in which it is working. The term real-time is usually applied to systems where the feedback is more immediate or direct e.g. process control system in factories, missile tracking systems for defense.

Security, safety and immediacy are of paramount importance in real-time systems.

LECTURE TWO

Overview of Operating system

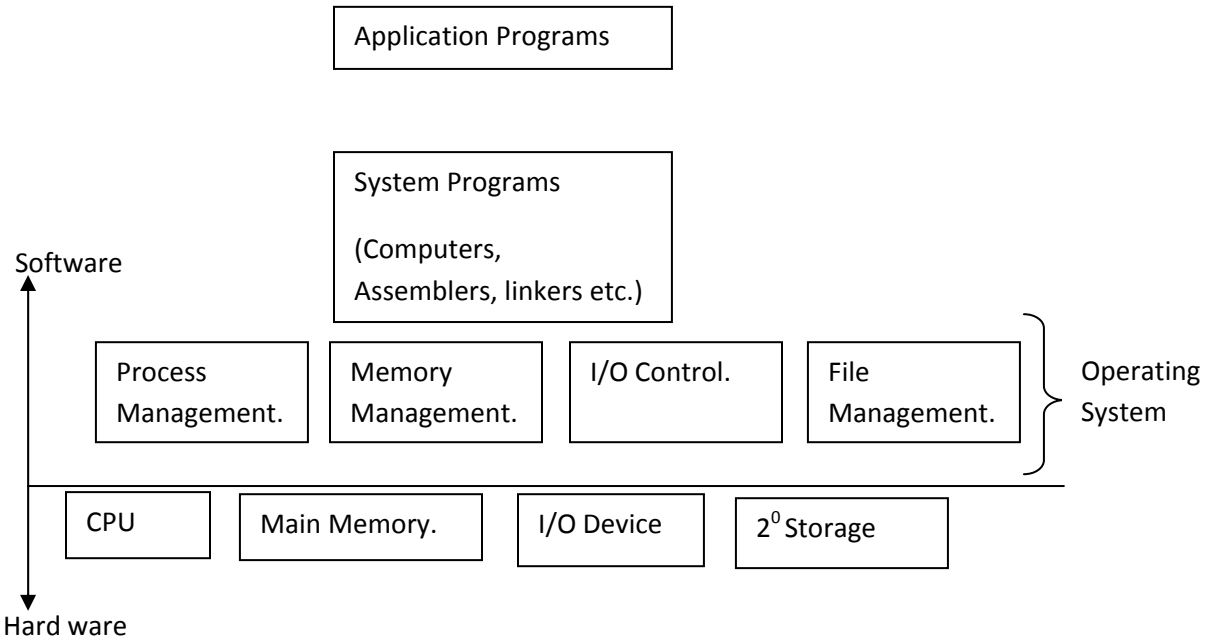
The operating system is primarily a provider and manger of machine resources; the physical resources of a computer are the processor, main memory, secondary storage, I/O devices.

Access of these resources is centralized and controlled by various modules of the system. In addition, the O/S provides other services such as user interface, data security, etc.

A typical list of an O/S components include

1. System programs e.g. program loader, command interpreter etc.
2. Language processors e.g. compilers, assembler, linker

3. Utilities: font editors, end filters, and filters, document formatters, terminal emulator etc.
4. Subroutine libraries: standard C library, utilities etc.



Operating System Structure

Types of O/S

1. Single-user system.

   Single-user system provide a virtual machine for only one user at the time and are suitable for computers which are dedicated to a single function or are inexpensive to make sharing unworthy while. The major emphasis is on the provision of an easily used command language, a simple file system and I/O facilities for terminal and disc.

2. Process Control

   Process control refers to the control by computer of an industrial process (e.g. reefing of oil) and can be extended to include such things as environmental control is a space capsule or monitoring of a patient's condition.

   The main function of the operating system in process control is to provide maximum reliability with the minimum of operator intervention, and to fail safe in the event of any hardware malfunctions.

3. File interrogation system or Database Mgt system (DBMS)

This types O/S is characterized by large set of data which can be interrogated for information. The DBMS must be capable of being modified as information is updated, and should have query facilities to C it can respond to.

4. Transaction processing

This O/S are also characterized by DBMS C is frequently being modified, perhaps several times a second. The major constraint is the necessity of keeping the DBMS up-to-date. In distributed systems, the O/S must resolve the problem of concurrent updated (transaction) without the user's knowledge.

5. General Purpose Systems

General purpose O/Ss are used in computers characterized by a large no of users performing wide range of tasks. Such systems are designed to handle a continuous flow of work in the form of jobs to be run the computer.

Characteristics of O/S

1. Concurrency: The existence of several simultaneous or parallel activities. This raise problems of switching from one activity to another, of protecting one activity against the effect of another, and synchronizing activities C are mutually dependent.

2. Sharing:

   Concurrent activities may be required to share resources or information due to the following reasons: Cost, Building on the work of others, sharing data, and removing redundancy.

   Associated sharing constrains are resources allocation simultaneous access to data, simultaneous program execution, protection against corruption.

3. Long-term Storage

   The need for sharing programs and data implies the need for long-term storage of information. Long-term storage also allows a user, the convenience of keeping his programs or data in the computer rather than on some external medium.

   The problems arising are those of providing easy access, protection against interference, and protection against system failure.

4. Non-determinacy

   An O/S must facilitate the running of the same program to produce the same output with the same date set irrespective of when and how many times the program is run.

However, an O/S (is indeterminate) must respond to events which will occur in an up predictable order. These events are such things as resource requests, runtime errors in programs, and interrupts from peripheral devices. An O/S must be written to handle any sequence of events.

