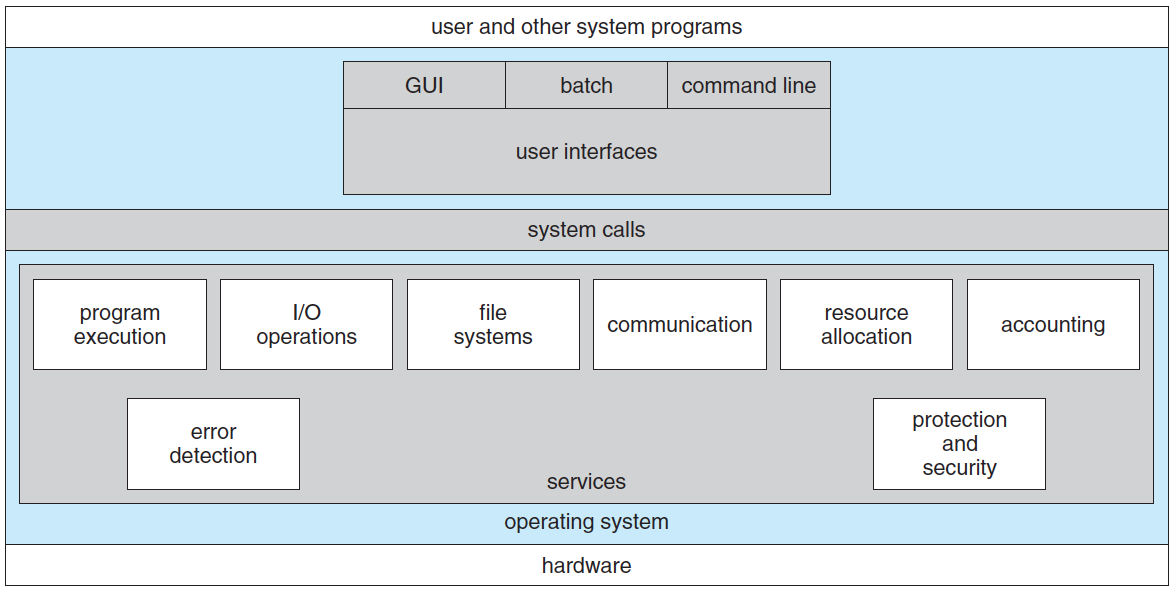
**Operating system services**

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

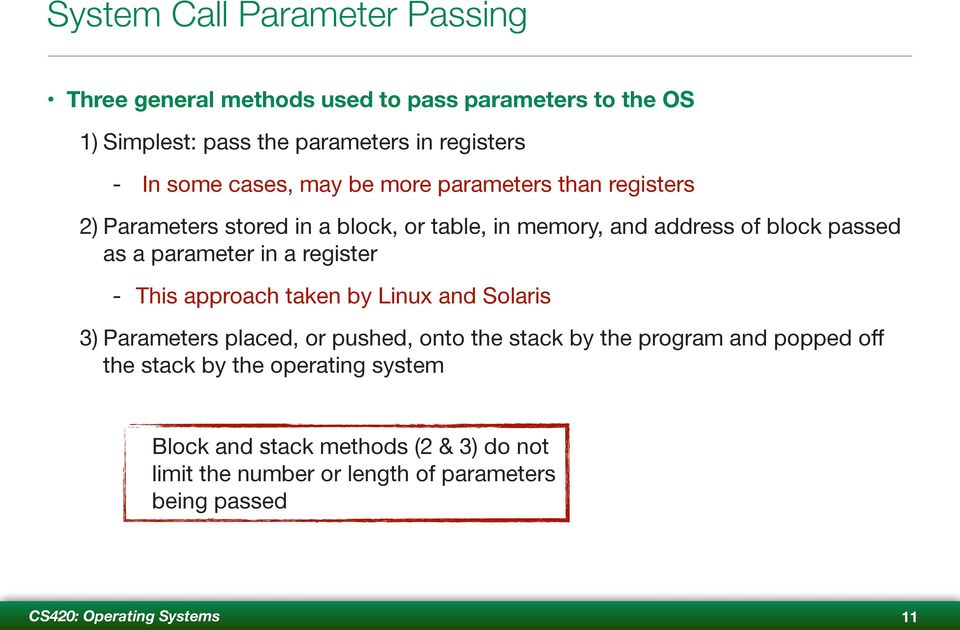


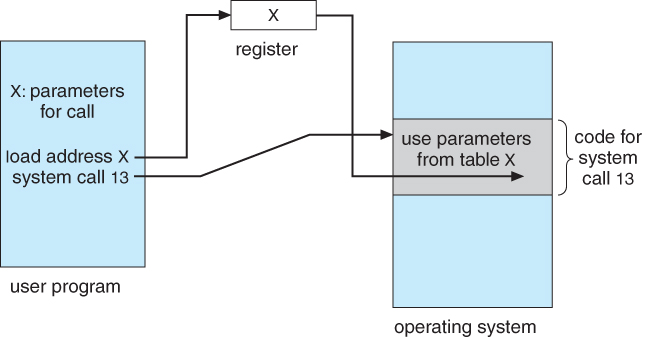
1. User Interface
2. Program execution
3. I/O operations
4. File-system manipulation
5. Communication
6. Error Detection
7. Resource allocation
8. Accounting
9. Protection and Security

* ****User interface****.This interface can take several forms. One is a ****command-line interface(CLI)****, which uses text commands and a method for entering them. 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.
* ****Program execution****. The system must be able to load a program into memory and to run that program.
* ****I/O operations****. A running program may require I/O, which may involve a file or an I/O device.
* ****File-system manipulation****. Obviously, programs need to read and write files and directories.
* ****Communications****. There are many circumstances in which one process needs to exchange information with another process.
