Operating System

Unit – 1
(Part-B)
Introduction

Mayank Mishra
School of Electronics Engineering
KIIT-Deemed to be University

Virtualization

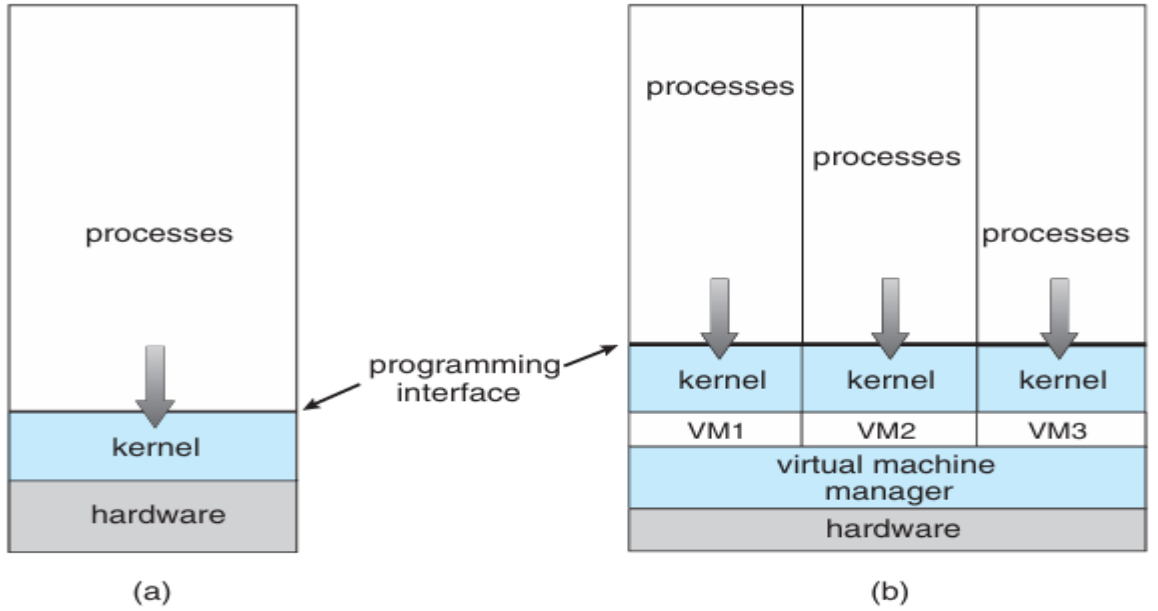

Fig. A computer running (a) a single operating system and (b) three virtual machines.

- **Virtualization** is a technology that allows us 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 environment is running on its own private computer.

- These environments can be viewed as different individual operating system (for example, Windows and UNIX) that may be running at the same time and may interact with each other.

- A **Virtual Machine (**VM) is a virtualized environment of a physical computer. It can perform almost all of the same functions, including running applications and operating systems.

- In its simplest form, a **virtual machine**, or VM, is a digitized version of a physical computer. Virtual machines can run programs and operating systems, store data, connect to networks, and do other computing functions. However, a VM uses entirely virtual resources instead of physical components.

- A more straightforward way to understand what a virtual machine is to think of it as a virtual computer within another computer. But instead of a physical computer like a server, laptop, or smartphone, a VM is defined by software.

- Virtualization refers to create virtual instances of OS that run **concurrently** on the same physical machine. These instances are isolated from each other and managed by a virtualization layer, such as **hypervisor (for example, VMware, VirtualBox, or Hyper-V)**

- Each VM operates independently. One VM’s crash or failure does not impact others.

Computing Environments

We will focus in this section to study, how operating systems are used in a variety of computing environments.

Traditional Computing (Personal Computing):

- As computing has matured, the lines separating many of the traditional computing environments have blurred.

- Consider the "typical office environment." Just a few years ago, this environment consisted of PCs connected to a network, with servers providing file and print services. Remote access was awkward, and portability was achieved by use of laptop computers.

- Today, web technologies and increasing WAN bandwidth are stretching the boundaries of traditional computing. Companies establish **portals**, which provide web accessibility to their internal servers.

- **Network computers** (or thin clients)—which are essentially terminals that understand web-based computing—are used in place of traditional workstations where more security or easier maintenance is desired.

- Mobile computers can synchronize with PCs to allow very portable use of company information. Mobile devices can also connect to wireless networks and cellular data networks to use the company's web portal.

- At home, most users once had a single computer with a slow modem connection to the office, the Internet, or both. Today, network-connection speeds once available only at great cost are relatively inexpensive in many places, giving home users more access to more data.

- These fast data connections are allowing home computers to serve up web pages and to run networks that include printers, client PCs, and servers. Many homes use firewall to protect their networks from security breaches. Firewalls limit the communications between devices on a network.

Mobile Computing:

- **Mobile computing** refers to computing on handheld smartphones and tablet computers. These devices share the distinguishing physical features of being portable and lightweight.

