Operating System Structure

Md. Asifur Rahman

Lecturer

Dept. of CSE, RUET

Operating System

An **operating system** is a program that manages a computer's hardware and provides a basis for application programs that acts as an intermediary between the computer user and the computer hardware.

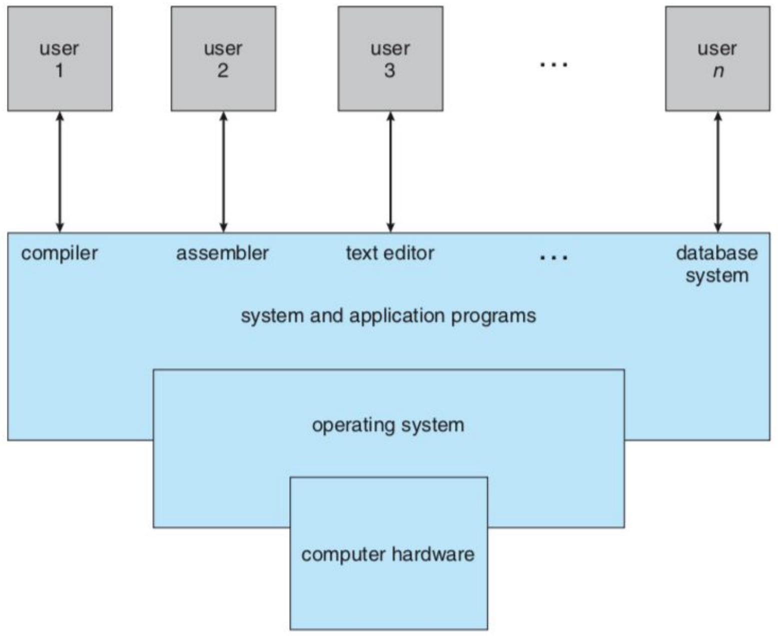
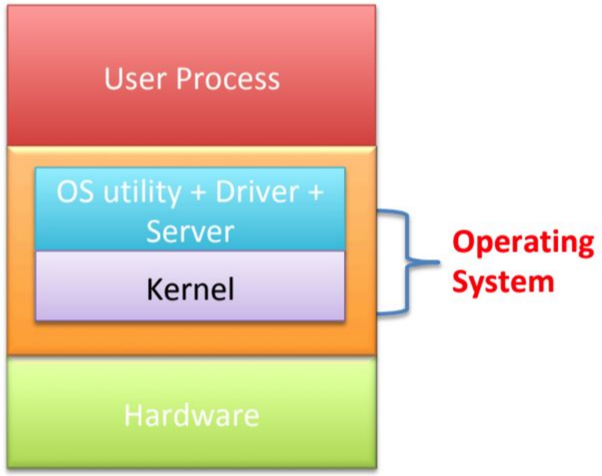


Figure Abstract view of the components of a computer system.



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.

Operating System Services

- User Interface:
    - Command line interface
        - Command line interpreter Bourne Shell, Korn shell etc
    - Graphical User interface (GUI)
    - Batch interface
- Program Execution:
    - Load , Run, Terminate , Indicate error (Memory Dump , Error Message)
- I/O Operation:
- File system manipulation
- Communication:
    - Share memory, Message Passing

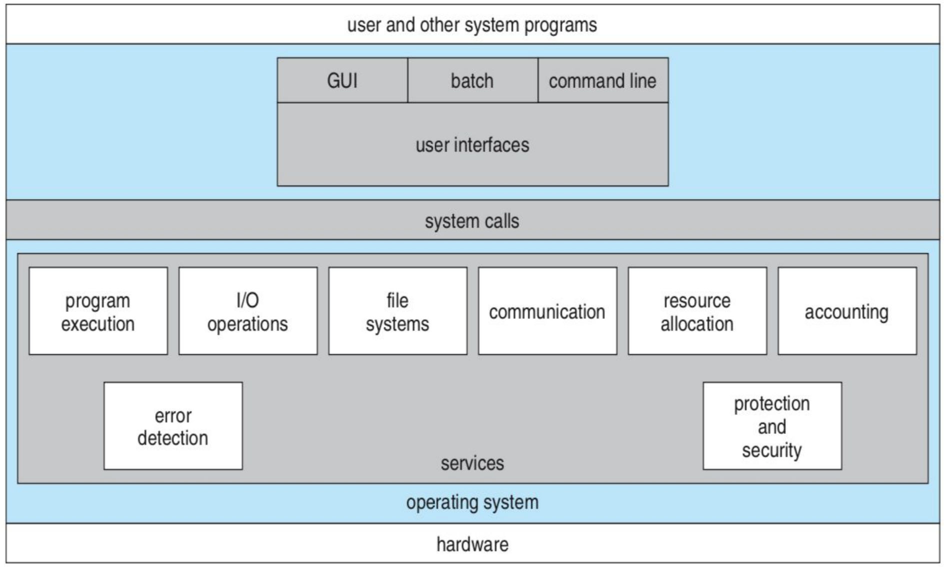


Figure A view of operating system services.