Desirable features for a general purpose O/S include

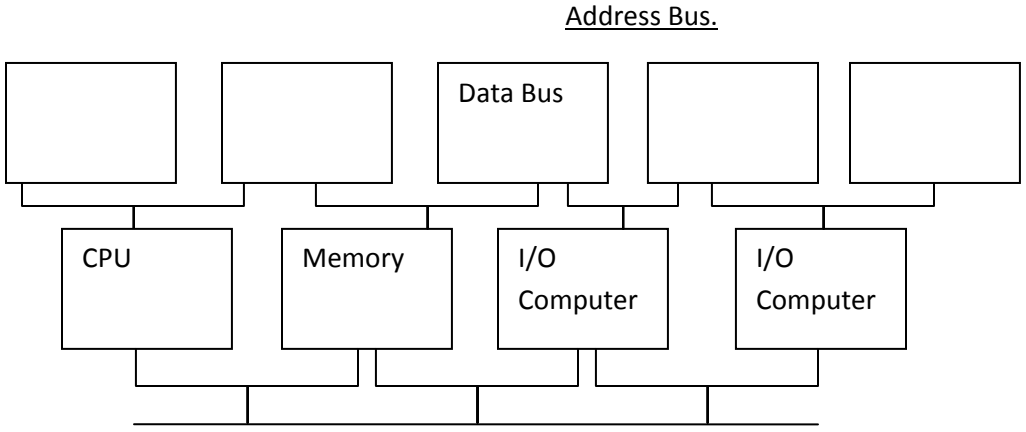
1. Efficiency: An O/S must perform its desired tasks without stress.
2. Reliability: An O/S should be completely error-free, and able to handle all contingencies.
3. Maintainability: an O/S should be modular in construction, with clearly defined interface between the modules, and that it should be well documented.
4. Small storage space: An O/S should be compactable and portable.

LECTURE THREE

Hardware features

General machine structure.

A computer system generally consists of a group of modules, interconnected by a number of signal line sets called buses. The most common arrangement is a three bus configuration shown below



Computer Bus System

Each bus consists of a number of signal line, is the case of the data and address buses, the number of lines is referred to as the WIDTH of the bus. This bus width is importance because it affects the performance of the computer. The I/O controllers are hardware modules used to facilitate communication with I/O devices.

The Data Bus is used to transfer data between the connected unit. The address bus is used to specify the sources or destination of data. The addressing scheme references both main memory locations and I/O controllers.

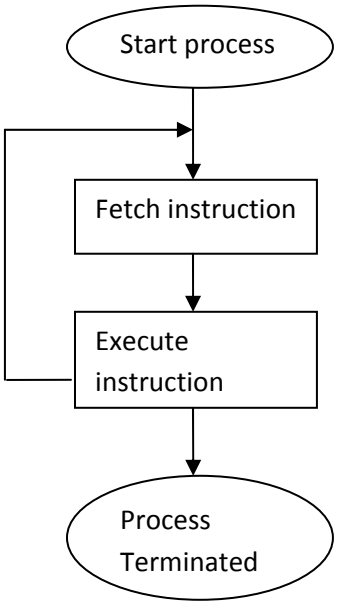
The control bus is used to transmit timing and control signals between modules of particular interest within the control bus is the interrupt line which carries an interrupt request from the I/O controllers to the processor.

ACTIVITY: ***Students to read more about the Buses.***

Interrupts

The interrupt system is totally essential for the functioning of any O/S. its purpose is to alert the O/S when any of a number of "events" occurs, so that it can suspend its current activity and deal appropriately with the new situation. This means that the processor can be used to sustain several executing programs and I/O transfers simultaneously, servicing each as the need arises.

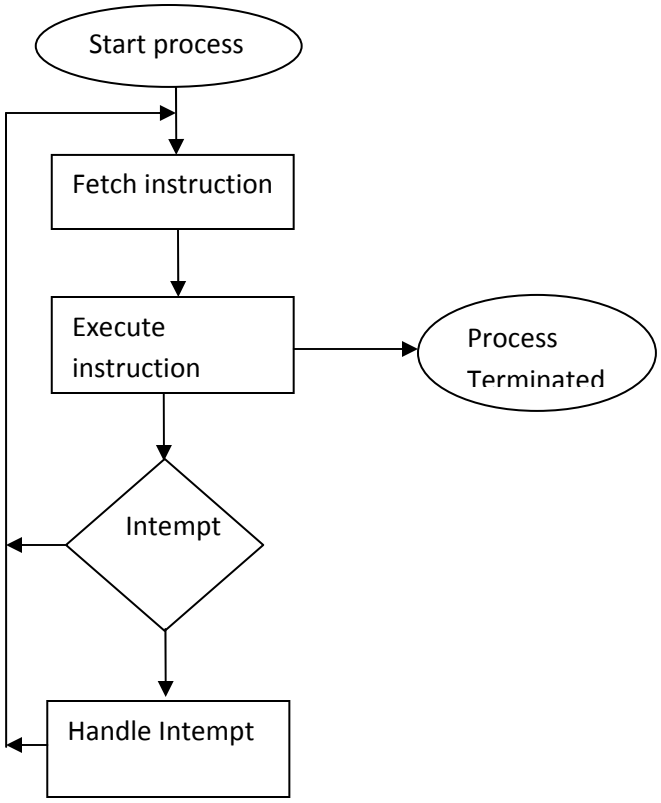
The simples test view of processor action is that it consists of a simple fetch-execute cycle as shown below



Simple fetch-Execute Cycle

If the execute instruction consist of an I/O operation, the assumption must be that using simple model, the processor would be suspended waiting for completion of the I/O transfer. This is termed programmed I/O clearly, this is inefficient, since the processor could spend this time executing some other program. In particular the huge disparity between the processor speed and the I/O speed makes programmed I/O impractical for general systems.

