User Operating system interface:

1. Almost all operating systems have interface (UI). This interface can take several Forms one is Command-Line Interface, Batch Interface, Menu Driven User Interface and Graphical User Interface (GUI).
2. The important of user interface are:
    a. To assist users interacting with a software.
    b. To control how a user enters data instructions.
    c. TO control how information is displayed.

A. Command-line User interface:

1. The command-line user interface requires a user to type commands or press special keys on the keyboard to enter data and instructions that instruct the operating system what to do. It has to be typed one line at a time.
2. The command-line user interface also requires memorization. The advantage of command-line interface is, it helps the user to operate the computer quickly after memorizing the keywords and syntax.

B. Menu Driven User Interface:

1. It avoids memorizing the commands. Also it is much easier than Command Line Interface.
2. Here user is displayed with menus and submenus by clicking the appropriate options specific tasks carried out.

C. Graphical User Interface:

1. It is a software interface that user interact with a pointing device, such as mouse.
2. These are very user friendly because they are easy to interact with the system and to execute instructions. For example, when you are looking at the GUI of web browser.

Concepts of system calls:

1. They provide an interface between the process and operating system.
2. System calls allow user-level processes to request some services from the operating system which process itself is not allowed to do.
3. System calls provide an interface to the services made available by an operating system.
4. These calls are generally available as routines written in C and C++.
5. A system call is a request by the user to the operating system to do something on user's behalf. System Call provides an interface between a running program and an operating system.
6. In handling the trap, the operating system will enter in the kernel mode, where it has access to privileged instructions, and can perform the desired service on the behalf of the user-level process. It is because of the critical nature of operations that the operating system itself does them every time they are needed.
7. For example, for I/O a process involves a system call telling the operating system to read or write particular area and this request is satisfied by the operating system.

What are system calls?

1. System calls provide the interface between a process and the operating system.
2. System calls are instructions that generate an interrupt that causes the operating system to gain control of the processor.
3. The operating system then determines what kind of system call it is and performs the appropriate services for the system caller.

How to make a System Call:

1. A system call is made using the system call machine language instruction. These calls are generally available as assembly language instructions and are usually listed in the manuals used by assembly- language programmer.
2. Certain systems allow system calls to be made directly from a higher language program, in which case the calls normally resemble predefined function or subroutine calls. They may generate a call to a special run-time routine that makes the system call.
3. Several languages such as, C, C++ have been defined to replace assembly language for systems programming. These languages allow system calls to be made directly.
4. For example, Unix system calls may be invoked directly from a C or C++ program. System calls for Microsoft Windows Platform are part of the Win 32 API, which is available for use by all the compilers written for Microsoft Windows.

i. File and I/O System Calls:

Open	Get reading to read or write a file.
Create	Create a new file and open it.
Read	Read bytes from an open file.
Write	Write byte to an open.
Close	Indicate that you are done reading or writing a file.

ii. Process Management System Calls:

Create process	Create a new process.
Exit	Terminate the process making the system call.
Wait	Wait for another process to exit.
Fork	Create a duplicate of the process working the system call.
Execv	Run a new program in the process making the system call.

iii. Interprocess Communication System calls:

createMessageQueue	Create a queue to hold message.
SendMessage	Send a message to a message queue
ReceiverMessage	Receive a message from a message queue.

5. System calls can be roughly grouped into following major categories:
   i. Process or Job control.
   ii. File Management.
   iii. Device Management.
   iv. Information Maintenance.

A. <u>Process Or Job control:-</u>

- A running program needs to be able to halt its execution either normally (end) or abnormally (abort).
- If a system call is made to terminate the currently running program abnormally, or if the program runs into a problem and causes an error trap.
- A process or job executing one program many want to load and execute another program
- This allows the controller card interpreter to execute a program as directed by, for example, a user command, the click of a mouse, or a batch command.
- If we create a new job or process, we should be able to control its execution. We may also want find to terminate a job or process that we create (terminate process)
- If we find that it is incorrect or no longer needed we may require waiting time to finish execution (wait time). Another set of system calls are helpful in debugging a program.
- System calls related to Process control

