Introduction to Operating System (OS)

Course Content:

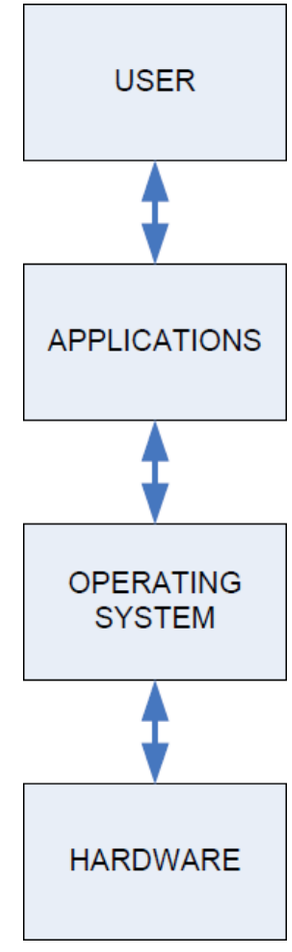
- What is an OS.
- What are its key functions.
- The evaluation of OS.
- What are the popular types of OS.
- Basics of UNIX and Windows.
- Advantages of open source OS like Linux.
- Networks OS.

What is an Operating System?

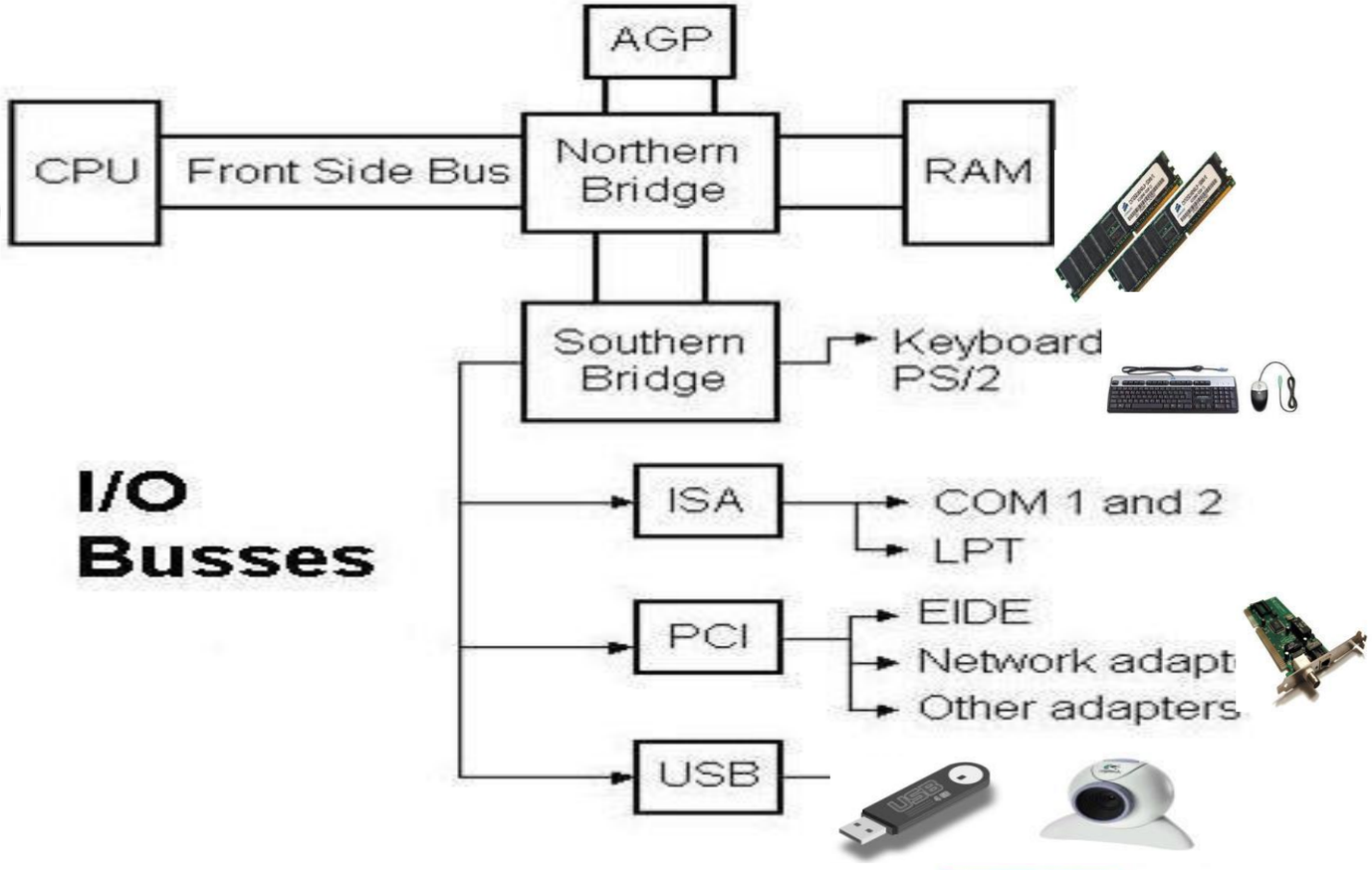
- Computer System = Hardware + Software
- Software = Application Software + System Software(OS)
- An Operating System is a system Software that acts as an intermediary/interface between a user of a computer and the computer hardware.
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

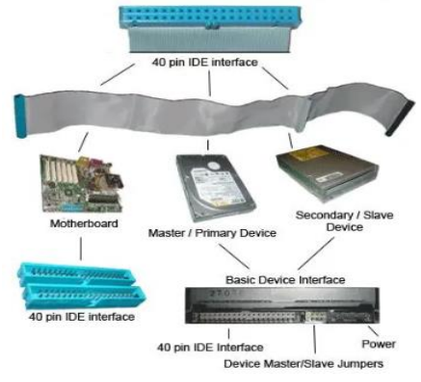
The Structure of Computer Systems

- Accessing computer resources is divided into ***layers.***
- Each layer is isolated and only interacts directly with the layer below or above it.
- If we install a new hardware device
  - ✓ No need to change anything about the user/applications.
  - ✓ However, you do need to make changes to the operating system.
  - ✓ You need to install the ***device drivers*** that the operating system will use to control the new device.
- If we install a new software application
  - ✓ No need to make any changes to your hardware.
  - ✓ But we need to make sure the application is supported by the operating system
  - ✓ user will need to learn how to use the new application.
- If we change the operating system
  - ✓ Need to make sure that both applications and hardware will compatible with the new operating system.



Computer Architecture





Special busses (roads) connecting all input/output devices to motherboard.

Accelerated Graphics Port (AGP) was a high-speed interface that connected graphics cards to a computer's motherboard

PS/2, or Personal System/2, is a 6-pin mini-DIN connector that was used to connect keyboards and mice to a PC compatible computer system

An Instruction Set Architecture (ISA) is part of the abstract model of a computer that defines how the CPU is controlled by the software.

Line Printer Terminal

Peripheral Component Interconnect (PCI) slots are utilized to install sound cards, Ethernet and remote cards

Enhanced Integrated Drive Electronics (EIDE) is a technology that connects a computer's motherboard to storage devices like hard drives and CD-ROM drives.

CPU – Central Processing Unit

- This is the brain of your computer.
- It performs all of the calculations.
- In order to do its job, the CPU needs commands to perform, and data to work with.
- The instructions and data travel to and from the CPU on the system bus.
- The operating system provides rules for how that information gets back and forth, and how it will be used by the CPU.

RAM – Random Access Memory

- This is like a desk, or a workspace, where your computer temporarily stores all of the information (data) and instructions (software or program code) that it is currently using.
- Each RAM chip contains millions of address spaces.
- Each address space is the same size, and has its own unique identifying number (address).
- The operating system provides the rules for using these memory spaces, and controls storage and retrieval of information from RAM.
- Device drivers for RAM chips are included with the operating system.

Problem: *If RAM needs an operating system to work, and an operating system needs RAM in order to work, how does your computer activate its RAM to load the operating system?*

Computer System: Software Systems

1. System Software
2. System Program Software/ Programming Software
3. Application Software

1. System Software

1. System Software:

System software coordinates the activities and functions of hardware and software, and it controls the operations of computer hardware. It consists of variety of programs that supports the operation of a computer. System software is computer software that provides the infrastructure over which programs can operate. This makes it possible for the users to focus on an application or other problems to be solved, without needing to know the details of how machine works internally.

Device Driver

- **Device Drivers:** A device driver is a computer program that controls a particular device that is connected to your computer. Typical devices are keyboards, printers, scanners, digital cameras and external storage devices. Each of these need a driver in order to work properly.

  Device drivers act as a translator between the operating system of the computer and the device connected to it. For many types of devices, the necessary drivers are built into the operating system. When you plug in a device, the operating system starts looking for the right driver, installs it and you are ready to start using the device. This is referred to as plug-and-play and is much preferred over having to manually install the correct drivers.

  There are so many different devices, however, that not all of them are built into the operating system. As an alternative, the operating system can look online to find the right driver to install. Many hardware devices, however, come with the necessary drivers. For example, if you buy a printer, it may come with a CD that typically will include the correct driver. The advantage of this is that the hardware manufacturer can make sure you have the right driver for the printer.

Operating System (OS)

- **Operating Systems:** OS is concerned with the allocation of resources and services, such as memory, processors, devices, and information. OS correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs and a file system. It is the software that runs in the background and brings the separate, physical parts of the computer together in order to provide the seamless stream of activity that a user experiences. Some of its responsibilities include the transfer of data between the memory and disks (on your hard drive) as well as providing the information needed to display icons, text, cursors and other visible necessities on the display screen. This display is called the graphical user interface or GUI and is entirely the result of the OS on the computer.

2. System Program Software/ Programming Software

System programs are the set of programs that are required for the effective execution of the application programs.

It usually provides tools to assist programmer in writing computer programs, and software using different programming languages

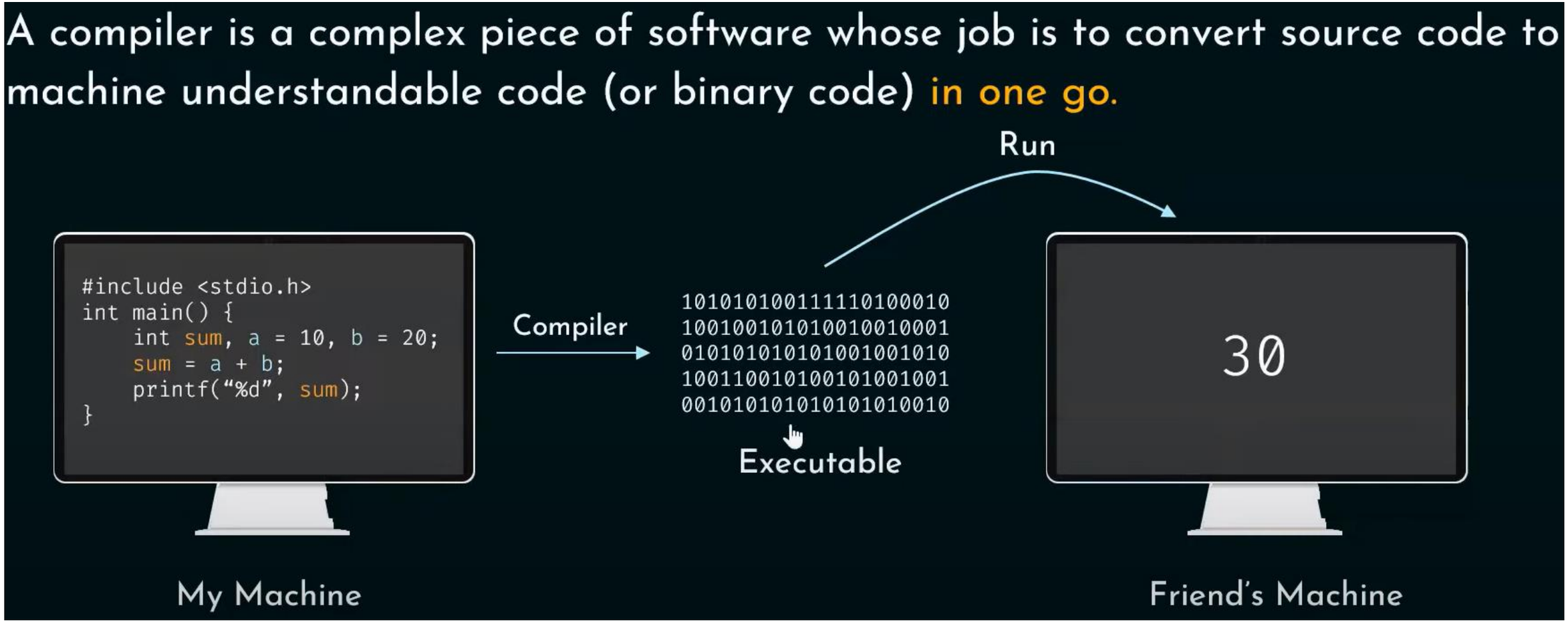
Programming Software Tools

- **Compiler:** Compiler is program that takes as input a program written in its source language (usually high level language like C++) and produces an equivalent program written in its target language (object code or machine code)

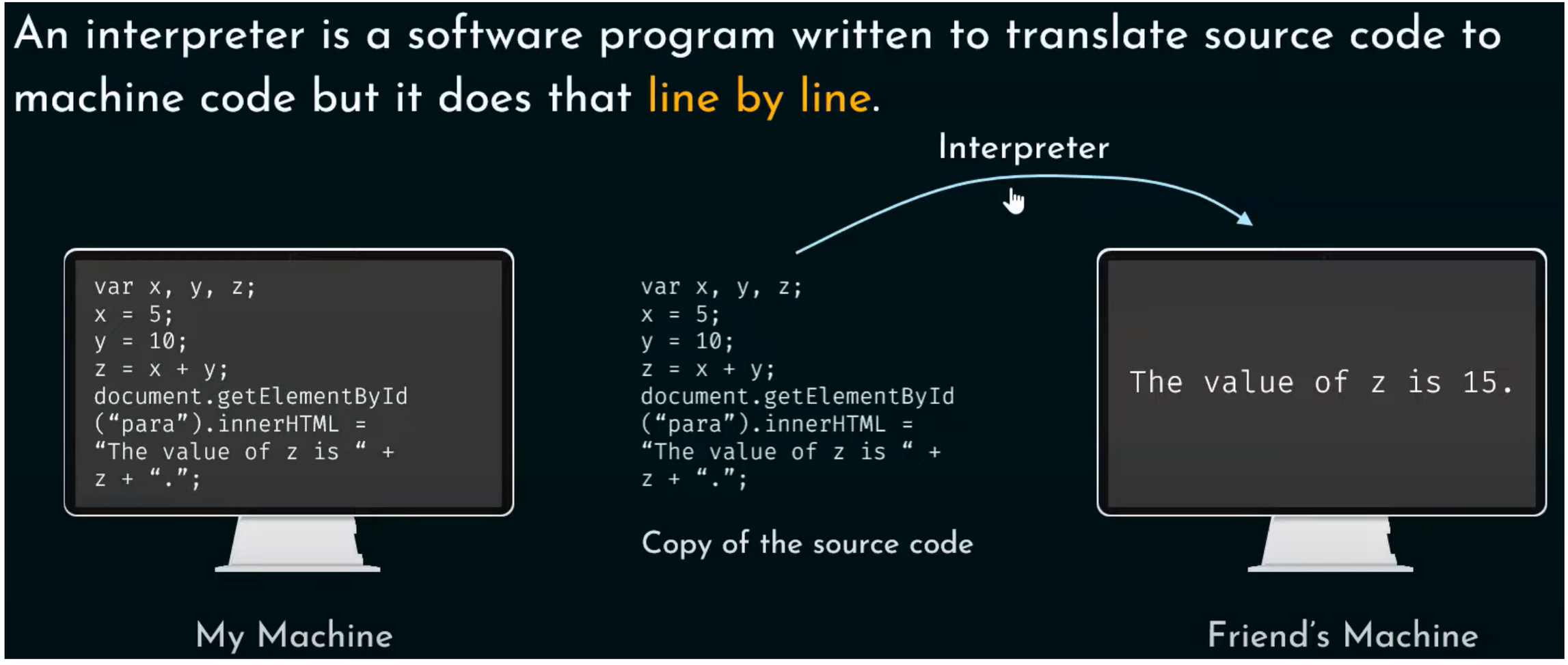
- **Assembler:** An assembler is a language translator for the assembly language of a particular computer. Input to an assembler is an assembly language program. Output is an object program plus information that enables the loader to prepare the object program for execution.

- **Linkers:** Both the compiler and assemblers rely on linker which a program that collects code separately compiled or assembled into a file that is directly executable. Linker also connects an object program to the code for standard library functions and to resources supplied by the OS of the computer, such as memory allocators and I/O devices.

Programming Software Tools



Programming Software Tools



Programming Software Tools

- **Loaders:** Compiler, assembler or linker will produce a code that is not yet completely fixed or ready to execute, but whose principal memory references are all made relative to an undetermined starting location that can be anywhere in the memory. Such code is said to be relocatable and a loader will resolve all relocatable addresses relative to a give base or starting address. In short, a loader is a program that places programs into memory and prepares them for execution. There are various loading schemes: absolute, relocating, and direct-linking. In general, the loader must load, relocate and link the object program.

- **Macro Processor:** A macro call is an abbreviation (or name) for some code. A macro definition is a sequence of code that has a name (macro call). A macro processor is a program that substitutes and specializes macro definitions for macro calls.

Programming Software Tools

- In C Programming: Concept of **Function.**
- Function is block of statements related to such task which we want to execute repeatedly in program.
- The function is defined once & can be repeatedly call whenever necessary.
- Suppose function call 100 times, Compiler transfer control call to definition & definition to call 100 times.

