CIS-350
INFRASTRUCTURE TECHNOLOGIES

Operating Systems - An Overview (Chapter 15)

Read chapter 15 thoroughly.

Without an Operating System (OS)

- Program instructions must be loaded into memory by hand
- No user interface except for I/O routines provided with executing program
- System is idle when waiting for user input
- No facility to store, retrieve, or manipulate files
- No ability to control peripheral devices
- Can run only one program at a time; computer halts at end of each program

THE OPERATING SYSTEM (OS) CONCEPT

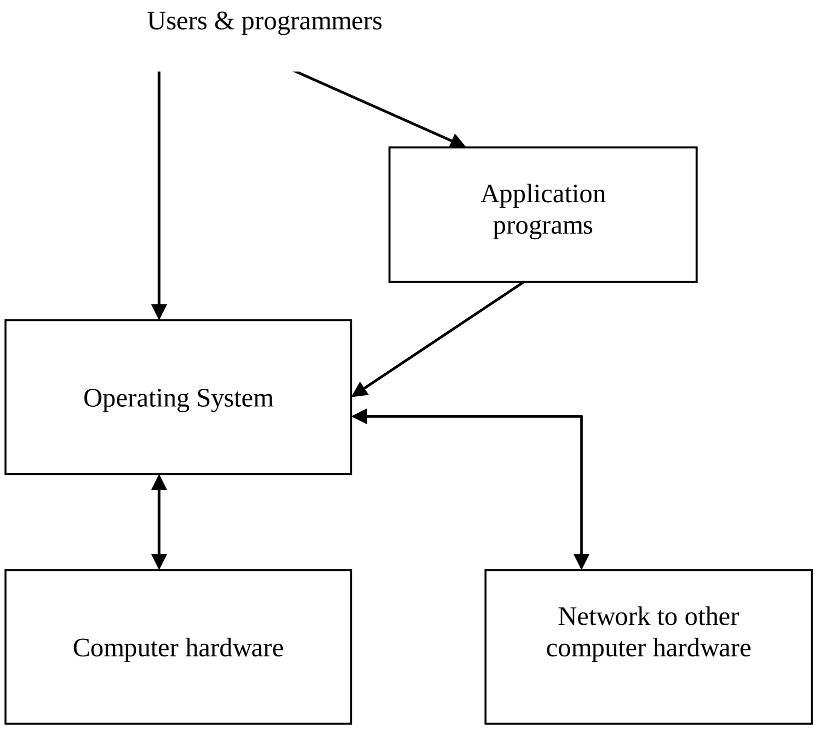
The Operating System (OS) is a program that acts as an intermediary between a user of a computer system and the computer hardware.

The purpose of the OS is to provide a convenient environment in which a user can execute programs.

There are two goals which the OS should meet:

1. make a computer system convenient to use
2. use the computer hardware in an efficient manner

A computer system can be roughly divided into 4-5 components:
- Hardware
- Software (System software - OS and Application software)
- Data
- Communication
- Users and programmers

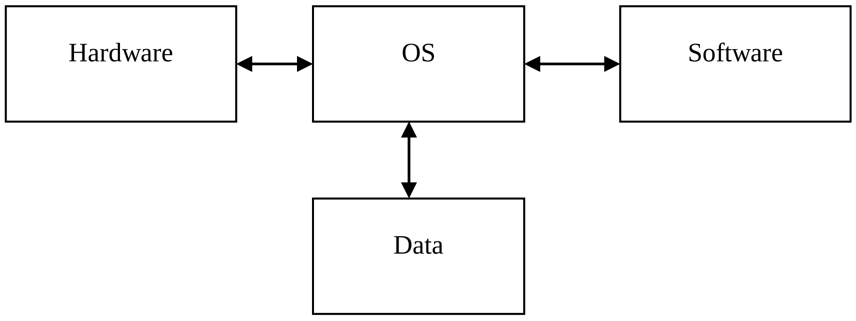


The OS controls and coordinates the use of hardware among the various programs for the various users. It manages basic hardware resources: CPU time, memory, and I/O devices.

There is a strong relationship between the OS and computer architecture. Changes in the design of hardware may cause changes in the OS and vice versus.

A computer system has many resources:

1. <u>hardware</u>
2. <u>software</u>
3. <u>data</u>



These resources are required to solve a problem. <u>The OS manages the allocation of:</u>

1. <u>hardware</u> (CPU time, memory space, file storage space, I/O devices)
2. <u>software</u> (editors, interpreters, compilers, linkers, utilities, libraries)
3. <u>data</u>

<u>to specific programs and users.</u>

There may be many, possibly conflicting, requests for resources, so the OS has to decide which requests are allocated resources to operate a computer system efficiently and fairly. The OS has to decide when, to whom, and in what amount these resources should be granted.