Incorporating an interrupt mechanism into the cycle would produce the alternative scheme shown below



As each instruction terminates, the processor checks for the occurrence, since point in the previous cycle, of an interrupt from an I/O device. If an interrupt has been received, the processor is diverted to an interrupt handling routine with the O/S, which has to determine the source and cause of the interrupt and then take appropriate action to deal with it. In the meantime, the program C was executing is temporarily suspended. Typically, an interrupt might be caused by the completion of an I/O transfer C was initiated by some other program. When the interrupt has been serviced, the execution will return to the interrupted program.

The importance of the interrupt system is that it permits several programs and I/O activities to proceed independently and a synchronously, while enabling over control to be retained by the O/S. this implies

i. Control of the processor may be handed over from the O/S to a user program, while enabling the O/S to regain control when any critical event occurs

ii. The O/S can allow an I/O operation to proceed without constant supervision, because it will be alerted on completion of the transfer.

It is important to appreciate that the O/s is sharing the processor with the user programs; once execution of a user program is started, it would, is the absence of the interrupt system, continue until its

normal termination. Interrupts therefore guarantee that the control will revert back to the O/S at frequent intervals and at crucial times.

Types of interrupt

Interrupts may be generated from a number of sources, which may be classified as below.

S/N	Types	Name
1.	I/O	Generated by the controller of an I/O duce to signal normal completion or the occurrence of an error, or failure condition
2.	Timer	Generated by an internal clock within the processor, used to alert the O/S at pre-determined intervals to attend to time critical activities.
3.	Hardware Error	Generated hardware faults e.g. memory parity or power failure.
4.	Program	Generated by error conditions arising with user programs e.g. address violation execution of an invalid instruction.

Direct Memory Access (DMA)

DMA is a technique used to transfer data between memory and I/O devices with less effort on behalf of the O/S. the DMA unit has access to the data bus and can transfer data autonomously in and out of memory. In practice, a program would instruct the DMA unit to, transfer a specified block of data from memory to a peripheral device.

The DMA unit operates by suspending the CPU and accessing the memory system itself to obtain the data required. This technique is called CYCLE STEALING because machine cycles are effectively stolen from the CPU and used by the DMA unit to transfer data along the data bus.

Note that this is not an interrupt; the current program context is not saved and the CPU does not do something else. In effect, the speed of execution of the program is slowed down since it is losing processor cycles. The DMA unit can be part of a peripheral control unit for a device or it may be a separate module serving several devices.

ACTIVITY: Read up Memory addressability

LECTURE FOUR

THE PROCESS CONCEPT

The concept of a process is an essential part of the theory and practice of modern O/S. though abstract is nature, a process can be defined to mean

- The execution of a program or
- A task which is executed by their processor.

Process creation and states

When a user initiates execution of a new process, the O/S creates a data structure, some times called a process control Block (PCB) which gives substance to the process and serves to control it. The PCB has an identification Number (Process id (PID)) by which it can be referenced by the O/S and by other processes.

Kernel mode

We normally assume that the operating system is "just another program" running in the computer most computers however, have a special operating mode called KERNEL (or supervisor mode) into which the machine switches when the O/S is running. This enables the execution of certain privileged instructions, particularly is the area of I/O operations, which are not available to normal programs (i.e. user mode).

The processor can only be switched into kernel mode by actions C involve the O/S (I.e. interrupts or software system calls) so that there is no way that a user program can execute them. This is to allow the O/S to exercise control over certain aspects of the computer, while in the interests of system security, this facility is denied to any user program.

Process life cycle

A process can be viewed as the execution of program and is used as a unit of work for a processor. In other words, the process simultaneously manages several activities at one time each corresponding to one process. The processor at any instant can only be executing one instruction from one program but several processes can be sustained over a period of time by assigning each process to the processor at intervals, while the remainder becomes temporarily inactive.

The process takes complete control of the processor and continues so until either

(i) The process issues an I/O request by means of a system call

(ii) An interrupt occurs.

The O/s regains control at this time and the processor may be re-assigned to another process.

Process creation

A process consists of the machine code image of the program in memory plus the PCB structure used to manage the process during its lifetime, Process can also be initiated by a user process so a single program activated by a user could result eventually in several separate processes running simultaneously. The process creating a new process is called the parent, while the created process is called the child. The act of creating a new process is referred to as SPAWNNING a process. A child process could itself spawn a process, resulting in a tree of processes.

When a process terminates, it will return to its parent, supplying some return code which indicates the success/failure of the child process's mission. A process spawned by the O/S its will report back to the system. It is possible for a parent process to terminate before a child; in this case the return code of the child will be collected by the next higher level parent or by the O/S.

In addition to creating the process data structures, the O/S. will also have to allocate resources to a new process. These include physical resources such as memory space, I/O devices and 2^0 storage space and files. Information regarding resources allocated to a process is managed within the PCB and associated structures.

Process state diagrams

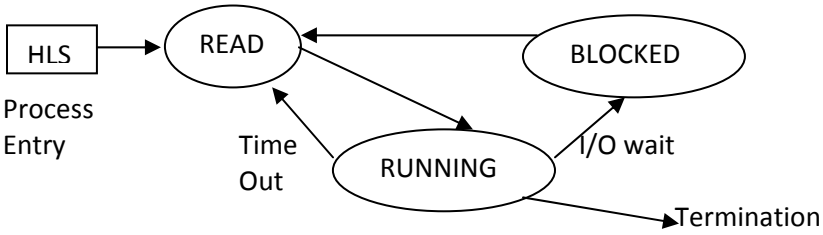
A process can be in any of the follow P3 modes at any given point in time, namely

(a) READY state: The process is able to use the processor when it is assigned to it;

(b) RUNNING state: The process is being executed

(c) BLOCKED state:

We can visualized the process as existing in one of a number of different state using the process state Diagram, illustrated below: I/O completion.

