OPERATING SYSTEM SERVICES

- An operating system provides an environment for the execution of programs.
- It provides certain services to programs and to the users of those programs.
- The specific services differ from one operating system to another.
- These operating-system services are provided for the convenience of the programmer, to make the programming.

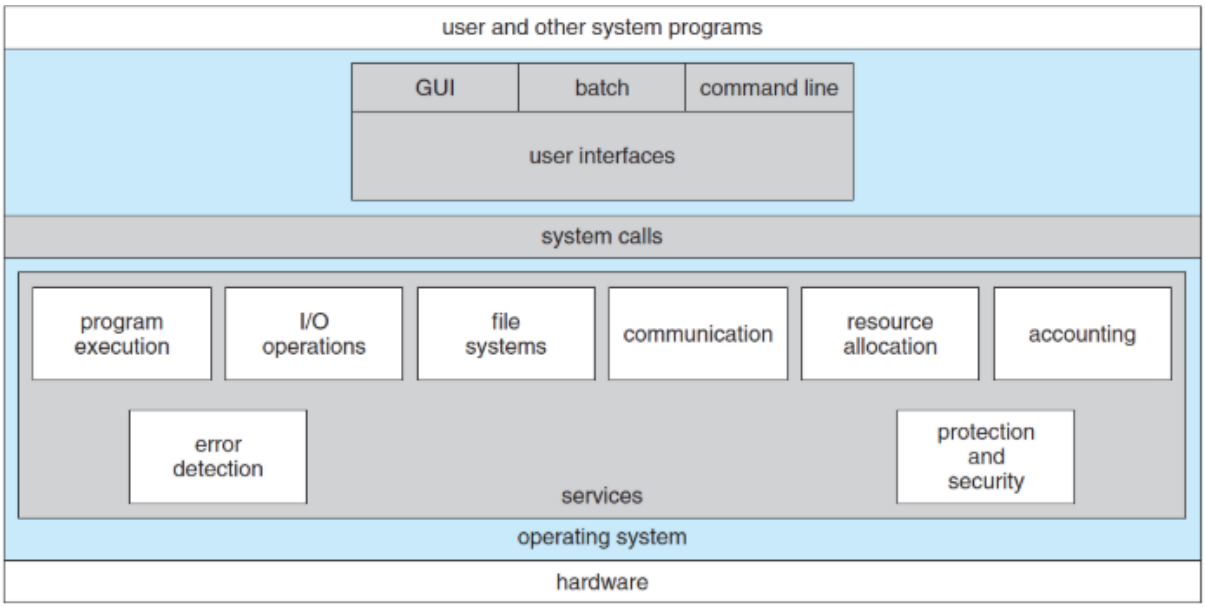


Fig : **A View Of Operating System Services**

One set of operating-system services provides functions that are helpful to the user.

1. **User interface**:
   Almost all operating systems have a **user interface (UI)**.

   ✓ **Forms of UI:**

   - One is a **command-line interface (CLI)**, which uses text commands and a method for entering them.
     Ex: keyboard for typing in commands in a specific format with specific options.
   - 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.

2. **Program execution**.
    - The system must be able to load a program into memory and to run that program.
    - The program must be able to end its execution, either normally or abnormally (indicating error).

3. **I/O operations**.
    - A running program may require I/O, which involve a file or an I/O device.
    - For specific devices, special functions may be desired (such as recording to a CD or DVD drive or blanking a display screen).
    - For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide a means to do I/O.

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.
    - Some operating systems include permissions management to allow or deny access to files or directories based on file ownership.
    - Many operating systems provide a variety of file systems, sometimes to allow personal choice and sometimes to provide specific features or performance characteristics.

5. **Communications**.
    - If one process needs to exchange information with another process, Such communication may occur between processes that are executing on the same computer or between processes that are executing on different computer systems connected together by a computer network.
    - Communications may be implemented via
        - **shared memory**, in which two or more processes read and write to a shared section of memory, or
        - **message passing**, in which packets of information in predefined formats are moved between processes by the operating system.

6. **Error detection**.
    - The operating system needs to be detecting and correcting errors constantly.

- Errors may occur in the CPU and memory hardware (such as a memory error or a power failure), in I/O devices (such as a parity error on disk, a connection failure on a network, or lack of paper in the printer), and in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location, or a too-great use of CPU time).
- For each type of error, the operating system should take the appropriate action
- ✓ to ensure correct and consistent computing,
- ✓ Sometimes, it has no choice but to halt the system.
- ✓ At other times, it might terminate an error-causing process or return an error code to a process for the process to detect and possibly correct.