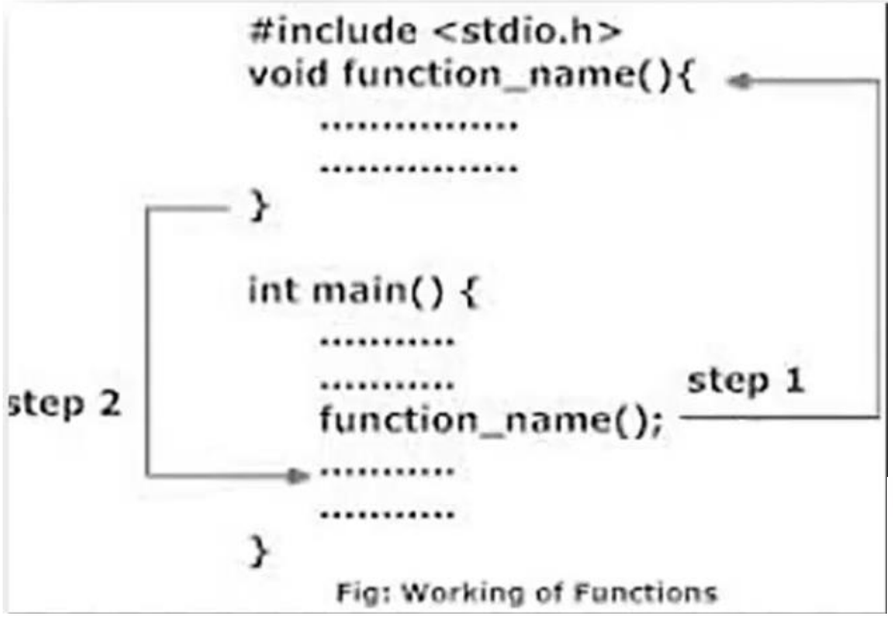


Fig: Working of Functions

```
MACRO    --------  Start of definition
INCR     --------  Macro name
A 1,DATA
A 2,DATA           ------Sequence of instructions to
  be
A 3,DATA                 abbreviated
MEND     --------  End of definition
```

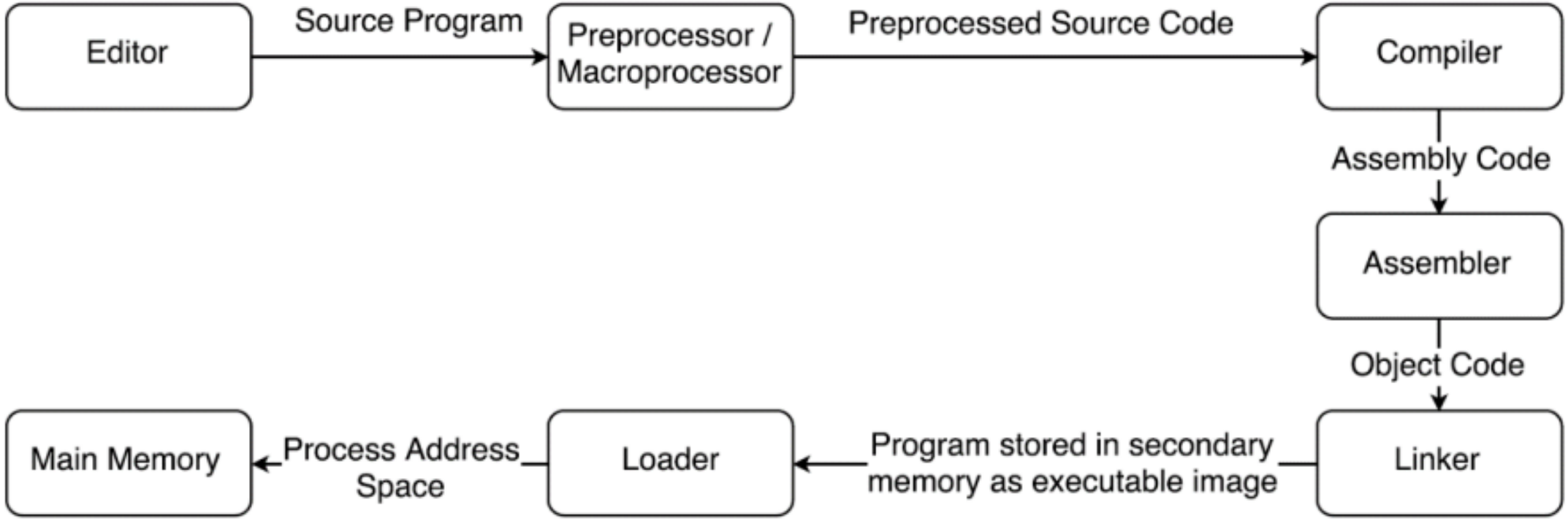
- Macro instructions are an extension of the basic assembly language that can simplify debugging and program modification during translation.
- To process macro instructions, most assembler use pre-processors known as Macro process

Programming Software Tools

- **Interpreter:** An interpreter translates high level instructions into an intermediate form. An interpreter may be program that either
  - Executes the source code directly
  - Translates source code into some efficient intermediate representation and executes them.

  E.g.: Interpreter of LISP, BASIC, etc.

Relative Locations of System Programming Software

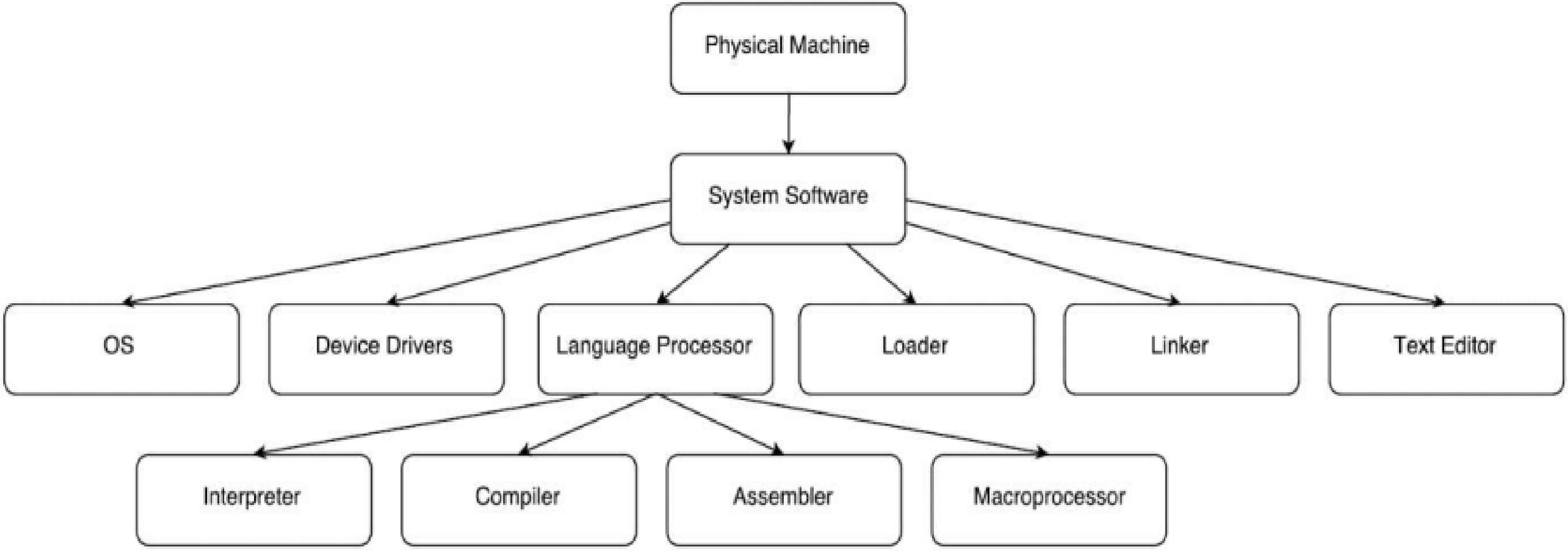


Comparison

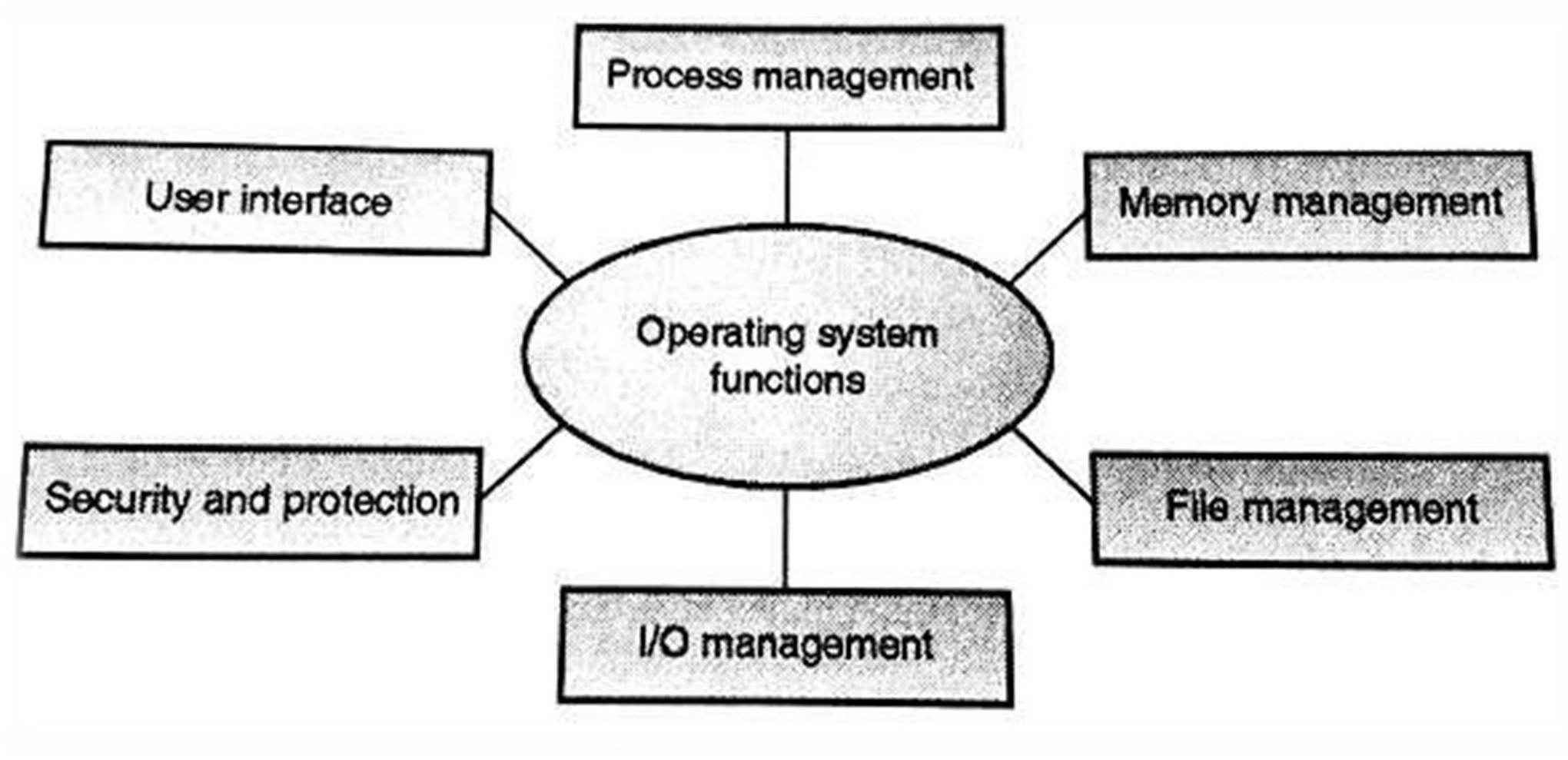
Comparison between Assembler, Compiler & Interpreter:

Assembler	Compiler	Interpreter
It converts the Assembly language into its machine language equivalent	It converts the source program to target program.	It converts line by line of the source program written in HLL into its equivalent machine code and executes if the program is error free.
Speed of execution is fast	Speed of execution is fast	Speed of execution id slow
Translation is done of the entire program	Translation is done of the entire program	Translation is done line by line
Program need be assembled	Program need be compiled only once and can be executed repeatedly	For every run of the program the program needs to be translated.
It creates an object file	It creates an object file	It does not create any object file
Microsoft Assembler (MASM)	MS-DOS C Compiler	Basic Interpreter.

System Software



Functions of Operating System



1. Process Management

- **A *process* is a program in execution.**
- A process needs certain resources, including CPU time, memory, files, and I/O devices to accomplish its task.
- Simultaneous execution leads to **multiple processes**. Hence creation, execution and termination of a process are the most basic functionality of an OS.
- If processes are dependent, then they may try to share same resources. thus task of **process synchronization** comes to the picture.
- If processes are independent, then a due care needs to be taken to avoid their **overlapping** in memory area.
- Based on priority, it is important to allow more important processes to execute first than others.

2. Memory management

- Memory is a large array of words or bytes, each with its own address.
- It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a **volatile** storage device. When the computer made ***turn off*** everything stored in RAM ***will be erased automatically***.
- In addition to the physical RAM installed in your computer, most modern operating systems allow your computer to use a ***virtual memory system***. *Virtual memory allows your computer to use part of a permanent storage device (such as a hard disk) as extra memory.*
- The operating system is responsible for the following activities in connections with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and de-allocate memory space as needed.

3. File Management

- A file is a **collection of related information** defined by its creator.
- ***File systems*** *provide the conventions for the encoding, storage and management of data on a storage device such as a hard disk.*
  - FAT12 (floppy disks) File allocation table
  - FAT16 (DOS and older versions of Windows)
  - FAT32 (older versions of Windows)
  - NTFS (newer versions of Windows) New Technology file system
  - EXT3 (Unix/Linux)
  - HFS+ (Max OS X) Hierarchical file system
- The operating system is responsible for the following activities in connections with file management:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - File backup on stable (nonvolatile) storage media.

4. Device Management or I/O Management

- ***Device controllers*** *are components on the motherboard (or on expansion cards) that act as an interface between the CPU and the actual device.*
- ***Device drivers,*** *which are the operating system software components that interact with the devices controllers.*
- A special device (inside CPU) called the **Interrupt Controller** handles the task of receiving interrupt requests and prioritizes them to be forwarded to the processor.
- **Deadlocks** can occur when two (or more) processes have control of different I/O resources that are needed by the other processes, and they are unwilling to give up control of the device.
- It performs the following activities for device management.
  - Keeps tracks of **all devices connected** to system.
  - Designates a program responsible for every device known as Input/output controller.
  - Decides which process gets access to a certain device and for how long.
  - Allocates devices in an effective and efficient way.
  - Deallocates devices when they are no longer required.

5. Security & Protection

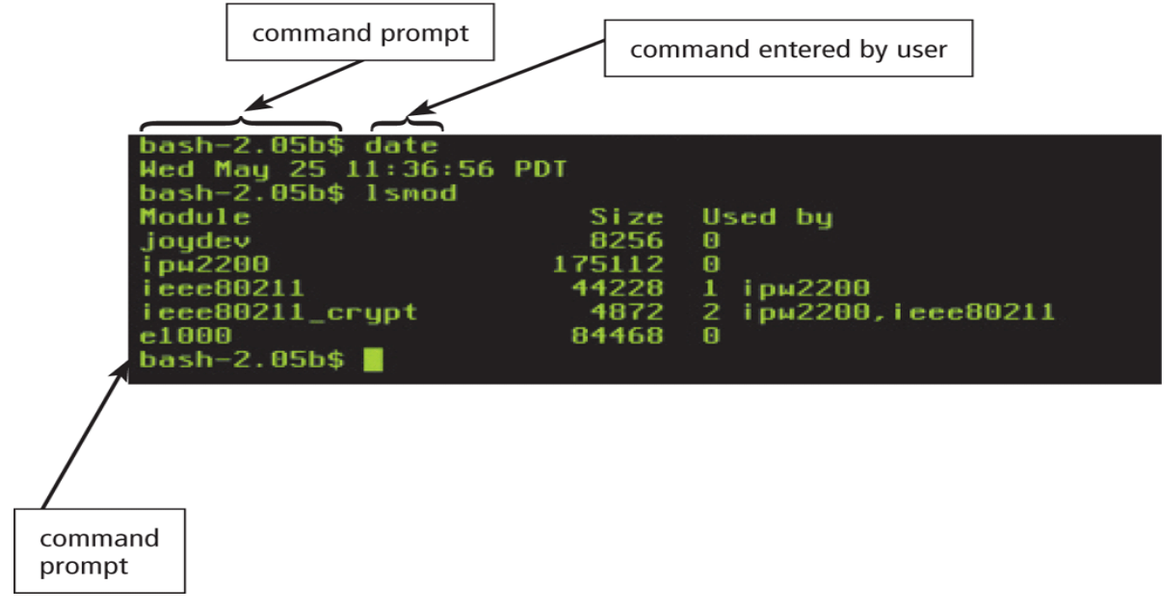
- The operating system uses **password protection** to protect user data and similar other techniques.
- It also prevents unauthorized access to programs and user data by assigning access right permission to files and directories.
- The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other.

6. User Interface Mechanism

- A **user interface** (**UI**) controls how you enter data and instructions and how information is displayed on the screen
- There are two types of user interfaces
  1. Command Line Interface
  2. Graphical user Interface

6.1. Command-line interface

- In a **command-line interface**, a user types commands represented by short keywords or abbreviations or presses special keys on the keyboard to enter data and instructions



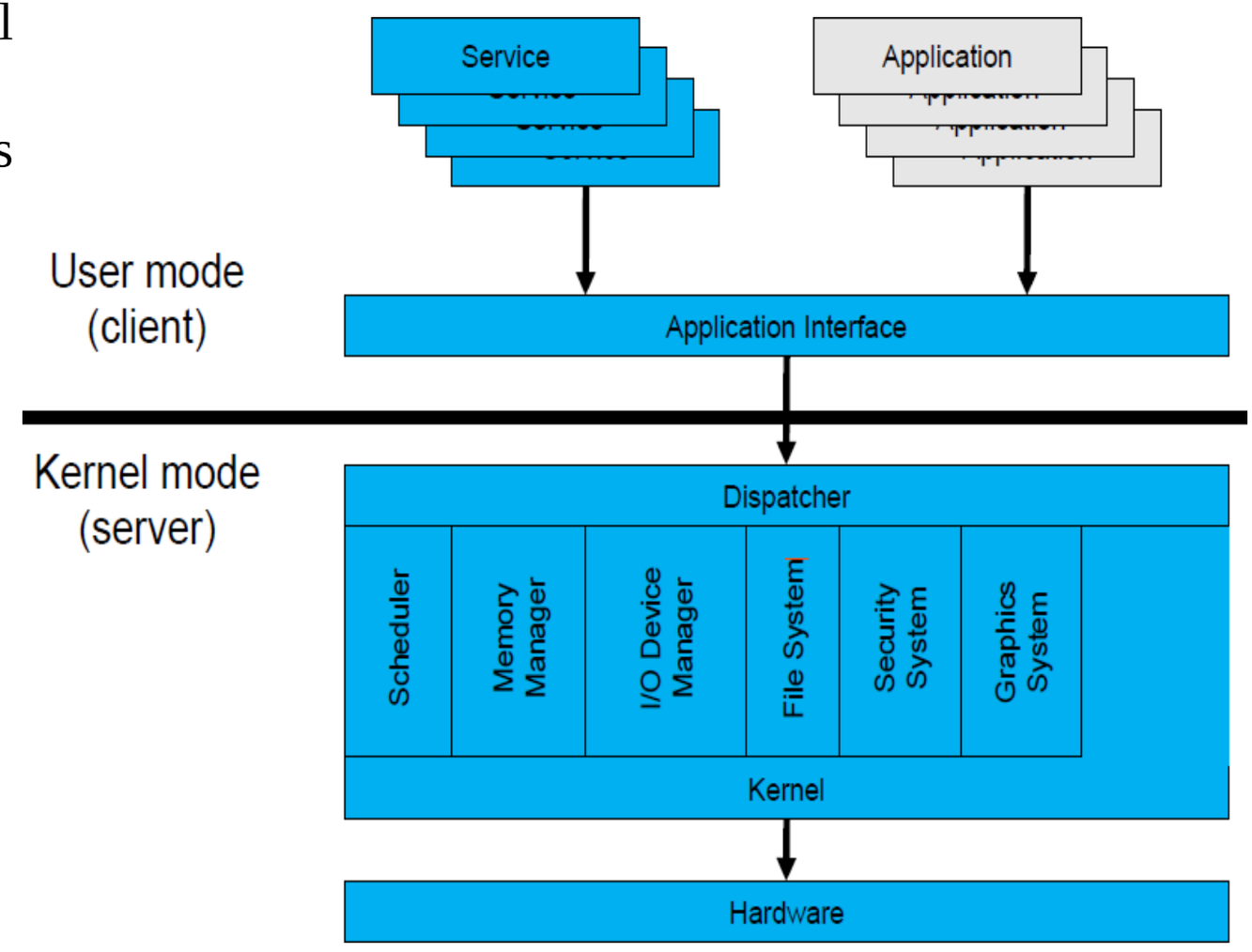
6.2. Graphical User Interface

- With a graphical user interface (GUI), you interact with menus and visual images

Operating System Mode

- The *User Mode* is concerned with the actual interface between the user and the system.
- It controls things like running applications and accessing files.

