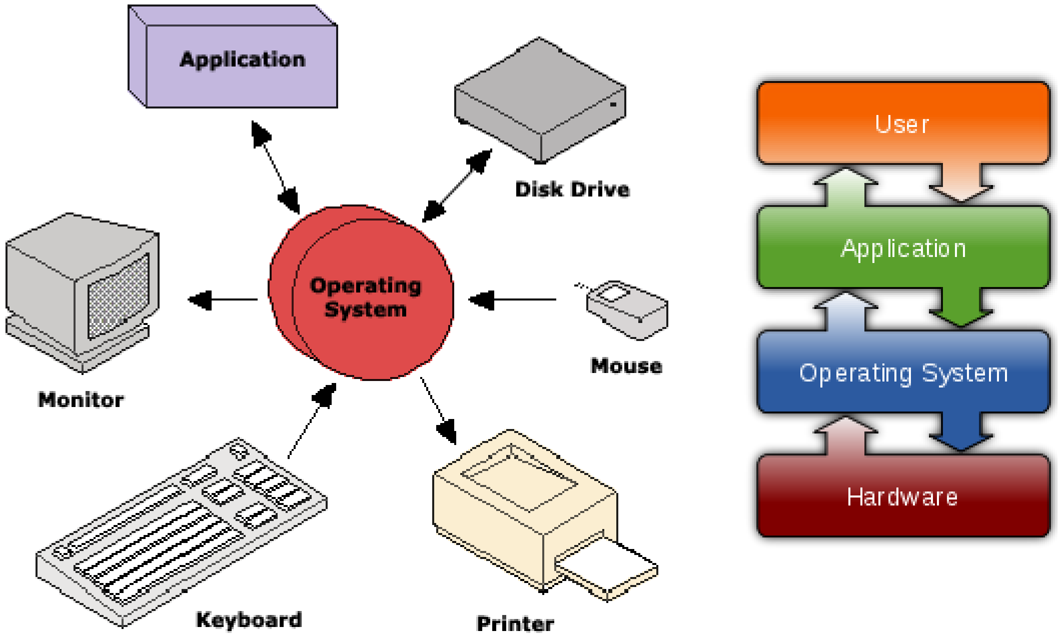
Module 4: Operating Systems

4.1 What is an Operating System?

The operating system is the most important program that runs on a computer. Every general-purpose computer must have an operating system to run other programs. Operating systems perform basic tasks, such as:

- Recognizing input from the keyboard.
- Sending output to the display screen.
- Keeping track of files and directories on the disk.
- Controlling peripheral devices such as disk drives and printers.



The Two Diagrams above Represent the Role of the Operating System

For large systems, the operating system has even greater responsibilities and powers.

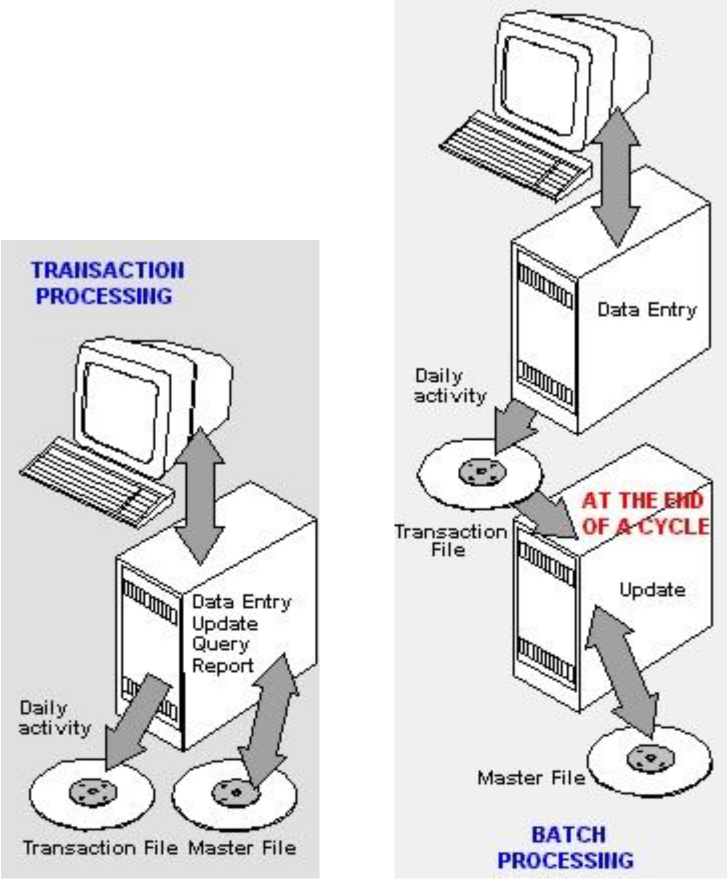
- It is like a traffic cop - it makes sure that different programs running at the same time do not interfere with each other.
- It is also responsible for security, ensuring that unauthorized users do not access the system.

Operating systems provide a software platform on top of which application programs can run. The application programs must be written to run on top of a particular operating system. Your choice of operating system, therefore, determines to a great extent the applications you can run.

4.2 An Overview of different kinds of Operating Systems

Batch processing involves executing a series of non-interactive jobs all at one time. The term originated in the days when users entered programs on punch cards. They would give a batch of these programmed cards to the system operator, who would feed them into the computer.

Batch jobs can be stored up during working hours and then executed during the evening or whenever the computer is idle. Batch processing is particularly useful for operations that require the computer or a peripheral device for an extended period of time. Once a batch job begins, it continues until it is done or until an error occurs. Note that batch processing implies that there is no interaction with the user while the program is being executed.



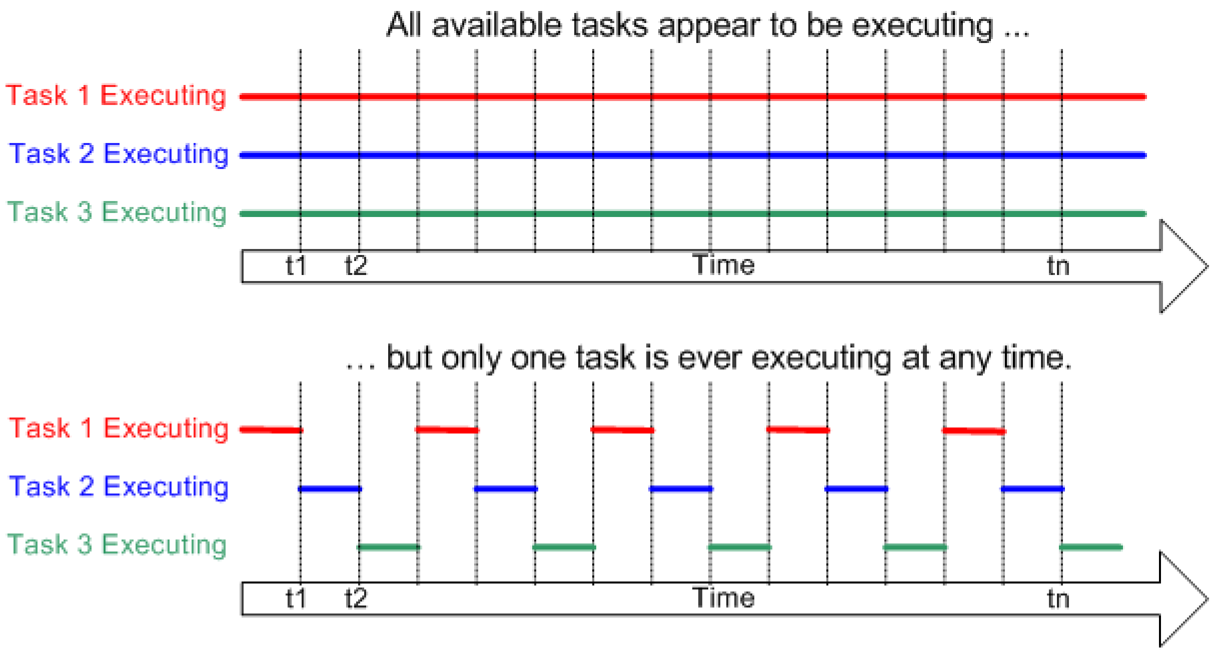
Transaction Processing and Batch Processing

An example of batch processing is the way that credit card companies process billing. The customer does not receive a bill for each separate credit card purchase but one monthly bill for all of that month’s purchases. The bill is created through batch processing, where all of the data are collected and held until the bill is processed as a batch at the end of the billing cycle.