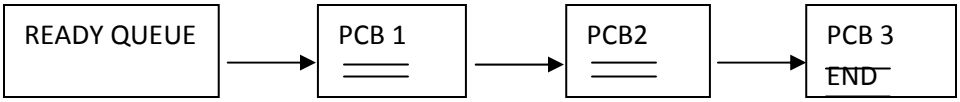


A 3 State Process State Diagram.

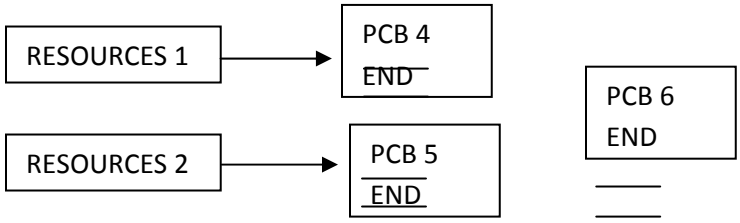
Ellipse ⟶ state

Live ⟶ transition from one state to another

Two significant transitions are the initial entry into the system which is controlled by the High level scheduler (HLS) and the transition from Ready to RUNNING is controlled by the low level scheduler (LLS). At any instant, a process is in one and only one of the three states. For single processor computer, only one process can be in the RUNNING state at any one instant. There may be many process in the READY and BLOCKED state; each of these sate will have an associated queue of processes such queues are implemented by linking the process PCB into a linked list as illustrated the diagram below: In the case of the BLOCKED state, several queues can be employed, each representing one event for which processes in the queue are waiting.



BLOCKED QUEUES



Note that:

(1) Processes entering the system must go initially into the READY state.

(2) Processes can only enter the RUNNING state. (normally) though processes can be aborted by the system or by the user C could catch the process is the READY or BLOCKED state.

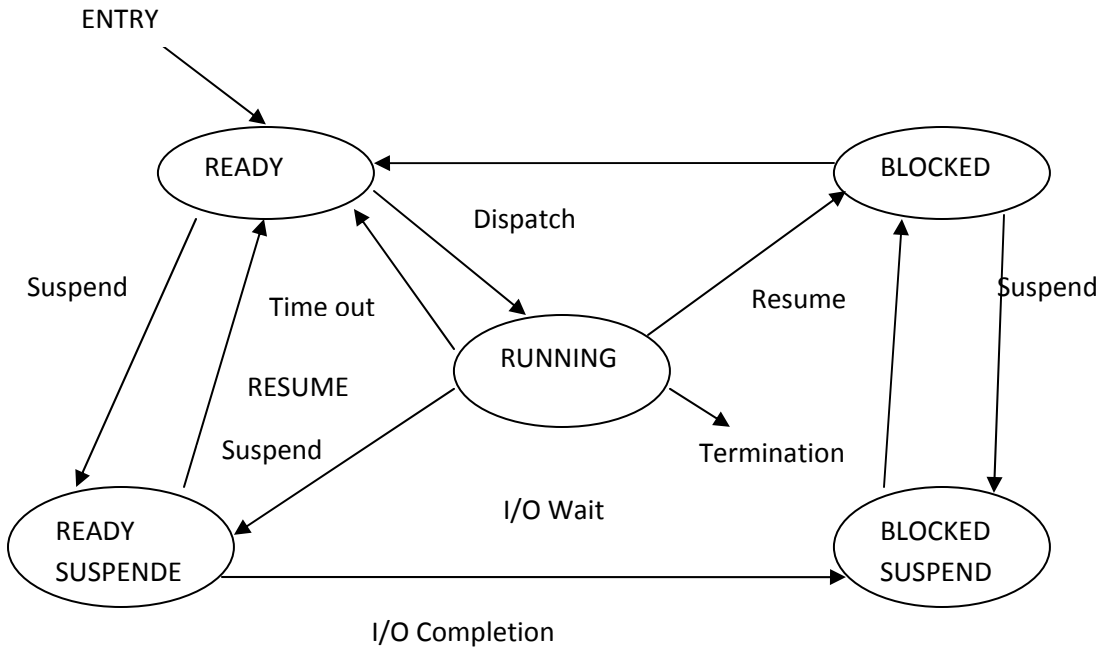
The five state (Process state Diagram) Model

The three sate model can be extended to allow for other possible events, namely suspending and resuming a process.

These events, are controlled by the medium level scheduler (MLS)

A process can be suspended for different reasons, chiefly when a process is being swapped out of memory by the memory management system in order to free memory for other process. A suspended process remains dormant until it is awoken (resumed) by the system or the user.

The process could be suspended while in the RUNNING, READY or BLOOCKED state, giving rise to two other sates, namely READY SUSPENDED and BLOCKED SUSPENDED. A RUNNING process C suspended becomes READY suspended.



5 – States PSD

Threads

A thread is an independent sequence of execution within the content of a parent process. Threads share the resources of the parent process but are separately and independently scheduled. Threads can perform many of the functions of the processes & are much cheaper in terms of system overheads. However, since they share the same resources, they may conflict in their accessing of these resources. This creates the need for meditation between competing threads by the applications such as a database server. A request would be allocated to a thread within the server process while the server takes advantage of its knowledge of the database to optimize the allocation of time between the competing threads.

LECTURE FIVE

Introduction Scheduling

Scheduling refers to the determination of the optimum sequence and timing of assigning processes to the processor by the O/S. The overall scheduling effort is intended to meet some objectives in terms of the system's performance and behaviour. Thus, the scheduling system should

(1) Maximize the system through put;

(2) Be "fair" to all users consistently relative to the importance of the work being done.

(3) Provide tolerable response (for on-line users) or turn-around time (for batch users).

(4) Degrade performance gratefully. If the system becomes overloaded, it should not "collapse" but avoid for their loading and/or temporarily reduce the level of service (response time)

(5) Be consistent and predictable. The response &turn-around time should be relatively stable from day to day.