Operating System Services

- Error Detection
- Resource allocation
- Protection and security
- Accounting

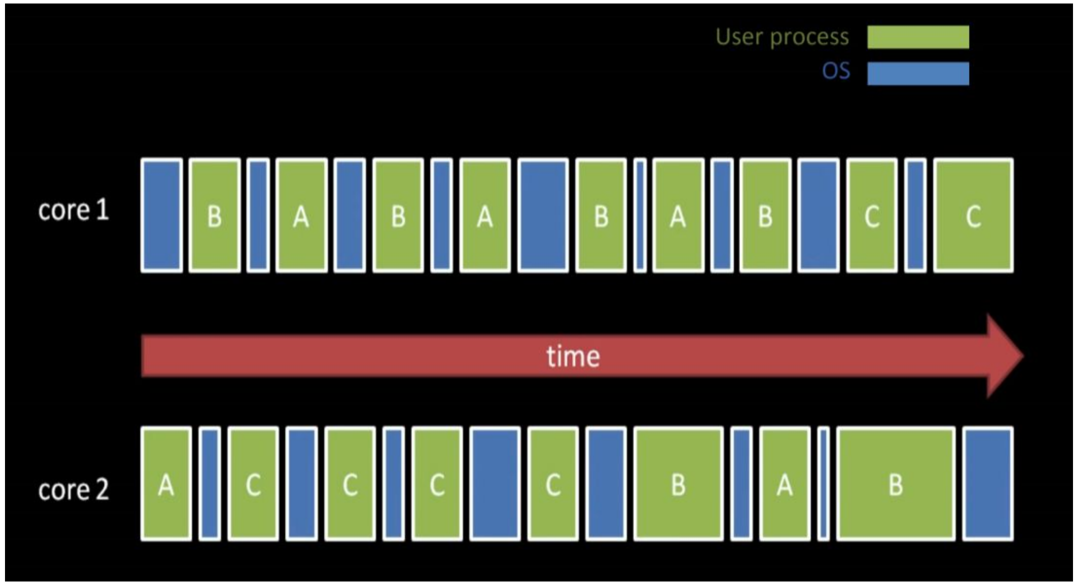


Fig: Resource Allocation

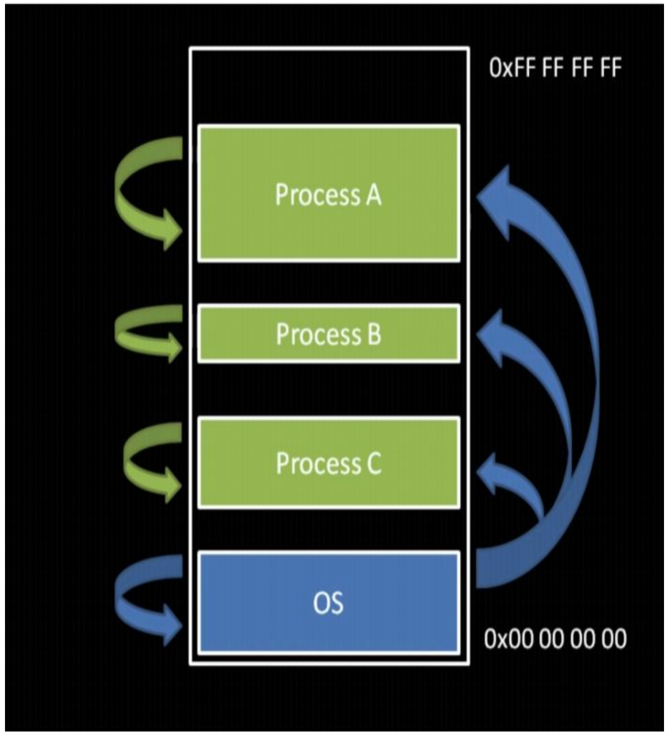
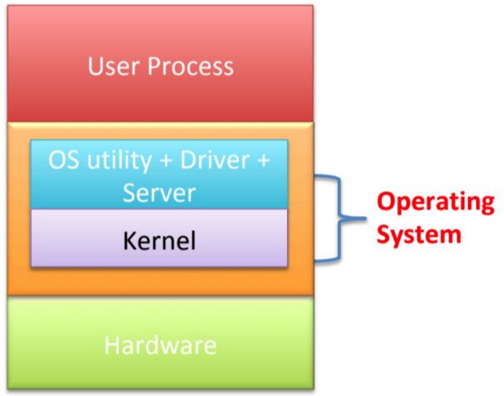


Fig: System call service

Kernel

Usually an operating system can be specifically traced back to one core program named **Kernel** which is running at all times on the computer. A Kernel is core component of an operating system. It contains the **privileged instruction** as well as other general instructions which allows it to provide core features such as:

- Program execution.
- Memory management and protection.
- Resource allocation
- Communication.



- Usually there are three types of program in a system: **Kernel**, **System program**, **Application Program**.
- System programs are associated with the operating system but are not necessarily part of the kernel.
- Application programs (User program) includes all programs that are not associated with the operating system.
- Mobile operating systems comes along with another program named **middleware** beside the core kernel that contains a set of software frameworks that provide additional services (databases, multimedia, and graphics) to application developers.

CPU Execution Mode

There are two modes of execution, known as **user mode** and **kernel or supervisor mode**.

Kernel Mode/ Supervised Mode:

- Execution of all machine instructions is allowed.
- Reference to all memory locations is allowed.

User mode:

- Only a subset of instructions is allowed to be executed.
- Reference to a certain (own address space) memory location is permitted.

CPU Execution Mode

Most computer systems provide hardware support that allows us to differentiate among user mode and kernel mode of execution.

A bit, called the **mode bit**, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).