* ****Error detection****. The operating system needs to be constantly aware of possible errors. Errors may occur in the CPU, 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. This includes CPU cycles, main memory, file storage, and I/O devices.
* ****Accounting & Protection and security.****This features are for managing multiple users and keep eyes on them.

**System Calls**

1. System calls provide an interface to the services made available by an operating system.
2. A system call is a way for programs to **interact with the operating systems**.
3. A computer program makes a system call when it makes a request to the operating system’s kernel.
4. **Services Provided by System Calls:**
   1. Process creation and management
   2. Main memory management
   3. File Access, Directory and File system management
   4. Device handling(I/O)
   5. Protection
   6. Networking, etc.

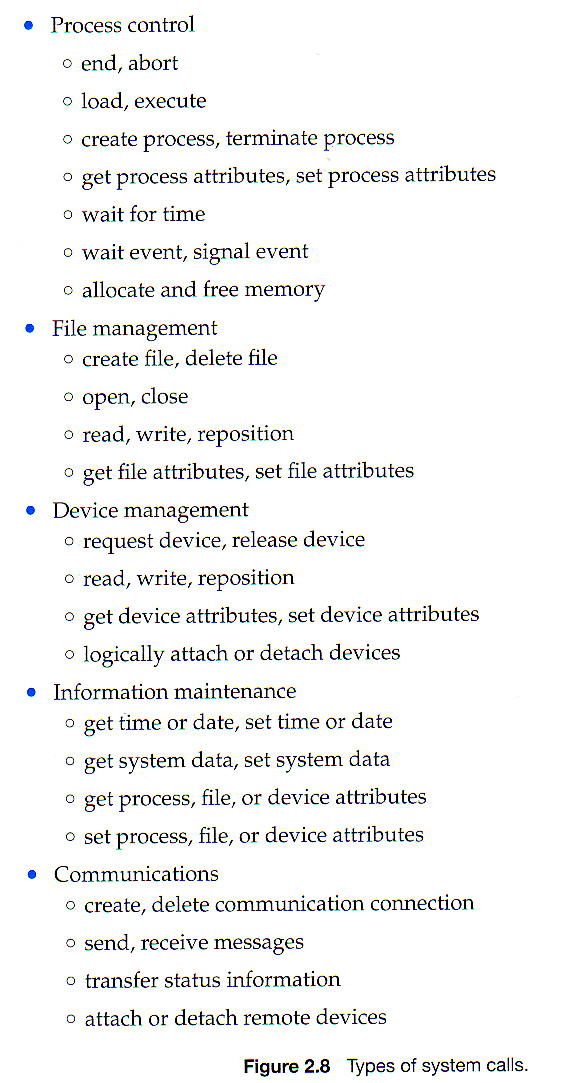
****

****

Parameter passing as a table

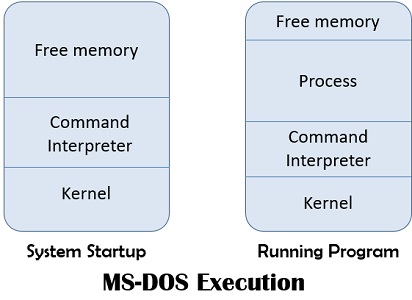
**Types of System Calls**

1. Process control,
2. File manipulation,
3. Device Manipulation,
4. Information Maintenance
5. Communications
6. Protection