Criteria for Scheduling

The following criteria are considered in scheduling of processor work.

(1) Priority assigned to job

(2) Class of job e.g. batch or on-line or real-time.

(3) Resource requirements: e.g. expected run-time, memory required

(4) I/O or CPU bound

(5) Resources used to date e.g. the amount of processor time considered

(6) Waiting time to date e.g. the amount of time spent

Levels of Scheduling

Scheduling can be exercised at three distinct levels namely

(i) High level scheduling (HLS) (or long-time or job scheduling): deals with the decision as to whether to admit another new job to the system.

(ii) Medium – level scheduling (MLS) or intermediate scheduling: is concerned with the decision to temporarily remove a process from the system (in other to reduce the system load) or to re-introduce a process.

(iii) Low – level scheduling (or short-term or processor scheduling): handles the decision of which ready process is to be assigned to the processor. This level is often called the Dispatcher but the term refers to the actual kernel activity of transferring control to the selected process.

*Students are to read up (Preemptive scheduling, Non-preemptive scheduling and Cooperative scheduling

***Assignment (to be submitted at a date to be announced**)

Describe briefly, the following scheduling policies and classify each as preemptive, non-preemptive or cooperative scheduling.

(i) First Come First Served (FCFS)

(ii) Shortest-Job First (SJF)

(iii) Round-Robin (RR)

(iv) Shortest Remaining Time (SRT)

(v) Highest Response Ratio Next (HRN)

(vi) Multi – level Feedback Queues (MFQ)

LECTURE SIX

DEVICE HANDLERS

1. Organization of I/O software and hardware

The I/O system is relatively slow and lack consistency when compared with other hardware resources [processor, main memory &file system]. This is due to the nature of I/O devices and their role in trying to provide communication between the microsecond domain of the computer with the somewhat more leisurely outside world. The range of I/O devices and the variability of their inherent nature, speed, specific design etc make it difficult for the O/S to handle them with any generality. The table below indicates the range of characteristics found in I/O devices.

	Characteristics	Examples
1.	Data Rate	Disk : 2Mbytes/sec
		Keyboard : 10-15 bytes/s
2.	Unit of Transfer	Disk : blocks of 512, 1024 etc by ten
		Screen : single characters
3.	Operators	Disk : read, write, seek etc
		Printer : write, move, paper, select font
4.	Error conditions	Disk : read errors
		Printer : paper out
5.	Data Representation	

Characteristics of I/O Devices

The computer communicates with I/O devices by means of an I/O bus system. Each I/O device has an associated hardware controller unit attached to this bus system which can transmit and /or receive data to / from the computer main memory. Devices on the bus are assigned addresses which enable the processor to identify the device to which a specific interchange of data is to be directed. To enable the I/O devices to run asynchronously with respect to the processor (i. e the I/O device performs its activity independently of and simultaneously with the processor activity), a system of interrupts is used. The device sends an interrupt signal to the

processor when it has completed an assigned task, enabling the processor to provide further work for the device.

Most computers use DMA which enables much faster data rates, since the processor is only involved in initiating the transfer; thereafter the data is transferred directly between memory and device without processor involvement. The processor is notified by interrupt only on completion of the whole transfer.

2. Objectives of the I/O system.

Efficiency: Perhaps the most significant characteristics of the I/O system is the speed disparity between it and the processor and memory. This is because I/O devices involve mechanical operations, they cannot compete with the microsecond or nanosecond speed of the processor & memory.

The design of the I/O system largely reflects the need to maximize the problems caused by this disparity. Of central importance is the need to make the most of the I/O devices operates at maximum efficiency.

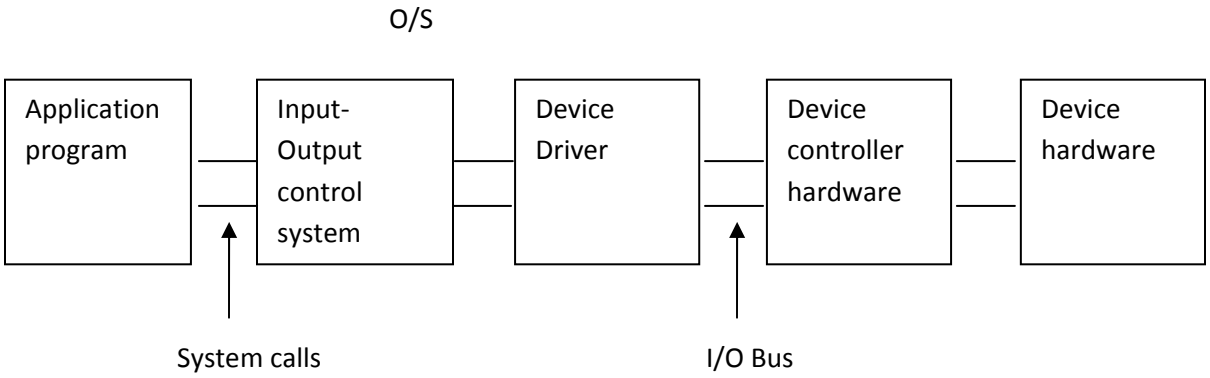
3. Generality and device independence

I/O devices are typically quite complex mechanically and electronically. Much of this complexity is in the realm of the electronic engineering and is of no interest to the user/ programmer of the computer. The average user is not aware of the complexities of positioning the heads on a disk drive, reading the signal from the disk surface, waiting for the required sector to rotate into position.

