CH-1 Operating System

- An operating system is a system software which manages, operates and communicates with the computer hardware and software.
- OS acts as an interface between the user and hardware of the computer.
- Os controls the execution of application programs.
- Os is known as resource manager.

Need of OS.

- To perform tasks

  Example, Identify input from the input devices like keyboard, mouse. Send output to the output devices like moniter, printer.
- Keep track of files and directories
- Controlling peripheral devices such as secondary storage devices, printer, scanner, etc.
- Heart of computer system → Processing unit (CPU).
- OS allocates memory to user program as per need.

Objectives of OS

- ↳ Convenience
- ↳ Efficiency
- ↳ Ability to evolve

Functions of OS

→ Process Management
→ Memory Management
→ File Management
→ Device management
→ Protection and Security
→ User Interface or Command Interpreter
→ Booting the Computer
→ Performs basic computer tasks.

① Process Management

- Control access to shared resources like file, memory, I/O CPU
- Control the execution of user applications
- Creation, execution, deletion
- Scheduling of a process
- Synchronization, communication, deadlock handling

② Memory Management

- Allocates memory to the user and system processes
- Reclaim the allocated memory from the completed process
- Once used block becomes free, OS allocates it again
- Monitoring and keeping track of memory usage.

③ File Management

- Creation and deletion of files and directories.
- OS offer the service to access the files.
- Keeps back-up of files.
- Offers the security for files.

④ Device Management

- Device drivers are opened, closed, written by OS.
- Communicate, control and monitor

⑤ Protection and security
- Resources are protected by OS.
- OS makes use of user authentication for protection
- Read, write, encryption, backup of data.

⑥ User interface
- Offers set of commands or GUI.
- Software and hardware interaction

⑦ Booting the computer
- Process of starting or restarting the computer.
- Switched off → Turned on → Cold booting
- Using OS → Restart → Warm booting

⑧ Performs basic computer tasks
- By input and output devices
- Plug and play → Automatic recognition

Evolution of OS

	Decade	Machines	OS and Features
①	1970	Mainframes	- MULTICS - Multiuser, timesharing
②	1980	Minicomputers	- UNIX. - Multiuser, timesharing, networked
③	1990	Desktop Computers	- DOS, MS-WINDOWS Windows-95, Windows 98, Windows XP (For Desktop users), Windows NT, UNIX / LINUX, etc. - Multiuser, timesharing, networked, clustered.
④	2000	Handheld Computers	- Windows XP (For Desktop users) Windows 2000, Linux. - Multiprocessor, Multiuser, Networked.

Types of OS

- Multiuser
- Multiprocessing
- Multitasking
- Multithreading
- Real time

① Multiuser

- No. of users to work on same computer at the same time
- Some OS permits 100 or 1000 users working concurrently
- Multiuser OS → Linux, UNIX, Windows 2000, Windows NT.

② Multiprocessing

- Multiple CPU in one system.
- Parallel execution of threads, efficiency, performance achieve
- Multiprocessing OS → UNIX, Windows 2000

③ Multitasking

- Execution of multiple softwares at the same time.
- Multiple task handling
- Multitasking OS → UNIX, Windows 2000

④ Multithreading

- Single programs can run multiple threads parallely
- OS divides program in threads
- OS → Linux, UNIX, Windows 2000

⑤ Real Time

- Quick response
- UNIX and Windows are not real time OS

OS Services

→ User interface

- Program execution
- I/O operations
- File system manipulation
- Communications
- Error Detection

① User Interface
- Essential
- Command line interface or graphical user interface
- mouse-based and menu system.

② Program Execution
- OS provides environment
- OS provides resources required for execution.
- Memory allocation, processor allocation, scheduling, multitasking.

③ I/O operations
- Till I/O completes, program goes in waiting state.

④ File System manipulation.
- Files are stored in secondary storage device and manipulated in main memory
- User need not to worry about secondary storage devices
- Program execution speed is fundamental to I/O operations

(5) Communications:

- Hardware → OS → Software
- User → Applications

(6) Error Detection

- Detect errors
- User program kept free
- OS needs to carry complex tasks includes memory process.