The opposite of batch processing is transaction processing or interactive processing. In interactive processing, the application responds to commands as soon as you enter them.

A real-time operating system processes inputs simultaneously, fast enough to affect the next input or process. Real-time systems are usually used to control complex systems that require a lot of processing like machinery and industrial systems.

In a multi-tasking OS several applications may be simultaneously loaded and used in the memory. While the processor handles only one application at a particular time it is capable of switching between the applications effectively to apparently simultaneously execute each application. This type of operating system is seen everywhere today and is the most common type of OS, the Windows operating system would be an example.



Multitasking

In pre-emptive multitasking the processing time is shared with all running programs. Pre-emptive multitasking creates a time-shared environment in which running programs receive a recurring slice of time from the CPU. Depending on the operating system, the time slice may be the same for all programs or it may be adjustable to meet the current mix of programs and users. For example, background programs can be given more CPU time no matter how heavy the foreground load and vice versa. In addition, the OS is able to grab the machine cycles that a modem or network program needs for uninterrupted processing.

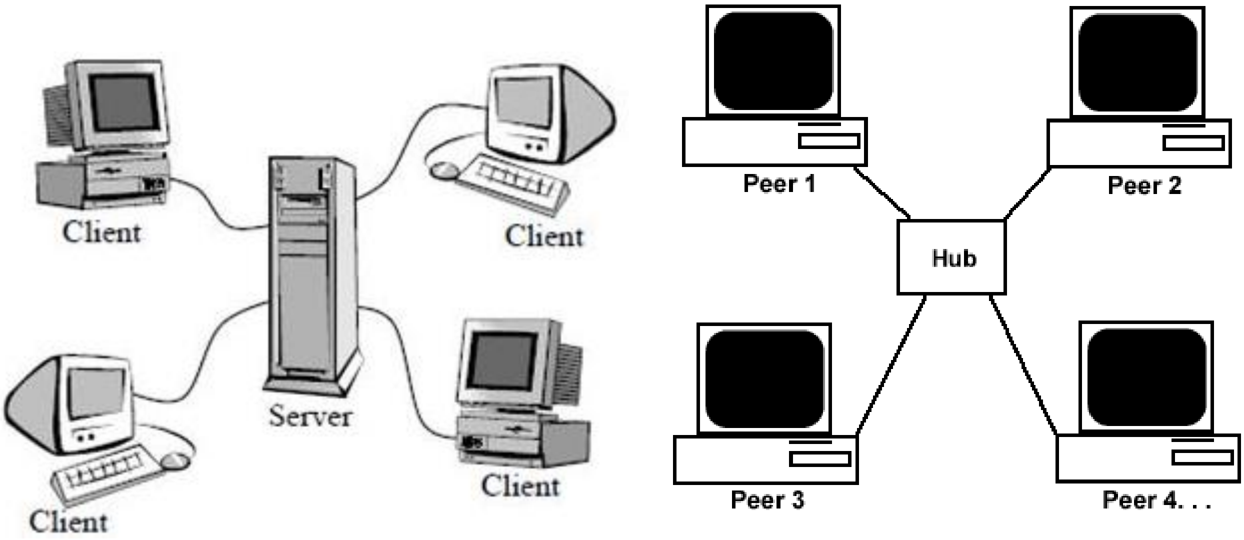
Mainframe operating systems have employed pre-emptive multitasking for decades. Desktop operating systems began to utilize this architecture starting with Windows 95 and Mac OS X.

In non-pre-emptive multitasking an application gives up control of the CPU to another application only at certain points, such as when it is ready to accept keyboard input. Under this method, one program performing a large number of calculations can dominate the machine and cause other programs to have limited access to the CPU. For example, if a communications program is running in the background and another application has appropriated the CPU, the communications program cannot keep up with the incoming data.

Also called cooperative multitasking, programs must be designed to leave the CPU to each other regularly in order to work effectively in this environment. Windows, prior to Windows 95, and the Mac, prior to Mac OS X, were non-pre-emptive multitasking operating systems.

A Network Operating System (NOS) is an operating system that is designed for a server. Normally, it is a complete operating system with file, task and job management; however, with some earlier products, it was a separate component that ran under the OS; for example, LAN Server required OS/2, and LANtastic required DOS.

Unix, Linux, Solaris and the server versions of Windows are common network operating systems designed for use in stand-alone servers. Such products may also include a Web server, directory services, messaging system, network management and multiprotocol routing capabilities.



Client-Server and Peer-To-Peer Networks

An NOS manages concurrent requests from clients and provides the security necessary in a multiuser environment. A file sharing component is installed in each client machine that interacts with the server to share files and applications as well as devices on the network such as printers, faxes and modems.

With operating systems such as Windows 98, XP, Vista or Win7, each client machine can also be a server (this architecture is called a Peer-to-Peer Network). The OS might still be considered a network operating system, but it is more lightweight than a full-blown NOS.

In a distributed system, software and data may be distributed around the system, programs and files may be stored on different storage devices which are located in different geographical locations and maybe accessed from different computer terminals.

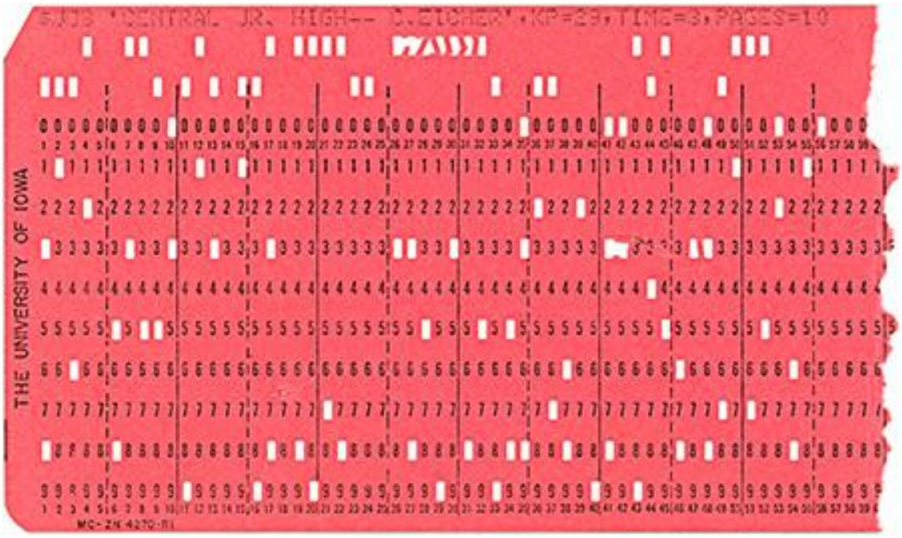
While we are mostly accustomed to seeing multi-tasking and multi-user operating systems, the other operating systems are usually used in companies and firms to power special systems.

A single user OS as the name suggests is designed for one user while a multi-user OS allows multiple users to simultaneously use the system. In the latter case the processor splits its resources and handles one user at a time. The speed and efficiency at which it does this makes it apparent that users are simultaneously using the system.

Multiprocessing supports the running of a program on more than one CPU. Multithreading allows different parts of a single program to run concurrently. The term ‘online operating system’ is reserved to systems that allow connections through communication lines. Multiprogramming is an old term and some use it to mean multitasking.

JCL (Job Control Language) is a command language for mini and mainframe operating systems that launches applications. It specifies priority, program size and running sequence as well as the files and databases used. JCL is a scripting language used on IBM mainframe operating systems to instruct the system on how to run a batch job or start a subsystem.