- Historically, compared with desktop and laptop computers, mobile systems gave up screen size, memory capacity, and overall functionality in return for handheld mobile access to services such as e-mail and web browsing.

- Over the past few years, however, features on mobile devices have become so rich that the distinction in functionality between, say, a consumer laptop and a tablet computer may be difficult to discern. In fact, we might argue that the features of a contemporary mobile device allow it to provide functionality that is either unavailable or impractical on a desktop or laptop computer.

- Today, mobile systems are used not only for e-mail and web browsing but also for playing music and video, reading digital books, taking photos, and recording and editing high-definition video. Accordingly, tremendous growth continues in the wide range of applications that run on such devices.

- Many developers are now designing applications that take advantage of the unique features of mobile devices, such as global positioning system (GPS) chips, accelerometers, and gyroscopes. An embedded GPS chip allows a mobile device to use satellites to determine its precise location on Earth.

- An accelerometer allows a mobile device to detect its orientation with respect to the ground and to detect certain other forces, such as tilting and shaking. In several computer games that employ accelerometers, players interface with the system not by using a mouse or a keyboard but rather by tilting, rotating, and shaking the mobile device! **Perhaps more a practical use of these features is found in augmented-reality applications.**

- To provide access to on-line services, mobile devices typically use either IEEE standard 802.11 wireless or cellular data networks. The memory capacity and processing speed of mobile devices, however, are more limited than those of PCs.

- The memory capacity and processing speed of mobile devices, however, are more limited than those of PCs. Whereas a smartphone or tablet may have 256 GB in storage, it is not uncommon to find 8 TB in storage on a desktop computer. Similarly, because power consumption is such a concern, mobile devices often use processors that are smaller, are slower, and offer fewer processing cores than processors found on traditional desktop and laptop computers..

- Two operating systems currently dominate mobile computing: Apple iOS and Google Android.

- iOS was designed to run on Apple iPhone and iPad mobile devices.

- Android powers smartphones and tablet computers available from many manufacturers.

Client Computing:

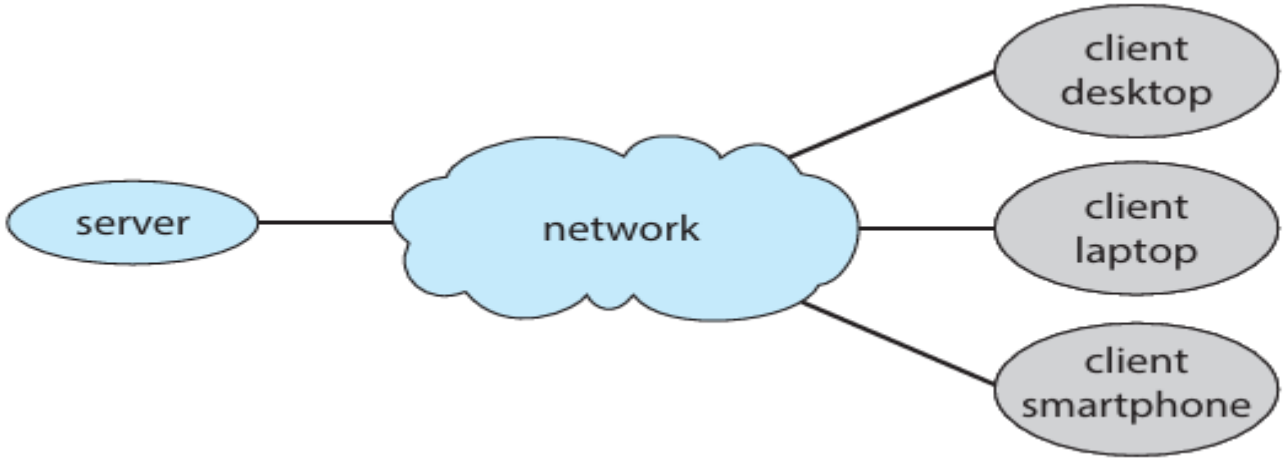

Fig. General structure of a client–server system.

- Contemporary network architecture features arrangements in which **server systems** satisfy requests generated by **client systems**. This form of specialized distributed system, called a **client–server system**.

- Server systems can be broadly categorized as **compute servers** and **file servers.**

- The **compute-server system** provides an interface to which a client can send a request to perform an action (for example, read data). In response, the server executes the action and sends the results to the client. A server running a database that responds to client requests for data is an example of such a system.

- The **file-serve system** provides a file-system interface where clients can create, update, read, and delete files. An example of such a system is a web server that delivers files to clients running web browsers. The actual contents of the files can vary greatly, ranging from traditional web pages to rich multimedia content such as high-definition video..