- The *Kernel Mode* is concerned with everything running in the background.
- It controls things like accessing system resources, controlling hardware functions and processing program instructions.
- System calls are used to change mode from User to Kernel.



Kernel

- Kernel is a software code that reside in central core of OS. It has complete Control Over System (COS).
- When operation system boots, kernel is first part of OS to load in main memory.
- Kernel remains in main memory for entire duration of computer session. The kernel code is usually loaded into protected area of memory.
- Kernel performs it's task like executing processes and handling interrupts in kernel space.
- User performs it's task in user area of memory.
- This memory separation is made in order to prevent user data and kernel data from interfering with each other.
- Kernel **does not interact directly with user**, but it interacts using SHELL and other programs and hardware.

Kernel cont…

- Kernel includes:-

  1. **Scheduler:** It allocates the Kernel's processing time to various processes.

  2. **Supervisor:** It grants permission to use computer system resources to each process.

  3. **Interrupt handler** : It handles all requests from the various hardware devices which compete for kernel services.

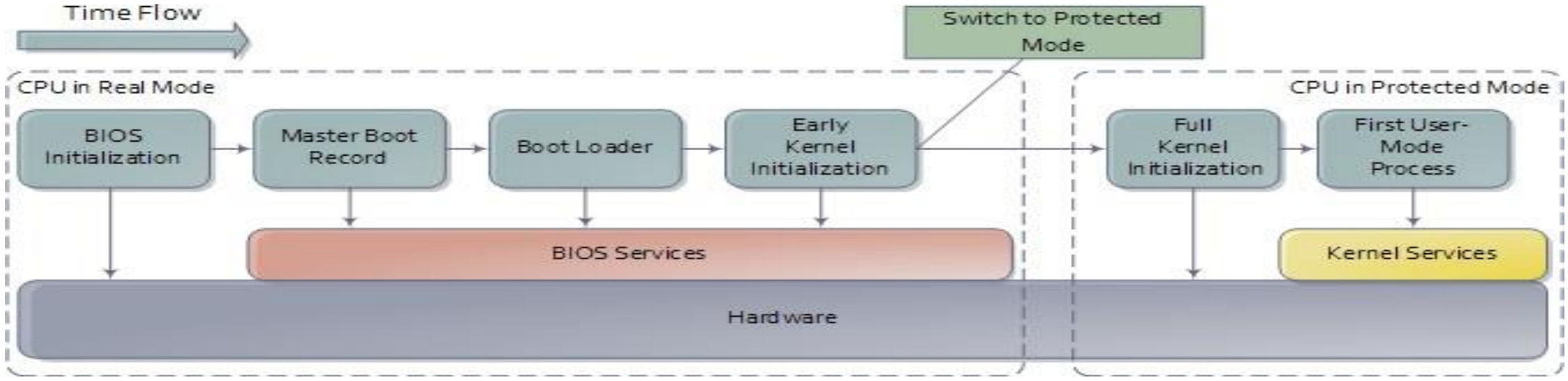
  4. **Memory manager** : allocates space in memory for all users of kernel service.

- kernel provides services for process management, file management, I/O management, memory management.
- System calls are used to provide these type of services.

System Call

- **System call** is the programmatic way in which a computer program/user application requests a service from the kernel of the operating system on which it is executed.
- Application program is just a user-process. Due to security reasons , user applications are not given access to privileged resources(the ones controlled by OS).
- When they need to **do any I/O** or have **some more memory** or **spawn a process** or wait for **signal/interrupt,** it requests operating system to facilitate all these. This **request is made through System Call.**
- System calls are also called **software-interrupts.**

Starting an Operating System(Booting)



- ✓ Power ON Switch sends electricity to the motherboard on a wire called the *Voltage Good* line.
- ✓ If the power supply is good, then the BIOS (Basic Input/Output System) chip takes over.
- ✓ In Real Mode, CPU is only capable of using approximately 1 MB of memory built into the motherboard.
- ✓ The BIOS will do a **Power-On Self Test (POST)** to make sure that all hardware are working.

- ✓ BIOS will then look for a small sector at the very beginning of your primary hard disk called MBR (Master Boot Record). The MBR contains a list, or map, of all of the partitions on your computer‘s hard disk (or disks).
- ✓ After the MBR is found the Bootstrap Loader follows basic instructions for starting up the rest of the computer, including the operating system.
- ✓ In Early Kernel Initialization stage, a smaller core of the Kernel is activated.
- ✓ This core includes the device drivers needed to use computer‘s RAM chips.

BIOS

- BIOS firmware was stored in a ROM/EPROM (Erasable Programmable Read-Only Memory) chip known as **firmware** on the PC motherboard. (firmware helps devices boot up and communicate, software is focused more on user interaction.)
- BIOS can be accessed during the initial phases of the boot procedure by pressing del, F2 or F10.
- Finally, the firmware code cycles through all storage devices and looks for a boot-loader. (usually located in first sector of a disk which is 512 bytes)
- If the boot-loader is found, then the firmware hands over control of the computer to it.

UEFI

- UEFI stands for **U**nified **E**xtensible **F**irmware **I**nterface. It does the same job as a BIOS, but with one basic difference: ***it stores all data about initialization and startup in an .efi file, instead of storing it on the firmware***.
- This ***.efi*** file is stored on a special partition called **EFI System Partition (ESP)** on the hard disk. This ESP partition also contains the **bootloader**.
- UEFI was designed to overcome many limitations of the old BIOS, including:
  - UEFI supports drive sizes upto 9 **zettabytes (1ZB = 10^{21}Bytes)**, whereas BIOS only supports 2.2 terabytes.
  - UEFI provides **faster boot time.**
  - UEFI has discrete driver support, while BIOS has drive support stored in its ROM, so updating BIOS firmware is a bit difficult.
  - UEFI offers security like **"Secure Boot",** which prevents the computer from booting from unauthorized/unsigned applications. This helps in preventing **rootkits** *(a type of malware program that enables cyber criminals to gain access to and infiltrate data from machines without being detected.)*
  - UEFI runs in 32bit or 64bit mode, whereas BIOS runs in 16bit mode. So UEFI is able to provide a GUI (navigation with mouse) as opposed to BIOS which allows navigation only using the keyboard.

Concept of a Process and Process State

A program which is in execution is called as a ***'Process'***

- As a process executes, it changes state.
- The state of a process is defined in part by the current activity of that process.

Each process may be in one of the following states:

NEW	The process is being created.
RUNNING	Instructions are being executed.
WAITING	The process is waiting for some event to occur (Such as an I/O completion or reception of a signal).
READY	The process is waiting to be assigned to a processor
TERMINATED	The process has finished execution.

Concept of a Process and Process State

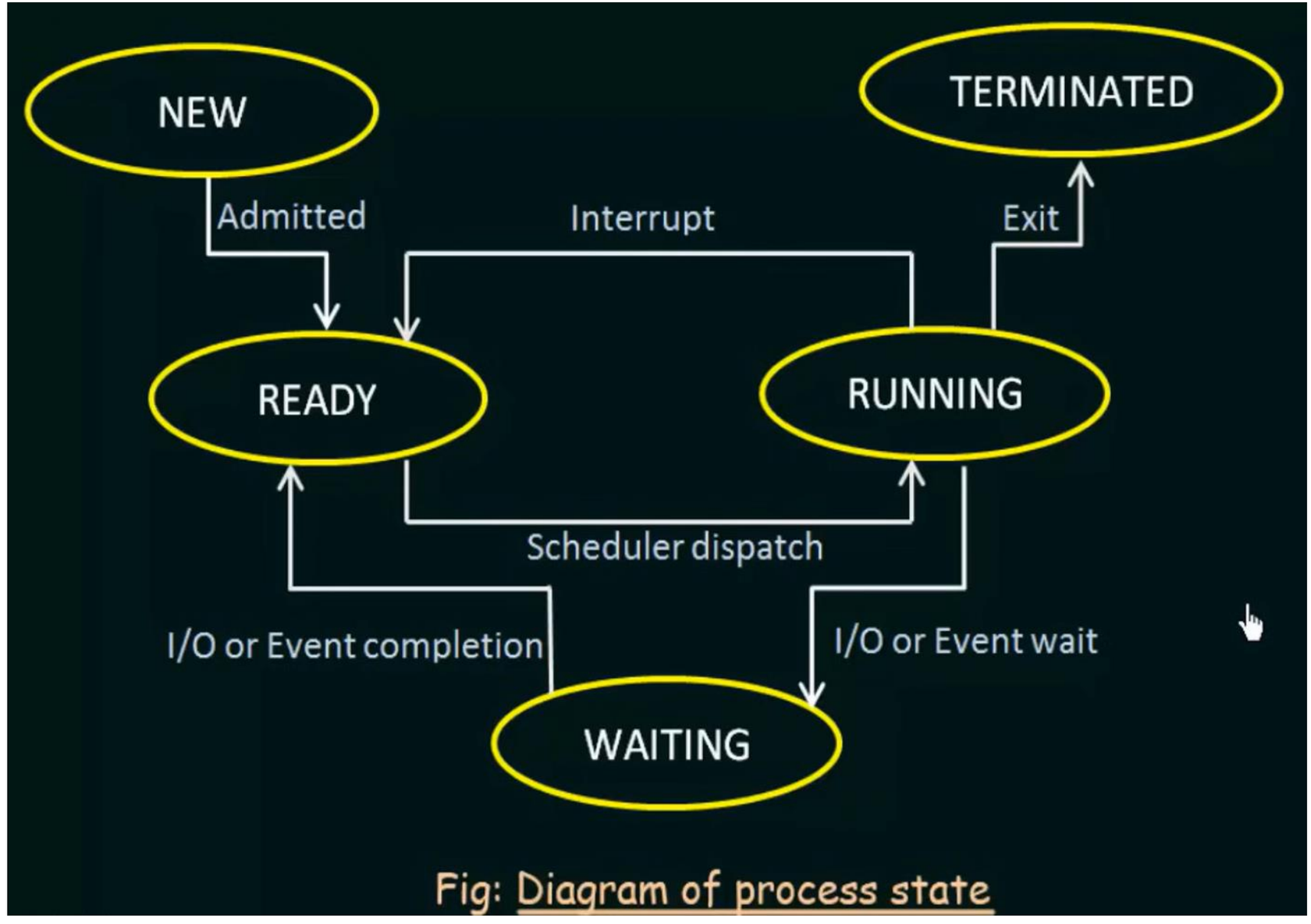
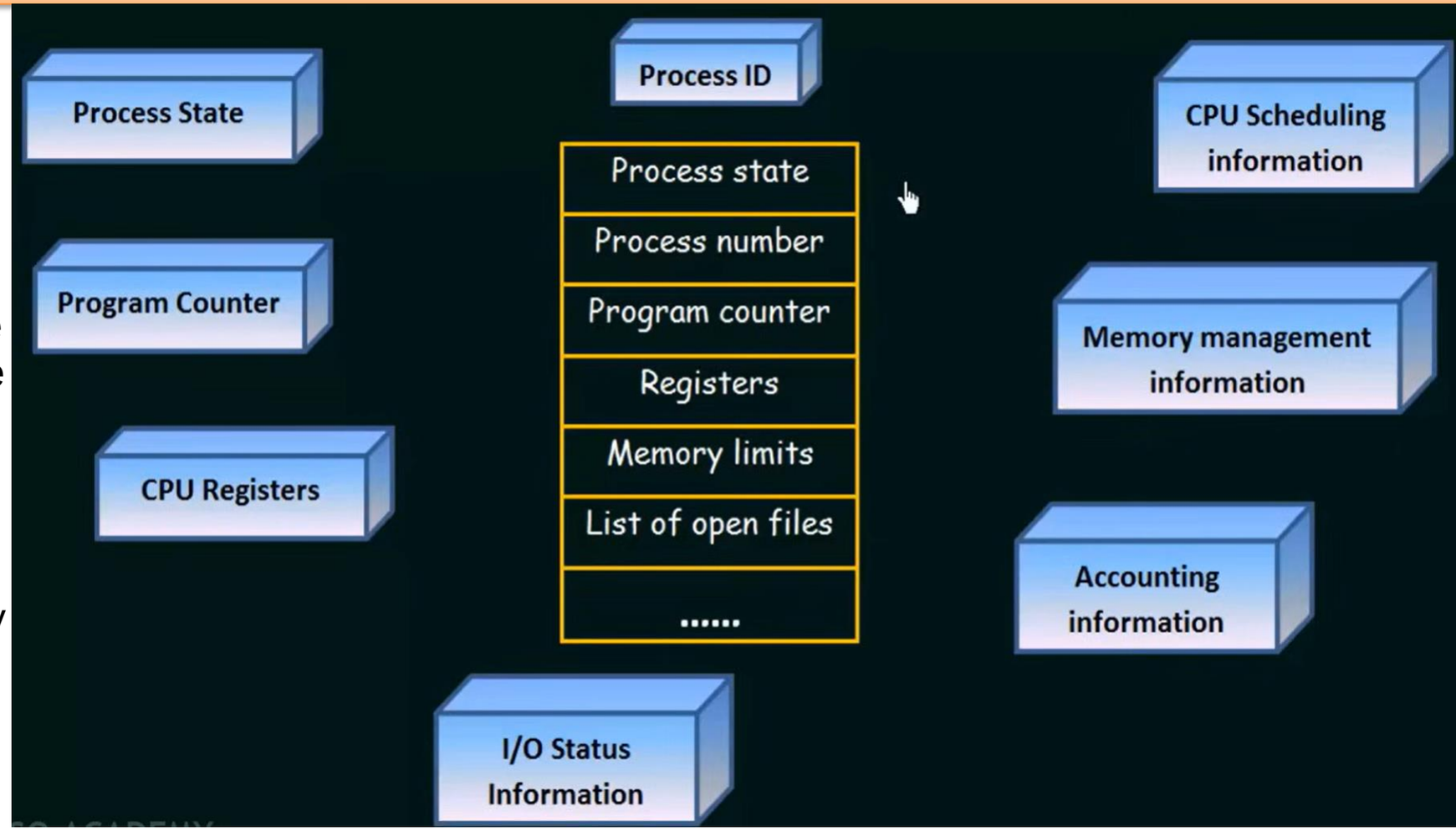


Fig: Diagram of process state

Process Control Block

Each *'Process'* is represented in the OS by a *'Process Control Block (PCB)'* OR *'Task Control Block'*



Address of the next line of the instruction in the program

CPU register being used by a particular process

It determines order / sequence of various processes to be executed (priority-wise)

Resources being used – CPU, Time, Memory

I/O Devices assigned for a process

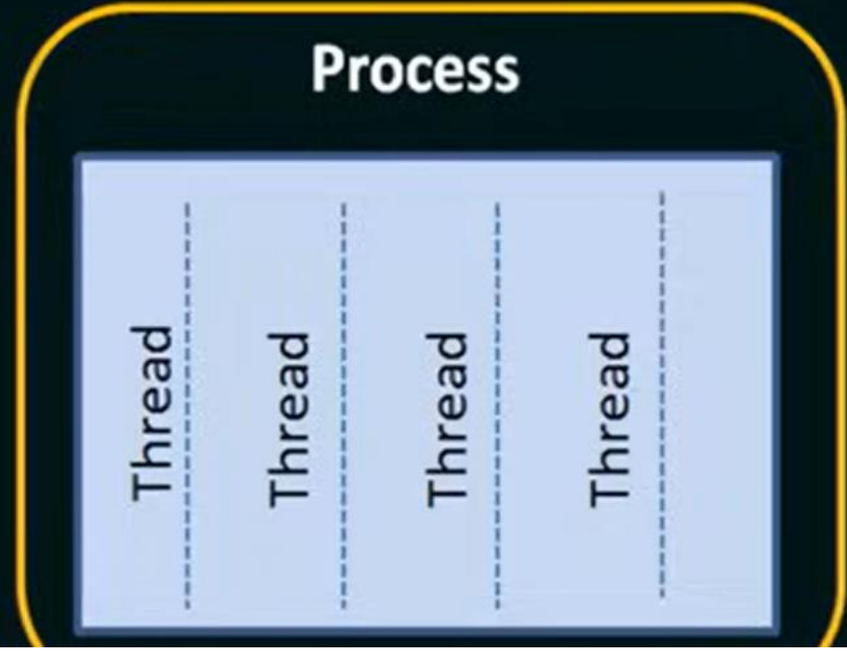
Threads: Definition and Types

Process:

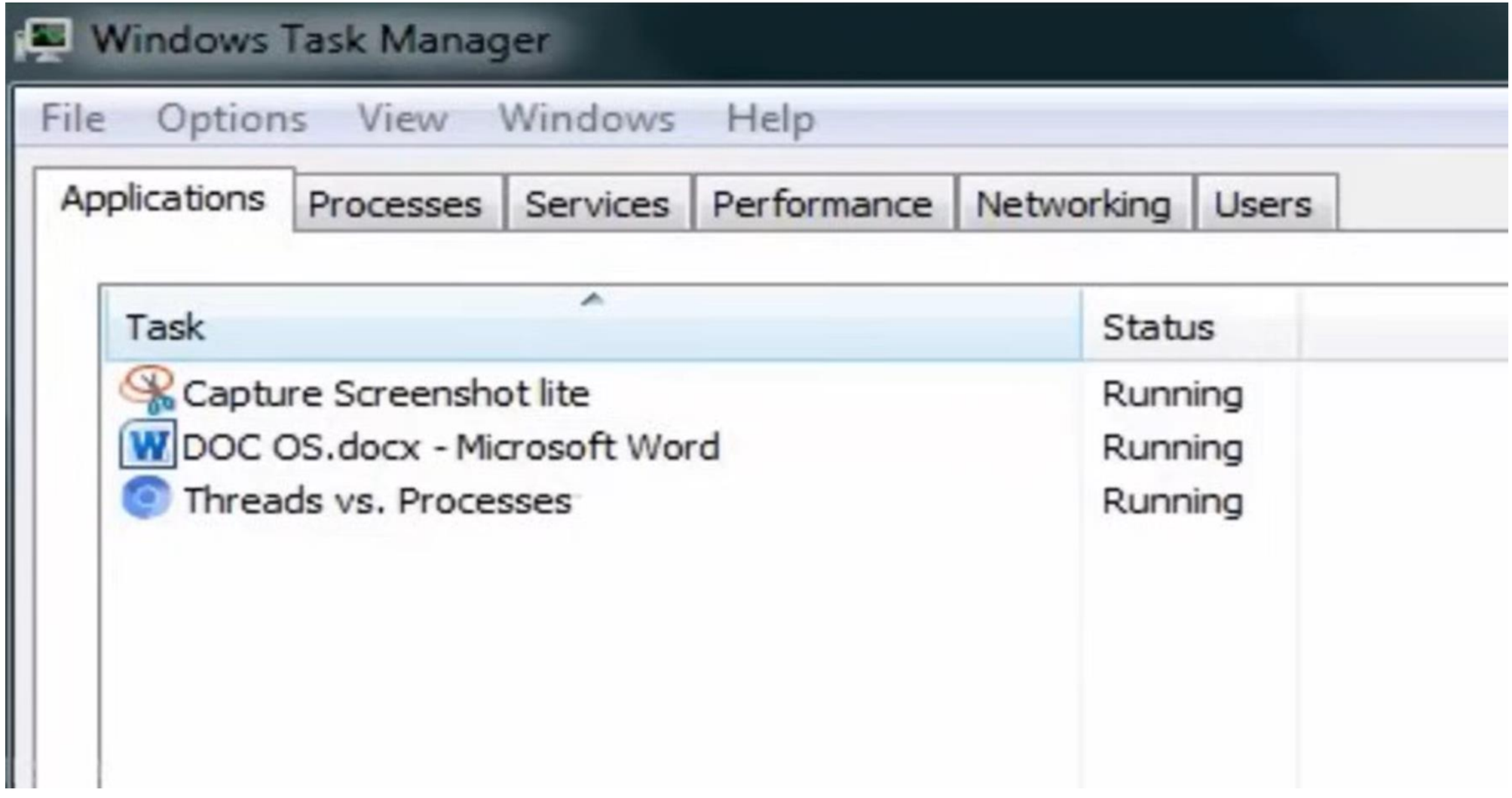
A process can be thought of as a program in execution.