Some examples of the OS: DOS, UNIX, Linux, IBM/z System (former OS/390 & MVS/JCL), Windows (1.0, 3.1, 3.11, 95, 98, NT, 2000, XP, Vista, 7), MAC OS.

The OS can also be viewed as a control program that controls the various I/O devices and user programs. The hardware alone is not very friendly and easy to use, so the OS is an interface between the users, application programs and the hardware.

The OS is not hardware. It is a set of software routines.

To summarize, communicating directly with the computer hardware is extremely tedious. The OS is a set of system software routines that provide an interface between hardware and application software and that manage system resources. The resources include hardware, software, and data. The OS sits between these basic resources.

The OS also provides more specific services in the following areas:

(1) Facilitates program creation, compiling and link-editing

The OS provides a variety of facilities and services to assist the programmer in:

- creating and modifying programs (editors – Unix: vi, emacs, pico, ee)
- compiling or interpreting them (compilers, linkage editors, interpreters)

(2) Manages, loads and executes programs, and supervises their execution

- loads instructions and programs into main memory
- initializes devices and files
- prepares/allocates other resources such as the CPU time, using a software module called a dispatcher

(3) Accepts and executes the requests from the user commands (DOS - *dir* command, UNIX - *ls* command) and from the user's programs (C++*/Java/C#: (1) open a file ("infile.dat"); (2) read from the file: x1, x2, x3, …; (3) close the file.*

(4) Manages hardware resources of the computer

(5) Supervises program execution

(6) Provides I/O services and accesses I/O devices

- each I/O device requires its own peculiar set of instructions. The OS takes care about the details so that the programmer can think in terms of simple reads and writes. The OS loads automatically I/O device driver programs for each device installed on the system: plug-and-play. Nowadays, most common device drivers are part of the OS kernel's module.

(7) Controls access to files - control must include not only the nature of the I/O device (disk, tape) but also the file format on the storage medium.

(8) Identifies users
- user id
- password

Reasons:

security - unauthorized access to computer resources should be denied
accounting – keep track of what users use how much, and what kinds of resources (CPU time, time in memory, disk or tape spaces used) for billing and statistics
setting priorities and limiting access

(9) Identifies programs

Programs, compilers, interpreters, editors, loaders, linkers, and other OS utilities are assigned names.

Ex. UNIX on the HPCC at U of L:

$> ***vi*** *prog1.c* or

$>***ee*** *prog1.c* invokes the vi or ee editor

sort *–nr random.dat > randsor.dat &*

(10) Enables run-time intervention

- if the program is in an endless loop or produces wrong results, the user can press CTRL-C keys to halt the program, Visual C++ , Java, C#: Windows (98, NT, 2000, XP, Vista, 7). The user can cancel his/her own program.
- if the program exceeded the CPU time assigned to it, the time out OS routine kicks in

(11) Responsible for security and protection

- no one can access/modify your files unless you or a system administrator gives permission
- all services such as file management or I/O requested by a program or a user must be done by the OS
- protects memory areas allocated to OS routines from being used by application programs

(12) Provides a means for starting the computer. This process is known as bootstrapping or Initial Program Load (IPL).