JCL is used for describing jobs (units of work) to the MVS, OS/390, and VSE operating systems, which run on IBM's S/390 large server (mainframe) computers. These operating systems allocate their time and space resources among the total number of jobs that have been started in the computer. Jobs in turn break down into job steps. All the statements required to run a particular program constitute a job step. Jobs are background (sometimes called batch) units of work that run without requiring user interaction (for example, print jobs). In addition, the operating system manages interactive (foreground) user requests that initiate units of work. In general, foreground work is given priority over background work.



```
$JOB 'CENTRAL JR. HIGH-- C.EICHER',KP=29,TIME=3,PAGES=10
```

This is a torn fragment of an IBM punch-card. It is a JCL control card that was put at the top of a deck of punched cards. (c1970)

One IBM manual compares a set of JCL statements to a menu order in a restaurant. The whole order is comparable to the job. Back in the kitchen, the chefs divide the order up and work on individual dishes (job steps). As the job steps complete, the meal is served (but it has to be served in the order prescribed just as some job steps depend on other job steps being performed first).

JCL statements mainly specify the input data sets (files) that must be accessed, the output data set to be created or updated, what resources must be allocated for the job, and the programs that are to run, using these input and output data sets. A set of JCL statements for a job is itself stored as a data set and can be started interactively.

4.3 The Main Functions of an Operating System

The main functions of an operating system are:

- Process control
- Memory management
- Protection and security
- User interface
- File Management
- Interrupts Handling

4.3.1 Process Control

4.3.1.1 What is a Process?

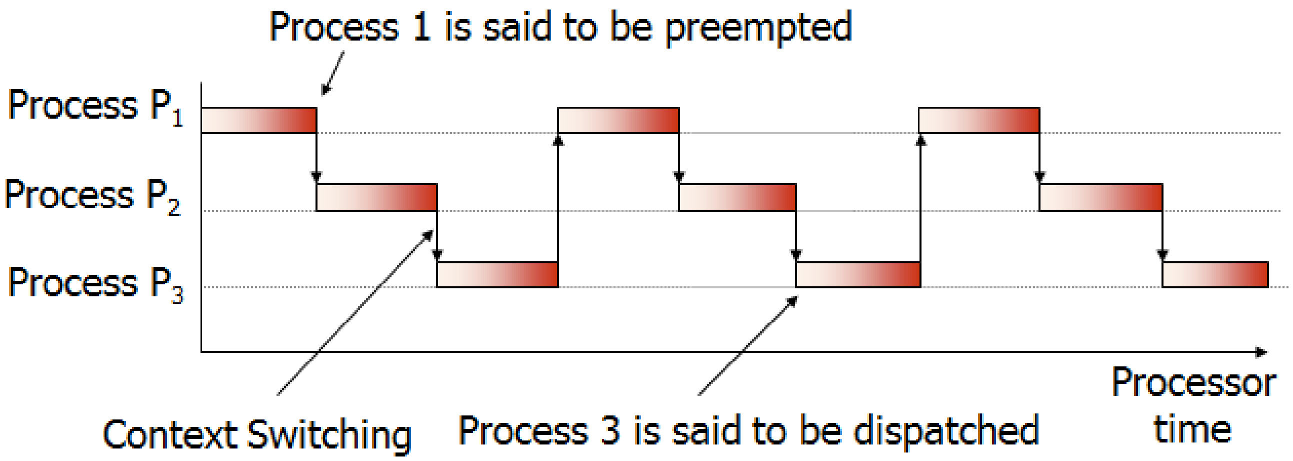
An important concept in operating systems is that of a Process. A process is a program during execution i.e. while a program is a sequence of instructions a process includes temporary intermediate values, the values in the CPU registers, the state of the process etc.

When a machine can have more than one process running at the same time, the system is called a multiprocessing system. A system can have more than one processor, but the number of processes running on a machine is generally greater than the number of processors.

4.3.1.2 Process Management

In a system where there are more processes than processors the operating system decides which processes use which processors and which processes have to wait. This is called process management or process control.

In the diagram below the graph shows how the processor's time is distributed between three processes.



CPU sharing by three processes

Pre-emption in the above diagram simply means that Process P_1 is being interrupted by the OS so that process P_2 starts (or continues) execution.

A context switch is the process of storing and restoring the state (context) of a CPU so that execution can be resumed from the same point at a later time.

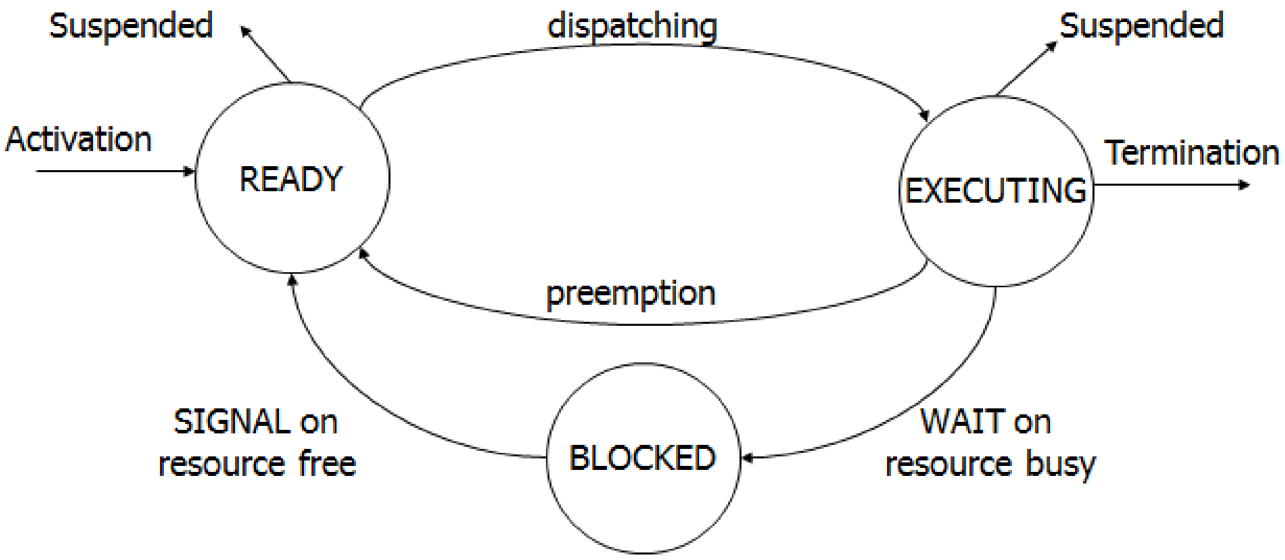
Dispatching in diagram 8 means sending (a process) to be executed.

Note that pre-emption, context switching and dispatching occur every time a process is stopped and another is given access to the CPU.

4.3.1.3 States of a Process

A process can be in one of three states:

- Executing
- Ready (waiting)
- Blocked (suspended)



The three states of a process

Moving from one State to Another	**Executing**	**Ready**	**Blocked**
Executing	-----	Pre-emption stops temporarily a process.	The process stops because it is waiting for an event to happen e.g. input.
Ready	The dispatcher sends an order for the execution of the process.	-----	-----
Blocked	-----	The event that the process was waiting for to continue has happened.	-----

Table showing reasons why a process changes state.

The dispatcher is an OS module that selects the next process for execution. The dispatcher should be as fast as possible, since it is invoked during every process switch. The time it takes for the dispatcher to stop one process and start another running is known as the dispatch latency.

4.3.1.4 Scheduling

Scheduling involves the timing of when to introduce new processes into the system (activation) and the order in which processes should run or be stopped (dispatching and pre-emption).

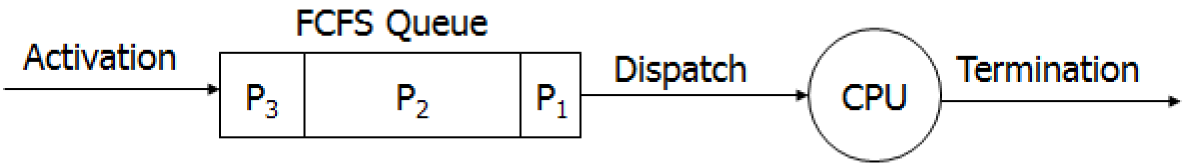
Given a set of processes (or tasks) = {P_1, P_2, P_3 … P_n}, a scheduling algorithm involves a plan for assigning the CPU to each process. Among the most popular scheduling algorithms which exist are:

- First Come First Served (FCFS).
- Priority Scheduling.
- Round Robin (RR).