Users and programmers are distanced from these realities by layers of software which provide various levels of abstraction in addition to these inherent complexities, devices also vary enormously in their design, mode of operation, interface protocols etc., producing yet more potential headaches for the O/S. The O/S must be insulated from the complexities since it is not reasonable for the O/S to be cognizant of the detailed operation of every device with which it may have to communicate

4. Structure of the I/O system

To address the problems highlighted above, the hardware & software elements of the computer's I/O system consists of a number of layers separating the user at one end to the physical devices at the other end. Moving in this direction, we see a progressive increase in the level of details or specialization. The overall picture is shown in the diagram below



Structure of the I/O System

(i) Application program

Within the application, I/O activity is expressed in user-oriented terms such as read record 21 from file xy2". Such instructions in a high level language are translated into corresponding system calls which invoke O/S functions. The instructions (system calls) are expressed in logical terms and are largely independent of the device used.

(ii) Input – Output Control System (IOCS)

This term denotes that part of the O/S which deals with I/O related system calls. It performs initial processing and validation on the request and rontes it to the appropriate handler at the next stage. It is also where the general management of I/O interrupts takes place.

(iii) Device Driver or Handlers

A device driver (device handler) is a software module which manages the communication with, and the control of a specific I/O device, or type of device. It is the task of the device driver to convert the logical requests from the user into specific commands directed to the device itself. For example, a user request to write a record to a floppy disk would be realized within the device driver as a series of actions, such as checking for the presence of a disk in the drive, locating the file via the disk directory, positioning the heads etc.

Although device drivers are in effect "add-on" modules, they are nevertheless considered to be part of the O/S since they integrate closely with the IOCS. System differs in the degree to which drivers are structurally bound into the rest of the O/S, in UNIX for instance, the driver code has to be compiled and linked with the kernel object code, while in MS-DOS, and device drivers are installed and loaded dynamically.

(iv) Device controllers

A device controller is a hardware unit which is attached to the I/O bus of the computer and provides a hardware interface between the computer and the I/O device. Since it connects to the computer

bus, the controller is designed for the purpose of a particular computer system while at the same time it conforms in interface terms with the requirements of the actual I/O device.

(v) Device

I/O devices are generally designed to be used in a wide range of different computer system. For example, the laser printer could be used on windows, MS-DOS, Apple and UNIX systems. The O/S is insulated from the specific characteristics of the device by the other elements of the I/O pathway.

(vi) Block and character Devices

A block device transfers data in groups of characters at a time, while a character device transfers data one character at a time. Block devices are magnetic or other forms of secondary storage and character devices are any other device.

Block devices are inherently more complex and hence require more extensive services from the O/S, to deal with actions such as random accessing of data, bidirectional data flow etc.

Character devices on the other hand are relatively simple.

ACTIVITY: Students are to study the implementation of Device Drivers in

UNIX
WINDOWS } for comparative sake.
MS-DOS

LECTURE SEVEN

Memory Organization

1. Objectives

Main memory is essential within a computer first and foremost to enable processes to exist. It is within the main memory that instructions reside which is interpreted by the processor. However, in addition to holding program code, the memory is also a work space and transient storage repository for various kinds of data. Consequently, the memory of a typical computer is occupied by a wide range of different "objects" such as O/S code, O/S data (process table, file description table etc), user program code and data, video storage space etc.

Our goal is to explore how the O/S manages the memory space for the loading and execution of its own and user program code. We shall also see how users get as much use out the installed memory as possible using virtual storage technique.

}

Objects of memory management the principal problems to be handled by the O/S's memory manager are to:

- Provide the memory space to enable several processes to be executed at the same time; (Relocation)
- Provide a satisfactory level of performance (i.e. process execution speed) for the system users: (Physical organization)
- Protect each process from each other; (protection)
- Make the addressing of memory space as transparent as possible for the programmer; and (Logical organization)
- Enable sharing of memory space between processes where desired. (Sharing)
- Process loading: is the transfer of the program code from the secondary storage to main memory for it (process) to be executed.

Swapping is the transfer of a process from main memory to secondary storage in other to accommodate and consequently execute another process. The overall effect therefore is swamping of two processes.

2. Memory allocation methods:

A number of memory management techniques exist in order of increasing complexity. It will be seen that each method "solves" some shortcoming of the previous method and reflects the way in which the systems evolved historically. We shall make use of the concept of program relocation in which the need is to be able to position programs at arbitrary locations in the memory and also to be able to relocate the programs efficiently while the programs are still active.

a) Single process system

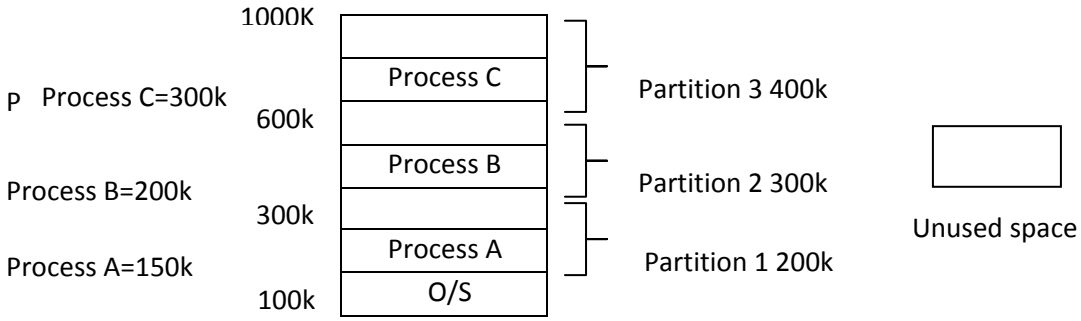
In a single processor system, the process to be executed is loaded into the free of the memory.

Unusual
User Process
O/S

Such an arrangement is clearly limited in capability and is found primarily in simple systems such as games computers. An extension to this is to permit the loading of more than one process into memory simultaneously which leads on to the schemes considered below

b) Fixed partition memory

In this scheme, the memory is divided into a number of separate fixed areas, each of which can hold one process.



