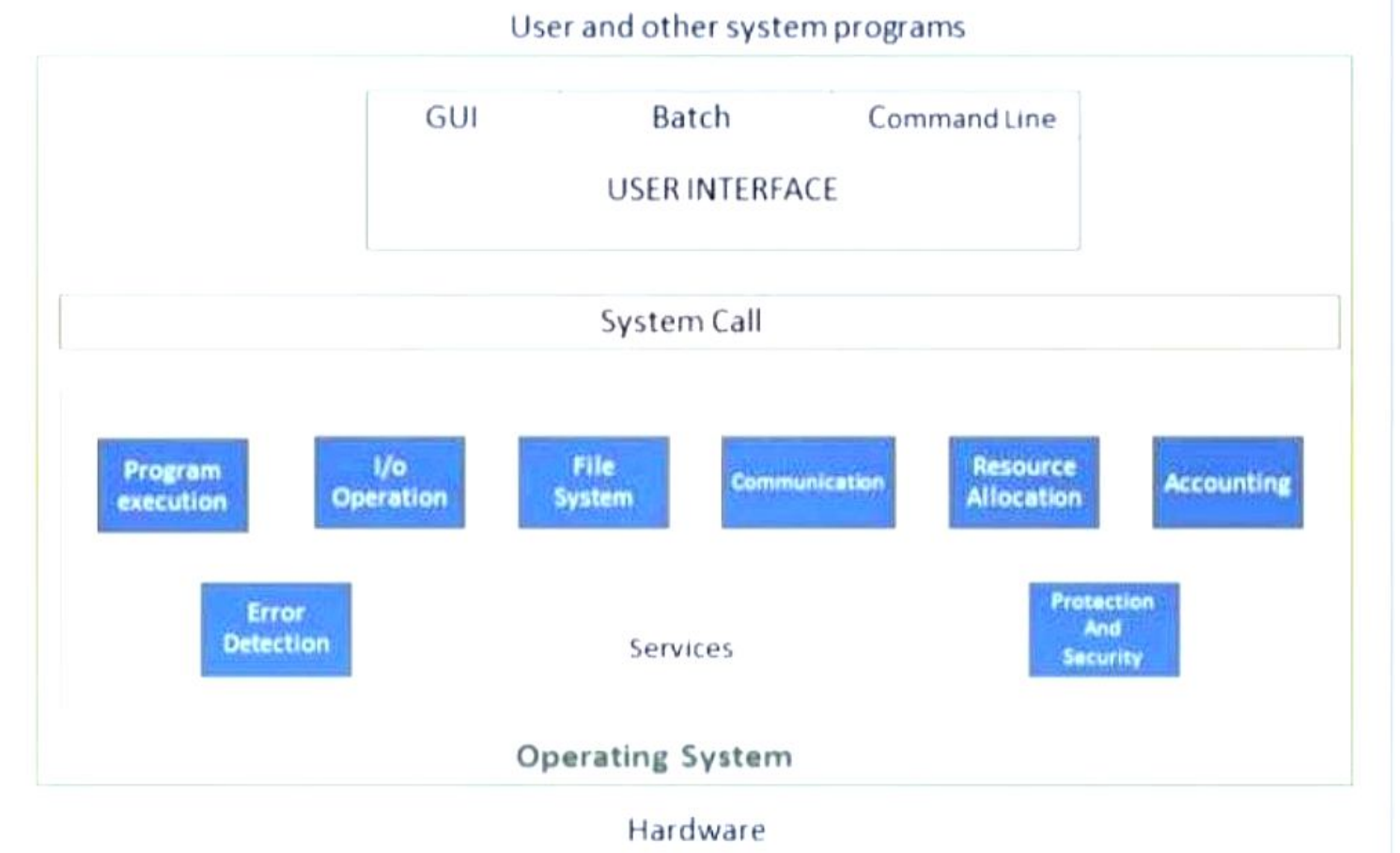
User and other system programs
GUI
Batch
Command Line
USER INTERFACE
System Call
Program execution
I/o Operation
File System
Communication
Resource Allocation
Accounting
Error Detection
Services
Protection And Security
Operating System
Hardware

1) User interface –

Almost all operating systems have a user interface (UI). One is a command line interface (CLI), which uses text commands. Another is a batch interface, in which commands and directives to control those commands are entered into files, and those files are executed.

Most commonly a graphical user interface (GUI) is used. Here, the interface is a window system with a pointing device to direct I/O, choose from menus, and make selections and a keyboard to enter text.

2) Program execution –

The system must be able to load a program into memory and to run that program.

3) I/O operations –

A running program may require I/O, which may involve an I/O device.

4) File system manipulation –

Programs need to read and write files and directories. They also need to create and delete them by name, search for a given file, and list file information.

5) Communications –

There are many situations in which one process needs to exchange information with another process. Communications may be implemented via shared memory or through message passing.

6) Error detection –

The operating system needs to be constantly aware of possible errors. Errors may occur in the CPU and memory hardware, in I/O devices, and in the user program.

) Resource allocation -

When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them.

Accounting –

Which users use how much and what kinds of computer resources. This record keeping may be used for accounting.

9) Protection and security –

Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important.