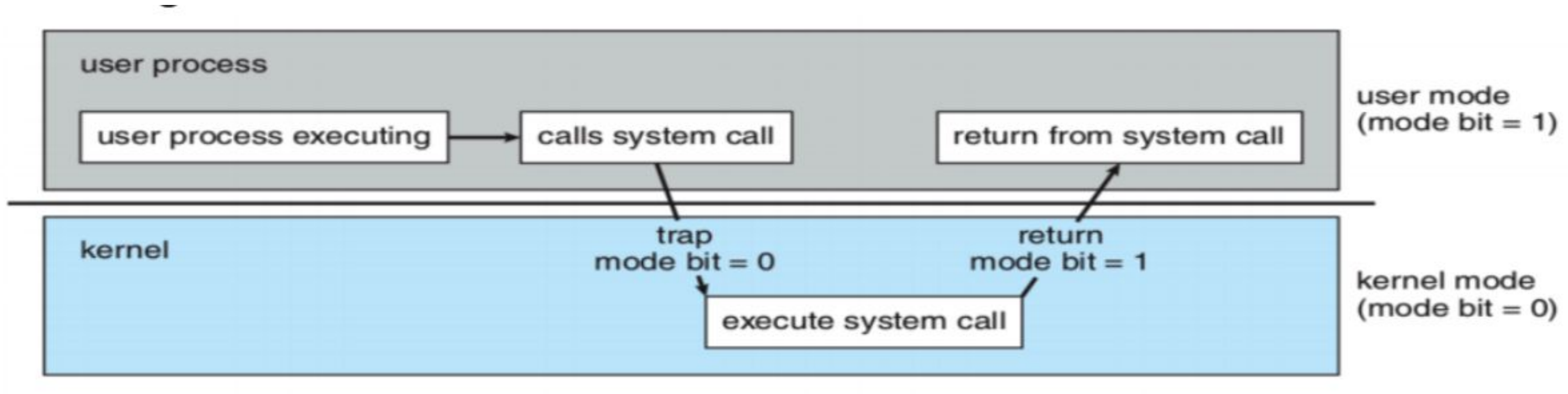


Figure Transition from user to kernel mode.

System Calls

System calls provide an interface to the services made available by an operating system. Since system calls are executed in kernel mode, they are allowed to execute any machine instruction and access any memory location.

- Operating system makes system calls available to the user program through the Application Programming Interface **(API)** or System call interface.
    - **API** specifies a set of functions that are available to an application programmer, including the parameters that are passed to each function and the return values the programmer can expect.
        - Three most common APIs are:
            - Windows API for windows system
            - POSIX API for POSIX based system (UNIX, LINUX, OS X)
            - JAVA API for java virtual machine.
    - **System call interface** is provided by the run-time support system of a programming language which, intercepts function calls in the API and invokes necessary system calls in the OS.

System Calls

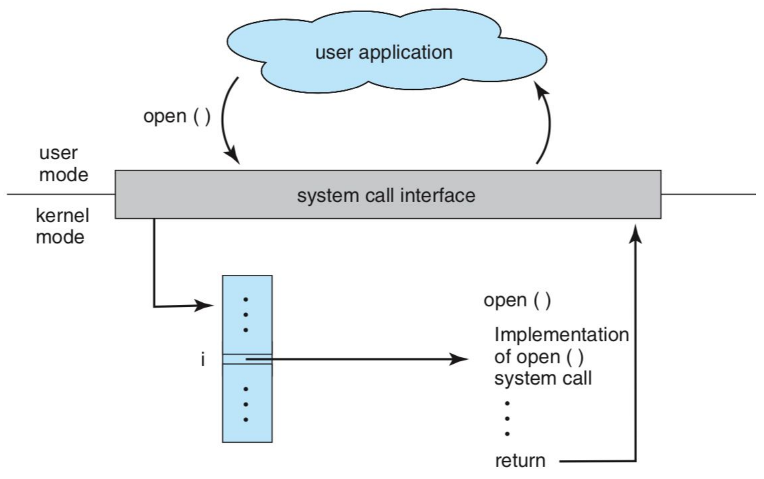


Figure 2.6 The handling of a user application invoking the `open()` system call.

- Why not use system call directly ?
  - Program Portability: Program written using an API can be expected to compile and run on any system that supports the same API.
  - Actual system calls are difficult to work with.

Overview of Computer System Operation

A brief overview of a computer System Operation (having Linux OS) is given below:

- When a CPU receives a reset event for instance, when it is powered up or rebooted the **instruction register** is loaded with instruction form a predefined memory location and starts execution there. At that location is the initial **BIOS (Basic Input Output System) program**. This program is usually stored in read-only memory (ROM) or (EEPROM), because the RAM is in an unknown state at system startup.

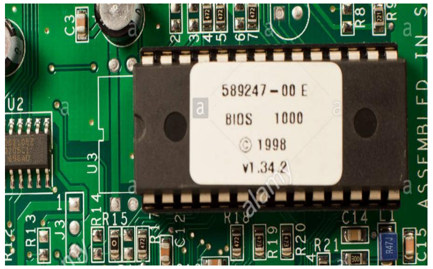


Figure: BIOS ROM Chip

ROM is convenient because it needs no initialization and cannot easily be infected by a computer virus. Program stored in ROM or EEPROM is called by the name **firmware.**

- During execution of the **BIOS** it usually run diagnostics to determine the state of the machine. If the diagnostics pass, the program can continue with the booting steps.