4.3.1.4.1 First Come First Served (FCFS)

The first-come-first-served scheme is the simplest algorithm and it is implemented by keeping a simple FIFO queue, with new processes inserted at the tail of the queue. Characteristics of this algorithm:

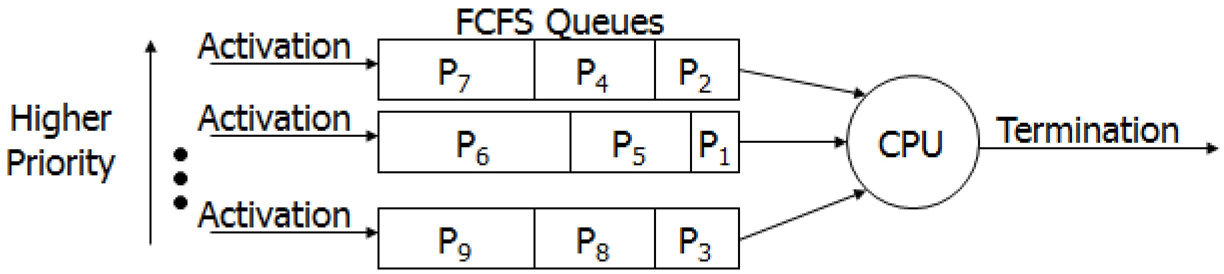
- Non Pre-emptive.
- Used mainly for batch non-interactive processes.
- Not used in modern operating system since it offers no real concurrent processing.



The FCFS scheme

4.3.1.4.2 Priority Scheduling

In priority scheduling each process is assigned a priority and the process with the highest priority is dispatched first. Processes with the same priority are dispatched on a FCFS bases. One problem in this type of scheduling is starvation, (i.e. where a process with low priority waits to be dispatched for an indefinite long time, due to other higher priority processes). One method to avoid starvation is by using aging in which the priority of a process is increased dynamically while waiting in the queue.



Priority Scheduling

There are two versions of priority scheduling – the pre-emptive and the non-pre-emptive.

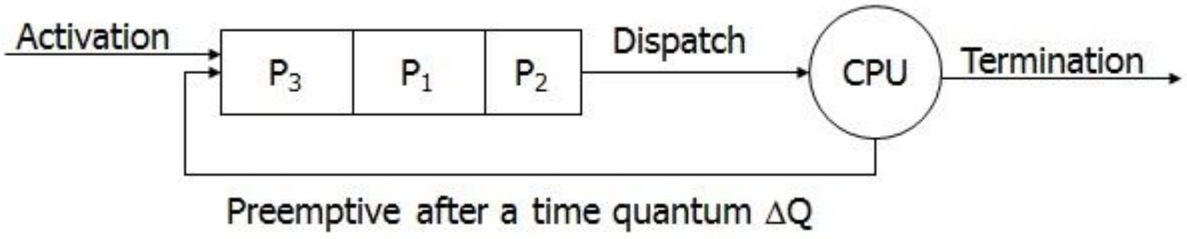
4.3.1.4.3 Round Robin

A round robin is an arrangement of choosing all elements in a group equally in some rational order, usually from the top to the bottom of a list and then starting again at the top of the list and so on.

In computer operation, one method of having different program process take turns using the resources of the computer is to limit each process to a certain short time period, then suspending that process to give another process a turn (or "time-slice"). This is often described as round-robin process scheduling.

In sports tournaments and other games, round-robin scheduling arranges to have all teams or players take turns playing each other, with the winner emerging from the succession of events.

This is particularly suitable for timesharing systems. Each process is allowed execution for a certain quantum of time ΔQ.



Round-robin process scheduling.

4.3.1.5 Deadlocks

Deadlock occurs when a process is waiting for an event that will never happen. As an example, consider the two processes A and B in the next diagram. Process A needs resource Y (held by B) to continue and B needs X (held by A). So, A and B are both waiting for each other and they can never continue their execution. This situation is called a deadlock.

Process A

.
.
.
gets resource X;
.
.
.
wants resource Y (process B has not released it yet);
.
.

Process B

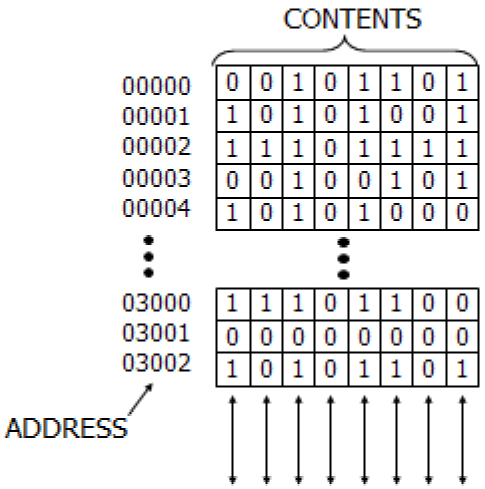
.
.
gets resource Y;
.
.
.
wants resource X (process A has not released it yet);
.
.
.

Deadlock

There are theoretical solutions to avoid deadlocks but these are costly in terms of overhead. So many operating systems do nothing to avoid deadlocks. They have a means of detecting them. When this happens a process is aborted (and thus this process releases the blocked resources) and it is then re-executed from the beginning.

4.3.2 Memory Management

Memory management (MM) deals primarily with main memory and its relation to secondary storage. The OS moves programs between disk and main memory.



00000	0	0	1	0	1	1	0	1
00001	1	0	1	0	1	0	0	1
00002	1	1	1	0	1	1	1	1
00003	0	0	1	0	0	1	0	1
00004	1	0	1	0	1	0	0	0
⋮	⋮							
03000	1	1	1	0	1	1	0	0
03001	0	0	0	0	0	0	0	0
03002	1	0	1	0	1	1	0	1

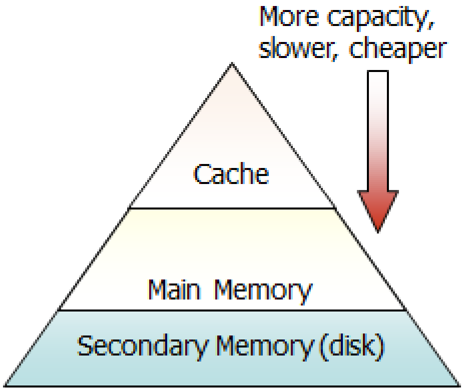
Memory Locations

Memory is a linear array of addressable locations. Main memory has direct (random) access. All locations have the same access time.

4.3.2.1 Uniprogramming and Multiprogramming

In uniprogramming main memory is divided in two parts (i) the currently executing program, and (ii) an OS kernel. In multiprogramming there are many processes (or parts of them) present in main memory. It is the job of the memory management unit to make use of the given memory so that all processes can be present in main memory when required.

4.3.2.2 Storage, Memory and Cache



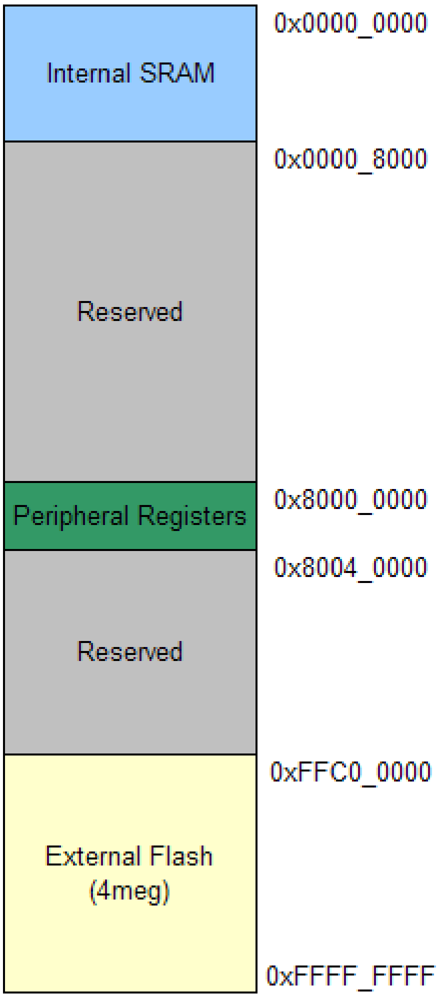
Cache, Memory and Storage

Secondary storage holds programs and data permanently but the computer requires that these be in main memory during processing. It is the job of the operating system to load programs, or parts from them, from secondary storage to main memory.