Peer-to-Peer Computing:

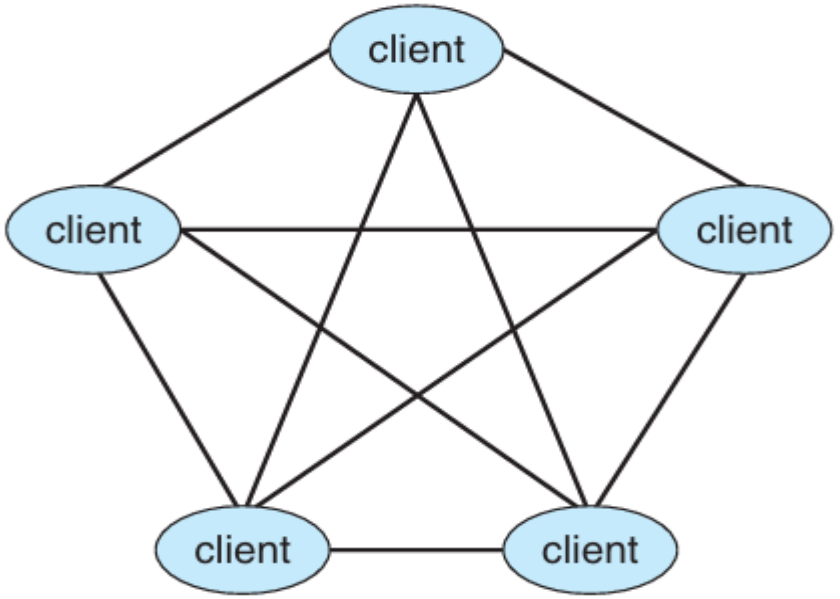

Fig. Peer-to-peer system with no centralized service.

- In this model, **clients and servers are not distinguished from one another**. Instead, all nodes within the system are considered **peers**, and each may act as either a client or a server, depending on whether it is requesting or providing a service.

- To participate in a peer-to-peer system, a node must first join the network of peers. Once a node has joined the network, it can begin providing services to—and requesting services from—other nodes in the network.

- **Skype** is example of peer-to-peer computing. It allows clients to make voice calls and video calls and to send text messages over the Internet using a technology known as **voice over IP (VoIP).** Skype uses a hybrid peer to-peer approach. It includes a centralized login server, but it also incorporates decentralized peers and allows two peers to communicate.

Cloud Computing:

- Cloud computing is a type of computing that delivers computing, storage, and even applications as a service across a network.

- In some ways, it's a logical extension of virtualization, because it uses virtualization as a base for its functionality.

- For example, the *Amazon Elastic Compute Cloud (ec2)* facility has thousands of servers, millions of virtual machines, and petabytes of storage available for use by anyone on the Internet. Users pay per month based on how much of those resources they use.

- Nowadays, **Cloud computing** is adopted by every company, whether it is an MNC or a startup many are still migrating towards it because of the cost-cutting, lesser maintenance, and the increased capacity of the data with the help of servers maintained by the cloud providers.

- **Cloud Computing** means storing and accessing the data and programs on remote servers that are hosted on the internet instead of the computer’s hard drive or local server.

- Cloud computing is also referred to as Internet-based computing, it is a technology where the resource is provided as a service through the Internet to the user. The data that is stored can be files, images, documents, or any other storable document.

- Cloud computing architecture refers to the components and sub-components required for cloud computing. These components typically refer to:

  - **Front end** (Fat client, Thin client)
  - **Back-end platforms** (Servers, Storage)
  - **Cloud-based delivery and a network** (Internet, Intranet, Intercloud)

- The User Interface of Cloud Computing consists of 2 sections of clients. The **Thin clients** are the ones that use web browsers facilitating portable and lightweight accessibilities and others are known as Fat **Clients** that use many functionalities for offering a strong user experience.