Overview of Computer System Operation

- Since **BIOS** is usually a very simple program it needs to load a more sophisticated program named **boot loader** into memory and execute it to load and execute the OS kernel.
- For this purpose **BIOS** has a bit of code that can load a single block from **Master boot record** (a fixed location say block zero) from **disk** into memory and execute the code from that **Master boot record. Master boot record** contains a **DOS** (Disk Operating System) **partition table** and **boot loader.**
- **Master boot record** start execution of the **boot loader** (for example linux **grub**).
- **Boot loader** locates the **OS kernel** location on the disk and loads the kernel of the operating system into memory and begin its execution.
- Once the kernel is loaded and executing, it can start providing services to the system programs and its users programs. In order to facilitate process load and execution, OS kernel upon its execution run a **system program** named **init** which is parent / root of all processes on linux. This enables creation of **processes** of any new **system programs** (system calls and others)

Overview of Computer System Operation

- Since OS in interrupt driven and there are three types of interrupt.
  - **Hardware Interrupts** are generated by hardware devices to signal that they need some attention from the OS.
  - **Software Interrupts** are generated by programs when they want to request a system call to be performed by the operating system.
  - **Traps** are generated by the CPU itself to indicate that some error or condition occured for which assistance from the operating system is needed.
- Next the program for **system program for interrupt handler** is executed and initiate the **interrupt table** (containing mapping of **interrupt number** with appropriate **interrupt service routine** (ISR)).
- Now system can facilitate execution in **kernel mode** and **user mode** separately since the ISR for setting the **hardware bit** has been initialized. Therefore OS kernel is now ready create user processes and provide services to **user program** in user mode through various **system call**.

▪ Change Processor's Mode of Execution – Kernel to User or vice versa

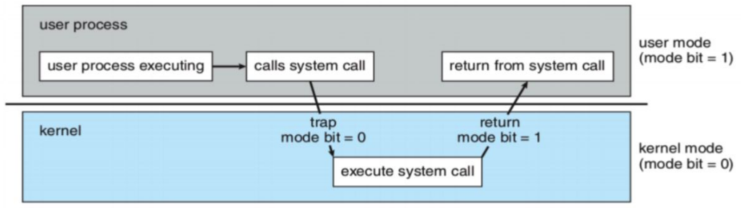


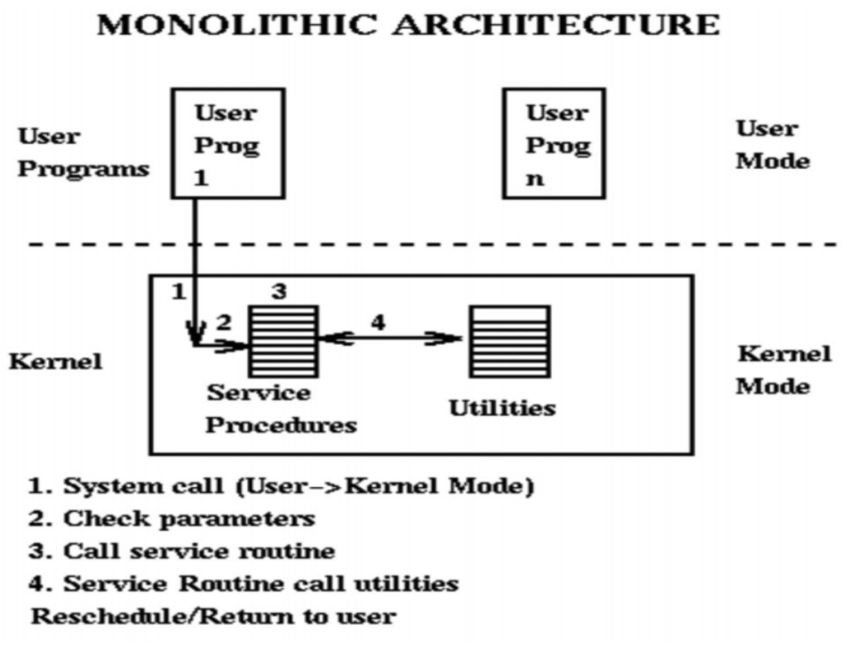
Figure Transition from user to kernel mode.

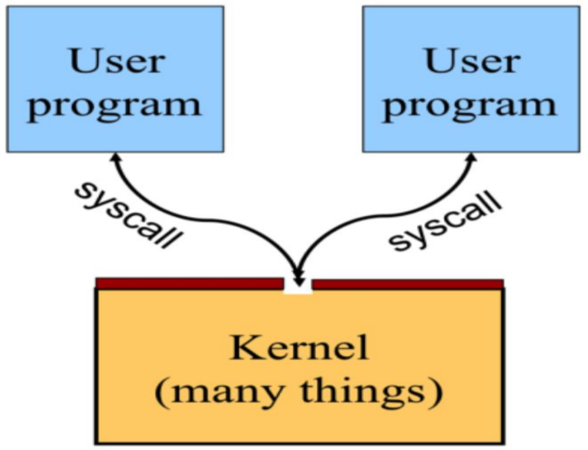
OS Organization

- **From here on by OS we will actually refer to the kernel itself.**
- OS can be categorized into the following six organization
  - Monolithic
  - Layered
  - Microkernel
  - Hybrid
  - Distributed
  - Virtual Machine Based

Monolithic

- The entire operating system runs as a single program in kernel mode. The operating system is written as a collection of procedures, linked together into a single large executable binary program.
- Each procedure in the system is free to call any other one. Hence thousands of procedures can call each other without restriction.