7. **Resource allocation**.
   - When there are multiple users or multiple jobs running at the same time, resources must be allocated to each of them.
   - The operating system manages many different types of resources like CPU cycles, main memory, and file storage.

8. **Accounting**.
   - To keep track of which users use how much and what kinds of computer resources. This record keeping may be used for accounting (so that users can be billed) or simply for accumulating usage statistics.

9. **Protection and security**.
   - The owners of information stored in a multiuser or networked computer system may want to control use of that information.
   - Protection involves ensuring that all access to system resources is controlled.
   - Security of the system from outsiders is also important. Such security starts with requiring each user to authenticate.

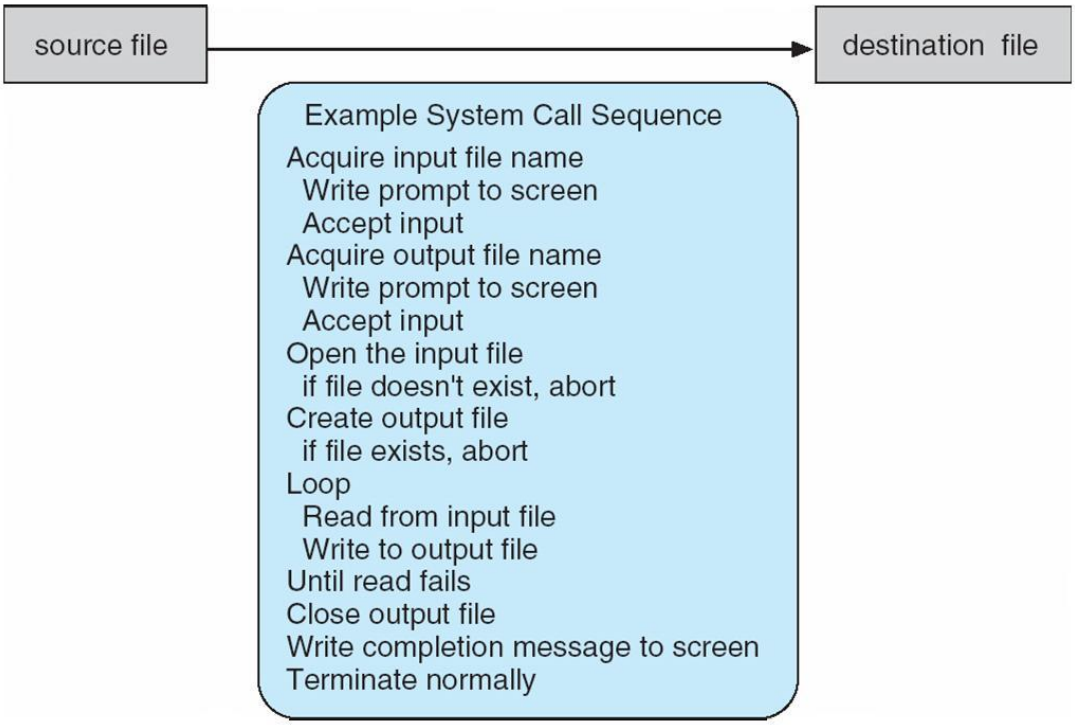
System Calls

- **System calls** provide an interface to the services made available by an operating system.
- These calls are generally available as routines written in C and C++.

- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use Three most common APIs are
  - ✓ Win32 API for Windows,
  - ✓ POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and
  - ✓ Java API for the Java virtual machine (JVM) .

Why use APIs rather than system calls?(Note that the system-call names used throughout this text are generic)

- **Example of System Calls**



System Call Implementation

- Typically, a number associated with each system call
- System-call interface maintains a table indexed according to these Numbers
- The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented Just needs to obey API and understand what OS will do as a result call
- Most details of OS interface hidden from programmer by API & Managed by run-time support library .

System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
- Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
- Simplest: pass the parameters in *registers*
- □ In some cases, may be more parameters than registers
- Parameters stored in a *block*, or table, in memory, and address of block passed as a parameter in a register
- This approach taken by Linux and Solaris
- Parameters placed, or *pushed*, onto the *stack* by the program and *popped* off the stack by the operating system
- Block and stack methods do not limit the number or length of parameters being passed .