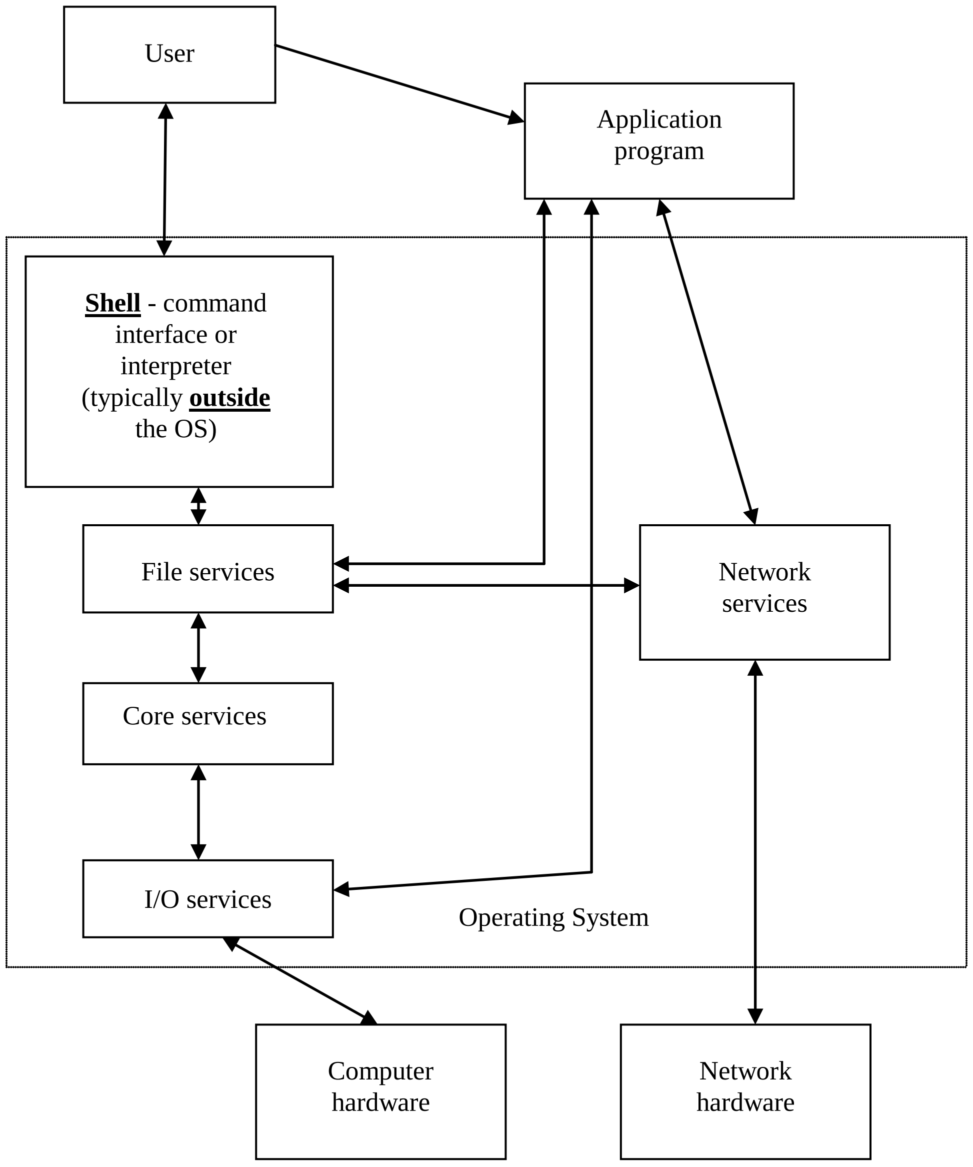
(13) Provides means for concurrent processing and multiprocessing.

To support concurrent processing, the OS has to allocate resources (memory, CPU time, I/O devices) to programs and users, and protect users and programs from each other as well as to make communication between programs possible.

(14) Handles all interrupt processing, including error handling and recovery

(15) Provides services for networking

A simplified diagram of operating system services is presented below.

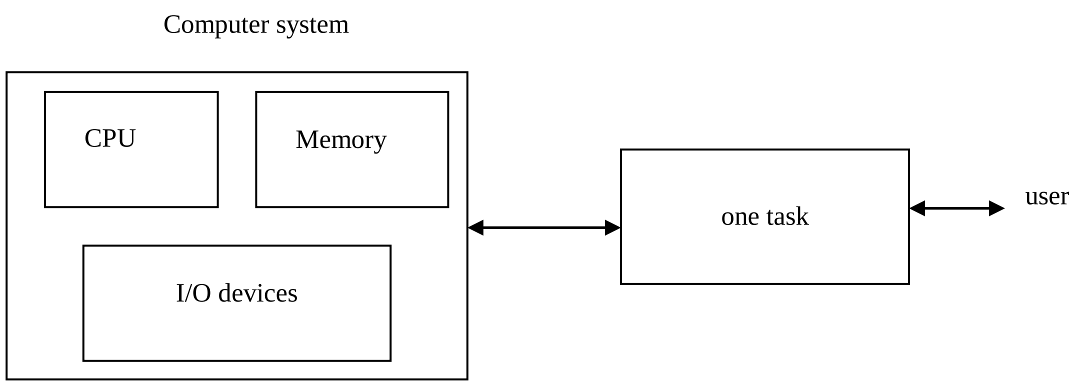


Core services include such services as memory management, CPU time scheduling, process and thread management, security, and interprocess communication.

Types of the OS:

1. ENIAC computer, 1945 - did not have the OS at all.

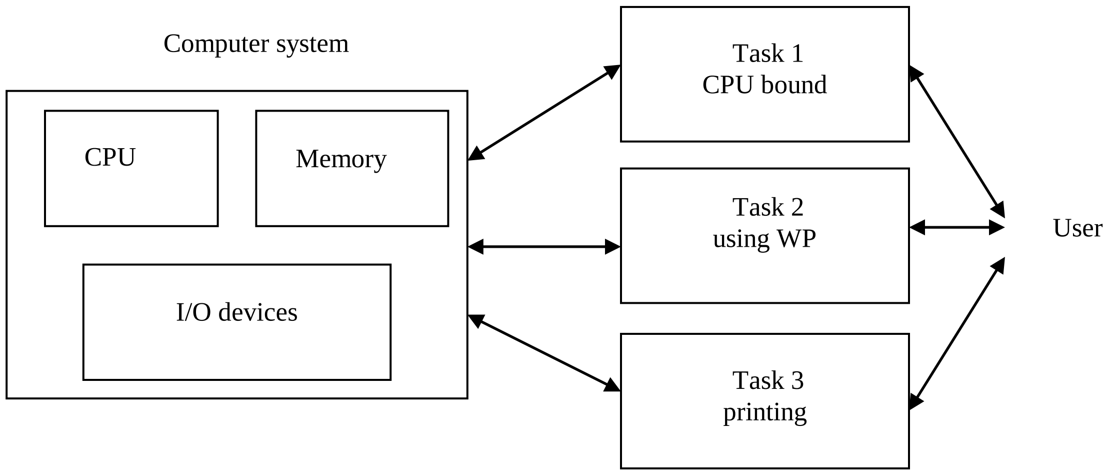
2. Single-tasking OS, 1950 - only one activity or task can be in progress at a time, DOS.



- one CPU – limited need for scheduling (one task, one user)
- limited memory management
- limited number of peripheral devices
- lacking security
- obviously processes interrupts
- in general, waste the system resources (CPU is idle much of the time)

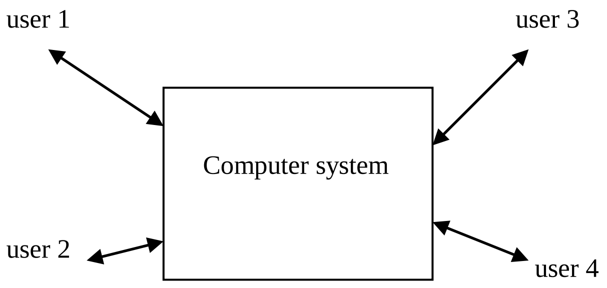
The OS will be simpler since it will have fewer modules.

3. Multi-tasking (single-user OS), 1970, Windows 95 through Windows 7



- has several activities in progress at a time (over the same period of time)
- printing runs in so called background
- CPU, memory is shared by two or more tasks
- the OS switches between tasks very quickly
- the OS is more complex than for single-tasking systems.

4. Multiuser (multiprogramming/multiprogramming OS), 1970: UNIX, IBM/z System (OS 390, MVS/JCL)



- each user may run multiple programs
- the OS is even more complex because it has to manage resources among the various programs for the various users.

5. Interactive OS, 1980, IBM/z (VM/CMS, TSO), Unix

   A user issues a command/request and awaits fast system response. A user can directly interact with the program to provide input data and guidance during program execution.

6. Batch OS, 1960, IBM/z (MVS/JCL, OS/390), Unix

   The data input for the program is collected together into a file on disk or tape. The user submits the program(s) or job(s) to the computer for processing - batch processing. The user does not interact with the program during batch processing.

7. Real-time OS, virtual machines (VM/CMS), distributed OS (client/server systems, Windows NT/2000/XP/Vista/7, Unix).

The OS should make a computer system easy to use and efficient. These two requirements are often contradictory, and typically much more stress has been put in to efficiency when the OS was designed or written (UNIX) -1970s. The OS became much more user friendly when the Graphical User Interface (GUI) concept was introduced in the 1980s.

Early computers did not have the OS at all. Today, all computers have the OS. On IBM mainframe computers, the OS multi-user multitasking/multiprogramming IBM/z System (OS/390, MVS/JCL) is a very sophisticated set of programs that the mankind has ever invented. The OS programs including utilities may occupy hundreds of megabytes (MB) of space. DOS, a single-user operating system, ran in 64KB of memory. IBM 370 needed at least 6MB of memory. Windows 7 requires 1GB RAM and at least 1GHz 32-bit processor.

When the computer is turned on, the memory content is empty or contains garbage; it does not contain user programs or OS. The OS is loaded by the bootstrap loader at start-up time. The bootstrap is stored in ROM and reads the rest of the OS routines.

The OS is divided into resident and nonresident parts. The memory resident components of the OS are commonly known as the kernel of the OS. For example, the routines that manage interrupts, allocate memory or dispatch the CPU are always resident because they are critical to the operation of the system. However, commands/routines that are used occasionally (such as the Format or Sort commands) are transient. Transient or nonresident routines are stored on hard disk.

The OS is event driven. This means that the OS sits idle and executes only if some event occurs that requires OS action. Events result from interrupts or from service requests by a program or a user. Events include file requests, I/O, keyboard input from the users, memory requests from programs, messages sent from one program to another, clock interrupts, etc.