Monolithic

Example : BSD Unix, Windows

Advantage:

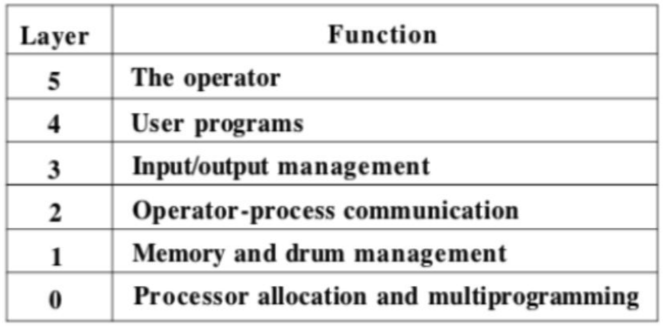
- Shared kernel space
- Easy to implement
- Good performance
- Low overheads

Disadvantage:

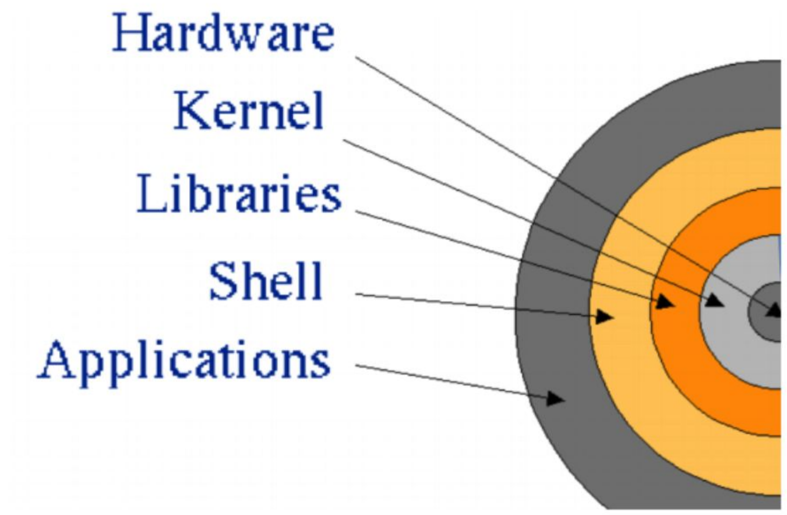
- Instability: crash in any procedure brings system down
- Inflexible / hard to maintain, extend
- One Single executable program.

Layered

- Components are divided into layers
  - ➢ Grouping similar components
- Interaction among layer:
  - ➢ The outer layer - request service
  - ➢ The inner layer – provide service



Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

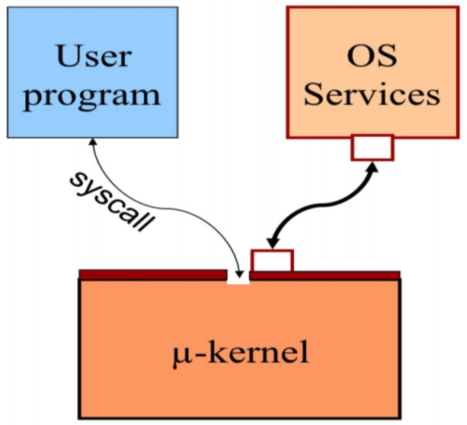


MULTICS

- Example: THE, MULTICS
- Advantages:
  - ▪ Layered Abstraction
- Disadvantages:
  - ▪ All the layers are linked together into a single executable program.

Microkernel

- The basic idea behind the microkernel design is to achieve high reliability by splitting the operating system up into small, well-defined modules - **microkernel** , **OS services**.
- In a microkernel, the **OS service** and **kernel** are implemented in different address space. The **OS services** are kept in user address space, and **microkernel** services are kept under kernel address space, thus also reduces the size of kernel and size of operating system as well.
- Only the **microkernel** runs in kernel mode and the rest run as relatively powerless ordinary user processes.