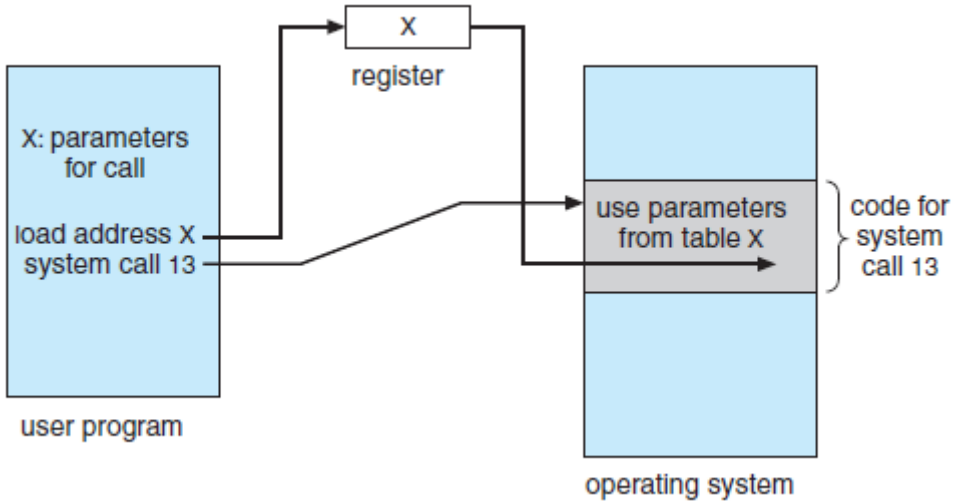


Fig : Passing of Parameters as a table.

Types of System Calls

System calls can be grouped roughly into six major categories:

- ✓ Process Control,
- ✓ File Manipulation,
- ✓ Device Manipulation,
- ✓ Information Maintenance,
- ✓ Communications, And
- ✓ Protection.

1. Process Control

- 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 dump of memory is sometimes taken and an error message generated.
- The dump is written to disk and may be examined by a
  - ✓ debugger—a system program designed to aid the programmer in finding and correcting errors, or
  - ✓ bugs—to determine the cause of the problem.
- Under either normal or abnormal circumstances, the operating system must transfer control to the invoking command interpreter.
- The command interpreter then reads the next command.

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

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(), write(), or reposition()
- Finally, we need to close() the file, indicating that we are no longer using it.
- File attributes include the file name, file type, protection codes, accounting information, and so on.
- At least two system calls, get file attributes() and set file attributes(), are required for this function.
- Some operating systems provide many more calls, such as calls for file move() and copy().

File management
◦ create file, delete file
◦ open, close
◦ read, write, reposition
◦ 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.
- The various resources controlled by the operating system can be thought of as devices.
- Some of these devices are physical devices (for example, disk drives), while others can be thought of as abstract or virtual devices (for example, files).
- A system with multiple users may require us to first request() a device, to ensure exclusive use of it.
- After we are finished with the device, we release() it.
- These functions are similar to the open() and close() system calls for files.

Device management
◦ request device, release device
◦ read, write, reposition
◦ get device attributes, set device attributes
◦ logically attach or detach devices

4. 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.

Information maintenance
◦ get time or date, set time or date
◦ get system data, set system data
◦ get process, file, or device attributes
◦ set process, file, or device attributes

5. Communication

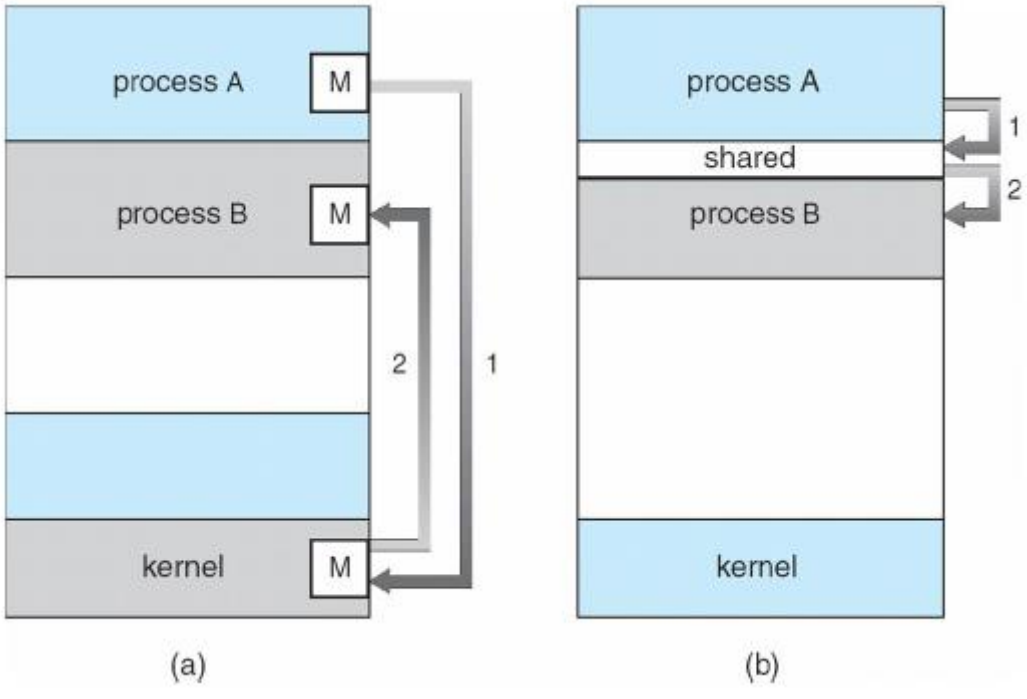
- There are two common models of interprocess communication:
  a. The message passing model and
  b. The shared-memory model.