The OS is a very important part of every computer system. All computers regardless the price and magnitude have the OS.

Most modern hardware vendors (IBM, Dell, Compaq) do not provide their own brand OS at all. Instead, their systems are supplied with a standard OS (UNIX/Linux or Windows (XP, Vista, 7) (Microsoft). UNIX can operate on different hardware platforms. Windows OS can operate on the IBM personal computers and their clones. The original version of UNIX OS has been written almost entirely in C language and can be ported easily to a new machine by recompiling the C code. Many modern operating systems are written in C, C++ or Java – they are often called system languages because they provide good facilities to interact with the hardware. Only small portions of the OS - the routines which communicate directly with the computer hardware are written in assembler. When the OS is to be ported to a different hardware platform, only portions of the OS written in assembler need to be rewritten. The part of the OS written in C, C++ or Java remains unchanged.

CONCURRENT OPERATIONS / PROCESSING, MULTIPROCESSING

A CPU can process only one instruction at a time. The simultaneous execution of two or more programs is obviously impossible with a single CPU. However, a CPU can execute concurrent programs, if the OS properly manages/controls them.

On a multitasking or multiprogramming system, a user can execute many programs concurrently on a single CPU.

A multiuser system allows each user to execute at least one program. In multiuser environments, many programs execute concurrently and many users use the system simultaneously.

Concurrent execution of programs vs. simultaneous execution of programs

The word "concurrent" means over the same period of time. Simultaneous means at the same time.

Multiprocessing - Actual simultaneous processing of multiple programs using either multiple CPUs or multiple CPU cores.

Users and programs compete for computer resources such as:

- processor's time
- memory space
- I/O devices

Different conflicts are inevitable. The OS has to resolve them in an efficient and fair manner. It has to decide to whom, in what amount, when and for how long these resources should be granted.

Multiprogramming/multitasking/multiuser is a technique in which we keep the CPU as busy as possible. The CPU utilization may vary from 0-100%. The CPU time can be utilized well (80-90%) when the CPU executes several programs concurrently. The CPU should never be idle (Figures 15.3 and 15.4, pp. 494-495).

Two simple strategies for concurrent processing (Figures 15.3 and 15.4):

- sharing the CPU during I/O breaks
- time-sharing the CPU

The process of selecting which program to run at any given instant is known as dispatching.

The OS is much more complex than the one for single-user systems. In addition to file management, dispatching, processing interrupts and other single-user OS functions, the OS

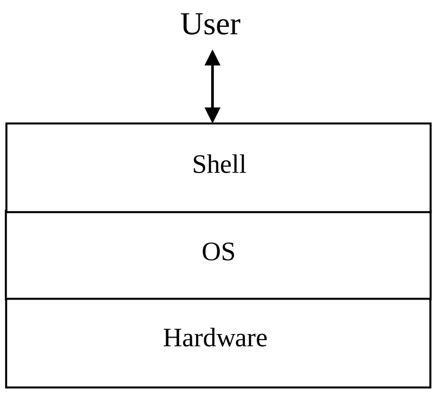
- must keep track of each program in the machine,
- must find and manage memory for each program,
- must schedule various I/O devices,
- must suspend and restart programs with all registers, data and instructions intact,
- must provide security and protection from other users and other programs, and
- perform a variety of resource management tasks
- prevent deadlocks from occurring
- perform spooling

Examples: IBM/z System, Unix, Linux, Windows (NT, 2000, XP, Vista, 7), MAC OS

SELECTED SERVICES AND FACILITIES

1. User Interface and Command Execution Services

To the user, the most important and visible service provided by the OS is the user interface and the capability that it provides to execute commands. The user interface is a separate shell that is usually not part of the OS. The user communicates with the OS through a shell. DOS's command processor is a shell.



The shell interprets commands issued, and the shell calls various OS modules to access hardware.

<u>Standard shells</u>:

- command processor
- no help on how to enter commands (you would have to memorize the syntax of commands or refer to the manuals)

<u>Custom shells</u>:

- user friendly (icons, menus), graphical user interface (GUI) (you do not have to remember commands)
- have the ability to work with more than one program at a time
- occupy more memory
- may be slower

In UNIX, there are three popular shells:

- the <u>Korn shell</u>,
- the <u>Bourne shell</u>, and
- the <u>C shell</u>.

Each of these shells provides different command structures and different capabilities.

The types of user interface:

- <u>Graphical User Interface</u> (GUI),
- <u>Menu-driven Interface</u>, and
- <u>Command Line Interface</u> (CLI).

GUI accepts commands primarily in the form of

- drop-down menus,
- mouse movements, and
- mouse clicks.

CLI relies on typed commands.

The OS allows for combining individual resident and nonresident commands into programs – a sequence of control statements.