Microkernel

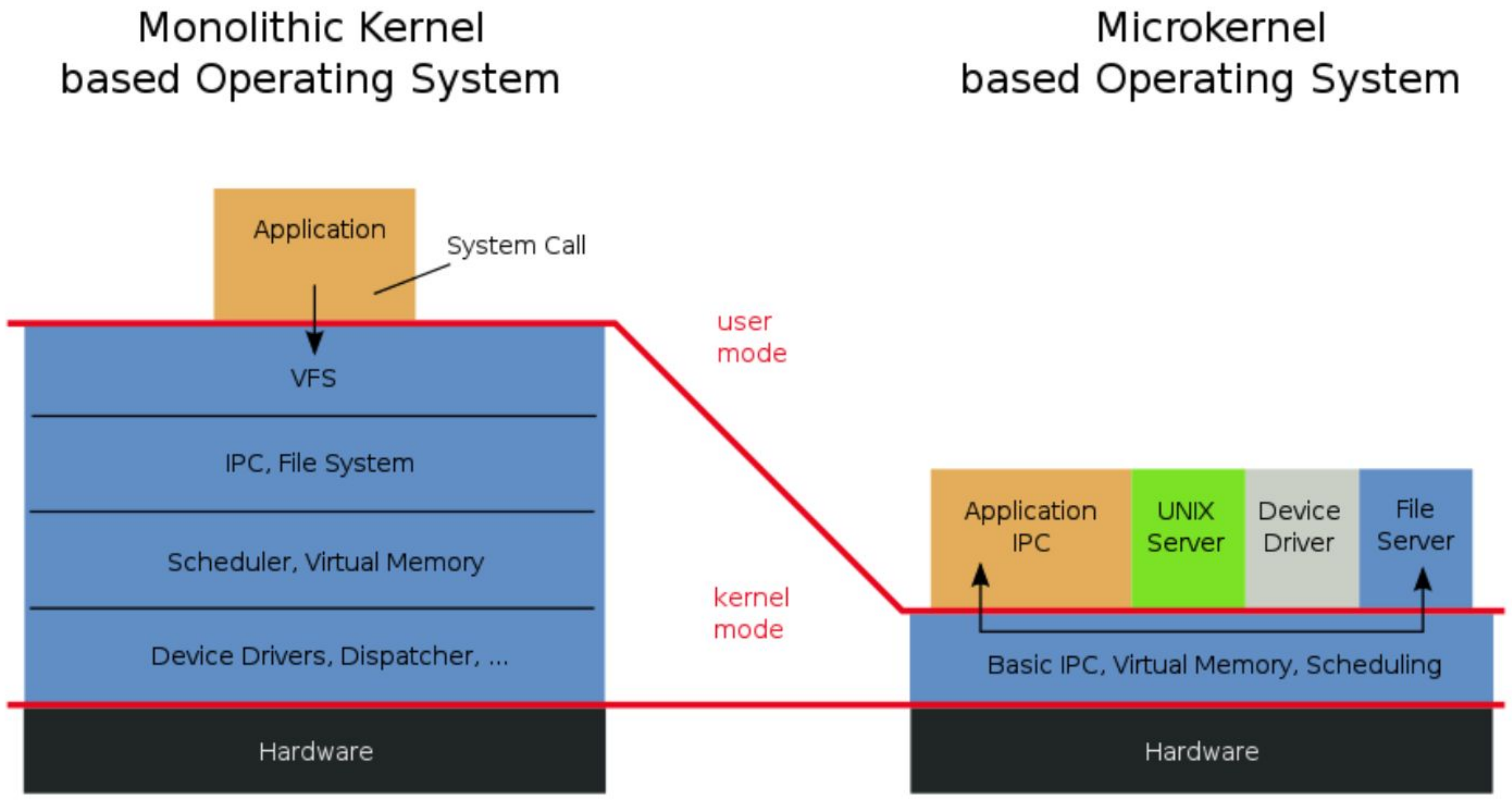
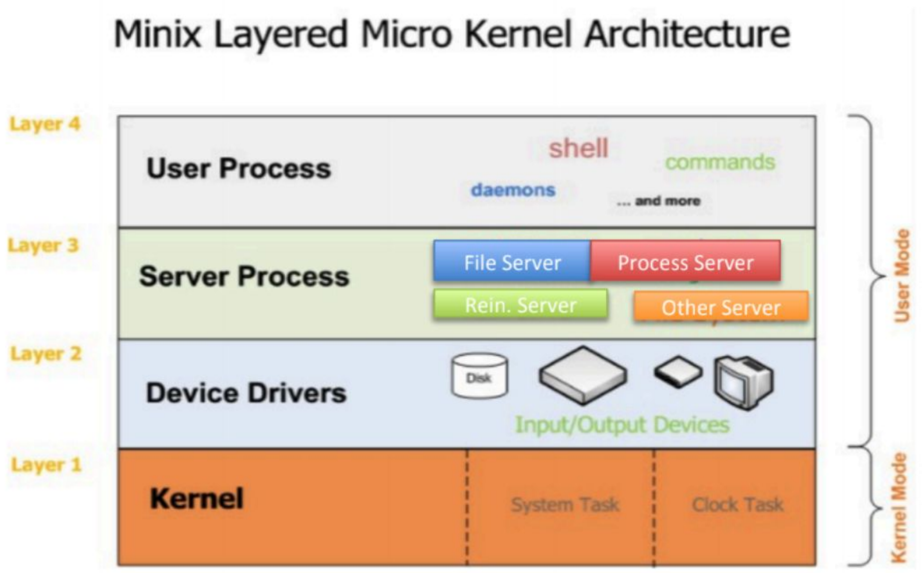

Monolithic Kernel based Operating System
Microkernel based Operating System
Application
System Call
VFS
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
user mode
kernel mode
Application IPC
UNIX Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling
Hardware

Figure : Structure of monolithic and microkernel-based operating systems, respectively (Source wikipedia)

Microkernel

Kernel Mode:

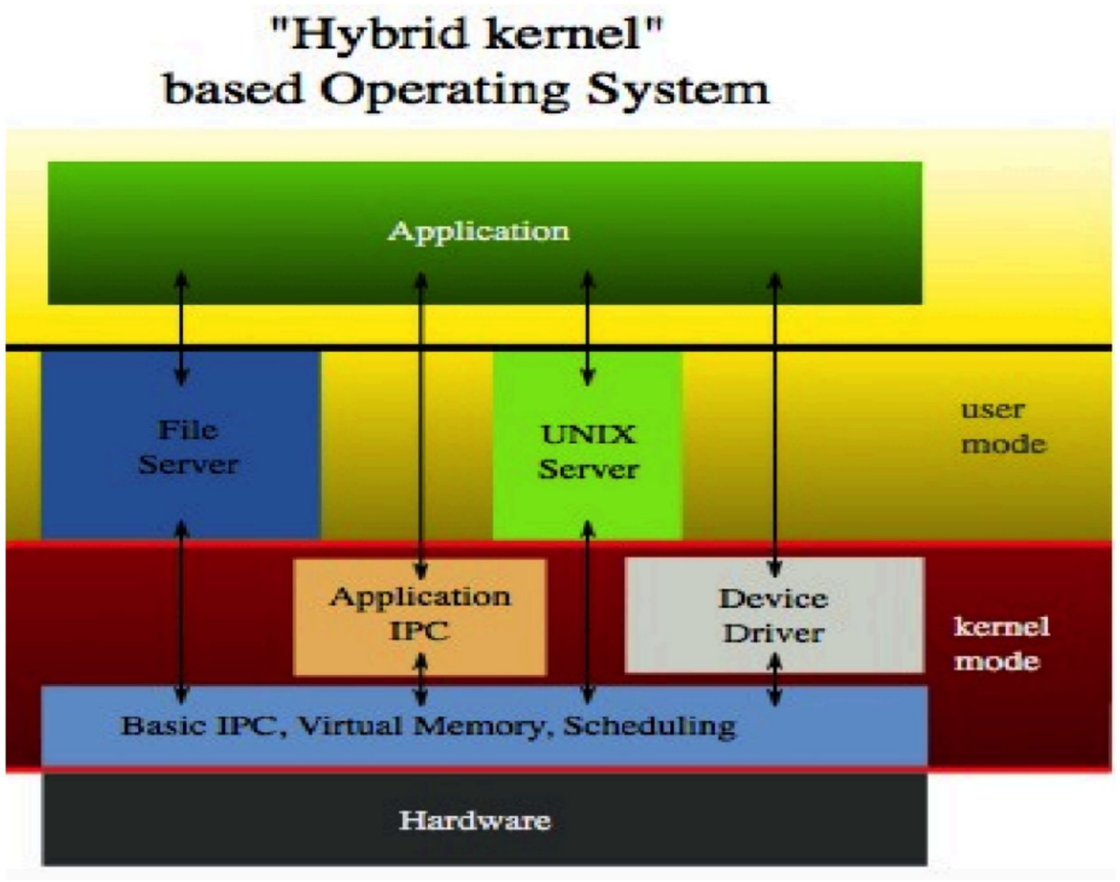
- **System Task:**
  - Catch interrupt
  - Switch process
  - Schedule process
  - Manage inter process communication
  - Provide other 35 system call service
- **Clock Task**: Contain driver for processor clock in order to assist scheduler.



User Mode:

- **Device Driver:** Do not have access to I/O port. Only packets the value to be written at a port.
- **Servers :**
  - File servers: manages file system
  - Process server: creates, destroys and manages process.
  - Reincarnation server:Checks if other server and driver are functioning correctly.

Hybrid


"Hybrid kernel"
based Operating System
Application
File Server
UNIX Server
user mode
Application IPC
Device Driver
kernel mode
Basic IPC, Virtual Memory, Scheduling
Hardware

Distributed

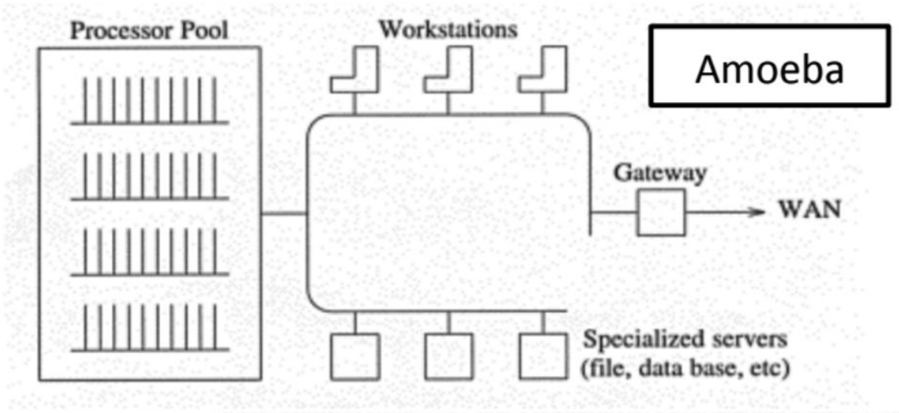
The Amoeba System Architecture Amoeba is a microkernel-based operating system. It offers multithreaded programs (shared memory) and a remote procedure call (RPC) mechanism for communication between threads, potentially across the network; even kernel- threads use this RPC mechanism for communication.

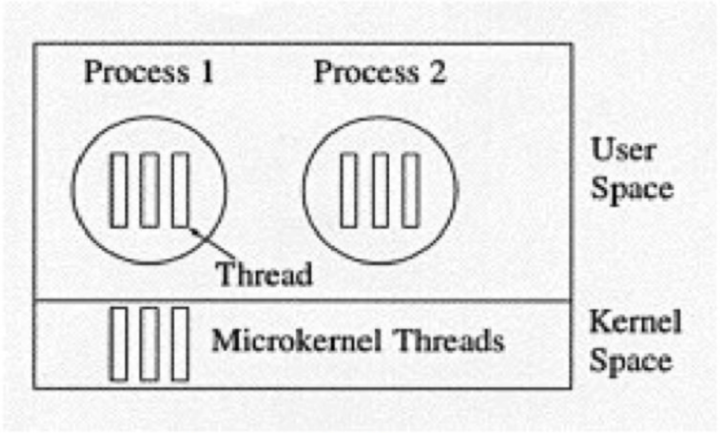
Four basic components:

1. **Workstation:** Each user has a access computer running Windows X or Unix Emulation
2. **Pool of processors** which are dynamically allocated to users as required.
3. **Specialized servers:** file, directory, database, etc.
4. **Interconnection:** These components were connected to each other by a fast LAN, and to the wide area network by a gateway.