Thread:

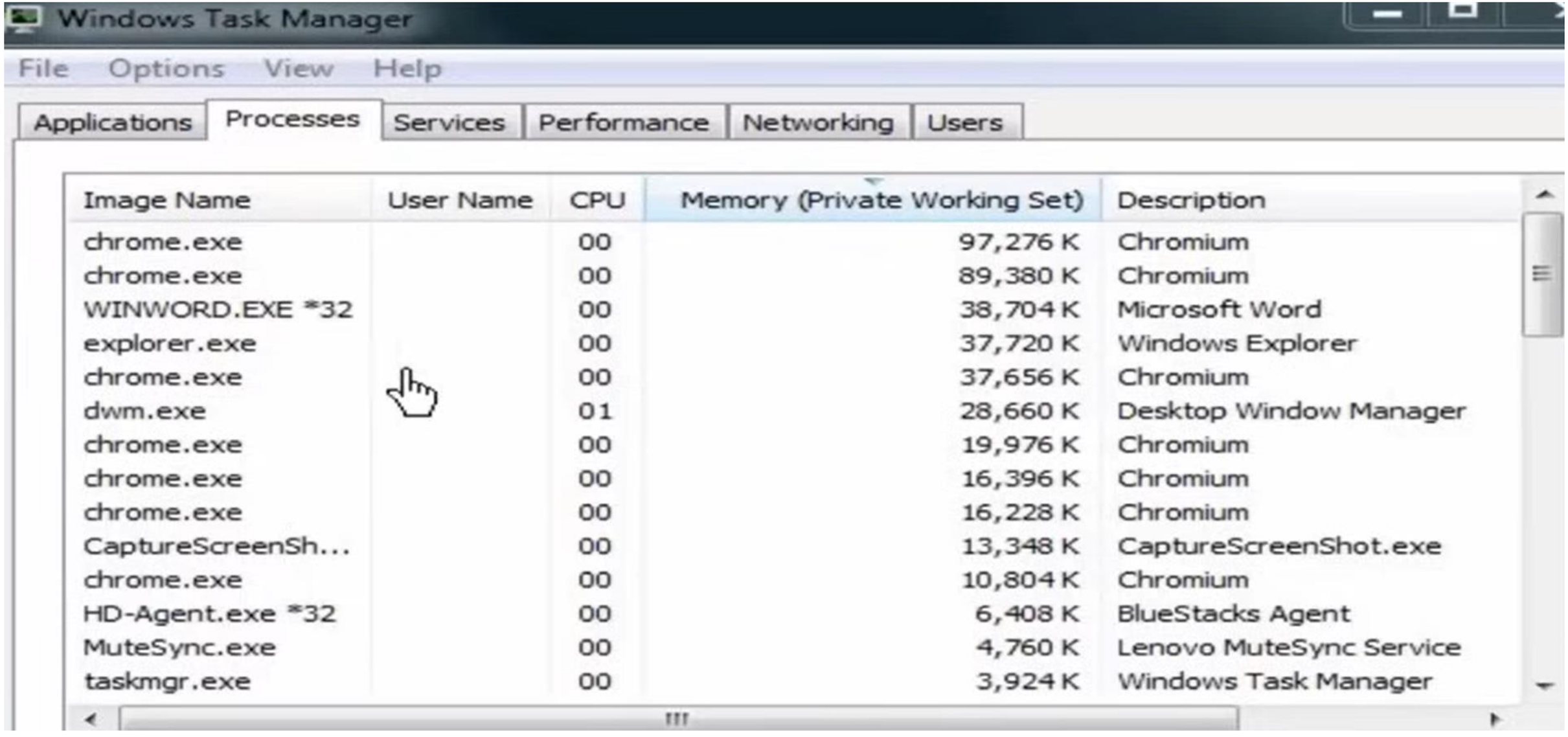
A thread is the unit of execution within a process. A process can have anywhere from just one thread to many threads.



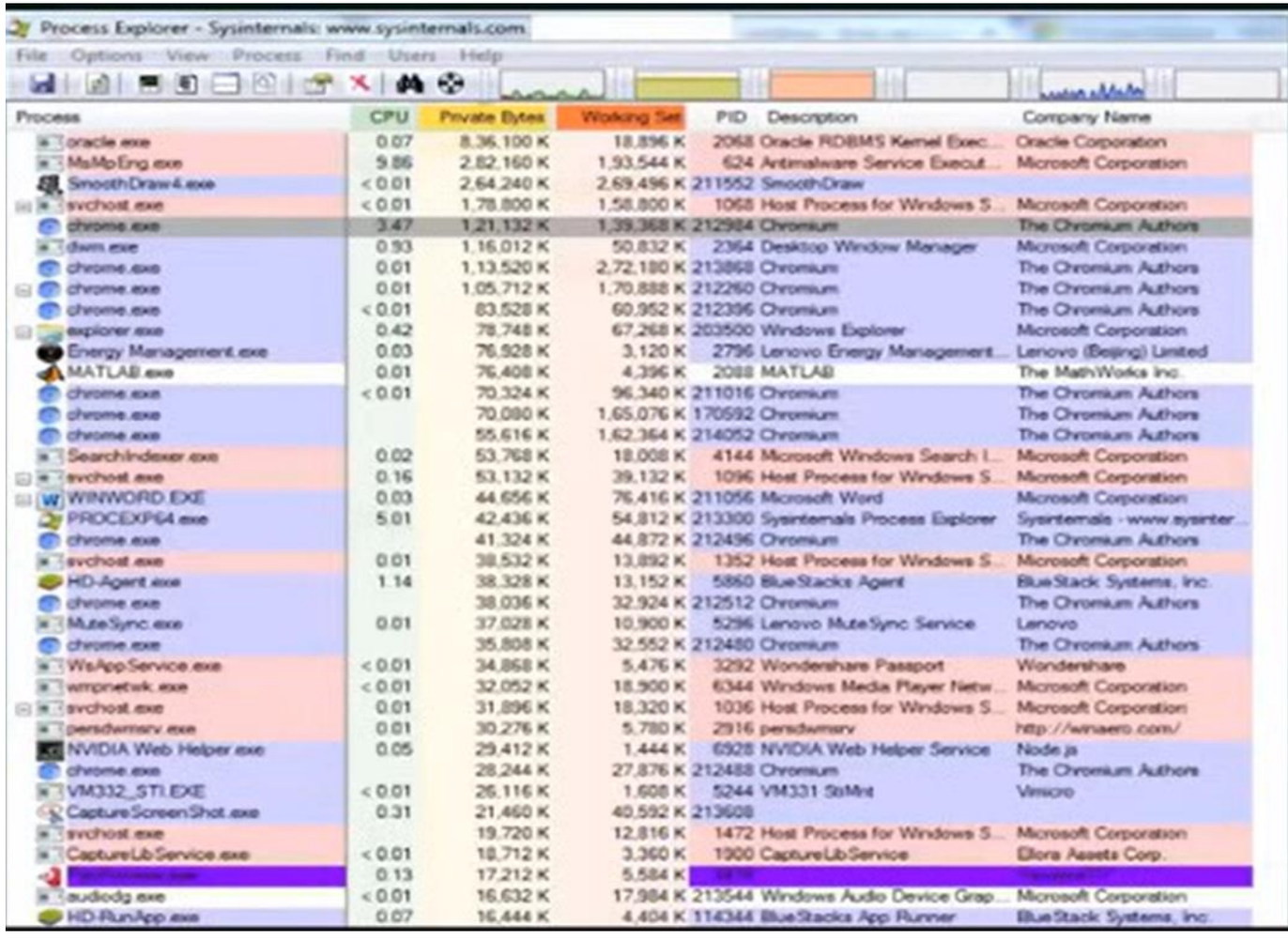
Threads: Definition and Types

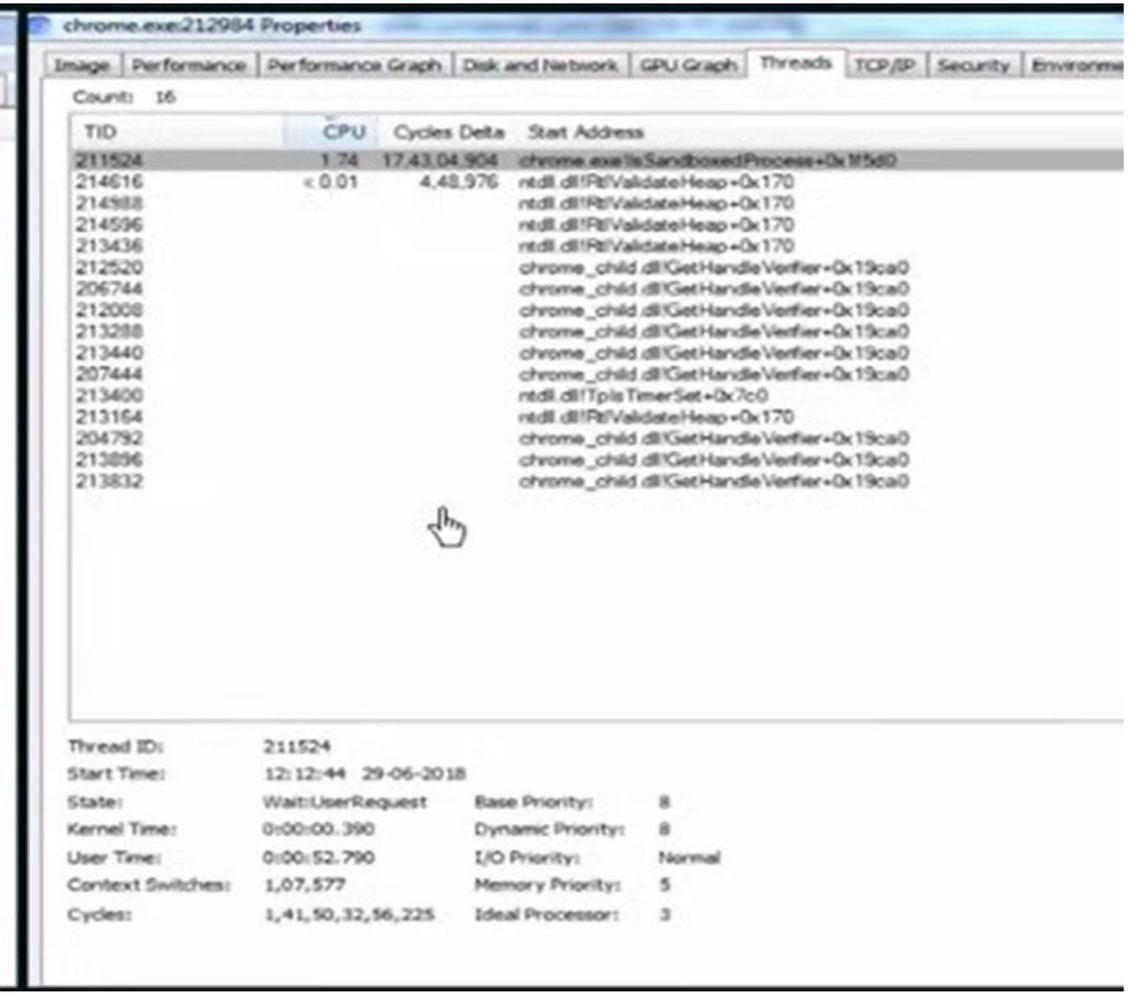


Threads: Definition and Types



Threads: Definition and Types





Single Threads / Heavy Weight Process

Traditional Heavy Weight process

- Has **a single thread of control.**
- Process performs a single thread of execution.

For Example,

- If a process is running a word processor program,
- A single thread of instructions is being executed.
- **The user could not simultaneously type in characters and run the spell checker within the same process.**

Multi Threads / Light Weight Process

Multithreaded Process

- If a process has **multiple threads of control**,
- It can do **more than one task** at a time.

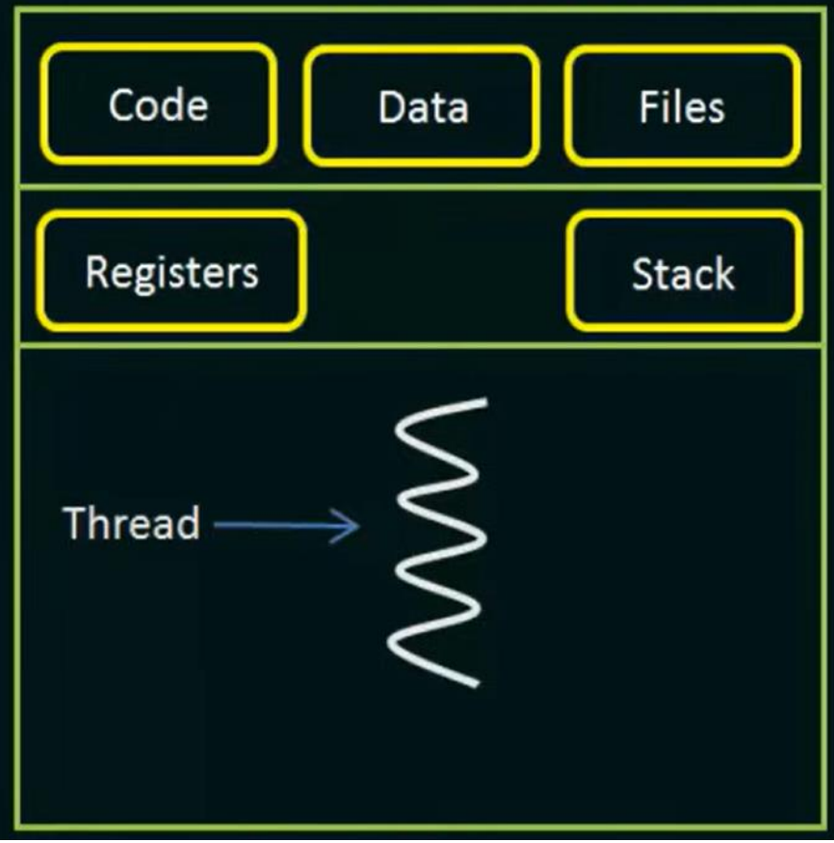
- A thread is also known as lightweight process.
- The idea is to **achieve parallelism by dividing a process into multiple threads.**

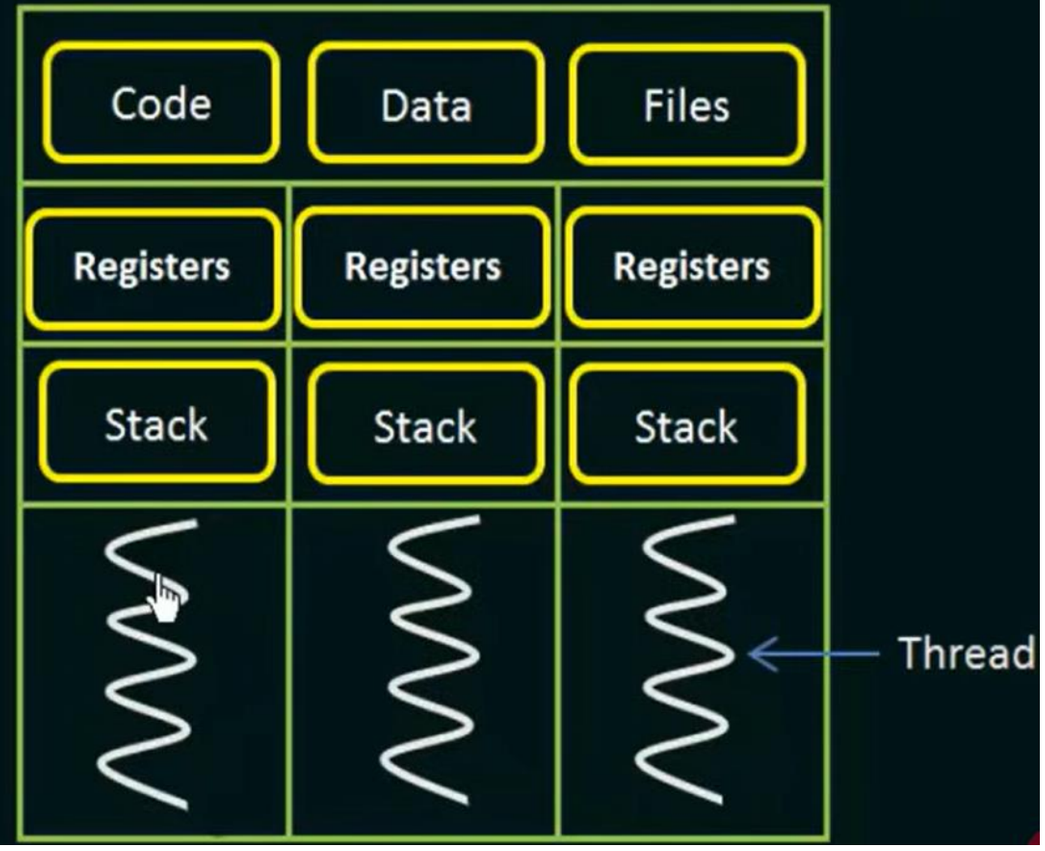
Concept of Multithreading

It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals.