- On the IBM mainframe in IBM/z (MVS/JCL or OS/390) OS, the set of commands/control statements is written in the <u>Job Control Language</u> (JCL). These control statements are interpreted and executed, one a time, to control the processing of a multistep job.

  Ex.

  ```
  JOB parameters
     EXEC PROG1 parameters
            DD define input file, parameters
            DD define output file, parameters
     EXEC PROG2 parameters
            DD define input file, parameters
  ```

```
            DD define output file, parameters
    …….
            (one DD statement for each input/output file)
END JOB
```

- In MS-DOS and Windows, one can group commands in <u>batch files</u> (BAT files) or Windows Powershell. (See labs #1 & #2 on DOS.) In addition to simple commands, batch files provide branch and loop commands, piping and redirection to name a few.

  Batch files are used for repetitive operations. If you create a batch file named MYFILE.BAT containing:

  ```
  command 1
  command 2
  command 3
  command 4
  ```

  and type the name of the batch file from the C> or E>, the commands from 1 through 4 will be carried out.

  ```
  C>MYFILE            or            E>myfile
  ```

  Some batch files, such as AUTOEXEC.BAT are executed automatically when DOS is booted.

- In UNIX and Linux, one can group commands in a <u>script file (shell script)</u>. They operate like MS-DOS and Windows batch files. (See labs 4 and 5 on Unix)

<u>2. Managing Processes</u>

Every executed program (the OS routine or application program) with the resources assigned to it is treated as a <u>process</u>. The resources include (memory, I/O devices, time for execution, contents of registers, and the like).

The list of resources used by the program is stored in the table known as <u>Program Control Block</u> (PCB). Each program has its own PCB. The OS synchronizes processes and must keep track of each process in memory. It also determines the state of each process: whether it is running, ready to run, or waiting for I/O to be completed.

Many modern systems break the process down into smaller units called <u>threads</u>. Each thread can be executed independently.

<u>3. Processing Interrupts</u>

The processor and main memory are synchronized by the processor’s clock. Interfaces and peripheral devices function independently and are <u>asynchronous</u>. The processor works as a supervisor and can always initiate the link with the peripheral device.

<u>To get the processor’s attention, the peripheral device sends the electronic signal called an interrupt which is sensed by hardware</u>.

The processor responds to an interrupt as follows:

1. It saves the control information needed to resume the current program in the PCB

2. It transfers control to the OS interrupt handling routine

3. The interrupt handling routine is executed to perform a requested service.

4. When it finishes, the OS reads the PCB and returns control back to the application program

<u>A hardware interrupt</u> originates in the computer hardware (for example, in a peripheral device. <u>Interrupts are detected (sensed) by hardware and handled by the OS software</u>. Hardware interrupts occur unexpectedly.

<u>A software interrupt</u> does not occur unexpectedly. It is intentionally requested by a program that needs a service of the OS routine.

We will be again discussing interrupts, their origins, and mechanisms how they are accomplished later in the course. Interrupts are very common and occur in the computer system all the time.

For example,

1. if you strike a key on your keyboard you are issuing an interrupt to your computer system.

2. your erroneous program may issue an <u>exception (program) interrupt</u> because it attempts to divide over 0, or tries to handle too large number (real overflow).

3. if an interrupt comes from an operator of a computer system it is called an <u>operator or external interrupt</u>

The mechanism through which an interrupt is accomplished is similar to calling a subroutine (function) from a program.

<u>4. Managing Memory</u>

Techniques:

- <u>fixed partition</u>
    - memory is divided into <u>equal</u> partitions (128KB each)
    - <u>simple</u> concept but memory space is <u>wasted</u>

- <u>dynamic</u>
    - memory becomes <u>fragmented</u>

- <u>segmentation</u>
    - segments are loaded into <u>noncontiguous</u> fragments in memory
    - the OS has to maintain the <u>beginning address</u> of each segment

- <u>paging</u>

- <u>segmentation and paging</u>

- <u>virtual memory</u>

<u>5. File System Management</u>

- File - logical unit of storage
- Basic file management system provides
    - Directory structures for each I/O device
    - Tools to copy, move, store, retrieve and manipulate files
    - Information about each file in the system and the tools to access that information
    - Security mechanisms to protects files and control access

    - Achieved by Managing File Directory and File Allocation Table

- Additional file management features
    - Backup, emergency retrieval and recovery
    - File compression
    - Journaling
    - Transparent network file access
    - Auditing

<u>6. Secondary Storage Management/Scheduling</u>
- Minimizing the disk seek time (see notes for chapter 10)

<u>7. Network and Communications Support Services</u>

<u>8. Scheduling and Dispatching Programs</u>

Two levels of scheduling:

- <u>high-level</u> - jobs are admitted to the system first, next they are placed in a queue, and then they are allocated memory

- <u>low-level</u>, called dispatching - allocating CPU time to programs

The OS module that manages the CPU time is called a <u>dispatcher</u>.