Cache is part of main memory but it is faster. The operating system sees that the programs, or parts of them, and the data mostly used by the CPU are placed in the cache to reduce overload.

4.3.2.3 Memory Maps

A memory map is a structure of data that indicates how memory is laid out. Look at the diagram “Memory Map”. It describes what is found in memory and where.

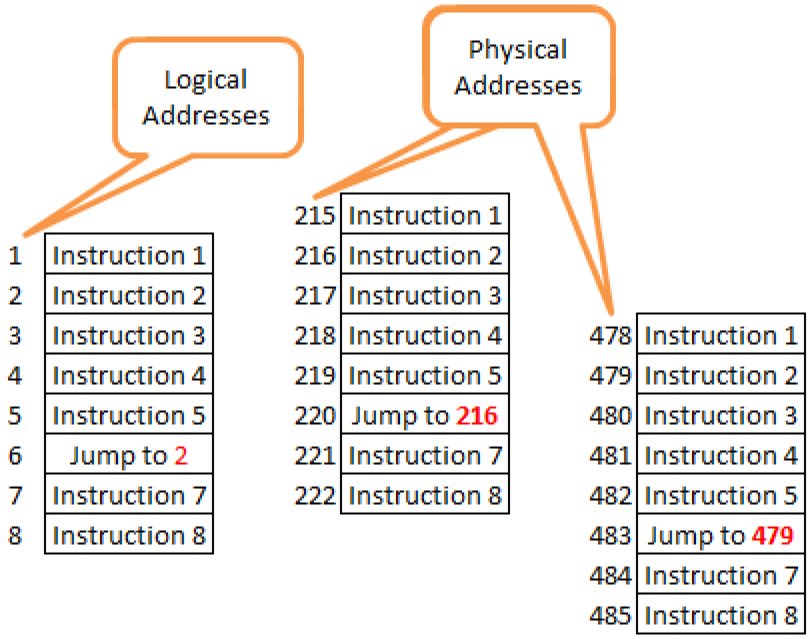


Memory Map

4.3.2.4 Logical and Physical Address Spaces

The translator (compiler, assembler etc.) does not know where the program will be placed in memory when it is executed. Furthermore, a process may be swapped in and out of memory, so it will occupy different physical locations at different times. Consequently, the translator cannot assign physical addresses to the instructions and data - it must assign logical (or virtual) addresses.

Relocation is the process of converting a logical address to a physical address. Its purpose is to let a program be executed from different areas of main memory at different times.



1	Instruction 1
2	Instruction 2
3	Instruction 3
4	Instruction 4
5	Instruction 5
6	Jump to 2
7	Instruction 7
8	Instruction 8

215	Instruction 1
216	Instruction 2
217	Instruction 3
218	Instruction 4
219	Instruction 5
220	Jump to 216
221	Instruction 7
222	Instruction 8

478	Instruction 1
479	Instruction 2
480	Instruction 3
481	Instruction 4
482	Instruction 5
483	Jump to 479
484	Instruction 7
485	Instruction 8

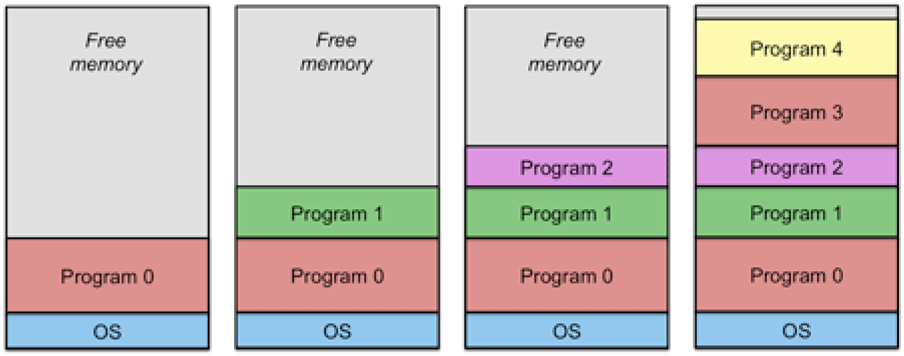
Program P

First relocation of program P

Second relocation of program P

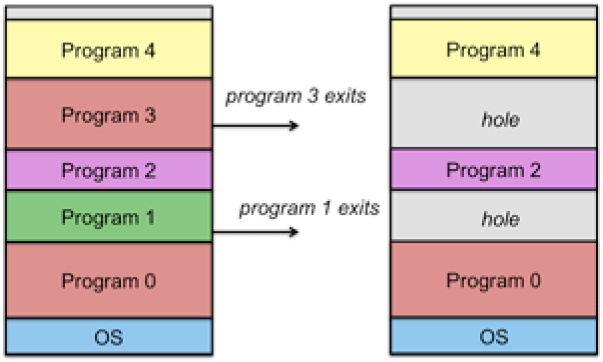
Logical and Physical Addresses

4.3.2.5 Memory Fragmentation and Compaction

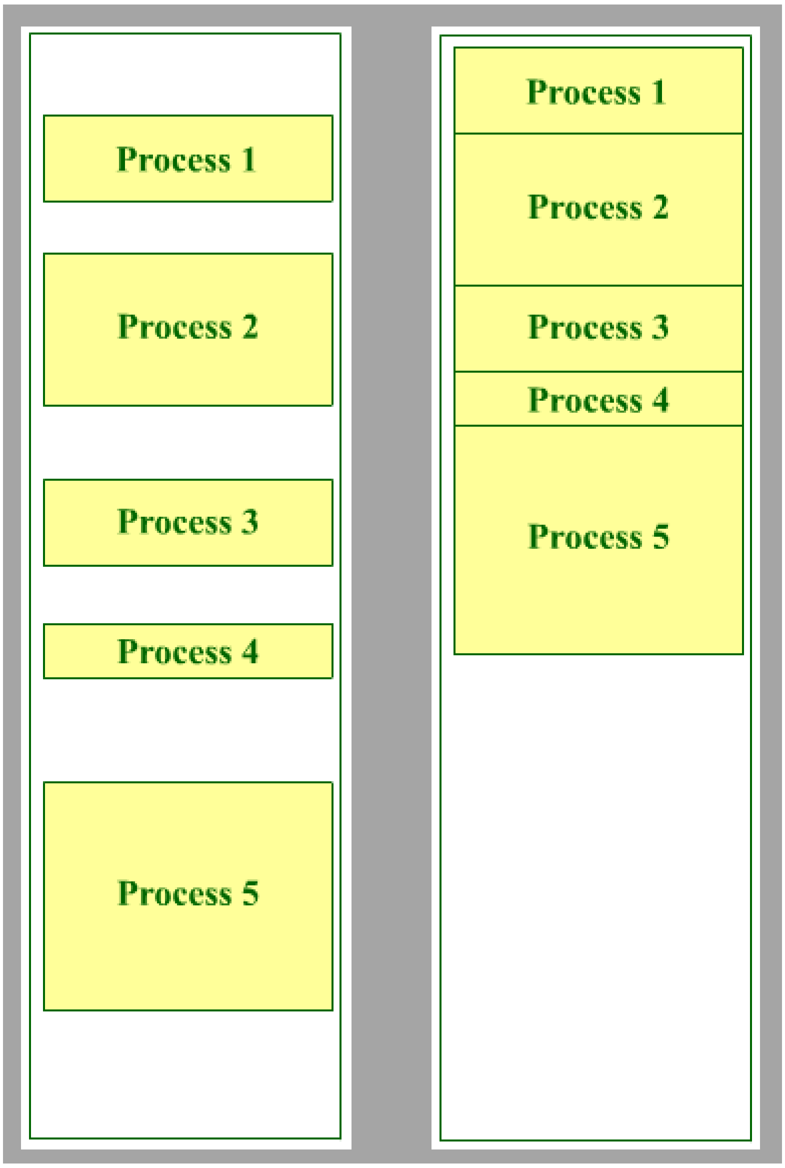


Memory holding a number of processes

The diagrams “Memory holding a number of processes” and “Closing programs” show a phenomenon called fragmentation. This means that ‘holes’ appear in memory and they are not utilized. Compaction is a process of relocating the programs so that the unutilized ‘holes’ are gathered together as one whole area. This is shown in the diagram “Before and after compaction”