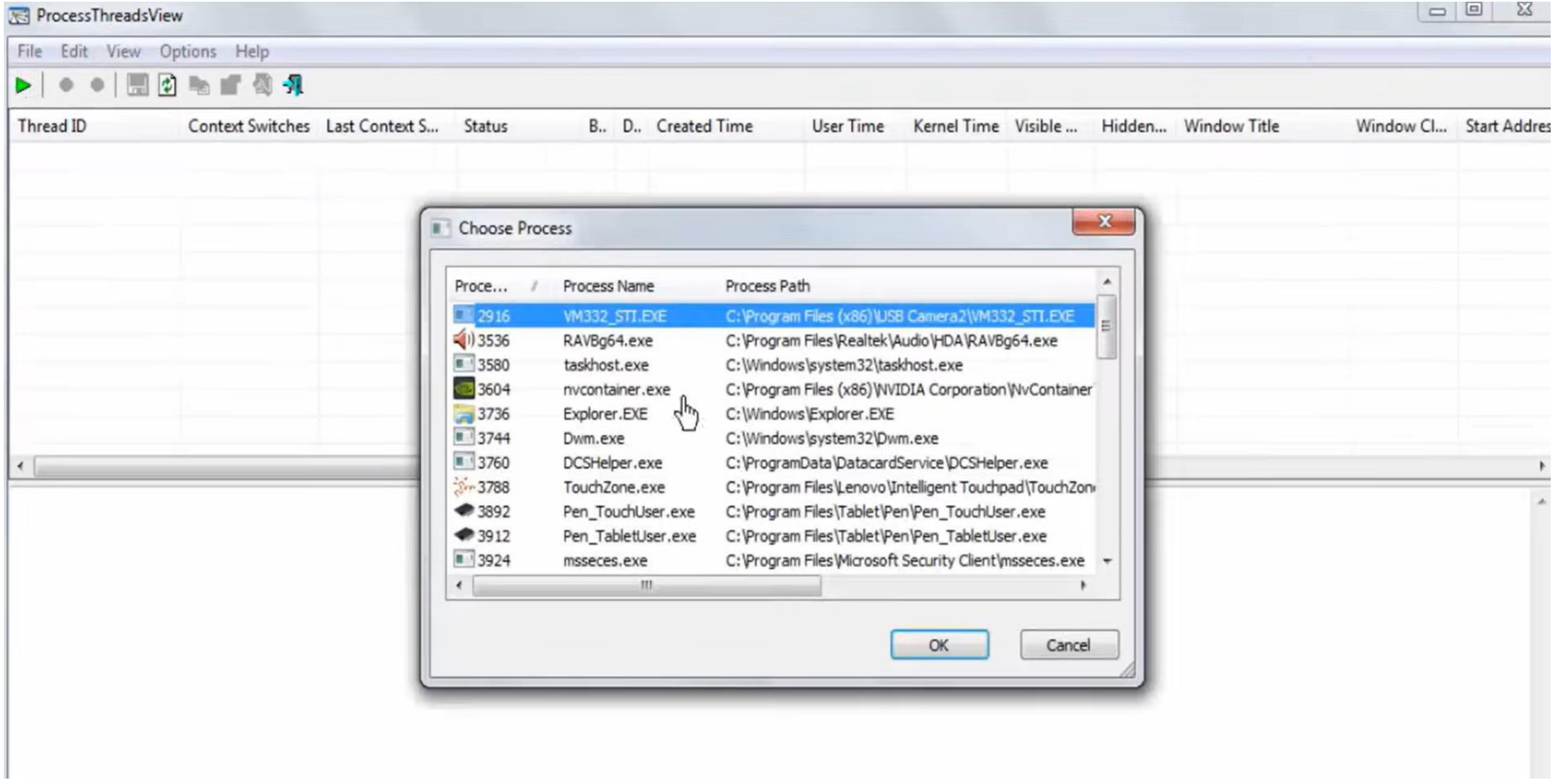
A traditional / heavyweight process has a single thread of control.

If a process has multiple threads of control, it can perform more than one task at a time.

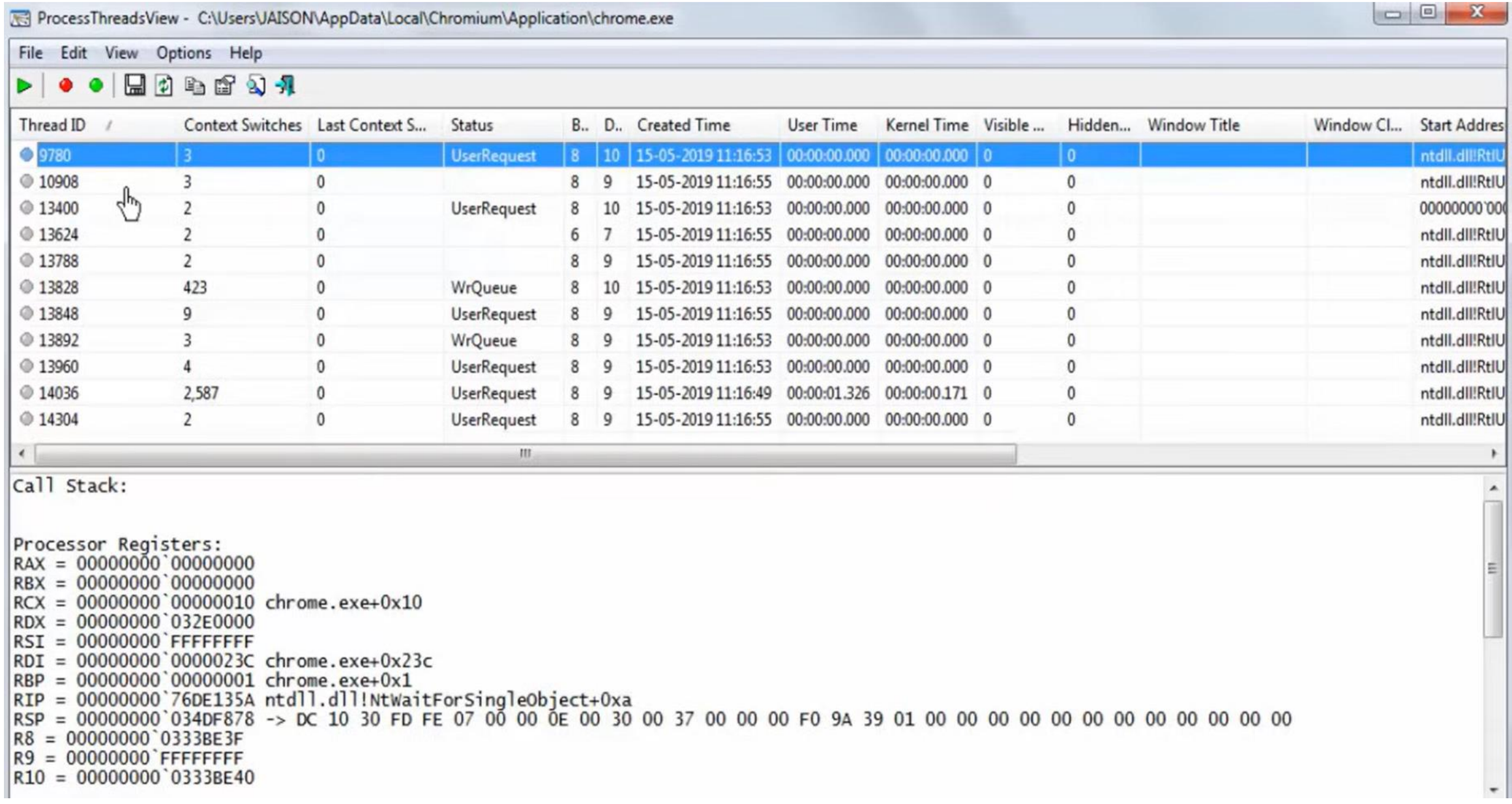




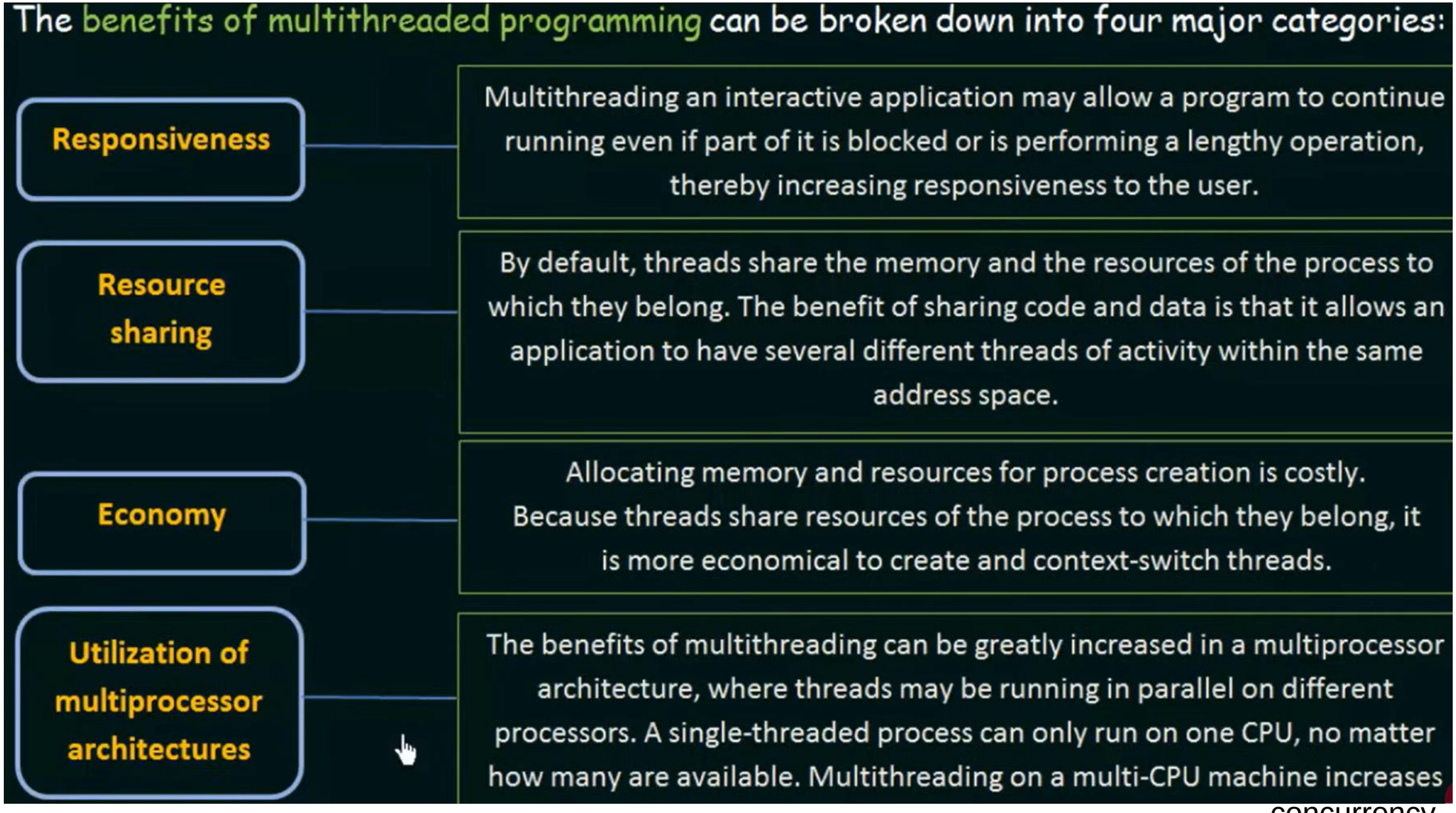
Concept of Multithreading



Concept of Multithreading



Concept of Multithreading



concurrency

Multithreading Models: One-to-One

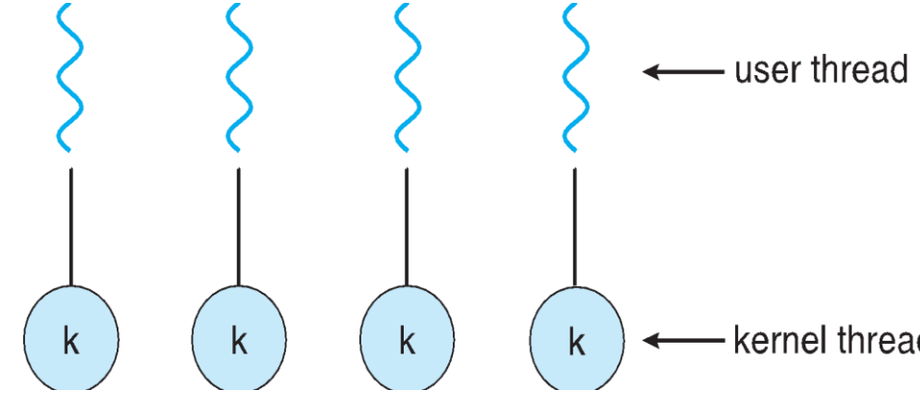
Each user-level thread maps to kernel thread

Creating a user-level thread creates a kernel thread

More concurrency than many-to-one

- **By allowing another thread to run when a thread makes a blocking system call**

Allows multiple threads to run in parallel on multiprocessors



Multithreading Models: One-to-One

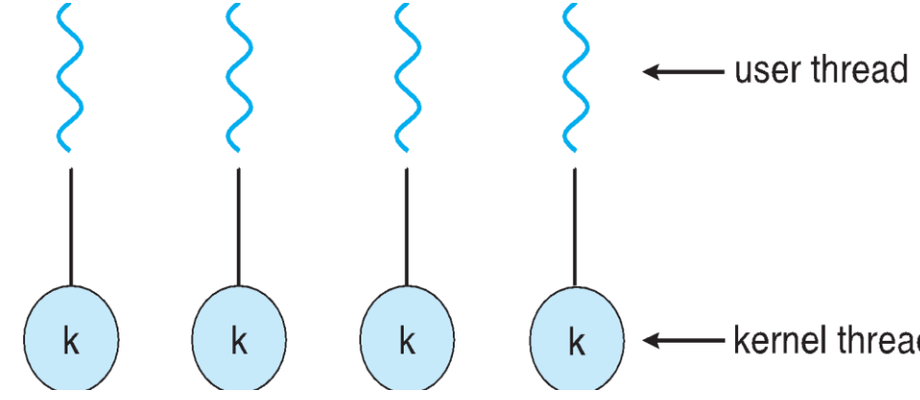
Creating a user thread requires creating the corresponding kernel thread.

Overhead of creating kernel threads can burden the performance of an application

Number of threads per process sometimes restricted due to overhead

Examples

- **Windows**
- **Linux**
- **Solaris 9 and later**



Multithreading Models: Many-to-One

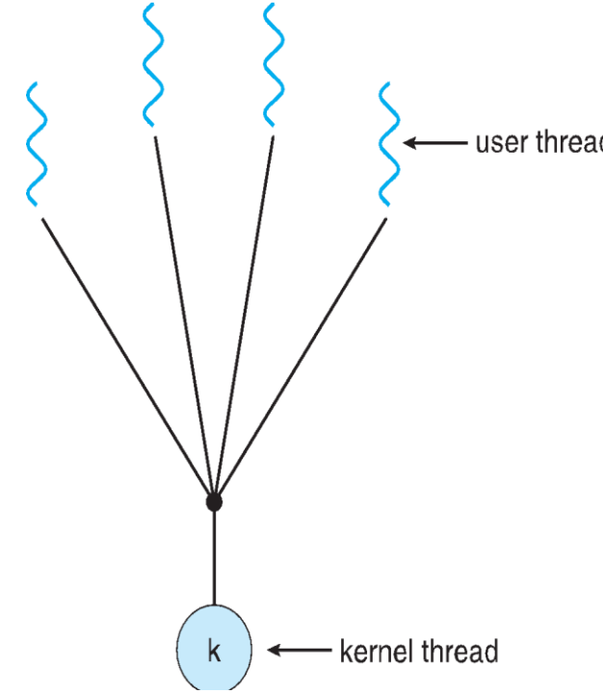
Many user-level threads mapped to single kernel thread

Only one thread can access the Kernel at a time.

Thread Management done in User space, so is efficient.

If one thread makes a blocking system call, the entire process will block.

One thread blocking causes all to block



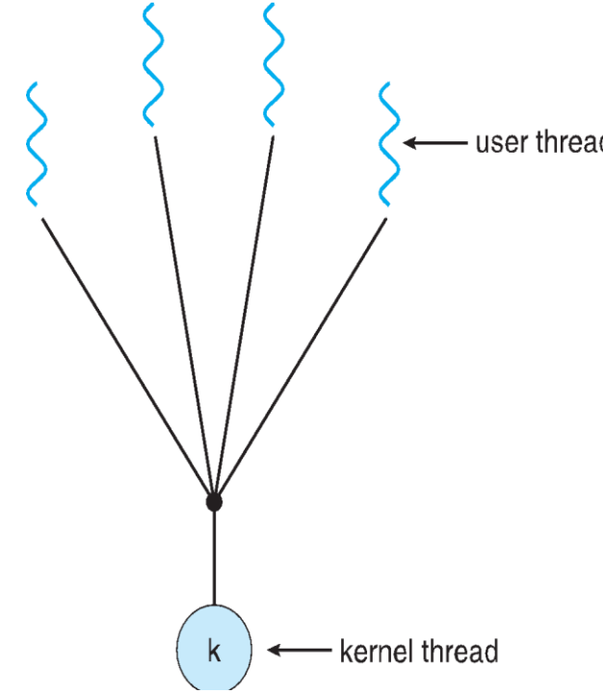
Multithreading Models: Many-to-One

Multiple threads are unable to run in parallel on multicore system because only one may be in kernel at a time

Few systems currently use this model

Examples:

- **Solaris Green Threads**
  - **Green thread, a thread that uses this model**
  - **Available for Solaris 2**
- **GNU Portable Threads**



Multithreading Models: Many-to-Many

Allows many user level threads to be mapped to many kernel threads

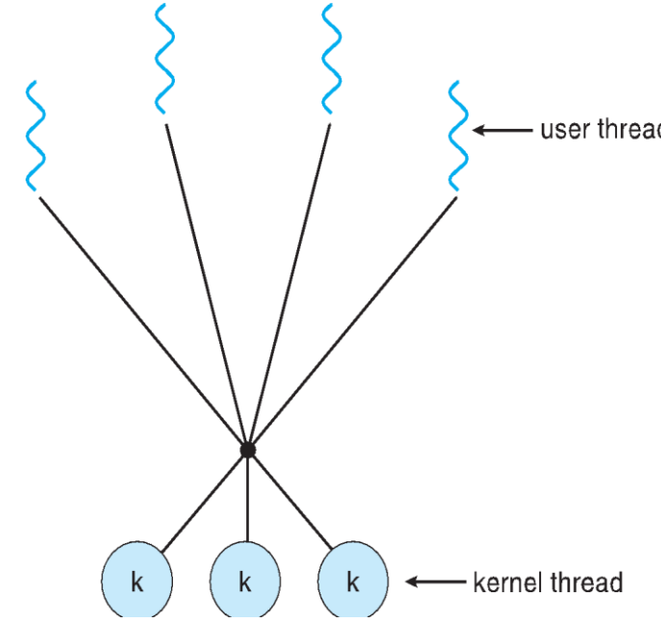
Allows the operating system to create a sufficient number of kernel threads