- o end, abort
- o load, execute
- o create process, terminate process
- o get process attributes, set process attributes
- o wait for time
- o wait event, signal event
- o allocate and free memory

B. Device Management

- Files can be thought of the system calls for files are also needed for device.
- If there are multiple users of the system however, the users must first request device to ensure that they have an exclusive use of it.
- After the users are finished with the device, they must release it. These functions are similar to the open/close system calls for files
- Once the device has been requested (and allocated to us), we can read, write, and (possibly) reposition the device, just as we can with files.
  - In fact, the similarity between I/0 devices and files is so great that many operating system merge the two into a combined file-device structure. In this case, a set of system calls is used on both files and devices
  - System calls related to Device management
    - o request device, release device
    - o read, write, reposition
    - o get device attributes, set device attributes
    - o logically attach or detach devices

C. File Management

- Operating system provides several common system calls dealing with files.
- We first need to be able to create and delete files.
- Once the file is created, we need to open it and to use it. We may also read, write, or reposition (rewinding or skipping to the end of the file, for example). Finally, we need to close the file, indicating that we are no longer using it.
- They may also need these same sets of operations for directories if we have a directory structure for organizing files in the file system.
- In addition, for either files or directories, we need to be able to determine the values of various attributes and perhaps to reset them if necessary.
- File attributes include the file name, file type, protection codes, accounting information, and so on. At least two system calls, get file attribute and set file attribute, are required for this function.
- System calls related to File management
  - o create file, delete file
  - o open, close
  - o read, write, reposition
  - o get file attributes, set file attributes

D. Information Maintenance

- Many system calls exist simply for the purpose of transferring information between the user program and the operating system.
- For example, most systems have a system call to return the current time and date. Other system calls may return information about the system, such as the number of current users, the version number of the operating system, the amount of free memory or disk space, and so on.

- In addition, the operating system keeps information about all its processes, and system calls are used to access this information.
- Generally, calls are also used to reset the process information (get process attributes and set process attributes).
- System calls related to Information maintenance are:
    - o get time or date, set time or date
    - o get system data, set system data
    - o get process, file, or device attributes
    - o set process, file, or device attributes

Communications

- There are two common models of interprocess communication one is the message-passing model and another is the shared-memory model.
    i. Message-passing model: In this model the communicating processes exchange message with one another to transfer information. Messages can be exchanged between the processes either directly or indirectly through a common mailbox. Before communication can take place, a connection must be opened. The name of the other communicator must be known, be it another process on the same system or a process on another computer connected by a communications network. Each computer in a network has a *host name* by which it is commonly known. A host also has a network identifier, such as an IP address. Similarly, each process has a *process name,* and this name is translated into an identifier by which the operating system can refer to the process. The get host id and get process id system calls do this translation. The identifiers are then passed to the general purpose and close calls provided by the file system or to specific open connection and close connection system calls, depending on the system's model of communication. The recipient process usually must give its permission for that will be receiving connections are special-purpose daemons, which are systems programs provide for that purpose They execute a wait for connection call and are awakened when a connection is made. The source of the communication, known as the *client*, and the receiving daemon, known as a *server,* then exchange messages by using read message and write message system calls. The close connection call terminates the communication.
    ii. Share memory model: In this model, processes used memory create and shared memory attach system calls to create and gain access to regions of memory owned by other processes. Recall that, normally, the operating system tries to prevent one process-from accessing another process's memory. Shared memory requires that two or more processes agree to remove this restriction. They can then exchange information by reading and writing data in the shared areas. The form of the data is determined by the processes and is not under the operating system's control. The processes are also responsible for ensuring that they are not writing to the same location simultaneously.
    - System calls related to communication are:
        - o create, delete communication connection
        - o send, receive messages
        - o transfer status information
        - o attach or detach remote devices

System Call Implementation:

- Basically, a number is associated with each system call. It is used to number the system calls.
- System call interface maintains a table indexed according to these numbers. The system call interface invokes intended system call is operating system kernel and return status of the system call and any return values.