a. In the **Message-Passing Model** :
   - ✓ The communicating processes exchange messages 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.
   - ✓ The get hostid() and get processid() system calls are used to get the **host name and process name**.
   - ✓ The identifiers are then passed to the general purpose open() 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 communication to take place with an **accept connection()** call.
   - ✓ 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.

b. In the **Shared-Memory Model** :

- ✓ processes use shared 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.

- **Communications**
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach or detach remote devices



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 manipulate 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.

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shm_open() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

System Programs

- **System programs**, also known as **system utilities**, provide a convenient environment for program development and execution.
- They can be divided into these 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, and PERL) are often provided with the operating system or available as a separate download.

• **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 connections among processes, users, and computer systems. They allow users to send messages to one another's screens, to browse Web pages, to send e-mail messages, to log in remotely, or to transfer files from one machine to another.

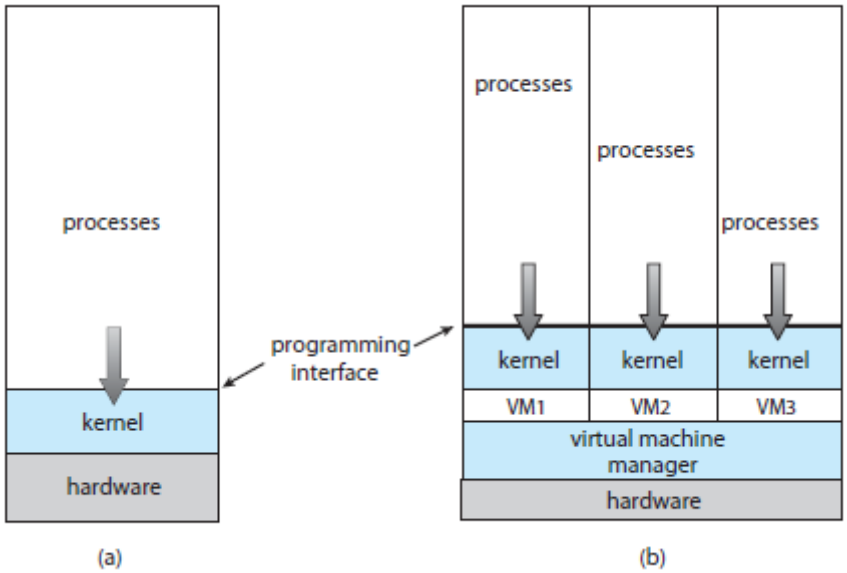
• **Background services**. All general-purpose systems have methods for launching certain system-program processes at boot time. Some of these processes terminate after completing their tasks, while others continue to run until the system is halted. Constantly running system-program processes are known as **services**, **subsystems**, or daemons.

Virtual Machines :

A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.

A virtual machine provides an interface *identical* to the underlying bare hardware.

The operating system host creates the illusion that a process has its own processor and (virtual memory) & each guest provided with a (virtual) copy of underlying computer .



Virtual Machines History and Benefits :

- First appeared commercially in IBM mainframes in 1972
- Fundamentally, multiple execution environments (different operating systems) can share the same hardware
- Protect from each other
- Some sharing of file can be permitted, controlled
- Commutate with each other, other physical systems via networking
- Useful for development, testing
- Consolidation of many low-resource use systems onto fewer busier systems
- "Open Virtual Machine Format", standard format of virtual machines, allows a VM to run within many different virtual machine (host) platforms

The Java Virtual Machine

Operating-System Debugging

- Debugging is finding and fixing errors, or bugs
- OS generate log files containing error information
- Failure of an application can generate core dump file capturing memory of the process
- Operating system failure can generate crash dump file containing kernel memory

- Beyond crashes, performance tuning can optimize system performance
- Kernighan's Law: "Debugging is twice as hard as writing the code in the rst place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."
- DTrace tool in Solaris, FreeBSD, Mac OS X allows live instrumentation on production systems
- Probes fire when code is executed, capturing state data and sending it to consumers of those probes .