Fixed partition allocation

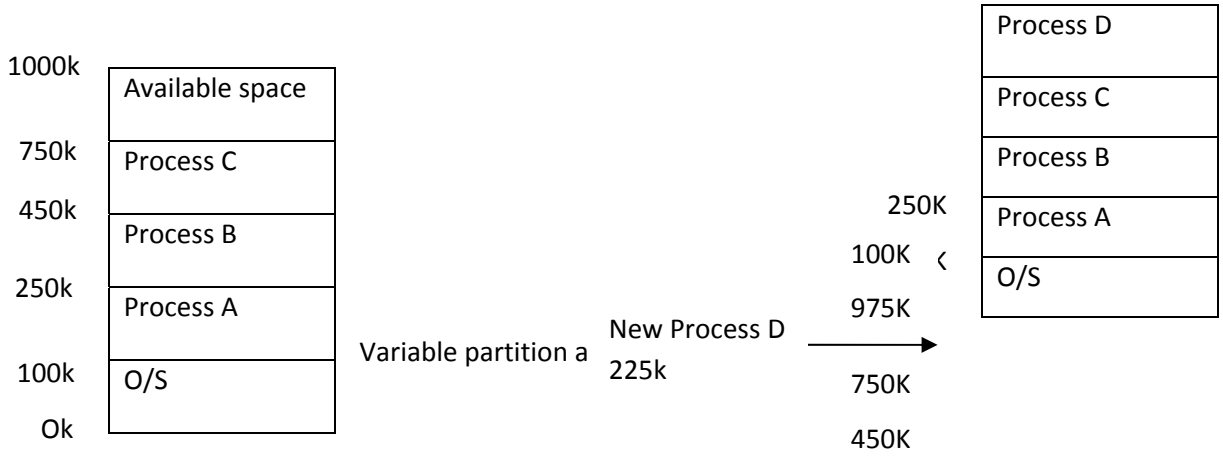
The diagram above illustrates a fixed partition allocation scheme consisting of B areas of size 200k, 300k, and 400k respectively each of which holds a process. Given that the O/S can support simultaneous execution of process, all three processes could be active at any time. The number of partitions in a practical system would be controllable, within limits, by the system manager and would depend on the amount of memory available and the size of processes to be run.

Typically the partitions would be set up with a range of partitions sizes, so that a mixture of large and small processes could be accommodated. However each partition will typically contain unused space which results to internal fragmentation (wasted space within the space allocated to a process).

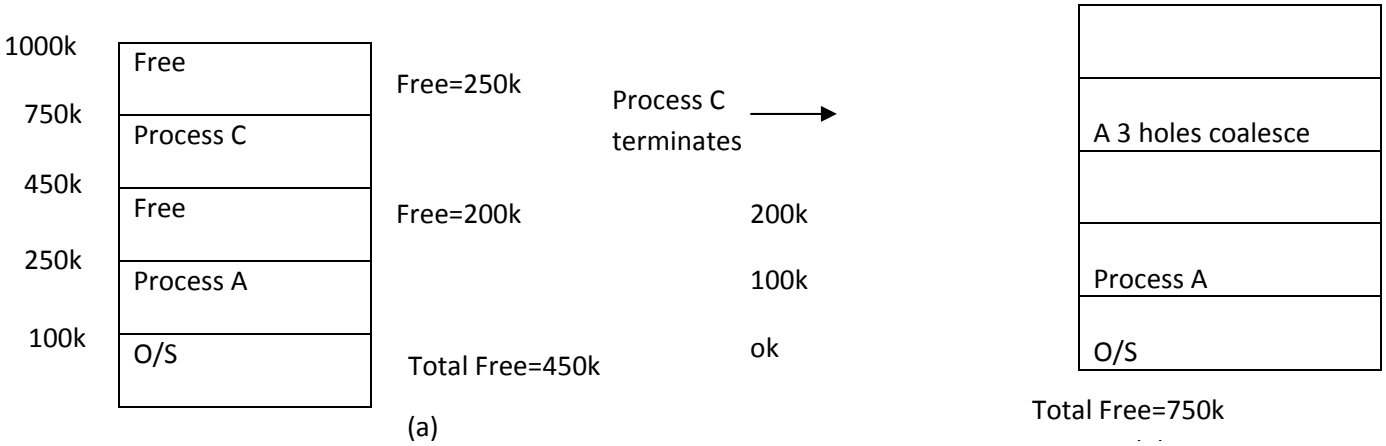
Normally the 3 running processes will be independent, so that process A, for example, should not reference addresses outside the bounds of partition1. However since there is no general way to prevent a process generating an invalid address, memory protection has to be implemented by the O/S and hardware.

c) Variable partition memory

The solution to the fixed partition problems is to allow the partitions to be variable in size at load time, in other words to allocate to the process the exact amount of memory it requires. Processes are loaded into consecutive areas until the memory is filled, or more likely the remaining space is too small to accommodate another process.



When a process terminates, the space it occupied is freed and becomes available for the loading of a new process. However, this reveals a flaw in the nature of the technique. As processes terminate and space is freed, the free space appears as a series of "holes" between the active memory areas. The O/S must attempt to load an incoming process into a space large enough to accommodate it. It can often happen that a new process cannot be started because none of the holes is large enough even though the total free space is more than the required size. This is illustrated in the diagram below in which processes B and D have terminated. Distribution of the free memory space in this fashion is termed external fragmentation.



Fragmentation and Coalescing

Coalescing of holes: It will frequently happen that a process adjacent to one or more holes will terminate and free its space allocation. This results in two or three adjacent holes, which can then be viewed and utilized as a single hole. The effect is referred to as coalescing of holes, and is a significant factor in maintaining fragmentation within usable limits.

Storage placement policies.

An algorithm termed placement policy is used to select the best locations in which to place new process in the variable partition scheme. Such placement policies could be:

i) Best -fit policy

An incoming process is placed in a hole in which it fits mostly "tightly"

ii) First- fit policy

An incoming process is placed in the first available hole which can accommodate it.

iii) Worst-fit policy

An incoming process is placed in the hole which leaves the maximum amount of unused space; which logically must be the current largest hole.