User Operating System Interface –

There are two fundamental approaches for users to interface with the operating system. One technique is to provide a command line interface or command interpreter that allows users to directly enter commands that are to be performed by the operating system.

The second approach allows the user to interface with the operating system via a graphical user interface or GUI.

1) Command Interpreter –

Some operating systems include the command interpreter in the kerne Others, such as Windows XP and UNIX, treat the command interpreter as a speci program that is running when a job is initiated.

The interpreters are also known as shells. For example, on UNIX and Lin systems, there are several different shells a user may choose from including t Bourne shell, C shell, Bourne – Again shell etc.

The main function of the command interpreter is to get and execute next user specified command. Many of the commands given at this level manipul files: create, delete, list, print, copy, execute, and so on.

There are two general ways in which these commands can implemented. In first approach, the command interpreter itself contains the cod execute the command.

An alternative approach, the command interpreter does not understand the command in any way; it just uses the command to identify a file to be loaded into memory and executed. Thus, the UNIX command to delete a file; rm sample.txt. Would search for a file called rm, load the file into memory, and execute it with the parameter sample.txt.

2) Graphical User Interface –

A GUI allows provides a mouse based window and menu system as an interface. A GUI provides a desktop where the mouse is moved to position its pointer on images, or icons, on the screen that represent programs, files and directories.

Depending on the mouse pointer's location, clicking a button on the mouse can invoke a program, select a file or directory – known as a folder – or pull down a menu that contains commands.

The first GUI appeared on the Xerox Alto computer in 1973. However, graphical interfaces became more widespread with the advent of Apple Macintosh computers in the 1980s. Microsoft's first version of Windows – version 1.0 – was based upon a GUI interface to the MS-DOS operating system

Traditionally, UNIX systems have been dominated by command line interfaces, although there are various GUI interfaces available. However, there has been significant development in GUI designs from various open source projects such as K Desktop Environment (or KDE).

The choice of whether to use a command line or GUI interface is mostly one of personal preference. As a very general rule, many UNIX users prefer a command line interface as they

often provide powerful shell interfaces.

Alternatively, most Windows users are pleased to use the Windows GUI environment

System Calls

A system call is a request by the user to the operating system to do something on behalf of user. System calls provides an interface between running program and an operating system.

Let's first use an example to illustrate how system calls are used: writing a simple program to read data from one file and copy them to another file.

Acquire input file name Acquire output file name Open the input file

```
if file doesn't exist, abort Create output file
        if file exists, abort Loop
        Read from input file
        Write to output file until read fails
        Close input and output file
Write completion message to screen Terminate normally
```

Figure – Example of how system calls are used

The first input that the program will need is the names of the two files: the input file and the output file. Once the two file names are obtained, the program must open the input file and create the output file.

There are also possible error conditions for each operation. When the program tries to open the input file, it may find that there is no file of that name. In these cases, the program should print a message on the console and then terminate abnormally.

If the input file exists, then we must create a new output file. We may find that

there is already an output file with the same name. This situation may cause the program to abort or we may delete the existing file and create a new one.

Now that both files are set up, we enter a loop that reads from the input file and writes to the output file. Finally, after the entire file is copied, the program may close both files, write a message to the console or window, and finally terminate normally.

Types of System Calls:

Six major categories of system calls described as follows,

1) Process Control –

When a process is running its execution is stop either normally or abnormally. If error occurs, then error massage is generated. We should able to control the execution of the newly created processes by assigning priority or reset the attributes of processes or sometimes terminate a created process. When the process is created, we should wait for its execution.

When a process creates memory is allocated or process terminates allocated memory is freed. System calls for the purpose of process control are as follows,

a. end, abort

b. load, execute

c. create process, terminate process

d. get process attributes, set process attributes

2) File Management –

We first need to be able to create and delete files. Either system call requires the name of the file and perhaps some of the file's attributes.

Once the file is created, we need to open it and to use it. We may also read or write.

Finally, we need to close the file, indicating that we are no longer using it.

File attributes include the file name, a file type, protection codes, accounting information, and so on. Get file attribute and set file attribute are required for this function. System calls for the purpose of file management are as follows,

a. create file, delete file

b. open, close

c. read, write

d. get file attributes, set file attributes

3) Device Management –

A process may need several resources to execute – main memory, disk drives, access to files, and so on. If the resources are available, they can be granted, and control can be returned to the user process.

Otherwise, the process will have to wait until sufficient resources are available. System calls for the purpose of device management are as follows,

a. request device, release device

b. read, write

4) Information Management –

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. System calls for the purpose of information management are as follows,

a. get time or date, set time or date

b. get system data, set system data

5) Communication –

There are two common models of inter-process communication: message passing model and the shared memory model. System calls for the purpo of communication are as follows,

a. create, delete communication connection

b. send, receive messages

c. transfer status information

6) Protection –

Protection provides a mechanism for controlling access to the resources provided by a computer system. Typically, system calls providing protection include set permission and get permission, which manipulates the permission settings of resources such as files and disks.

The allow user and deny user system calls specify whether particular users can or cannot be allowed access to certain resources. System calls for the purpose of protection are as follows,

a. set permission or get permission

b. allow or deny user