Closing Programs

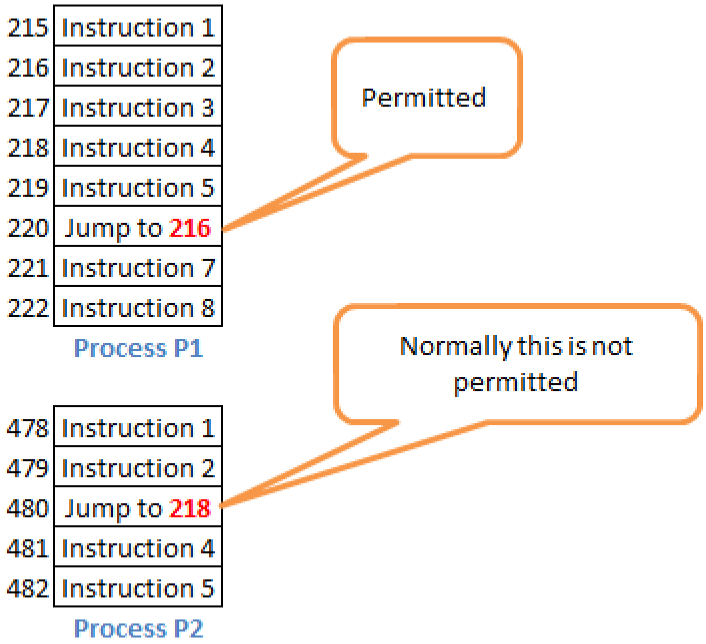


Before and after compaction

4.3.2.6 Memory Store Protection

For each process the OS keeps information in organised data structures. These include the process control block (PCB, i.e. information needed to manage processes e.g. identifier for the process, user identifier, the content of the CPU general-purpose registers, priority value etc.) and the data segment (contains variables initialized by the programmer).

The OS sees that processes do not reference memory locations belonging to other processes. Address references are checked at run time by hardware (software checks are too slow).



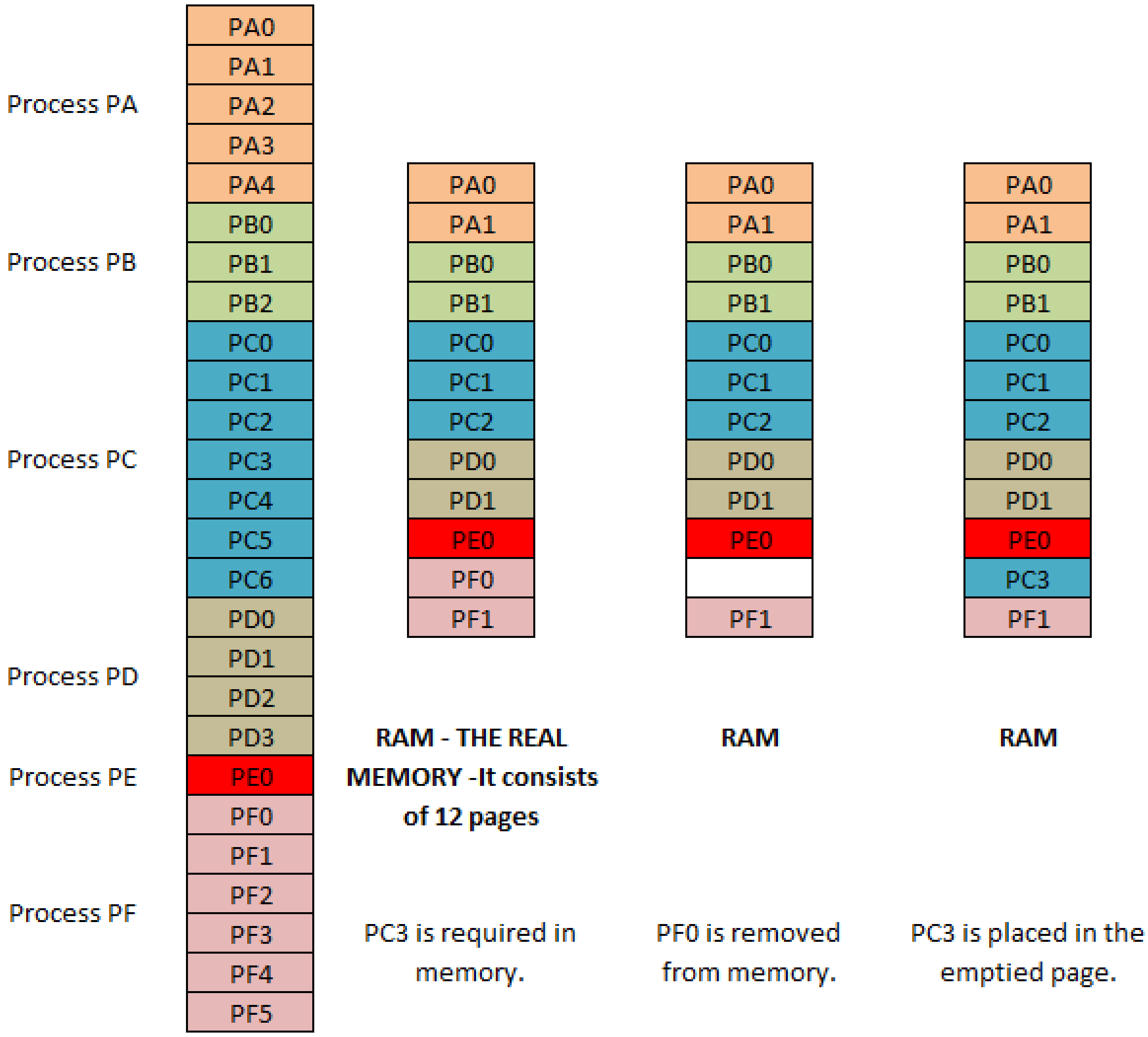
Protection in the RAM

4.3.2.7 Virtual Memory

‘Virtual memory’ means simulating more RAM than actually exists. This allows the computer to run larger programs and multiple programs concurrently. A common function in most every OS and hardware platform, virtual memory uses the hard disk to temporarily hold what was in real memory.

The RAM is broken up into smaller segments, called "pages," typically 4KB in size. When a program is opened it is not necessary that it be put all in memory. Only the parts that are being executed need to be put in the RAM. Later on if a part of a program is required to be placed in the RAM ‘swapping’ occurs. This means that a page that is not immediately required is put on the hard disk and the necessary page takes its place in the RAM.

Virtual memory dramatically increases the computer's capacity to do work. However, there is a penalty. When a user has too many open programs, there can be excessive amounts of page swapping, causing applications to slow down.



PC3 is required in memory.

PF0 is removed from memory.

PC3 is placed in the emptied page.