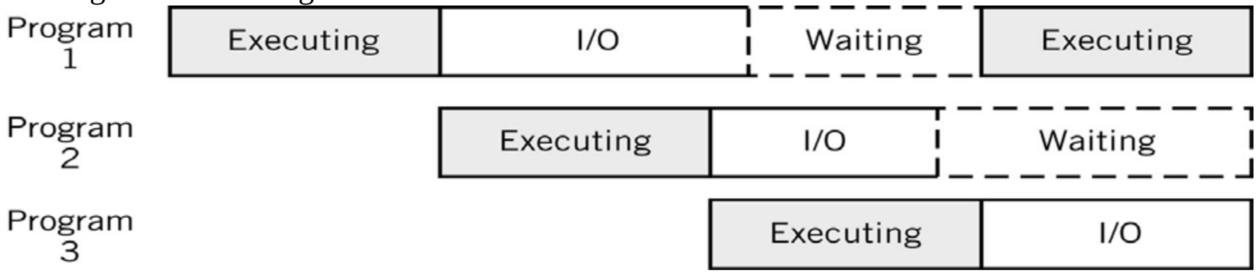
Dispatching concerns programs that are already in memory, not on disk.

Programs have priorities. The dispatcher scans the Program Control Block (PCB) table and determines which program is to be given the CPU time. If several programs are ready to execute, <u>the dispatcher is polling all programs PCBs in memory and decides which one should be granted the CPU time</u>.

A nonpreemptive system - the program is allowed to run until it is completed (voluntarily gives up the CPU) or blocked (issues I/O request). Used in older systems – batch systems. Figure 15.3, p. 494.

I/O bound programs do not consume too much of the CPU time. As a result, the surrender voluntarily control to the processor. They can execute in nonpreemptive environments.

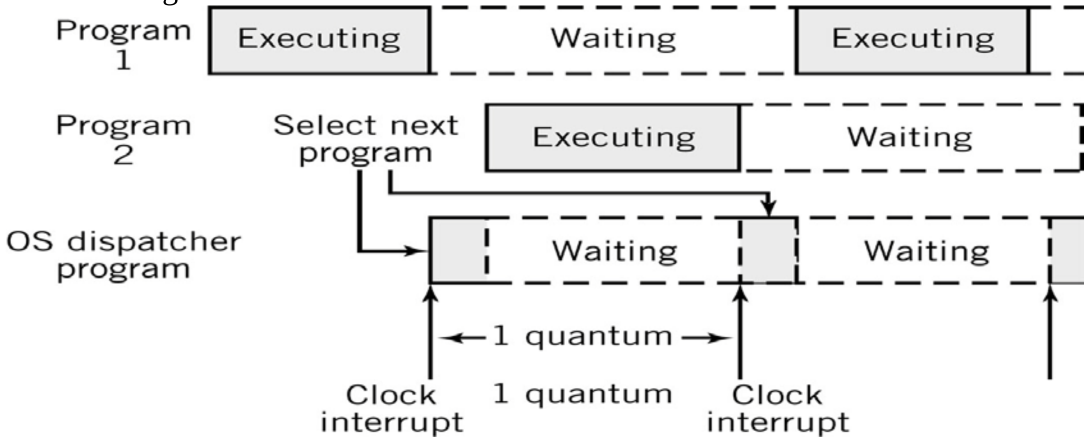
Sharing the CPU during I/O breaks.



A preemptive system - imposes the clock interrupt, uses time slicing. The program is given the CPU for a slice of time (10 ms, 20 ms, or 50 ms). If the program exceeds this time, the time out routine kicks in and the program is suspended. Figure 15.4, p. 495.

Many users and programs can share the computer simultaneously. Allowing a single user to dominate the system is unacceptable. All users have to be treated fairly. If users are to be treated fairly, 5 minutes of continuous CPU time cannot be granted to a single user.

Time-sharing the CPU.