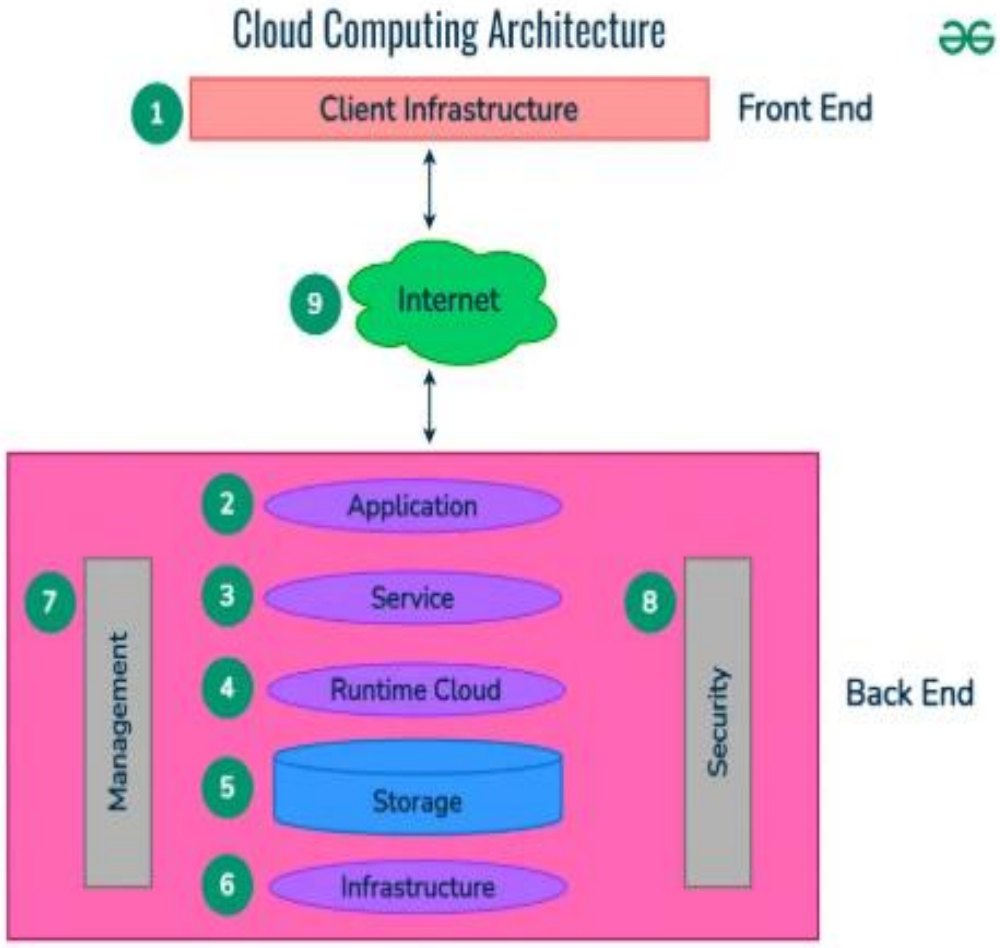

Fig. Architecture of Cloud Computing

- The core of cloud computing is made at **back-end** platforms with several servers for storage and processing computing. Management of Applications logic is managed through servers and effective data handling is provided by storage. The combination of these platforms at the backend offers the processing power, and capacity to manage and store data behind the cloud.

- On-demand access to the computer and resources is provided over the Internet, Intranet, and Intercloud. The **Internet** comes with global accessibility, the **Intranet** helps in internal communications of the services within the organization and the **Intercloud** enables interoperability across various cloud services. This dynamic network connectivity ensures an essential component of cloud computing architecture on guaranteeing easy access and data transfer..

➢ There are actually many types of cloud computing, including the following:

• **Public cloud**—a cloud available via the Internet to anyone willing to pay for the services

• **Private cloud**—a cloud run by a company for that company's own use

• **Hybrid cloud**—a cloud that includes both public and private cloud components

• **Software as a service (SaaS)—**one or more applications (such as word processors or spreadsheets) available via the Internet

• **Platform as a service (PaaS)—**a software stack ready for application use via the Internet (for example, a database server)

• **Infrastructure as a service (IaaS)—**servers or storage available over the Internet (for example, storage available for making backup copies of pro duction data)