The number of kernel threads may be specific to

- **Either a particular application**
- **Or a particular machine**

Solaris prior to version 9

Windows with the *ThreadFiber* package



Multithreading Models: Many-to-Many

Many to One

Allows developer to create as many user threads as needed

But true concurrency is not gained because kernel can schedule only one thread at a time

One to One

Allows for greater concurrency, but

the developer has to be careful not to create too many threads within an application

Many to Many

<u>Overcomes these shortcomings</u>

Developers can create as many user threads as necessary

Corresponding Kernel threads can run in parallel on a multiprocessor

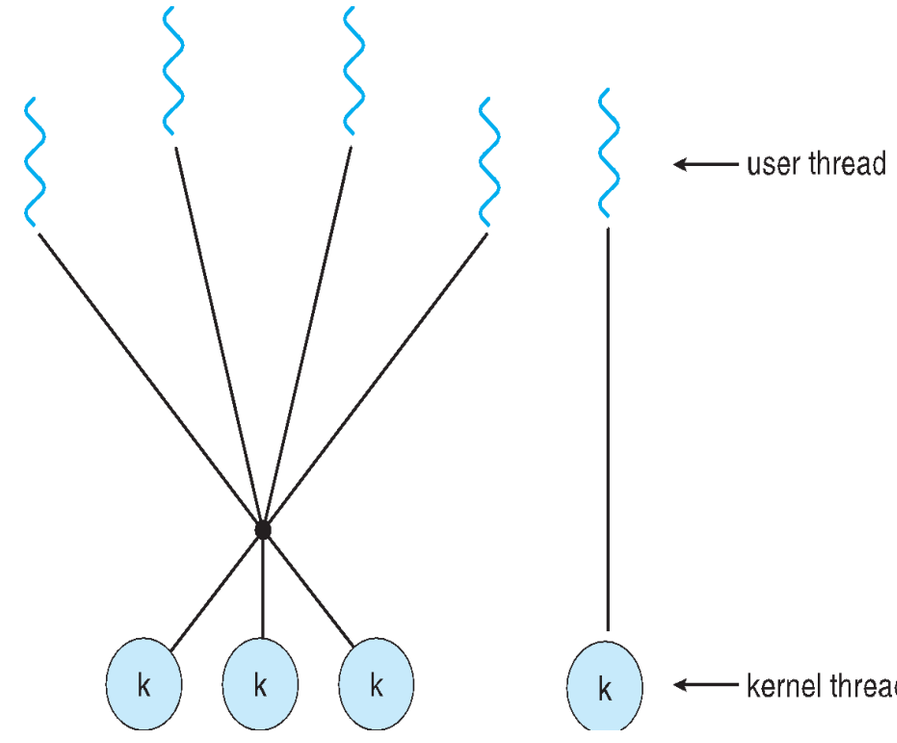
When a thread performs a blocking system call, the kernel can schedule another thread for execution

Two Level Models

Similar to M:M, **except that it allows a user thread to be bound to kernel thread**

Examples

- IRIX
- HP-UX
- Tru64 UNIX
- Solaris 8 and earlier



Advantages of Thread over Process

1. Responsiveness: If the process is divided into multiple threads, **if one thread completes its execution, then its output can be immediately returned.**

2. Faster context switch:

- **Context switch time between threads is lower compared to process context switch.**
- **Process context switching requires more overhead from the CPU.**

3. Resource sharing:

Resources like **code, data, and files can be shared** among all threads within a process.

Note: **stack and registers can't be shared among the threads.**

Each thread has its own stack and registers.

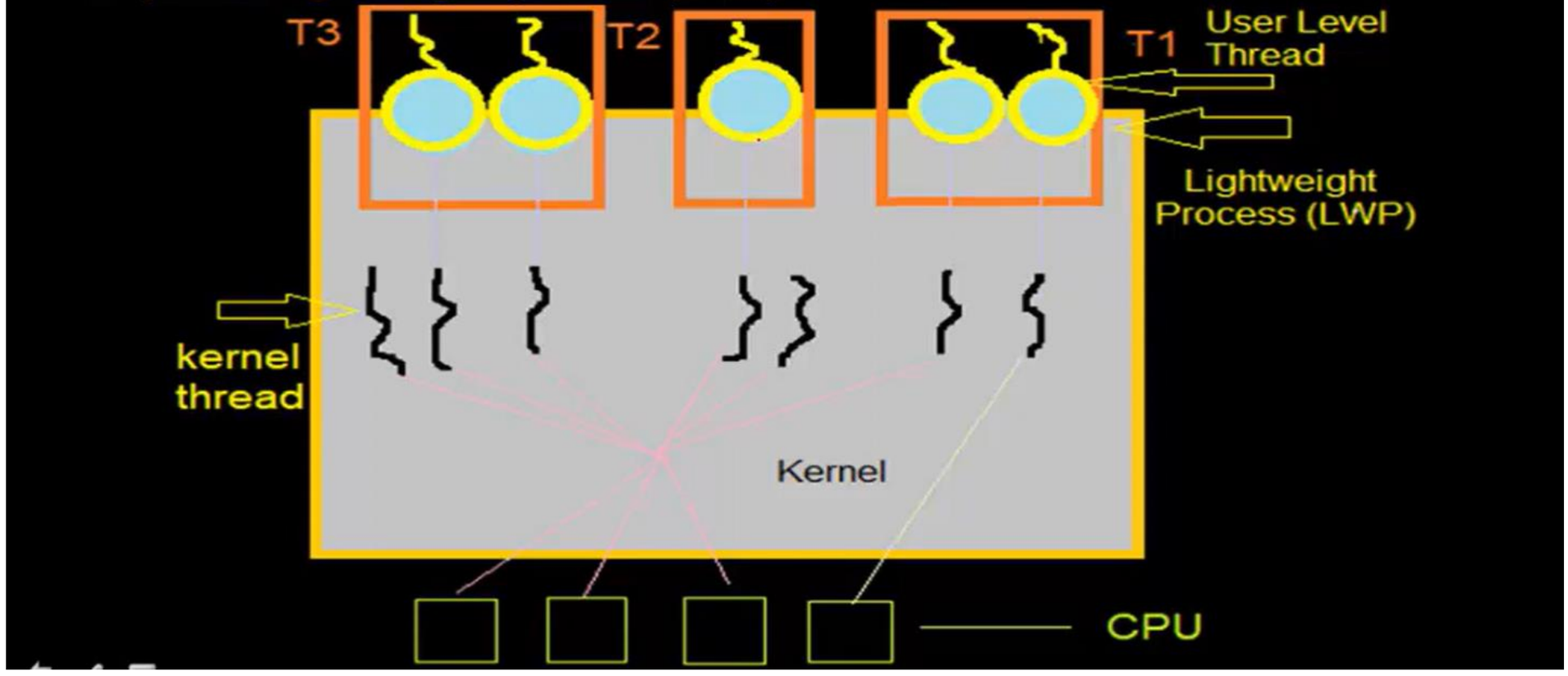
Advantages of Thread over Process

4. Communication:

- Communication between multiple threads is easier, as the **threads shares common address space,**
- while in process we have to follow some **specific communication technique** for communication between two process.

Introduction to Thread Scheduling

- On operating systems that support them, it is kernel-level threads—not processes—that are being scheduled by the operating system. User-level threads are managed by a thread library, and the kernel is unaware of them
- To run on a CPU, user-level threads must ultimately be mapped to an associated kernel-level thread, although this mapping may be indirect and may use a lightweight process (LWP)



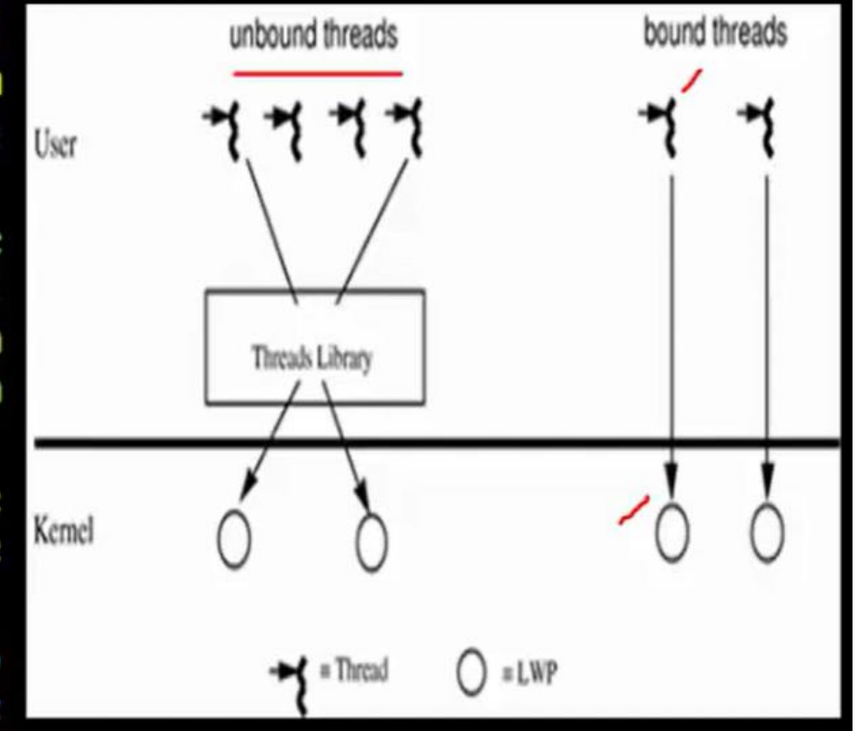
Introduction to Thread Scheduling

- A Thread Library provides the programmer with an API for creating and managing threads.
- There are two primary ways of implementing a thread library.
  - ✓ The first approach is to provide a library entirely in user space with no kernel support. All code and data structures for the library exist in user space. This means that invoking a function in the library results in a local function call in user space and not a system call.
  - ✓ The second approach is to implement a kernel-level library supported directly by the operating system. In this case, code and data structures for the library exist in kernel space. Invoking a function in the API for the library typically results in a system call to the kernel.
- Three main thread libraries are in use today: (1) POSIX Pthreads, (2) Win32, and (3) Java.
  - ➢ Pthreads, the threads extension of the POSIX standard, may be provided as either a user- or kernel-level library.
  - ➢ The Win32 thread library is a kernel-level library available on Windows systems.
  - ➢ The Java thread API allows threads to be created and managed directly in Java programs.

Introduction to Thread Scheduling

Contention Scope

- One distinction between user-level and kernel-level threads lies in how they are scheduled.
- process local scheduling (known as Process Contention Scope, or Unbound Threads—the Many-to-Many model) and system global scheduling (known as System Contention Scope, or Bound Threads—the One-to-One model).
- On systems implementing the many-to-one and many-to-many models, the thread library schedules user-level threads to run on an available LWP. This scheme is known as process contention scope (PCS).
- since competition for the CPU takes place among threads belonging to the same process.
- To decide which kernel-level thread to schedule onto a CPU, the kernel uses system-contention scope (SCS).
- Competition for the CPU with SCS scheduling takes place among all threads in the system.



Introduction to Thread Scheduling

Contention Scope

- The System Contention Scope is one of two thread-scheduling schemes used in operating systems. This scheme is used by the kernel to decide which kernel-level thread to schedule onto a CPU, wherein all threads in the system compete for the CPU.
- Systems using the one-to-one model, such as Windows, Linux, and Solaris, schedule threads using only SCS.
- Typically, PCS is done according to priority—the scheduler selects the run able thread with the highest priority to run.
- User-level thread priorities are set by the programmer and are not adjusted by the thread library, although some thread libraries may allow the programmer to change the priority of a thread.
- It is important to note that PCS will typically preempt the thread currently running in favor of a higher-priority thread; however, there is no guarantee of time slicing among threads of equal priority.

History of Operating System

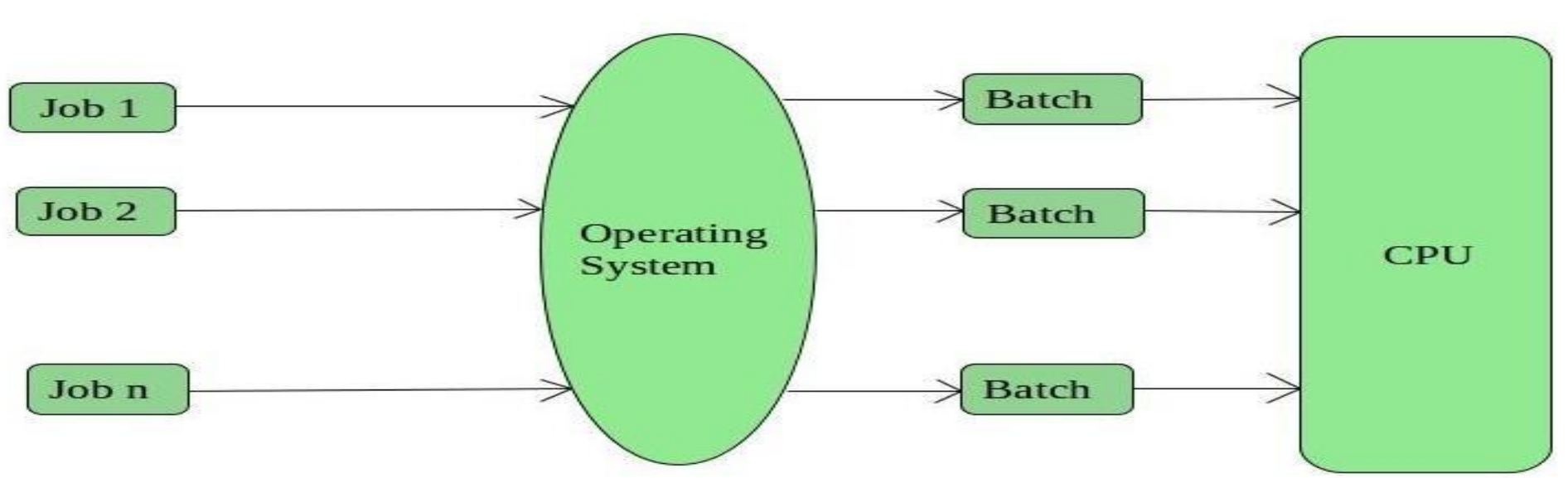
- **The First Generation (1940's to early 1950's)**
  - No Operating System
  - All programming was done in absolute machine language, often by wiring up plug-boards to control the machine’s basic functions.
- **The Second Generation (1955-1965)**
  - First operating system was introduced in the early 1950's.It was called GMOS
  - Created by General Motors for IBM's machine the 701.
  - Single-stream batch processing systems
- **The Third Generation (1965-1980)**
  - Introduction of multiprogramming
  - Development of Minicomputer
- **The Fourth Generation (1980-Present Day)**
  - Development of PCs
  - Birth of Windows / MaC OS

Types of Operating Systems

1. Batch Operating System
2. Multiprogramming Operating System
3. Time-Sharing / Multi-tasking OS
4. Multiprocessing OS
5. Distributed OS
6. Network OS
7. Real Time OS
8. Embedded OS

1. Batch Operating System

- The users of this type of operating system **does not interact** with the computer **directly**.
- Each user prepares his job on an off-line device like **punch cards / paper tape / magnetic tape** and submits it to the computer operator
- There is an operator which takes similar jobs having the same requirement and group them into **batches.**



1. Batch Operating System cont..

Advantages of Batch Operating System:

- Processors of the batch systems know how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time for the batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

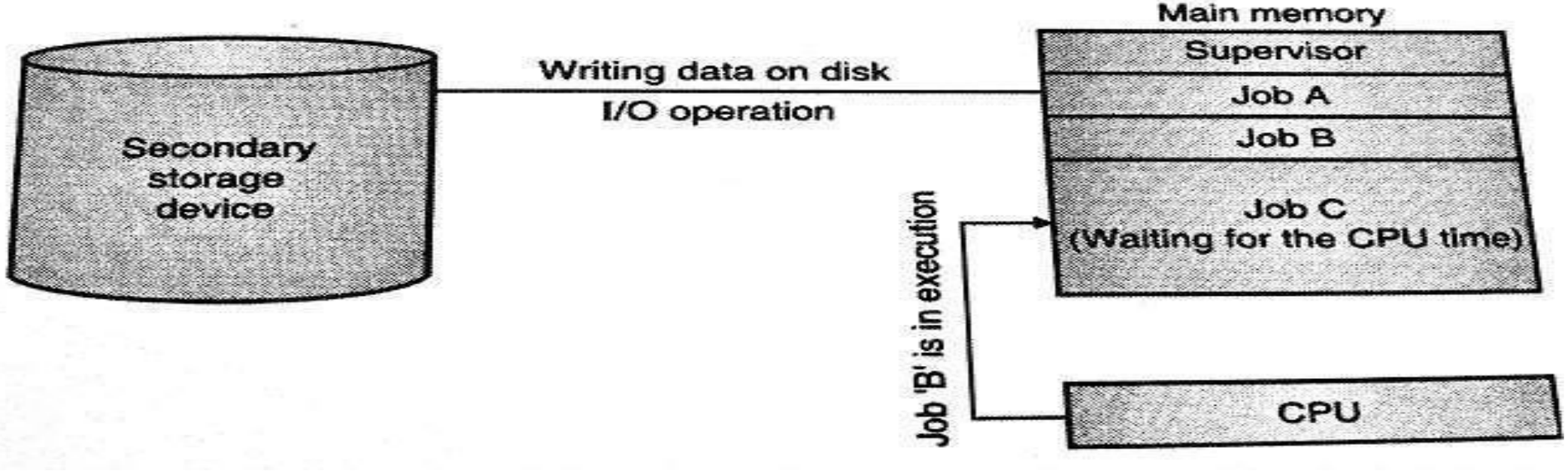
- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometimes costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System:

IBM's MVS (Multiple virtual storage)

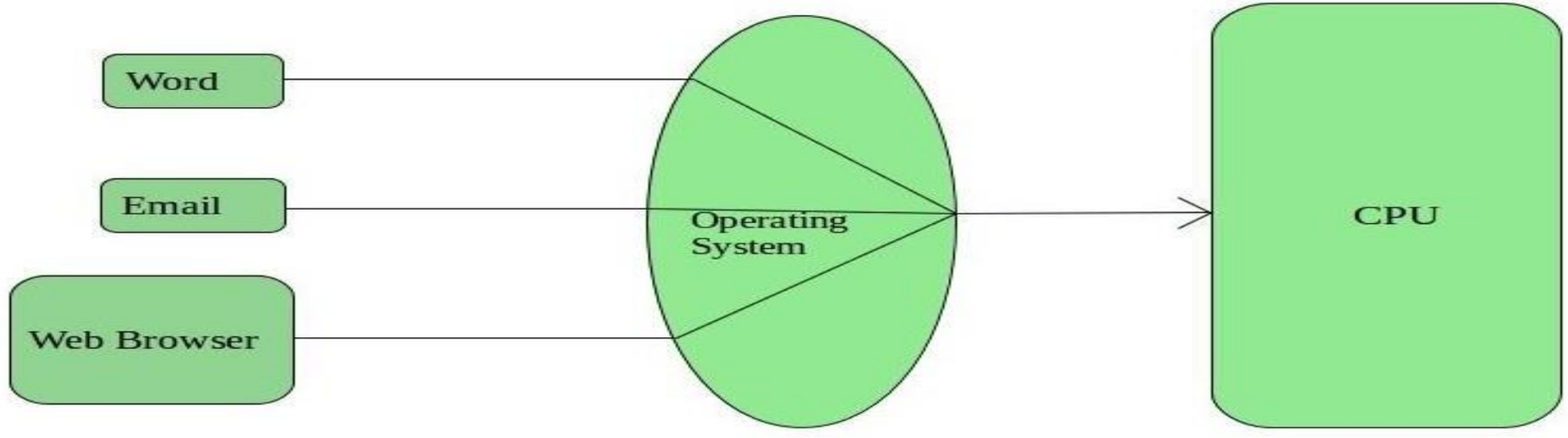
2. Multiprogramming Operating System:

- This type of OS is used to execute **more than one jobs simultaneously** by a single processor.
- It ***increases*** CPU utilization by organizing jobs so that the CPU always has one job to execute.
- Multiprogramming operating systems use the mechanism of job scheduling and CPU scheduling.