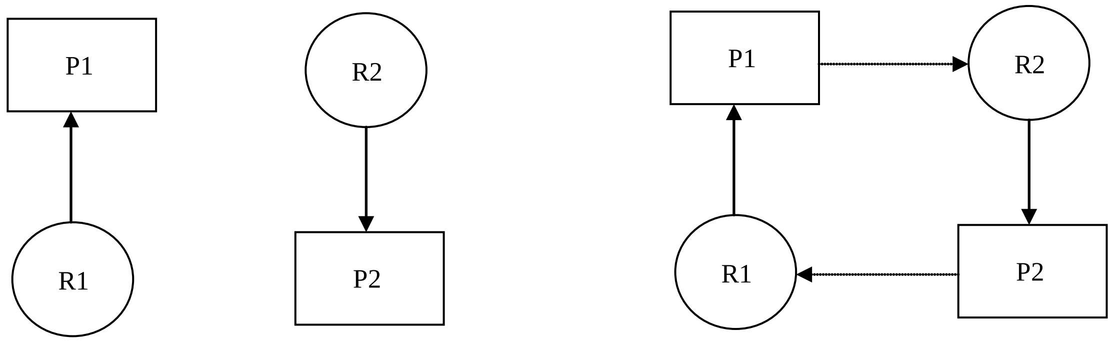
The processor switches so quickly between different tasks/programs that the user has got the impression that the computer works exclusively for him. In fact, it does not.

Compute (CPU) bound programs use little I/O and a great deal of the CPU, and do not generate interrupts too often. CPU bound programs need to be executed in preemptive environments.

Context Switching – transferring control to the program being dispatched.

9. Resolving Deadlocks or Preventing them from Occurring

When two programs P1 and P2 each need a resource R2 and R1, respectively, controlled by the other and neither is willing to give in, a deadlock occurs.



Deadlock must be corrected by the OS. If it is uncorrected, it may lead to a system failure. Some OSs allow deadlocks to occur and correct them, others prevent deadlocks from happening.

10. System Administration Support

The system administrators maintain the computer system. Their tasks include
- configuring the system,
- adding and deleting users to the system,
- changing user privileges,
- monitoring security,
- providing secure backups,
- mounting and unmounting file systems,
- installing new software and upgrading software,
- upgrading the OS, and
- recovering lost data, to name a few

For user administration, MS-DOS provided two simple text files
- CONFIG.SYS
- AUTOEXEC.BAT

CONFIG.SYS file sets the basic parameters for the system such as
- the size of stacks
- the location of I/O device drivers, and
- the # of files that could be opened at one time

Ex.
```
BUFFERS=20
FILES=20
LASTDRIVE=C
DEVICE=C:\DOS\MOUSE.SYS
```

Buffers=20 specifies the number of buffers allowed in memory for file transfers. Database programs doing numerous random R/W may require 20 or more buffers. The range is [1,99].

Files=20 specifies the number of files you can open at one time. The default is 8, and the range is [1,255] depending on the version of DOS.

Lastdrive=C is the maximum number of logical drives you can access. The default is E, regardless how many physical drives you have on the system.

The last command loads into memory the device driver you specify. The command installs the mouse attached to the system.

AUTOEXEC.BAT establishes user preferences at startup time.

Ex.
```
@ECHO  OFF
CLS
PROMPT  $p$g
PATH  C:\; C:\DOS; C:\WINDOWS
DATE
TIME
```

The "echo off" command causes that commands do not show up on the screen. In addition, the @ itself disables "echo off" display. The PROMPT command lets you customize the characters used in the MS-DOS command prompt. A "p" code displays the current directory of the default drive, and "g" code displays the > character.

When you type a command, DOS always looks for that command in the current directory. With the PATH command, you tell DOS to search other directories for a command that is not found in the current directory.

In addition to CONFIG.SYS and AUTOEXEC.BAT files, MS-DOS-based version of Windows added two additional files
- WIN.INI, and
- SYSTEM.INI

which provided for the configuration for the Windows subsystem.

Newer versions of Windows still provide all four of these files, but they are of limited capability. Much of the contents of the configuration files have been replaced with a registry system. Knowledgeable users can manipulate the system registry directly, when necessary.

On larger systems, the system administration is much more difficult. Many OSs provide tools that allow the administrator to tailor the system to optimize its performance. For example, throughput, turnaround time, response time, memory management allocation technique, scheduling method, the maximum # of programs to be executed concurrently, etc.

In UNIX, the system administrator is called a superuser with privileges that override all the restrictions and security built into the system. UNIX is equipped with tools that simplify the task of the superuser.

Examples of command line, menu-driven or GUI administrative commands

adduser - program for administering user accounts *newfs* – tool for building a file system

mount and *unmount* – for mounting and unmounting file systems

du and *df* – for measuring disk usage and free space, and

ufsdump and *ufsrestore* – for creating backups and recovering damaged files

Most OS including Windows (XP, Vista, 7) and Linux provide on-line real-time reports that indicate the load on the system and the efficiency of the system. For example, the CPU, I/O, and virtual storage usage.

<u>15.4. ORGANIZATION OF OS</u>

Read (pp. 502-505) about three configurations:
- monolithic,
- layered, and
- microkernel

<u>15.5. TYPES OF COMPUTER SYSTEMS</u> pp. 505-509 - Read