Running processes on the hard disk (i.e. the VIRTUAL MEMORY - 26 pages)

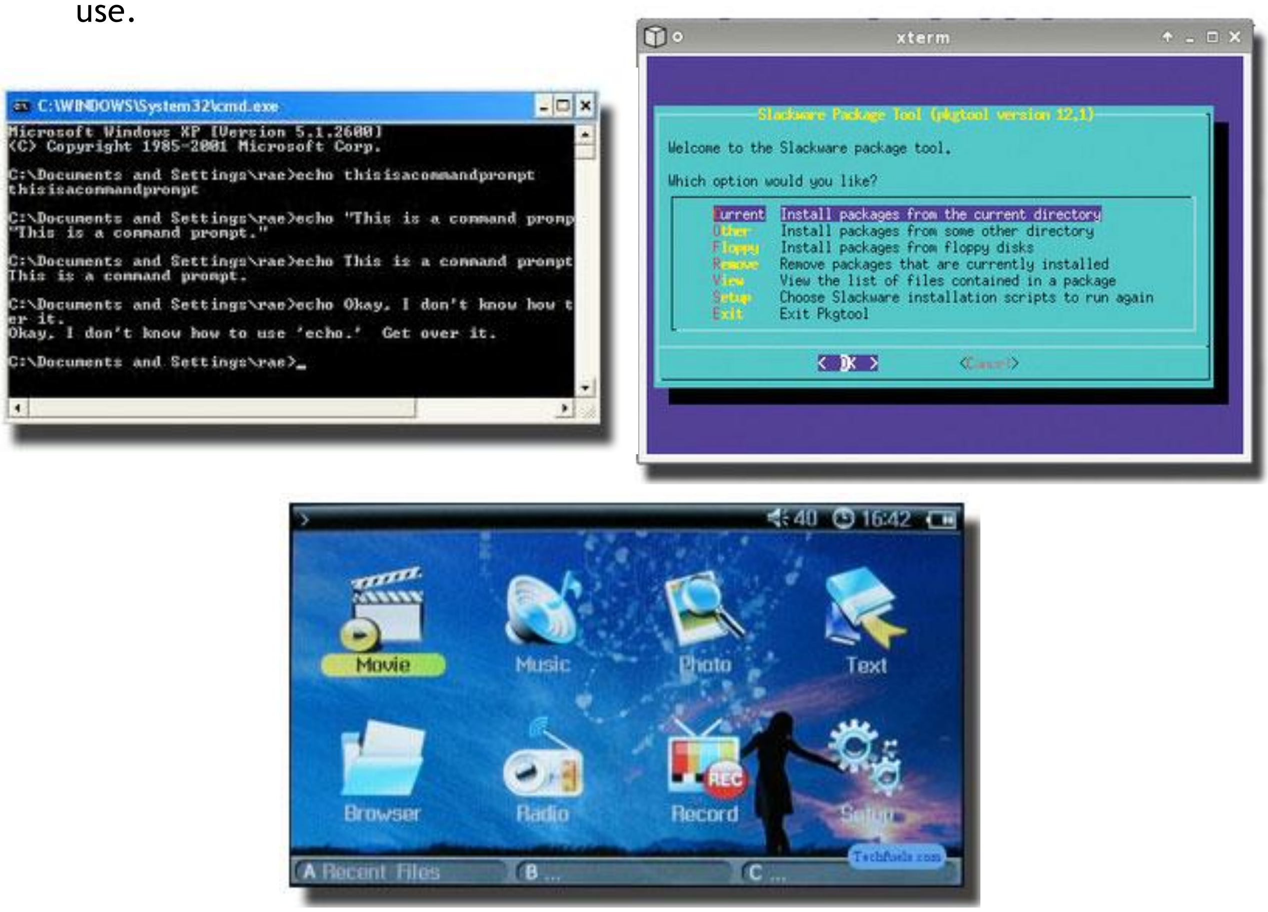
Virtual Memory

4.3.3 User Interfaces

A User Interface, abbreviated UI, is the junction between a user and a computer program. The most commonly used UIs are the following:

- Command-Driven Interface (CLI): The user types the commands from the keyboard. This is how computer interfaces started out. You must accurately type in the words that make up the instructions for the computer to complete.
- Menu-driven interface is one in which you select command choices from various menus displayed on the screen. This is the next step up from a command line interface and is considerably easier to use.
- Graphical User Interface (GUI): These use windows, icons, and pop-up menus. They have become standard on personal computers. Icons representing real

world objects can be moved round the screen and have actions performed on them e.g. drag and drop. Even novice users find these types of systems easy to use.



```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\rae>echo thisisacommandprompt
thisisacommandprompt

C:\Documents and Settings\rae>echo "This is a command promp
"This is a command prompt."

C:\Documents and Settings\rae>echo This is a command prompt
This is a command prompt.

C:\Documents and Settings\rae>echo Okay. I don't know how t
er it.
Okay. I don't know how to use 'echo.'  Get over it.

C:\Documents and Settings\rae>_
```

```
xterm
Slackware Package Tool (pkgtool version 12.1)

Welcome to the Slackware package tool.

Which option would you like?

    Current  Install packages from the current directory
    Other    Install packages from some other directory
    Floppy   Install packages from floppy disks
    Remove   Remove packages that are currently installed
    View     View the list of files contained in a package
    Setup    Choose Slackware installation scripts to run again
    Exit     Exit Pkgtool

            < OK >          <Cancel>
```

The Three Types of User Interfaces

4.3.4 File Management

The file management system provides support to help users and applications create, delete, and use files.

From the low-level perspective files are structured as byte streams. We will however look at them from a structured or high-level perspective. The file system provides one or more file structure types, each with a different access method (e.g. sequential, direct).

Files are divided into blocks. Blocks are the unit of storage both on disk and in main-memory buffers.

4.3.4.1 File Organisation and Access

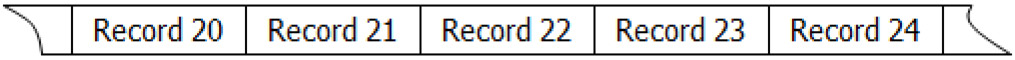
By file organization we mean the logical structure of a file (how records are organized and accessed). The basic file organisations are the following:

- serial files
- sequential
- indexed

- indexed sequential
- direct
- byte-stream files

4.3.4.1.1 Serial File

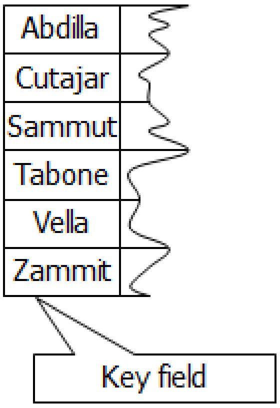
In this kind of file the records are not sorted so record access is done by exhaustive search.



A Serial File

4.3.4.1.2 Sequential File

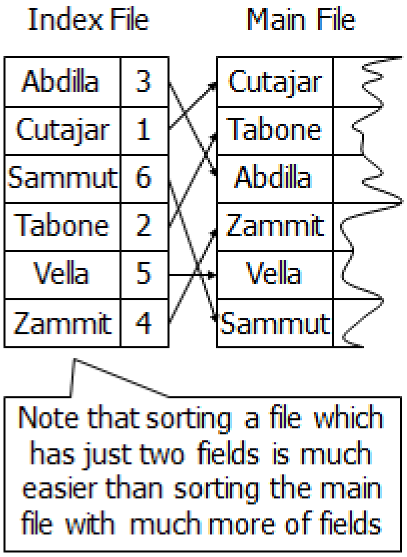
In the sequential file the records are sorted.



A Sequential File

4.3.4.1.3 Indexed File

This file system builds an index to help locate specific records within a file. Index entries consist of key value and a pointer into the file. The index may be a complex tree structure, or a simple list.



Note that sorting a file which has just two fields is much easier than sorting the main file with much more of fields

An Indexed File

4.3.4.1.4 Indexed-Sequential File

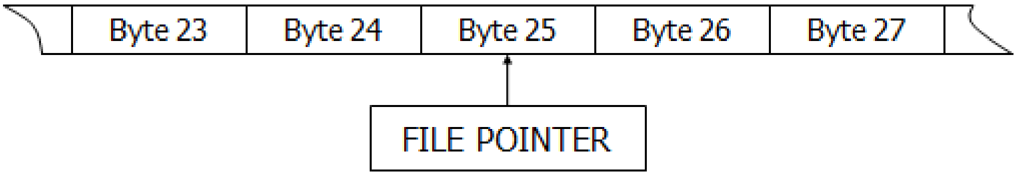
This kind of file is used sequentially however occasionally random access is required so an index is built so that this infrequent access is fast.

4.3.4.1.5 Direct Files

Direct files can be Indexed or Hashed. Indexed files can be searched on several keys: id number, name, etc. The access is direct. This organisation simplifies file updates because there is no need to restructure the file when new records are added. You just reorganize the index. A file can have more than one index.