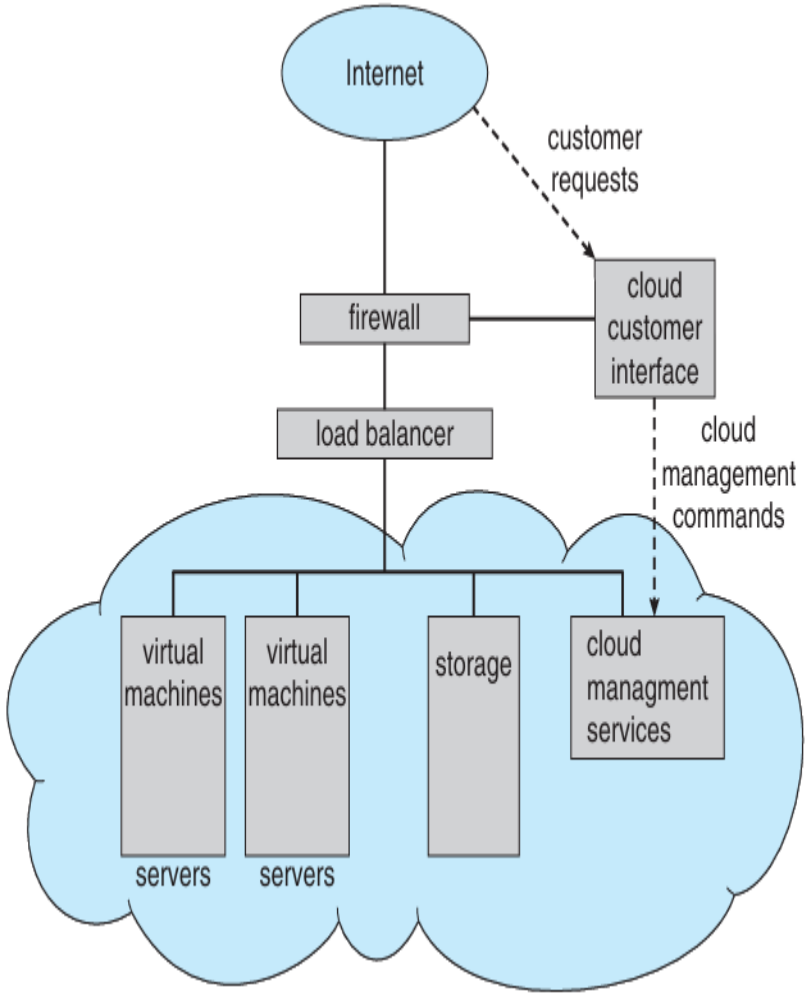

Fig. Illustration of a public cloud providing IaaS

Real-Time Embedded Systems:

- Embedded computers are the most prevalent form of computers in existence. These devices are found everywhere, from car engines and manufacturing robots to optical drives and microwave ovens.

- They tend to have very specific tasks. The systems they run on are usually primitive, and so the operating systems provide limited features.

- Usually, they have little or no user interface, preferring to spend their time monitoring and managing hardware devices, such as automobile engines and robotic arms.

- The use of embedded systems continues to expand. The power of these devices, both as standalone units and as elements of networks and the web, is sure to increase as well. Even now, entire houses can be computerized, so that a central computer—either a general-purpose computer or an embedded system —can control heating and lighting, alarm systems, and even coffee makers.

Operating System Structure

Simple Structure:

- It was the structure that was followed for most of the **Operating System** that were designed long ago.
- Simple structure operating systems do not have well-defined structures and are small, simple, and limited.
- The interfaces and levels of functionality are not well separated. MS-DOS is an example of such an operating system. In MS-DOS, application programs are able to access the basic I/O routines.
- These types of operating systems cause the entire system to crash if one of the user programs fails.

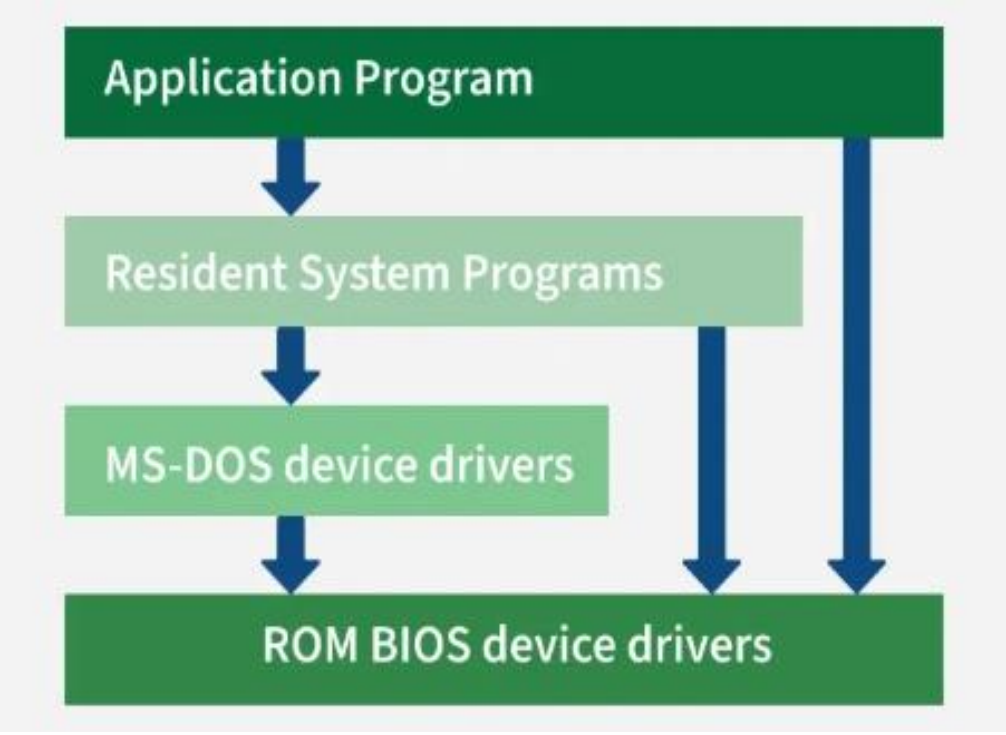

Fig. Simple OS Structure

Advantages of Simple Structure

- It is easy to develop as it contains only a few interfaces and levels.
- It executes faster because there are fewer layers between the hardware and applications.
- Direct access to hardware provides faster operations.
- Minimal consumption of system resources as the structure is very simple.
- Light in weight hence suitable for small or even embedded systems.