3. Time-Sharing / Multi-tasking Operating Systems

- Each task is given some time to execute so that all the tasks work smoothly.
- These systems are also known as **Multi-tasking Systems.**
- The task can be from a single user or different users also.
- The time that each task gets to execute is called ***quantum***.
- After this **time interval is over** OS switches over to the next task.



3. Time-Sharing / Multi-tasking Operating Systems

- **Advantages of Time-Sharing / Multi-tasking OS:**
  - Each task gets an **equal opportunity**
  - Fewer chances of duplication of software
  - CPU idle time can be reduced
- **Disadvantages of Time-Sharing / Multi-tasking OS:**
  - Reliability problem
  - One must have to take care of the security and integrity of user programs and data
  - Data communication problem
- **Examples of Time-Sharing / Multi-tasking OSs**

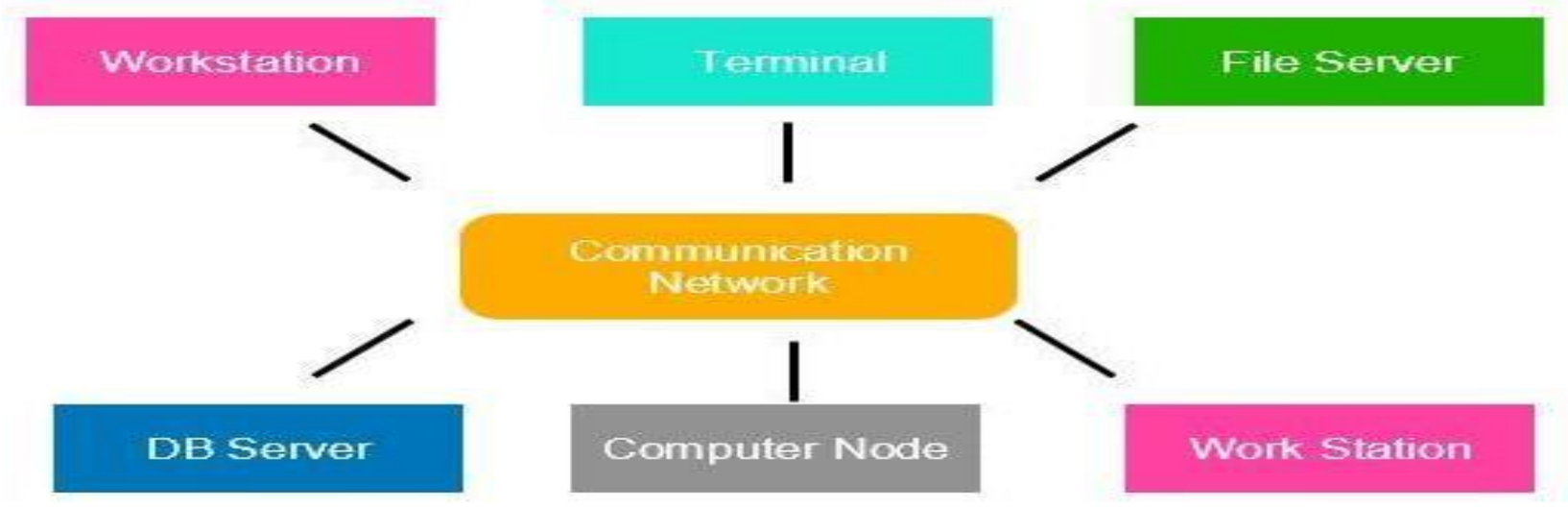
  Multics, Unix, etc.

4. Multiprocessor Operating Systems

- Multiprocessor operating systems are also known as parallel OS or tightly coupled OS.
- Such operating systems have more than one processor in close communication that sharing the *computer bus, the clock and sometimes memory and peripheral devices.*
- It executes multiple jobs at the same time and makes the processing ***faster.***
- It supports large physical address space and larger virtual address space.
- If one processor **fails** then other processor should retrieve the interrupted process state so execution of process can continue.
- Inter-processes communication mechanism is provided and implemented in hardware.

5. Distributed Operating System

- Various autonomous interconnected computers communicate with each other using a shared communication network.
- Independent systems possess their own memory unit and CPU.
- These are referred to as **loosely coupled systems**.
- Examples:- Locus, DYSEAC



6. Network Operating System

- These systems run on a **server** and provide the capability to *manage data, users, groups, security, applications, and other networking functions.*
- These types of OS allow shared access of **files, printers, security, applications, and other networking functions** over a small private network.
- The **“other"** computers are called client computers, and each computer that connects to a network server must be running client software designed to request a specific service.
- It is popularly known as ***tightly coupled systems.***

6. Network Operating System

Advantages of Network Operating System:

- Highly ***stable*** centralized servers
- Security concerns are handled through servers
- New technologies and hardware up-gradation are easily integrated into the system
- Server access is possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- Servers are costly
- User has to depend on a central location for most operations
- Maintenance and updates are required regularly

Examples of Network Operating System are:

Microsoft Windows Server 2003/2008/2012, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc.

7. Real-Time Operating System

- These types of OSs serve **real-time systems.**
- The time interval required to process and respond to inputs is very small.
- This time interval is called **response time**.
- **Real-time systems** are used when there are time requirements that are very strict like
  - missile systems,
  - air traffic control systems,
  - robots, etc.

8. Embaded Operating System

- An embedded operating system is one that is **built into the circuitry of an electronic device.**
- Embedded operating systems are now found in ***automobiles, bar-code scanners, cell phones, medical equipment, and personal digital assistants.***
- The most popular embedded operating systems for consumer products, such as PDAs, include the following:
  - Windows XP Embedded
  - Windows CE .NET:- it supports wireless communications, multimedia and Web browsing. It also allows for the use of smaller versions of Microsoft Word, Excel, and Outlook.
  - Palm OS:- It is the standard operating system for Palm-brand PDAs as well as other proprietary handheld devices.
  - Symbian:- OS found in " smart" cell phones from Nokia and Sony Ericsson

Popular types of OS

- Desktop Class
  - Windows
  - OS X
  - Unix/Linux
  - Chrome OS
- Server Class
  - Windows Server
  - Mac OS X Server
  - Unix/Linux
- Mobile Class
  - Android
  - iOS
  - Windows Phone

Desktop Class Operating Systems:-

- **Platform:** the hardware required to run a particular operating system
  - Intel platform (IBM-compatible)
    - Windows
    - DOS
    - UNIX
    - Linux
  - Macintosh platform
    - Mac OS
  - iPad and iPhone platform
    - iOS

Ms-DOS

- Single User Single Tasking OS.
- It had no built-in support for networking, and users had to manually install drivers any time they added a new hardware component to their PC.
- DOS supports only 16-bit programs.
- Command line user interface.
- So, why is DOS still in use? Two reasons are its size and simplicity. It does not require much memory or storage space for the system, and it docs not require a powerful computer.

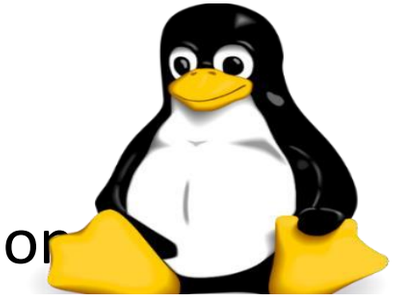
Microsoft Windows

- The graphical Microsoft operating system designed for Intel-platform desktop and notebook computers.
- Best known, greatest selection of applications available.
- Current editions include Windows 7, 8, 8.1 and 10,11.

Mac OS

- User-friendly, runs on Mac hardware. Many applications available.
- Current editions include: Sierra, High Sierra, Mojave, Catalina & Big Sur—Version XI(Released in Nov 2020)

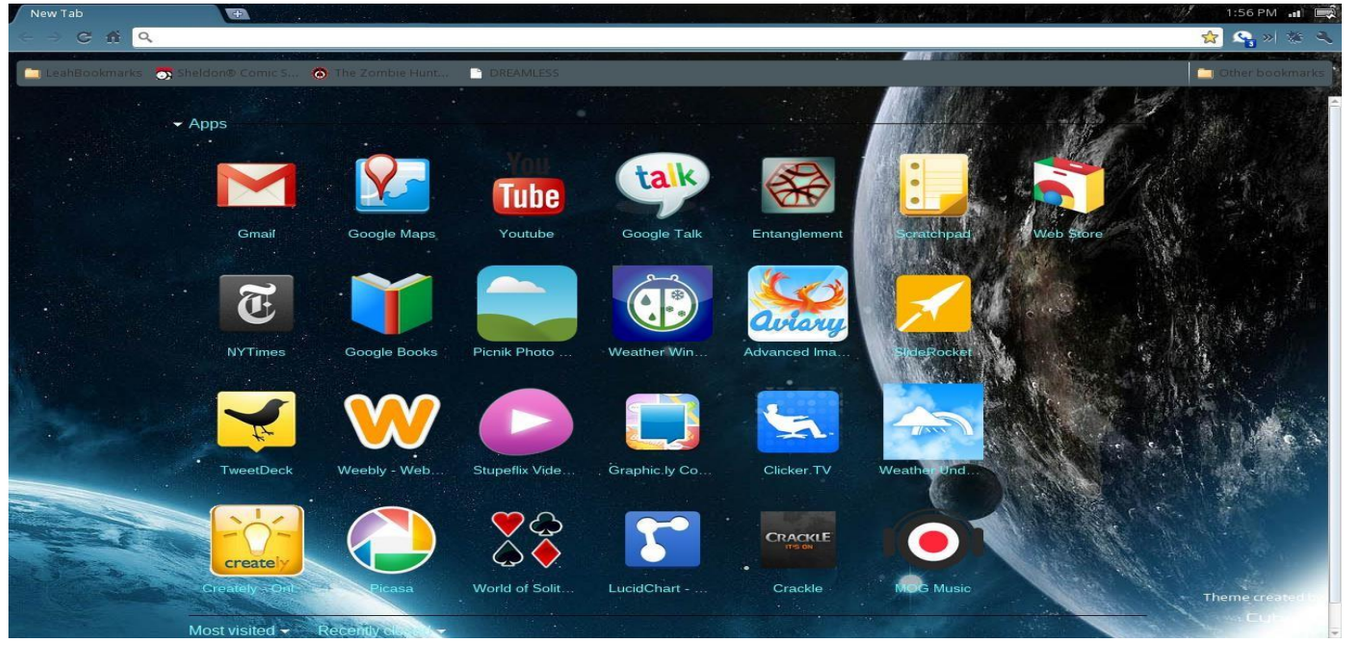
Linux



- **Linux:** An open-source, cross-platform operating system that runs on desktops, notebooks, tablets, and smartphones.
  - The name *Linux* is a combination *Linus* (the first name of the first developer) and *UNIX (another operating system*.
- Users are free to modify the code, improve it, and redistribute it.
- Developers are not allowed to charge money for the Linux kernel itself (the main part of the operating system), but they can charge money for **distributions** (**distros** for short).

Google Chrome OS

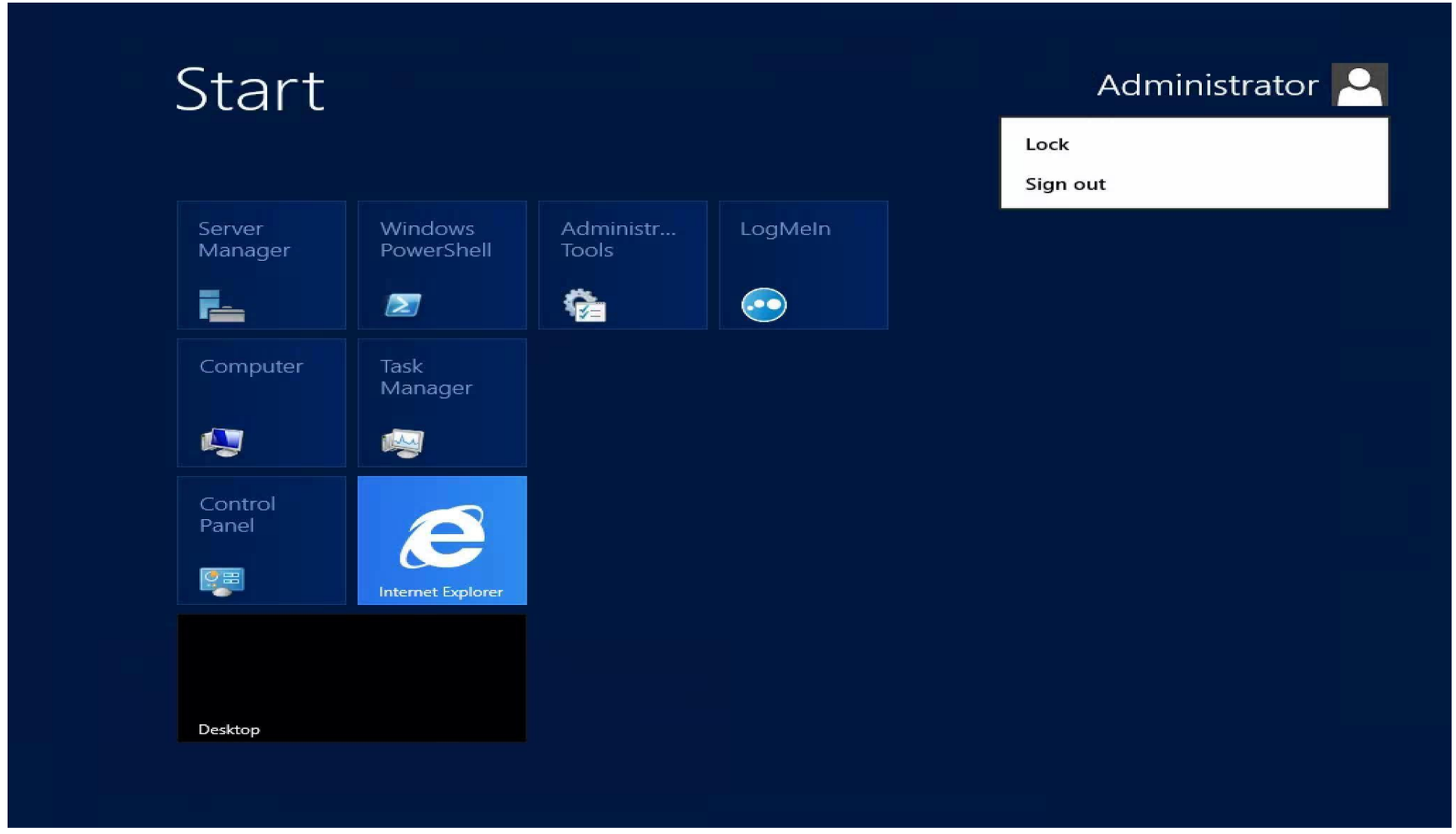
- **Chrome OS**. Is a popular thin client operating system.
- **Thin client** A computer with minimal hardware, designed for a specific task. For example, a thin web client is designed for using the Internet.



Server Operating Systems

- Windows Server
  - Familiar GUI interface for those experienced with Windows
- UNIX
  - Very mature server capabilities, time-tested, large user community, stable
- Linux
  - Free, customizable, many free services and utilities available

Windows Server



UNIX

```
mars@marsmain /usr/portage/app-shells/bash $ sudo /etc/init.d/bluetooth status
Password:
* status: started
mars@marsmain /usr/portage/app-shells/bash $ ping -q -c1 en.wikipedia.org
PING rr.esams.wikimedia.org (91.198.174.2) 56(84) bytes of data.

--- rr.esams.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 49.820/49.820/49.820/0.000 ms
mars@marsmain /usr/portage/app-shells/bash $ grep -i /dev/sda /etc/fstab | cut --fields=-3
/dev/sda1               /boot
/dev/sda2               none
/dev/sda3               /
mars@marsmain /usr/portage/app-shells/bash $ date
Sat Aug  8 02:42:24 MSD 2009
mars@marsmain /usr/portage/app-shells/bash $ lsmod
Module                  Size  Used by
rndis_wlan             23424  0
rndis_host              8696  1 rndis_wlan
cdc_ether               5672  1 rndis_host
usbnet                 18688  3 rndis_wlan,rndis_host,cdc_ether
parport_pc             38424  0
fglrx                2388128  20
parport                39648  1 parport_pc
iTCO_wdt               12272  0
i2c_i801                9380  0
mars@marsmain /usr/portage/app-shells/bash $
```

Tablet and Phone Operating Systems

- **System-on-chip (SoC):** An operating system that comes preinstalled on a chip on a portable device such as a smartphone.
- Popular SoC operating systems:
  - iOS: for iPad, iPhone
  - Android: for a variety of tablets and phones
- Downloadable applications (apps) from an App store, for example:
  - Apple App Store
  - Google Play Store

iOS on the iPhone and iPad

- The Apple-created operating system for Apple tablets and phones.
- The current stable version, iOS 14, was released to the public on September 16, 2020.

Android

- Android, a popular OS for smartphones and tablets, is based on Linux Kernel.
  - Developed by Google
- Current versions include:
  - Android 8 Oreo
  - Android 9 Pie
  - Android 10
  - Android 11 (released on Sep, 2020)

Advantage of Linux Operating System

1. Open Source

As it is open-source, its source code is easily available.
Anyone having programming knowledge can customize the operating system.
One can contribute, modify, distribute, and enhance the code for any purpose.

2. Security

The Linux security feature is the main reason that it is the most favourable option for developers.
It is not completely safe, but it is less vulnerable than others.
Each application needs to authorize by the admin user.
Linux systems do not require any antivirus program.

3. Free

Certainly, the biggest advantage of the Linux system is that it is free to use.
We can easily download it, and there is no need to buy the license for it.
It is distributed under GPL (General Public License).
Comparatively, we have to pay a huge amount for the license of the other OS

Advantage of Linux Operating System

4. Lightweight

The requirements for running Linux are much less than other operating system

In Linux, the memory footprint and disk space are also lower.