4.3.4.1.6 Byte-Stream File Organisation

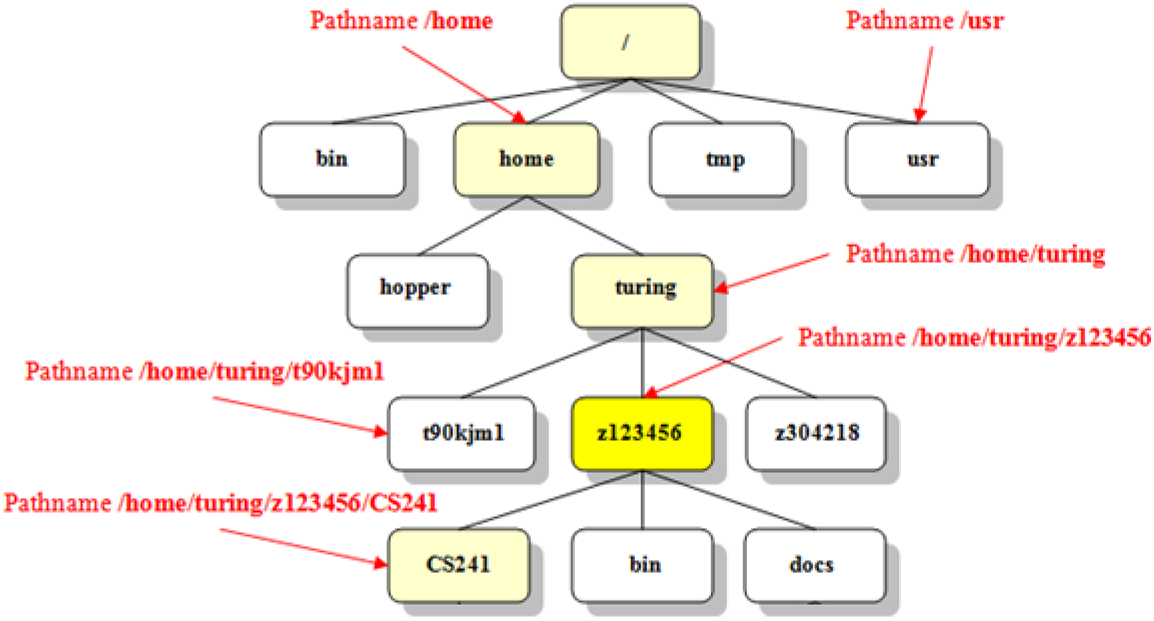
Some file systems treat files as a sequential stream of bytes. Each file has a file pointer holds the index of the next byte to be read. Some commands e.g. seek() can reposition the file pointer.



A Byte-Stream File

4.3.4.2 File Organisation

Files are organised in folders in a tree structure. A pathname describes where a file is found by going inside folders. File names are unique only within a subdirectory.



File Organisation

4.3.4.3 File Sharing

Most file systems allow users to share files, if the owner of the file agrees. File sharing introduces two issues:

- Determining access rights: who, besides the owner, can modify, execute, delete, etc.
- Managing simultaneous access: what if more than one user is trying to access a file at the same time?

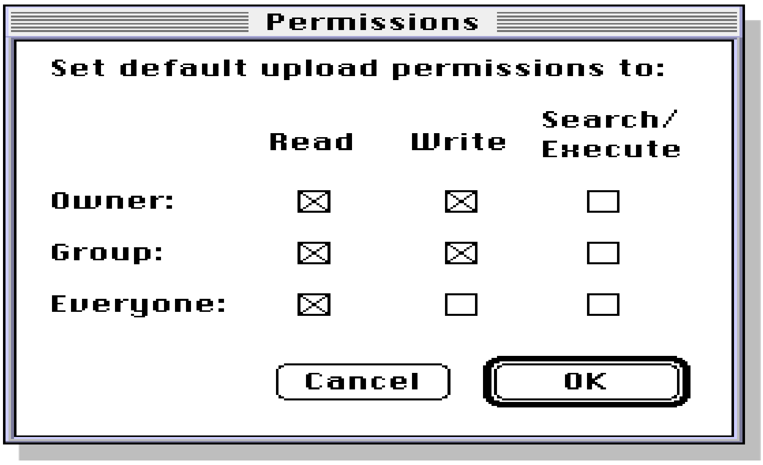
The most common access rights are:

- read: user can read and use the file, but can’t change it
- write: user can modify, delete or add to the file
- execute: user can load and run the program, cannot change or copy it

Other access rights include Append, Delete etc. These access rights are also called file attributes.

Access can be granted to different classes of users:

- individuals
- groups (a set of users, such as class members)
- all (includes everybody with access to the system - e.g., public files)



Permissions

Set default upload permissions to:

	Read	Write	Search/ Execute
Owner:	☒	☒	□
Group:	☒	☒	□
Everyone:	☒	□	□

File Permissions

4.3.4.4 Unauthorised Access

Gaining access to a website, a software program, a server, or some other type of system usually requires a password and username. If the username or password entered is incorrect, an “Unauthorized Access” message may be displayed. This means the user attempted to access the system and the authorization (username or password) was not valid and access was not granted.

Unauthorised Access

Unauthorized access could also occur if a user attempts to access an area of a system they have not permission to access.

Some system administrators set up alerts to let them know when there is an unauthorized access attempt, so that they may investigate the reason. These alerts can help stop hackers from gaining access to a secure or confidential system. Many secure systems may also lock an account that has had too many failed login attempts.

A firewall is designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both. Firewalls are frequently used to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

4.3.5 Interrupts Handling

4.3.5.1 What is an Interrupt?

In systems programming, an interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. The processor responds by suspending its current activities, saving its state, and executing a small program called an interrupt handler (or interrupt service routine, ISR) to deal with the event. This interruption is temporary, and after the interrupt handler finishes, the processor resumes execution of another process. There are two types of interrupts:

- Hardware interrupt
    - This is an electronic alerting signal sent to the processor from an external device, either a part of the computer itself such as a disk controller or an external peripheral.
    - Examples:
        - Pressing a key on the keyboard
        - Moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position.
    - Unlike the software type (below), hardware interrupts are asynchronous and can occur in the middle of instruction execution.
    - The act of initiating a hardware interrupt is referred to as an interrupt request (IRQ).
- Software interrupt
    - This is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed.
    - Software interrupt instructions function similarly to subroutine calls and are used for a variety of purposes, such as:
        - To request services from low level system software such as device drivers.
        - To communicate with the disk controller to request data that be read or written to the disk.
        - Each interrupt has its own interrupt handler.

Interrupts are a commonly used technique for computer multitasking, especially in real-time computing. Such a system is said to be interrupt-driven.

4.3.5.2 Polled and Vectored Interrupt

In a computer, a polled interrupt is a specific type of I/O interrupt that notifies the part of the computer containing the I/O interface that a device is ready to be read or otherwise handled but does not indicate which device. The interrupt controller must poll (send a signal out to) each device to determine which one made the request.

Polling is also called busy waiting. This is generally not as efficient as the alternative to polling, interrupt-driven I/O.

The alternative to a polled interrupt is a vectored interrupt, an interrupt signal that includes the identity of the device sending the interrupt signal.

4.3.5.3 Multiple Interrupts and Interrupt Priorities

A multiple interrupt is an interrupt that interrupts an interrupt handler. Two techniques to handle multiple interrupts are the following:

- Disable interrupts while an interrupt is being processed.
    - Advantage: solution is simple because interrupts are handled in strict sequential order.
    - Disadvantage: it does not take into account relative priority and time-critical needs.
- Define priorities for interrupts and allow an interrupt of higher priority to interrupt an interrupt-handler of lower priority.

The interrupt priorities define which of a set of pending interrupts is serviced first.

4.3.5.4 Interrupt Mask Register

An interrupt mask is an internal switch setting that controls whether an interrupt can be processed or not. The mask is a bit that is turned on and off by the program. An interrupt mask register is a register holding a series of interrupt masks. The following diagram shows an example of masking.

A	B	C	D	E	F	G	H	
1	0	0	0	0	1	1	0	Interrupt Vector
1	1	1	1	0	0	0	0	Interrupt Mask Vector
1	0	0	0	0	0	0	0	Masked Interrupts

Bits in this vector are formed by the AND operation on the Interrupt Vector and the Interrupt Mask Vector.