Disadvantages of Simple Structure

- All things are connected, and therefore, debugging is quite hectic; finding the error is very hard.
- Due to the lack of data abstraction, a security vulnerability is a strong possibility.
- It does not scale well to larger, more complex systems. Adding new features is particularly challenging in this architecture.
- One failure in one part might bring down the whole system.

Monolithic Structure:

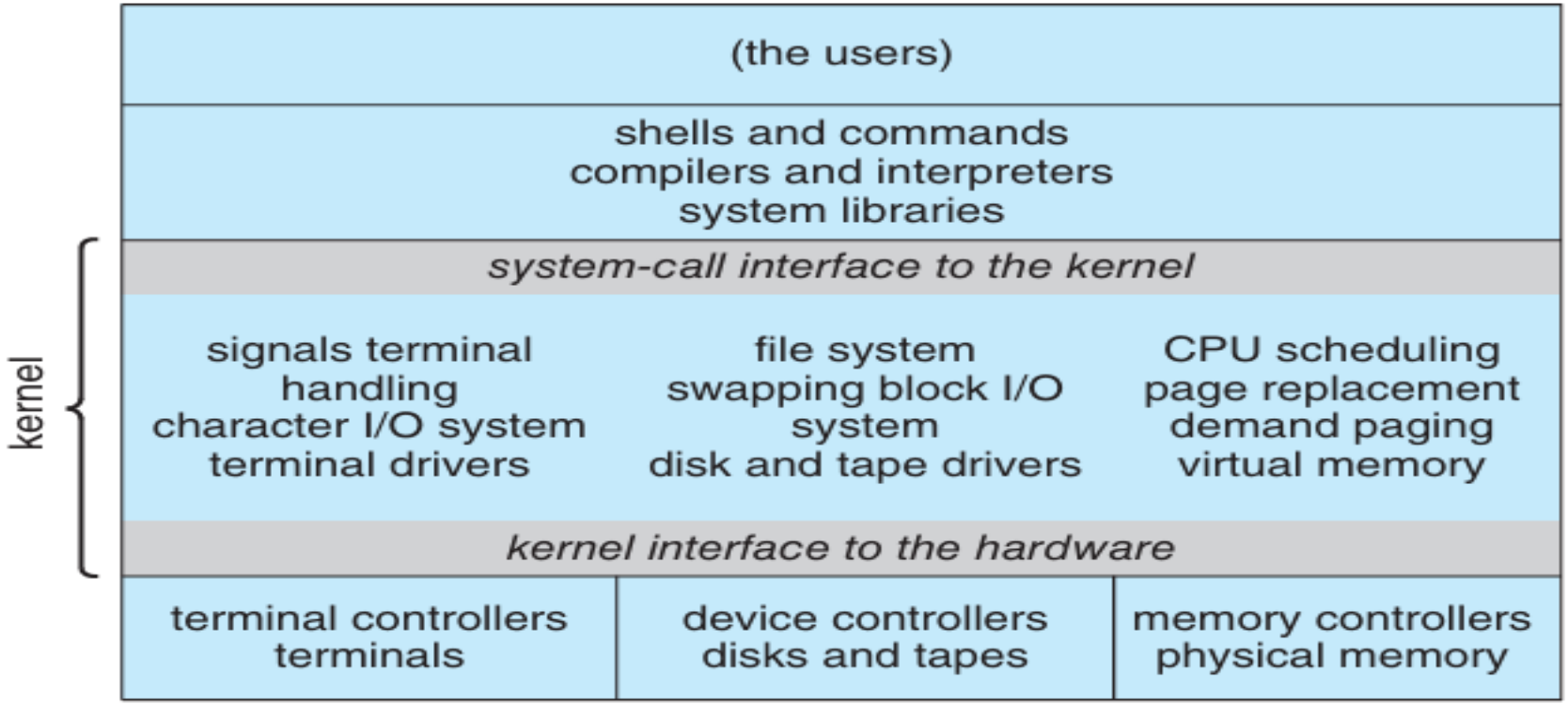

Fig. Traditional UNIX system structure

- A **monolithic structure** is a type of operating system architecture where the entire operating system is implemented as a single large process in kernel mode. Essential operating system services, such as process management, memory management, file systems, and device drivers, are combined into a single code block..

- The core of an operating system for computers is called the kernel (OS). All other System components are provided with fundamental services by the **kernel**.

- The operating system and the hardware use it as their main interface. When an operating system is built into a single piece of hardware, such as a keyboard or mouse, the kernel can directly access all of its resources.

- This is an old operating system that was used in banks to carry out simple tasks like batch processing and time-sharing, which allows numerous users at different terminals to access the Operating System.

- The monolithic approach is often known as a tightly coupled system because changes to one part of the system can have wide-ranging effects on other parts.