Distributed

- The Bullet Server
  - Used for file storage
- The Directory Server
  - Used for file naming
  - Maps from names to capabilities
- The Replication Server
  - Used for fault tolerence and performance
- The Run Server
  - Run server manages the processor pools
- The Boot Server
  - Ensures that servers are up and running
  - If it discovers that a server has crashed,
    - it attempts to restart it, otherwise selects another processor to provide the service





Virtual Machine

- Provide an interface that looks like independent hardware -
  - To multiple different Oss' running simultaneously on the same physical hardware.
  - Each OS believes that it has access to and control over its own CPU, RAM, I/O devices, hard drives, etc.

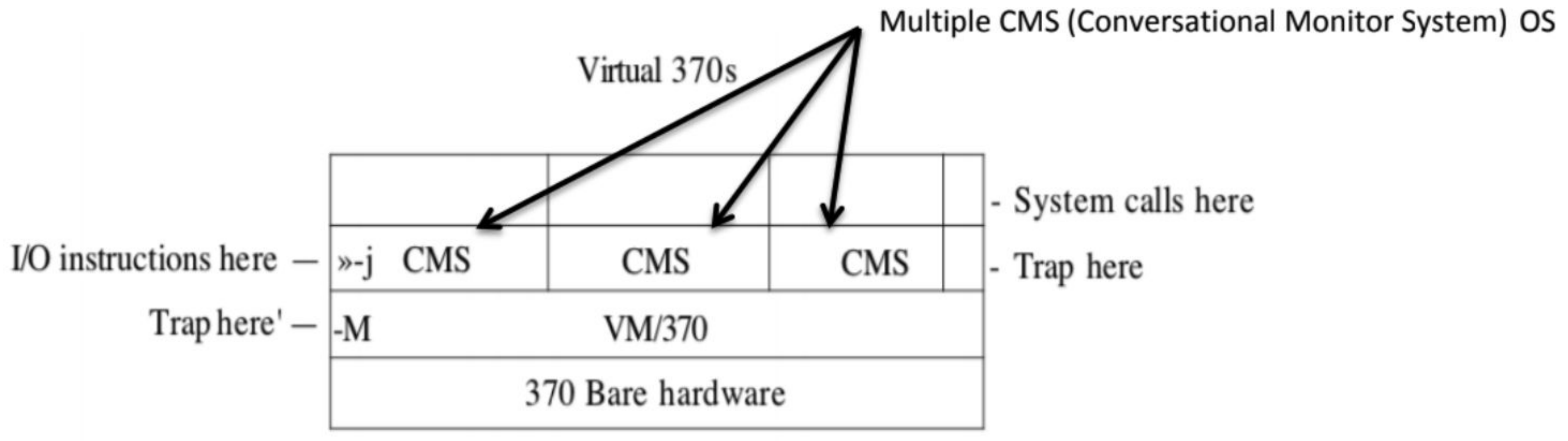


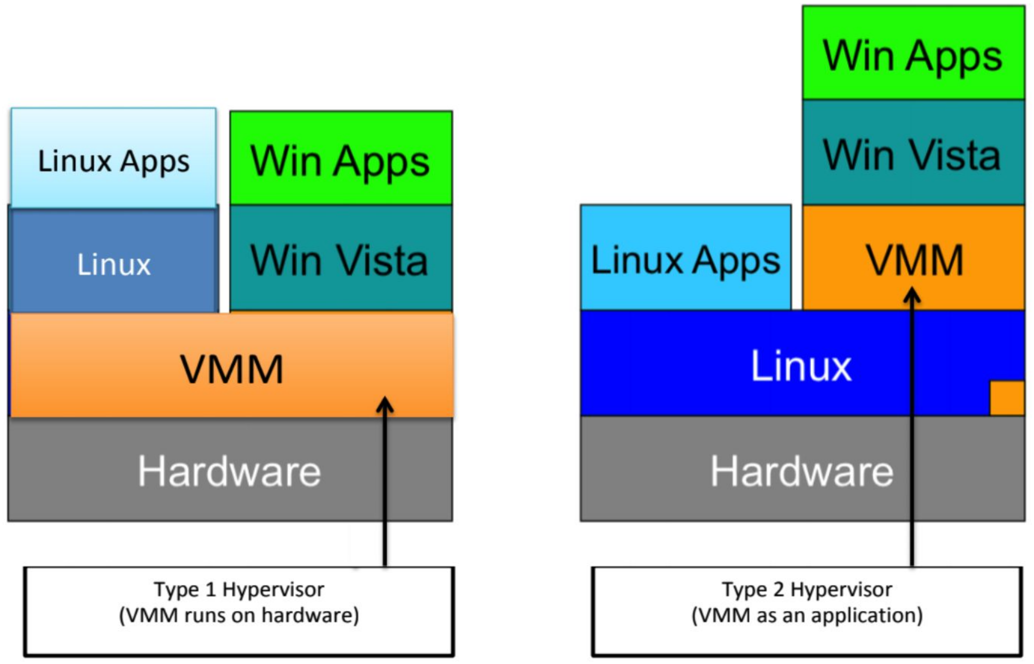
Figure The structure of ViM/370 with CMS.

Virtual Machine

• **Examples**: IBM VM/370, Java VM, VMWare

Two types of virtual machine organization is seen which are:

1. Type 1 hypervisor (VMM running on hardware)
2. Type 2 hypervisor (VMM as application running of OS)



References

Site reference:

- https://www.cs.princeton.edu/
- https://www.inf.ed.ac.uk
- https://fenix.tecnico.ulisboa.pt
- http://fsd-amoeba.sourceforge.net/amoeba.html
- http://www.yolinux.com/

Book Ref:

1. Modern Operating System – by Tanenbaum

2. Operating System Concepts - by Abraham Silberschatz