Generally, most of the Linux distributions required as little as 128MB of RAM around the same amount for disk space.

5. Stability

Linux is more stable than other operating systems.

Linux does not require to reboot the system to maintain performance levels.

It rarely hangs up or slow down. It has big up-times.

Advantage of Linux Operating System

6. Performance

Linux system provides high performance over different networks.

It is capable of handling a large number of users simultaneously.

7. Flexibility

Linux operating system is very flexible.

It can be used for desktop applications, embedded systems, and server applications too.

It also provides various restriction options for specific computers.

We can install only necessary components for a system.

8. Software Updates

In Linux, the software updates are in user control.

We can select the required updates.

There a large number of system updates are available.

These updates are much faster than other operating systems.

So, the system updates can be installed easily without facing any issue.

Advantage of Linux Operating System

9. Distributions/ Distros

There are many Linux distributions available in the market.

It provides various options and flavours of Linux to the users.

We can choose any distros according to our needs.

Some popular distros are **Ubuntu, Fedora, Debian, Linux Mint, Arch Linux,**

For the beginners, Ubuntu and Linux Mint would be useful.

Debian and Fedora would be good choices for proficient programmers.

10. Live CD/USB

Almost all Linux distributions have **a Live CD/USB** option.

It allows us to try or run the Linux operating system without installing it.

11. Graphical User Interface

Linux is a command-line based OS but it provides an interactive user interface like Windows.

Advantage of Linux Operating System

12. Suitable for programmers

It supports almost all of the most used programming languages such as C/C++, Java, Python, Ruby, and more.

Further, it offers a vast range of useful applications for development.

The programmers prefer the Linux terminal over the Windows command line.

The package manager on Linux system helps programmers to understand how things are done.

Bash scripting is also a functional feature for the programmers.

It also provides support for SSH, which helps in managing the servers quickly.

13. Community Support

Linux provides large community support.

We can find support from various sources.

There are many forums available on the web to assist users.

Further, developers from the various open source communities are ready to help us.

Advantage of Linux Operating System

14. Privacy

Linux always takes care of user privacy as it never takes much private data from the user. Comparatively, other operating systems ask for the user's private data.

15. Networking

Linux facilitates with powerful support for networking. The client-server systems can be easily set to a Linux system. It provides various command-line tools such as ssh, ip, mail, telnet, and more for connectivity with the other systems and servers. Tasks such as network backup are much faster than others.

16. Compatibility

Linux is compatible with a large number of file formats as it supports almost all file formats.

17. Installation

Linux installation process takes less time than other operating systems such as Windows. Further, its installation process is much easy as it requires less user input. It does not require much more system configuration even it can be easily installed on old machines having less configuration.

Advantage of Linux Operating System

18. Multiple Desktop Support

Linux system provides multiple desktop environment support for its enhanced use. The desktop environment option can be selected during installation. We can select any desktop environment such as **GNOME (GNU Network Object Model Environment)** or **KDE (K Desktop Environment)** as both have their specific environment.

19. Multitasking

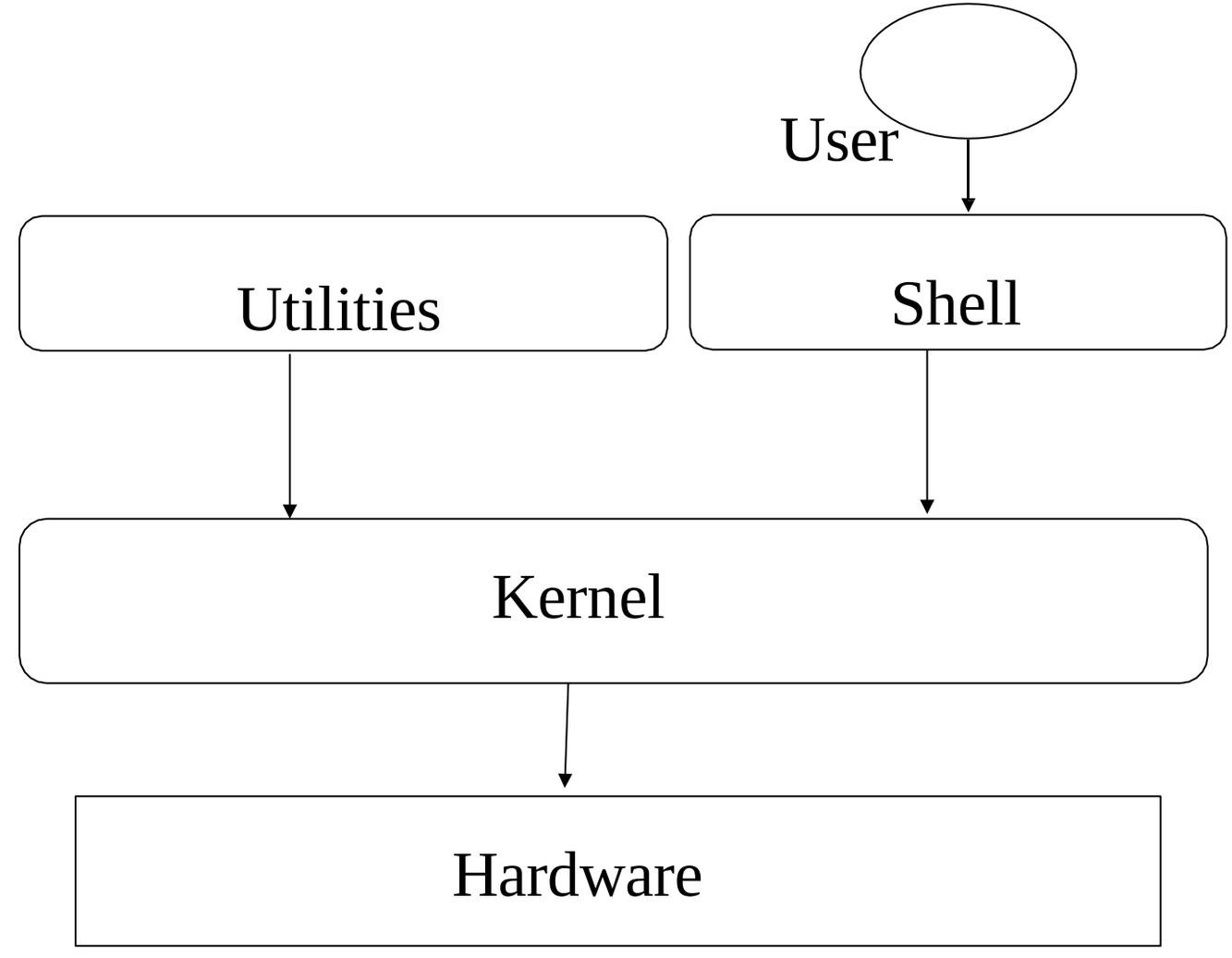
It is a multitasking operating system as it can run multiple tasks simultaneously without affecting the system speed.

20. Heavily Documented for beginners

There are many command-line options that provide documentation on commands, libraries, standards such as manual pages and info pages. Also, there are plenty of documents available on the internet in different formats, such as Linux tutorials, Linux documentation project, Serverfault, and more. To help the beginners, several communities are available such as **Ask Ubuntu**, Reddit, and **StackOverflow.**

UNIX Shell and Utilities

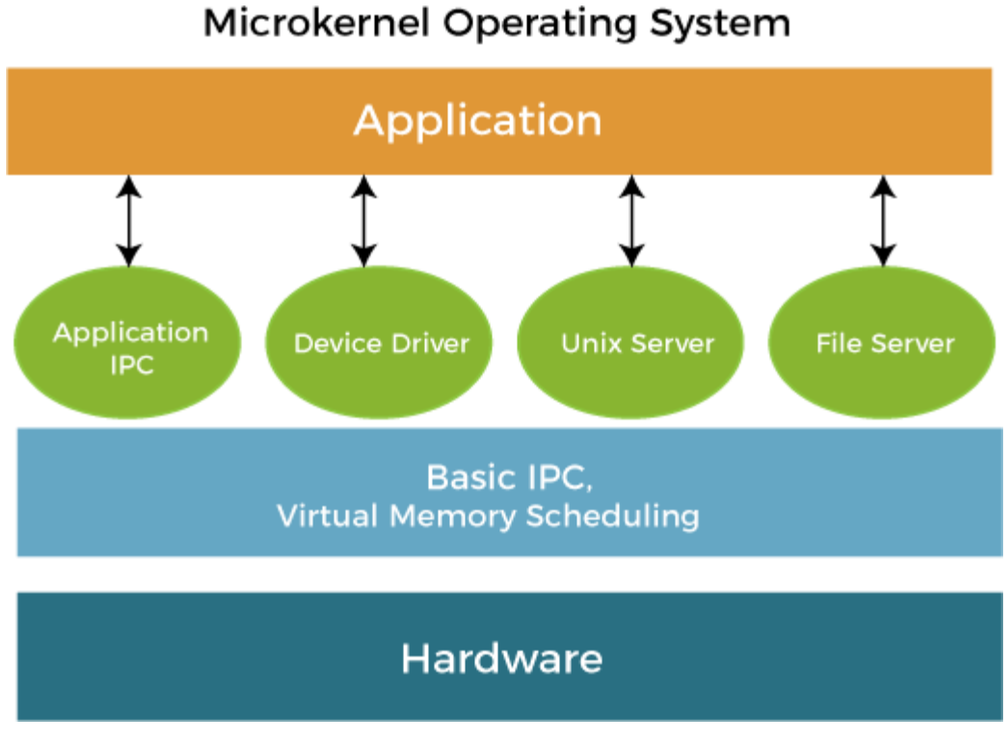
- The shell used to be in the kernel but now is a utility outside of it.
- Easy to change/debug.
- Many of them (sh, bsh, csh, ksh, tcsh, wsh, bash)
- Possible to switch between them (chsh)



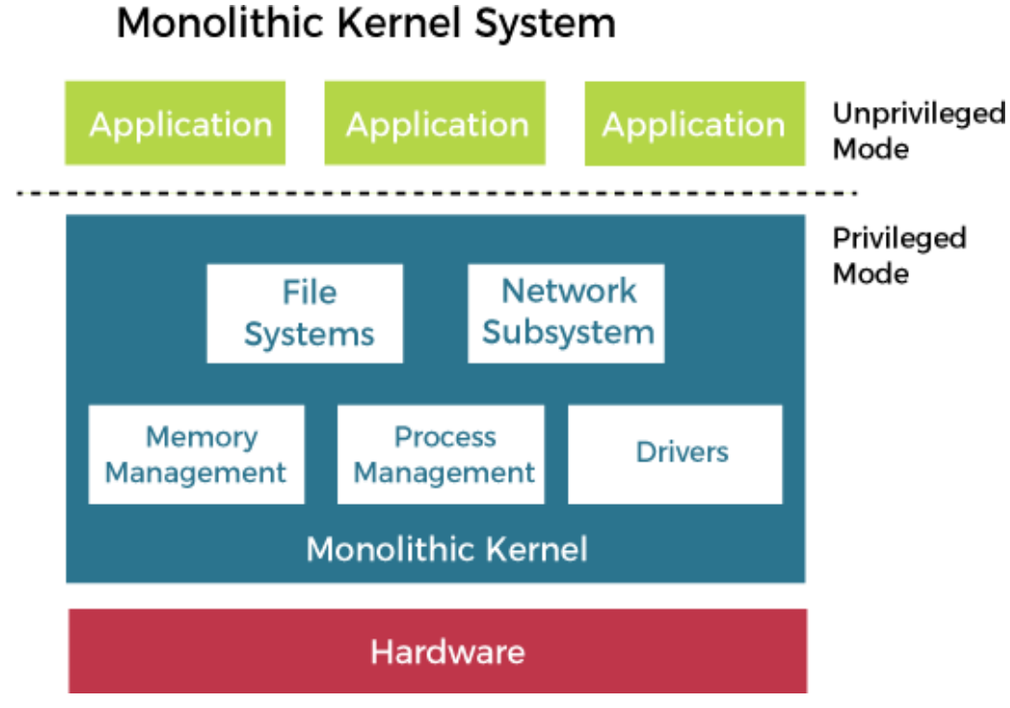
A very simplified Shell

```
#define TRUE 1

while (TRUE) {                                          /* repeat forever */
        type_prompt( );                                 /* display prompt on the screen */
        read_command(command, parameters);              /* read input from terminal */

        if (fork( ) != 0) {                             /* fork off child process */
             /* Parent code. */
             waitpid(–1, &status, 0);                   /* wait for child to exit */
        } else {
             /* Child code. */
             execve(command, parameters, 0);            /* execute command */
        }
}
```

Microkernel and Monolithic Operating System



Ref. https://www.javatpoint.com/monolithic-structure-of-operating-system

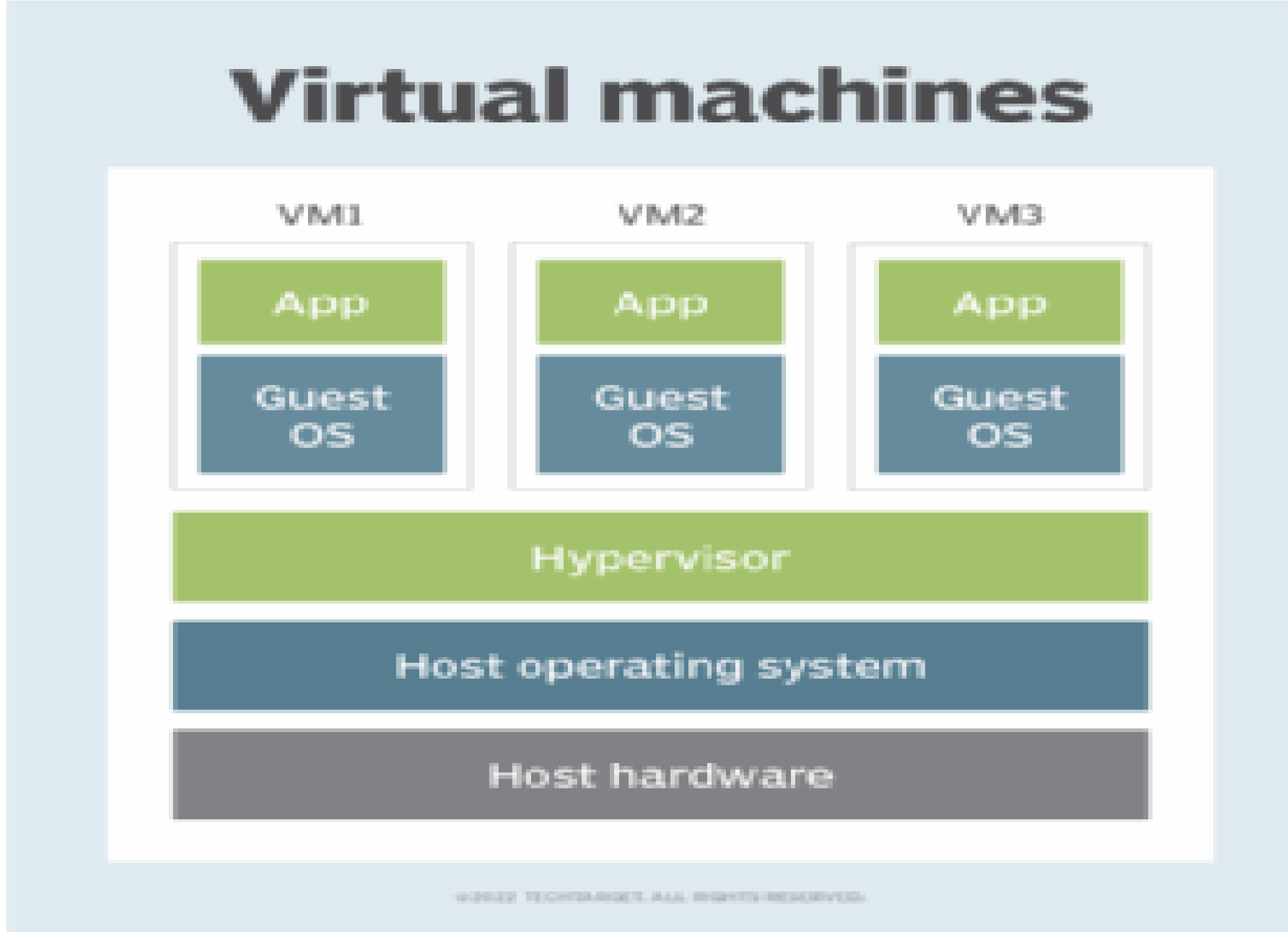


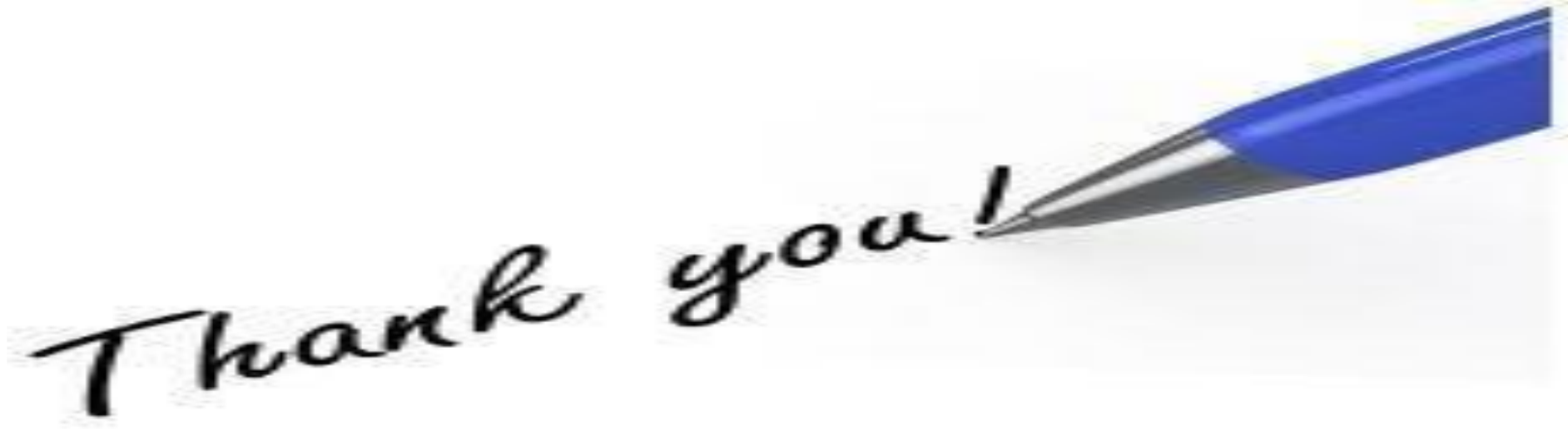
Ref. https://www.javatpoint.com/monolithic-structure-of-operating-system

Microkernel and Monolithic Operating System

Microkernel OS	Monolithic OS
Smaller in Size	**Larger in Size**
Slower	**Faster**
Less no of lines in Kernel	**Large no. of lines in Kernel**
Easy debug and Management	**Difficult**
Easy to add new functionality	**Difficult**
If one component crash, others keep resuming	**If one component crash, entire system will crash**

Virtual Machine




Thank you!