- The caller needs to know nothing about how the system call is implemented. Just needs to obey API and understand what operating system will do as a result call.
- Most details of operating system interface hidden from programmer by API It is managed by run-time support library.
- Some systems may allow system calls to be written in higher-level language program, in which the calls normally like predefined function or subroutine calls.
- They may be generated calls to a special run-time routine that makes the system calls, or the system call may be generated directly in line.
- Example; Consider two files: input and output. The program is to read data from input file and copy into output file. To execute this program, the sequence of system calls occurs as follows:
    - To obtain two file names from keyboard, system calls are required.
    - Then system call is open input file and creates an output file.
    - While opening, if file is not exit then program will print a message or terminate abnormally. This requires again system calls.
    - If the system exists, then for reading from the input file (a system call) and write to output file (another system call) calls are required.
    - For closing both files, we required two system calls.
- Some operating system provides API which invokes the actual system calls on behalf of the application programmer. Most of programming languages provides system call interface, which intercepts function calls in the API and invokes system calls within operating system.
- The relation between the API, the system call interface and operating system is shown in Fig.

- The parameter can be passed by using registers to implement system call. Sometimes parameters are stored in a block in memory and the address of block or table is passed as a parameter in register. UNIX uses such a mechanism.
- Implementation of system call very on different system. Generally, a unique number identifies the type of system call and a system call is made by calling particular number.
- Sometime there is a need of sending some additional information along with, when the call is made. That is called as parameters or parameters lists of the system calls.
- Special register is kept aside for the purpose of sending parameters. If the parameters are too many in number and are beyond the capacity of the register to hold, parameters are stored in a blocked or table in the memory and the base address of it is passed as parameter through the register.
- The system calls can categories into five major categories: process and job control, file management, device management, information maintenance and communications. The system calls vary from operating system to operating system.
- **System calls in UNIX and WINDOW:**

Types of System Calls.	**Unix**	**Windows.**
Process control.	Fork( ) Wait( ) Exit( )	CreateProcess( ) WaitForSingleObject( ) ExitProcess( )
File manipulation	Open( ) Read( ) Write( ) Close( )	CreateFile( ) ReadFile( ) WiteFile( ) CloseHandle( )
Device manipulation	Ioct( ) Read( )	SetConsoleMode( ) ReadConsole( )

	Write( )	WriteConsole( )
Information maintenance	Getpid( ) Alarm( ) Sleep( )	GetProcessId( ) SetTimer( ) Sleep( )
Communication	Pipe( )	CreatePipe( ) CreateFileMapping( ) MapViewFile( )
Protection and security	Chmod( ) Chown( )	SetFileSecurity( ) SetSecurityDesciptionGroup( )

- **Example:**

The standard C library provides a portion o£ the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the printf () statement The C library intercepts this call and invokes the necessary system call(s) in the operating system-in this instance, the write() system call. The C library takes the value returned by write() and passes it back to the user program. This is shown in Figure

System Program:

- System Program provides basic functioning to users so that they do not need to write their own environment for program development (editors, compilers) and program execution (shells).
- In some sense, system programming is bundles of useful calls.
- Modern system consists of a collection of system program.
- Most systems supply a large collection of system programs to solve commonly occurring problems and provide a more convenient and secure environment for program development and execution.
- System Program can be divided into several categories:
    - File management. These programs create, delete, copy, rename, print, dump, list, and generally manipulate files and directories.
    - Status information. Some programs simply ask the system for the date, time, amount of available memory or disk space, number of users, or similar status information.
    - File modification. Several text editors may be available to create and modify the content of files stored on disk or other storage devices. There may also be special commands to search contents of files or perform transformations of the text.
    - Programming-language support. Compilers, assemblers, debuggers, and interpreters for common programming languages (such as C, C++, Java, Visual Basic, and PERL) are often provided to the user with the operating system.
    - Program loading and execution. Once a program is assembled or compiled, it must be loaded into memory to be executed. The system may provide absolute loaders, relocatable loaders, linkage editors, and overlay loaders. Debugging systems for either higher-level languages or machine language are needed as well.
    - Communications. These programs provide the mechanism for creating virtual connection among processes, users, and computer systems. They allow users to send message to one another's screens, to browse Web pages, to send electronic-mail messages, to log in remotely, or to transfer files from one machine to another.