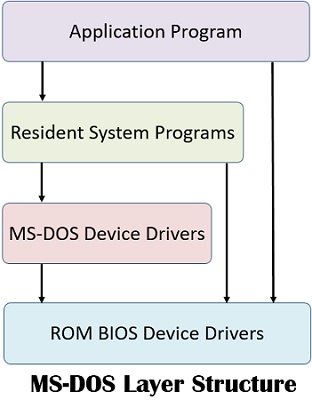
**How program run on MS-DOS**

* MS-DOS is single-tasking, it uses a simple method to run program and don’t create a new process.
* It loads program into memory, Next, it sets the instruction pointer to the first instruction of the program.
* The program then run and after execution a system call is occur to terminate program else the error is captured and showed to the screen.

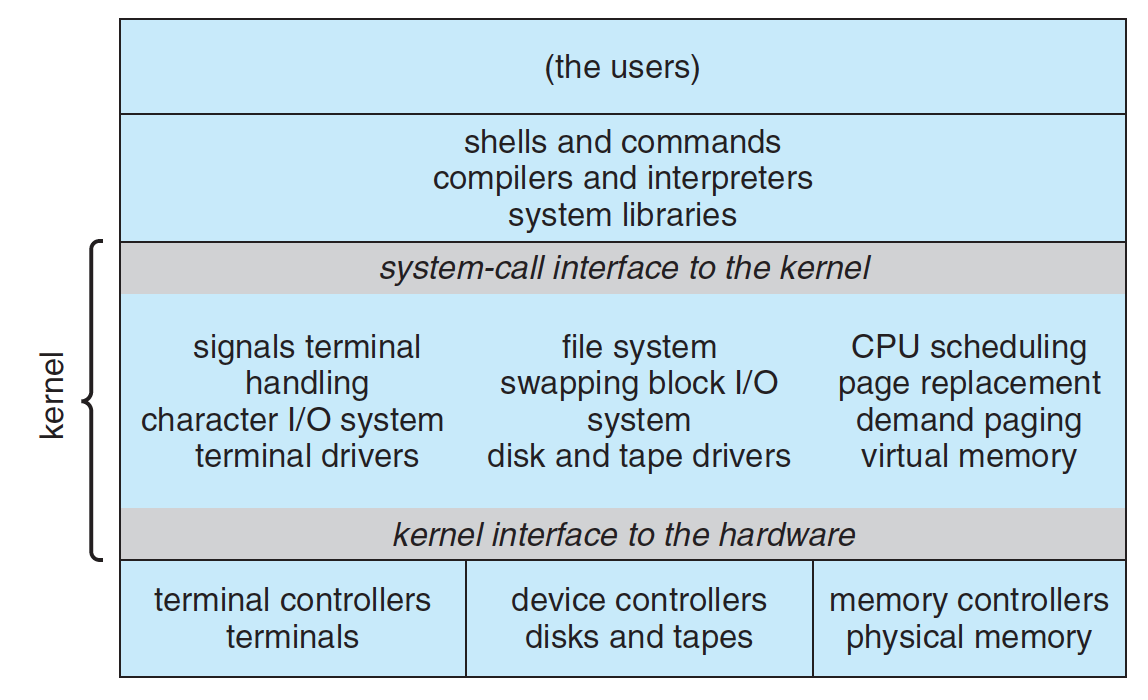


**Operating-System Structure**

**Simple structure:-**

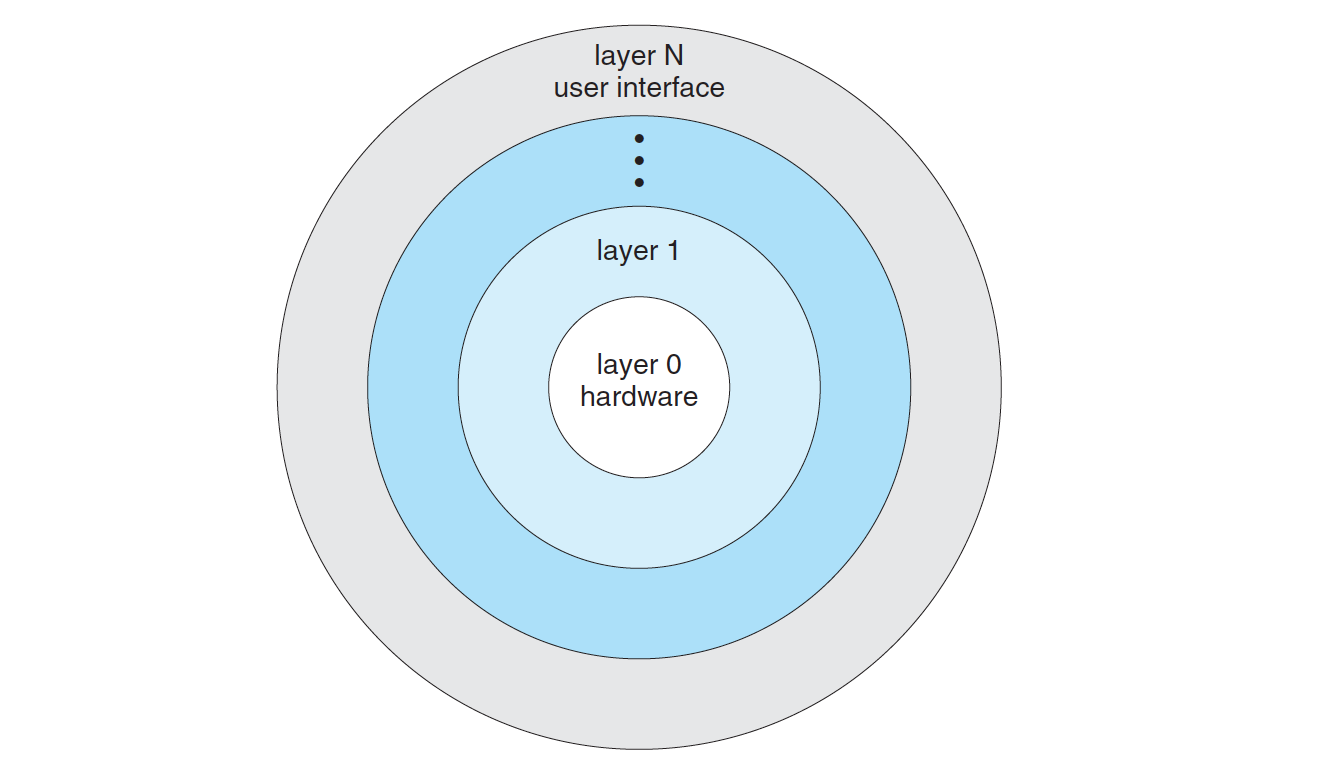


* Simple structure operating systems were used for OS which are designed in very beginning.
* It is followed by MS-DOS.
* As we can notice that ROM BIOS Device Drivers can be accessed by Device Drivers, Resident System Programs and by application program.
* So, the interface and levels of functionalities are not well separated which makes it vulnerable to malicious programs causing the entire system to crash.

**Monolithic structure**

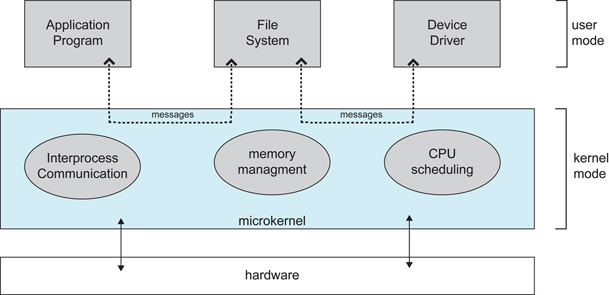
* Followed by earlier UNIX system.
* Consists of two separable parts: the kernel and the system programs.
* The kernel provides the file system, CPU scheduling, memory management, and other operating-system functions through system calls.
* All the functionalities are packed into one level i.e Kernel.
* Which make it implementation & maintainance very difficult.
* Suppose we want to change anything in kernel or debug some functionality ,for this we need to make touch all the parts of kernel.