Example

A variable partition memory system has at some point in time, the following hole sizes in the given order.

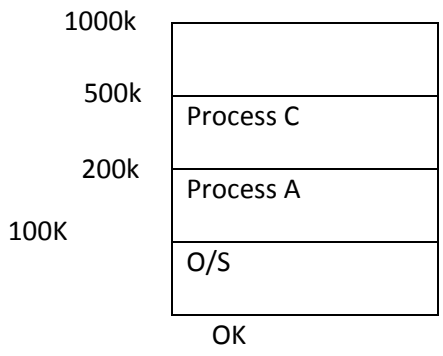
100k, 300k, 200k, 400k.

A new process is to be loaded of size 150k. Which hole size would be filled using best-fit, first-fit and worst-fit respectively?

The principal defect is the problem of fragmentation which reduces the effective capacity of the memory.

d) Variable partition allocation with compaction

The fragmentation problem encountered in the variable partition can be solved through a process called compaction. This process involve moving resident processes about the memory in order to close up the holes and hence bring the free space into a single large block.



Memory compaction is made

(i) As soon as any process terminates

(ii) When a new process cannot load due to fragmentation

(iii) At fixed intervals; and

(iv) When the users decides to.

In practice, the compaction scheme has seldom been used due to the fact that its overheads and added complexity tend to minimize its advantage over the non-compacted scheme.

e) <u>Simple paging</u>

In a paged system, each process is notionally divided into a number of fixed size "chunks" called pages, typically 4kb in length. The memory space is also viewed as a set of page frames of the same size. The loading process involves transferring each process page to some memory page frame. The diagram below illustrates 3 processes which have been loaded into contiguous pages in the memory.

Process A

PAGE 1
2
3
4
5
6

A1
A2
A3
A4
A5
A6
B1
B2
B3
B4
B5
C1
C2
C3
C4
-
-
-

Process B

PAGE 1
2
3
4
5

Process C

PAGE 1
2
3
4

A specific memory location is addressed using paging scheme in the form.

(p, d)

Where p is the number of the page containing the location and d is the displacement (or offset) of the location from the start of the page.

These parameters are derived from the actual memory address by sub diving the latter into two portions, representing the respective page and displacement values. The figure below illustrates this process, using as an example of a 16 bit address.

15 11 0

0101	010110011011

Page no Displacement

(p) (d)

The page no uses the top 5 bits and therefore has a value range of 0 t0 31 (i.e.2-1) or 32 papes. The displacement value uses 11 bits and therefore has a range of 0 to 2023 (i.e. 2”-11). This means a system based on this scheme would have 32 pages each of 2024 locations.

Command languages

Command languages are designed to allow a user to interact directly with the O/S from a terminal and the principal use made of the command language is to initiate the execution of programs. Hence, it is a facility for on-line user rather batch, although command languages often contain some provisions for batch working.

Job control languages (JCL)

JCL are used to specify and control the running of batch jobs generally on large computers. Recall that a batch operation implies that jobs are initiated by a user or computer operator and subsequently run with no (or minimal interaction or intervention). The principal characteristics of this mode of working are:

a) The execution is intensive in the sense that there is no delay caused by human interaction.

b) Most suitable for routine jobs typically found in commercial information processing such as payroll production.

c) Since the jobs runs on its own, responses to possible abnormal events must be planned.

d) The resources required by a job are generally more predictable and hence can be used in resource scheduling.

e) The work being done is often costed and charged to some cost-centre within organization.

A particular characteristic of batch-style running is the concept of a job (i.e. specification of the set-up, execution and control of a suite of programs which constitute a complete processing operation). A job is prefixed with a job command statement which supplies information such as

JOBNAME: Specifies the name of the user submitting the job plus a job name value to distinguish different jobs from the same user.

JCL are used to define the requirements & parameters of work submitted to a batch system & would generally be used by computer operations staff in a mainframe environment.

ACCOUNT: Specifies the name of an account to which the job is to be charged. This enables work done on the computer to be costed to specific company departments and/or projects.

RERUN: Indicates whether the job is re-startable after a system failure. For example, a job which is simply printing a report without updating any files could probably be re-started.

LOCATION: Specifies the terminal which is to receive system messages from the job.

STARTCLASS: Specifies the scheduling queue into which the job is to be initially placed. This affects the initial priority given to the job.

QUOTA: Specifies an allocation of memory space in kilobyte for the job.

OCP-TIME: Specifies the maximum amount of processor time required.

RANK: Provides a measure of the importance of the job relative to others and influences the allocation of scarce resources.

A job may be pre-defined and stored as a procedure for later use; a RUNJOB command is used to invoice such procedure.

JCL are designed for batch use supply information of relevance to this mode of working

(E.g. estimate of resources required, accounting information, error processing etc.)

Library Maintenance

To avoid loading many programs from external, input, installations usually make the commonly used programs available in a program library. With increase in speed it became necessary to simplify the complexity of rote programming. Large volumes of code had to be produced much of it repetitious and similar to previously written code. Conventions for writing standard codes and ways of using these codes were adopted for each machine so that any user had access to a LIBRARY of codes generated by the users' group.

Thus, the library is a collection of programs which can be used with other programs. To make the collection useful and accessible, it is put on a high-speed secondary storage device such as a disk, and the user can obtain a copy of any member for the loading process. The loader first combines the sections of code provided by the user, linking them by one of the above methods. If some of the external names are not defined in the user supplied programs, the loader attempts to find a library program of the same name. It does this by consulting an index, or directory, of the names of all programs in a library file. When a program is located, it is read by the loader and linked to the other user code. This means that the library can be retranslated each time it is used.