- Alternatively, we could design a loosely coupled system. Such a system i

Advantages of Monolithic Structure

- Because everything is packed into one big kernel, there is no need for multi-layer complexities. Hence, the design and bringing into practice of such a system is relatively easy.
- As long as all the components run in the same memory space, there is no need to have inter-process communication or to perform context switching between layers; thus, execution is faster.
- Direct access to hardware and little additional-abstraction overhead result in high performance.
- Development can thus be easier as developers have to deal with one codebase only, and no interactions among various layers or modules need to be considered.

Disadvantages of Monolithic Structure

- Fault in any single component has the potential to bring down the whole system since there is no isolation between different parts.
- This can be a hard and unsafe task, as changes in one part of the system may accidentally filter out into another part.
- Since the components are more interdependent, debugging is also more complex and always requires heavy testing and testing-related activities.
- This is an issue where scaling up or adjusting to the latest trends in a system involves significant changes in the kernel.
- Consequently, it will be cumbersome to update or upgrade the system because it involves a large codebase in a monolithic form where changes may have widespread effects.

Layered Structure:

- Alternatively to Monolithic Structure, we could design a **loosely coupled system**. Such a system is divided into separate, smaller components that have specific and limited functionality.

- All these components together comprise the kernel. The advantage of this modular approach is that changes in one component affect only that component, and no others, allowing system implementers more freedom in creating and changing the inner workings of the system.

- A system can be made modular in many ways. One method is the **layered approach**, in which the operating system is broken into a number of layers (levels).

Layered Structure:

- The OS is separated into layers or levels in this kind of arrangement. Layer 0 (the lowest layer) contains the hardware, and layer 1 (the highest layer) contains the user interface (layer N).

- The main advantage of the layered approach is simplicity of construction and debugging. The layers are selected so that each uses functions (operations) and services of only lower-level layers.

- This approach simplifies debugging and system verification.

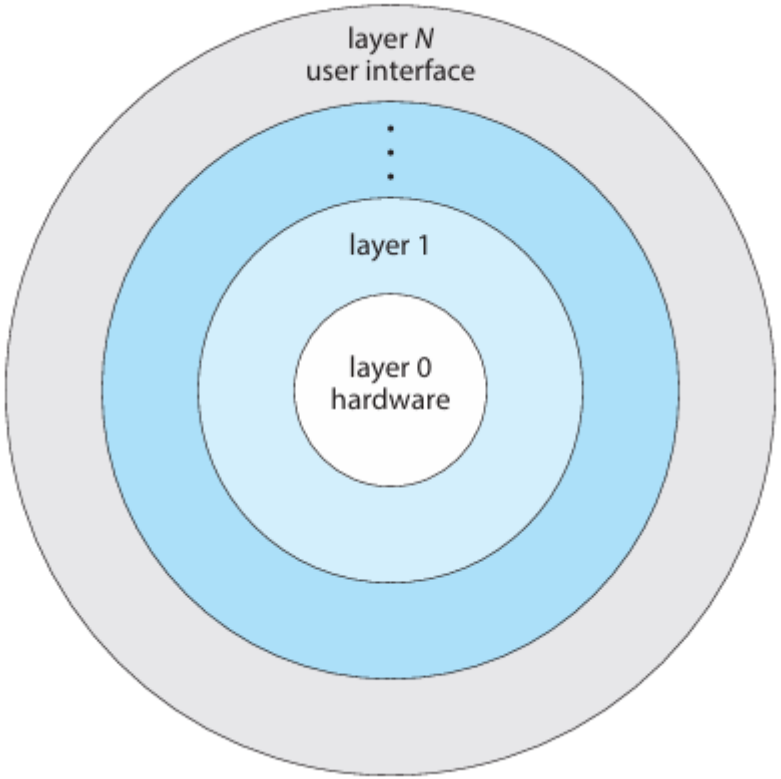

Fig. A layered operating system

- The first layer can be debugged without any concern for the rest of the system, because, by definition, it uses only the basic hardware (which is assumed correct) to implement its functions.

- Once the first layer is debugged, its correct functioning can be assumed while the second layer is debugged, and so on.

- If an error is found during the debugging of a particular layer, the error must be on that layer, because the layers below it are already debugged. Thus, the design and implementation of the system are simplified.

- Each layer is implemented only with operations provided by lower-level layers. A layer does not need to know how these operations are implemented; it needs to know only what these operations do. Hence, each layer hides the existence of certain data structures, operations, and hardware from higher level layers.

Advantages of Layered Structure

- Development, maintenance, and understanding are much easier, as every layer is assigned a specific role.
- It can isolate problems in individual layers because testing and debugging would be less demanding.
- Sensitive operations can be confined to specific layers; thus, unauthorized access can be minimized to such layers because the concerns have been clearly separated.
- Modifying or updating one layer will not affect other independent layers directly; thus, upgrading or fixing could be done more easily.
- With layers, basic operations can be entrusted to lower layers, and hence, the layers above need not be concerned with the minute details, thereby reducing the complexity of the overall management.