**Layered Structure**

****

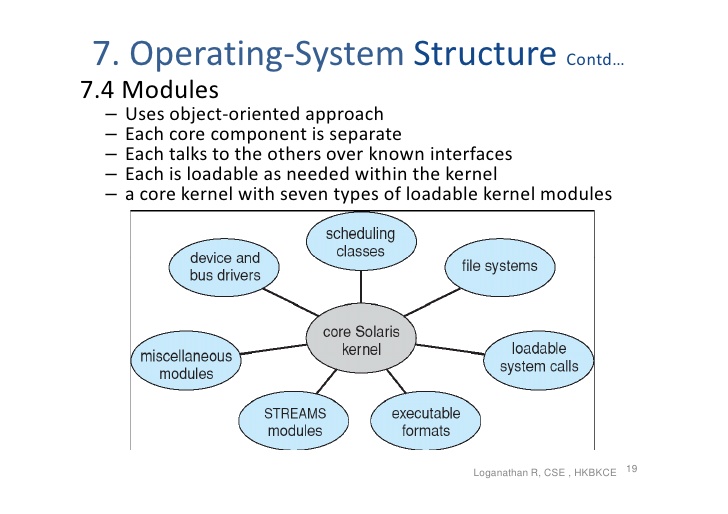
* In this structure, OS is divided into N number of layers (levels).
* At bottom layer (layer0) we have hardware and the Nth layer is user interface.
* A typical operating-system layer consists of data structures and a set of functions that can invoked by higher-level layers.
* The main advantage of the layered approach is simplicity of construction and debugging.
* The major difficulty with the layered approach involves appropriately defining the various layers. Because a layer can use only lower-level layers, careful planning is necessary.
* Less efficient than other structure.

**Microkernels Structure**

****

* In this structure we remove all non-essential components from the kernel and implementing them as system and user-level programs.
* Results in smaller kernel.
* The main function of the Microkernel is to provide a **communication facility** between the client process and the various services.
* Communication is provided by **message passing**.
* For example, if the client process wishes to access a file, it must interact with the file server. The client program and service never interact directly. Rather, they communicate indirectly by exchanging messages with the microkernel.
* Advantage whole system will not crash as most of the functionalities will be running in user mode.
* One benefit of the micro kernel approach is ease of extending the operating system.All new services are added to user space and consequently do not require modification of the kernel. When the kernel does have to be modified, the changes tend to be fewer, because the microkernel is a smaller kernel.
* Disadvantage suffer from performance decrease due to increase system function overhead.

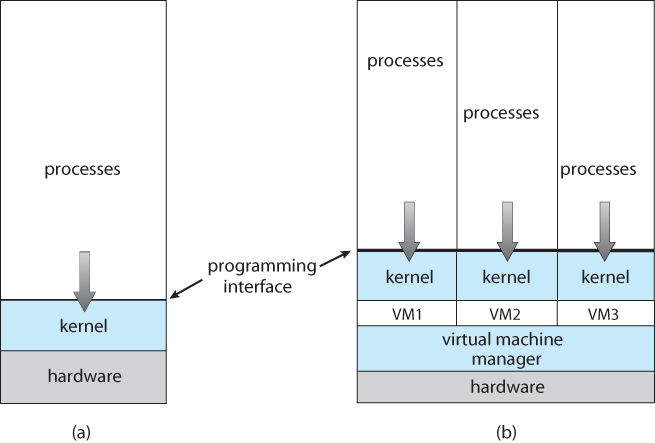
**Modular Structure**



* Best current methodology for operating-system design. Which involves using object oriented programming technique to create a modular kernel.
* In this methodology, we have core kernel which contain only core functionality of the kernel, and other functionalities are present in the form of modules and these functionalities will be loaded to the kernel either at boot time or run-time.
* uses dynamically loadable modules.
* This approach resembles to both layered structure and microkernel approach.
* It resembles to layered structure in such a way that each kernel section has defined protected interfaces. But more flexible than layered approach. For example, if one layerN want to communicate with layerM, system call need to pass through all the intermediate layers, but in case of modular approach , two modules can directly communicate with each other.
* It resembles to microkernel approach in such a manner that in both the approaches we have kernel separated from other functionalities. But its advantage over microkernel approach is that we don’t need message passing to communicate between two modules but in this approach functionalities will be directly loaded to core kernel itself when need. Hence system overhead is not there.

**Virtual Machines**

* The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards, and so forth) into several different execution environments, thereby creating the illusion that each separate execution environment is running its own private computer.



* The virtual machine provides an interface that is identical to the underlying hardware. Each process is provided with a (virtual) copy of the underlying computer.
* Usually, the guest process is in fact an operating system, and that is how a single physical machine can run multiple operating systems concurrently, each in its own virtual machine.
* Benefits of virtual machines
  + VM are robust security and continuous system development. It ensures robust security by completely separating system resources and allows continuous system development to be done without disrupting normal operation.
  + One of the disadvantages of VM is no direct sharing of resources. This often limits some functionalities of system and user programs.