Disadvantages of Layered Structure

- The layered approach adds delays due to the fact that data and commands must pass through several layers, thus slowing down the system in general.
- Layers are generally dependent on lower layers; thus skipping layers and directly accessing the lower-level functionalities becomes harder, hence reducing flexibility in many systems.
- System performance optimization becomes challenging due to the extra abstractions introduced by the layers, which harden the tuning of individual parts of the system to make them run more effectively.
- Unless the lower-layer design is lousy or inefficient, the whole system will suffer since the higher layer depends on them for functionality.

Operating System Services

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

- User interface
- Program execution
- I/O operations
- File-system manipulation
- Communications
- Error detection

Detailed Description

- **User interface**. Almost all operating systems have a **user interface** (UI). This interface can take several forms. Most commonly, a **graphical user interface** (GUI) is used. Here, the interface is a window system with a mouse that serves as a pointing device to direct I/O, choose from menus, and make selections and a keyboard to enter text. Mobile systems such as phones and tablets provide a **touch-screen interface**, enabling users to slide their fingers across the screen or press buttons on the screen to select choices. Another option is a **command-line interface** (CLI), which uses text commands and a method for entering them (say, a keyboard for typing in commands in a specific format with specific options). Some systems provide two or all three of these variations.
- **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).
- **I/O operations**. A running program may require I/O, which may involve a file or an I/O device. For specific devices, special functions may be desired (such as reading from a network interface or writing to a file system). For efficiency and protection, users usually cannot control I/O devices directly. Therefore, the operating system must provide a means to do I/O.
- **File-system manipulation**. The file system is of particular interest. Obviously, 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. Finally, 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,

Detailed Description

- **Communications**. There are many circumstances in which 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 tied together by a 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.

- **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 or an attempt to access an illegal memory location). 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.

Operating System Services

Another set of operating-system functions exists not for helping the user but rather for ensuring the efficient operation of the system itself. Systems with multiple processes can gain efficiency by sharing the computer resources among the different processes.

- Resource allocation.
- Logging
- Protection and security

Detailed Description

- **Resource allocation**. When there are multiple processes running at the same time, resources must be allocated to each of them. The operating system manages many different types of resources. Some (such as CPU cycles, main memory, and file storage) may have special allocation code, whereas others (such as I/O devices) may have much more general request and release code. For instance, in determining how best to use the CPU, operating systems have CPU-scheduling routines that take into account the speed of the CPU, the process that must be executed, the number of processing cores on the CPU, and other factors. There may also be routines to allocate printers, USB storage drives, and other peripheral devices.

- **Logging**. We want to keep track of which programs 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. Usage statistics may be a valuable tool for system administrators who wish to reconfigure the system to improve computing services.

Detailed Description

- **Protection and security**. The owners of information stored in a multiuser or networked computer system may want to control use of that information. When several separate processes execute concurrently, it should not be possible for one process to interfere with the others or with the operating system itself. 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 himself or herself to the system, usually by means of a password, to gain access to system resources. It extends to defending external I/O devices, including network adapters, from invalid access attempts and recording all such connections for detection of break-ins. If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

System Boot

After an operating system is generated, it must be made available for use by the hardware. But how does the hardware know where the kernel is or how to load that kernel? The process of starting a computer by loading the kernel is known as booting the system.

- A small piece of code known as the bootstrap program or boot loader locates the kernel.
- The kernel is loaded into memory and started.
- The kernel initializes hardware.
- The root file system is mounted

 # The root file system is the top of the hierarchical file tree. It contains the files and directories critical for system operation, including the device directory and programs for booting the system.

- Some computer systems use a multistage boot process: When the computer is first powered on, a small boot loader located in nonvolatile firmware known as **BIOS** is run. This initial boot loader usually does nothing more than load a second boot loader, which is located at a fixed disk location called the **boot block**.

- Many recent computer systems have replaced the BIOS-based boot process with **UEFI (Unified Extensible Firmware Interface)**. UEFI has several advantages over BIOS, including better support for 64-bit systems and larger disks. Perhaps the greatest advantage is that UEFI is a single, complete boot manager and therefore is faster than the multistage BIOS boot process.

- Whether booting from BIOS or UEFI, the bootstrap program can perform a variety of tasks. In addition to loading the file containing the kernel program into memory, it also runs diagnostics to determine the state of the machine —for example, inspecting memory and the CPU and discovering devices. If the diagnostics pass, the program can continue with the booting steps. The bootstrap can also initialize all aspects of the system, from CPU registers to device controllers and the contents of main memory.

- Sooner or later, it starts the operating system and mounts the root file system. It is only at this point is the system said to be **running.**

References

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, “Operating System Concepts,” Eleventh Edition (Willey).

2. https://www.geeksforgeeks.org/.

3. https://www.javatpoint.com/.

4. https://www.tutorialspoint.com/.