Types of OS considered for design consideration

→ Symmetric multiprocessor (SMP) OS

- Multicore OS

Key design issues of multiprocessor OS

- Simultaneous concurrent processes or threads.
- Scheduling
- Synchronization
- Memory management
- Reliability and fault tolerance

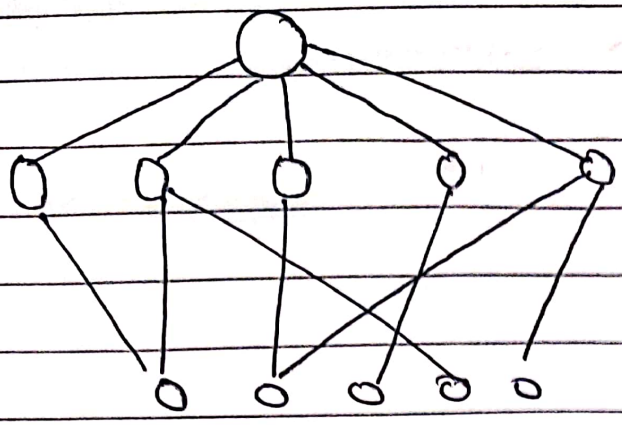
Extract parallelism from given application

- Parallelism with application
- Virtual Machine Approach.

OS structures

- Monolythic Systems
- Layered Systems
- Virtual machines
- Client Server Model

(1)



Structure of monolythic system

(2)

5

The Operator

↓ 4

User Programs

↓ 3

I/O Management

↓ 2

Operator-process communication

↓ 1

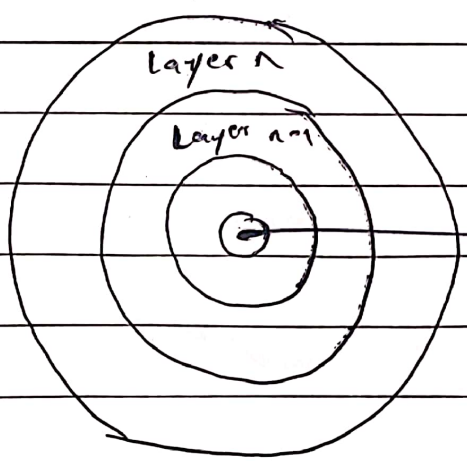
Memory and drum management

↓ 0

Process allocation and multiprogramming

Layered System

(3)



Monolythic Kernel	Micro Kernel
① It is a single large processes running entirely in a single address space	① The Kernel is broken down into separate processes known as servers
② All Kernel services exists and executes in kernal address space.	② Some servers run in Kernel space and some in user space. All servers are separate and runs in different address space
③ The Kernel can invoke function directly	③ Communication is done via message passing.
④ Advantage - Addition / removal is not possible less / zero flexible	④ Advantage - Flexible for changes.
⑤ Disadvantage - Inter component communication is better	⑤ Disadvantage - Communication overhead

System calls.

- Provides an interface to the services available by OS.
- Two modes of operation.
  1] User mode - All user processes are executed
  2] System mode - All privileged operations are executed
- Def^n.:
  System calls provide an interface to the services made available by an OS.
- It exposes all kernel functionalities that user mode program require.
- Advantage : Security
  Due to system calls user program is not able to enter into the OS.

Calls	Description
① Open()	A program initializes access to a file in a file system using the open system call
② Read()	A program that needs to access data from a file stored in a file system uses the read system call.
③ Write()	It writes data from a buffer declared by the user to a given device, maybe a file.
④ Exec()	exec is a functionality of an OS that runs an executable files in the context of an prerunning process.
⑤ Fork()	Fork is an operation whereby a process creates a copy of itself.

Kernel

- A Kernel is the central part of an OS.
- It manages the operations of the computer and the hardware, memory and CPU time.
- When a process makes requests of the kernel, It is called a System call.

Responsibilities of Kernel

→ Acts as a standard interface to the system hardware.
- Manage computer resources
- Put into effect isolation between processes.
- Implement multitasking

Types of System calls.

- Process Control
- Device Manipulation
- Communications
- File Manipulation
- Information Maintenance

Examples of system calls.

- Open
- Close
- Read
- Write
- Create Process

Group	Examples
① Process Control	end, abort, load, execute, create, terminate, get process attributes, set process attributes, wait for time, wait event, allocate and free memory
② Device Manipulation	request device, release device, read, write, reposition, logically attach or detach devices
③ Communications	create, delete communication connection, send, receive message, transfer status information
④ File manipulation	create, delete file, open, close, read, write, reposition
⑤ Information maintenance	get time, or date, set time or date, get system data, set system data, get process, set process

Linux Kernel

- Main component of a Linux OS.
- Core interface between hardware and its processes
- **Kernel jobs**
  - ↳ Memory management : Keep track on memory
  - ↳ Process management : Determine which processes can use the CPU
  - ↳ Device drivers : Act as mediator between hardware and processes
  - ↳ System calls and security : Receive requests for service from the processes.

Shell

- Shell is a UNIX term for the Interactive User Interface with an OS.
- Shell is the layer of programming that understands and executes the commands.
- Shell is called as Command Interpreter.

CH-2 Process concept and scheduling

Process

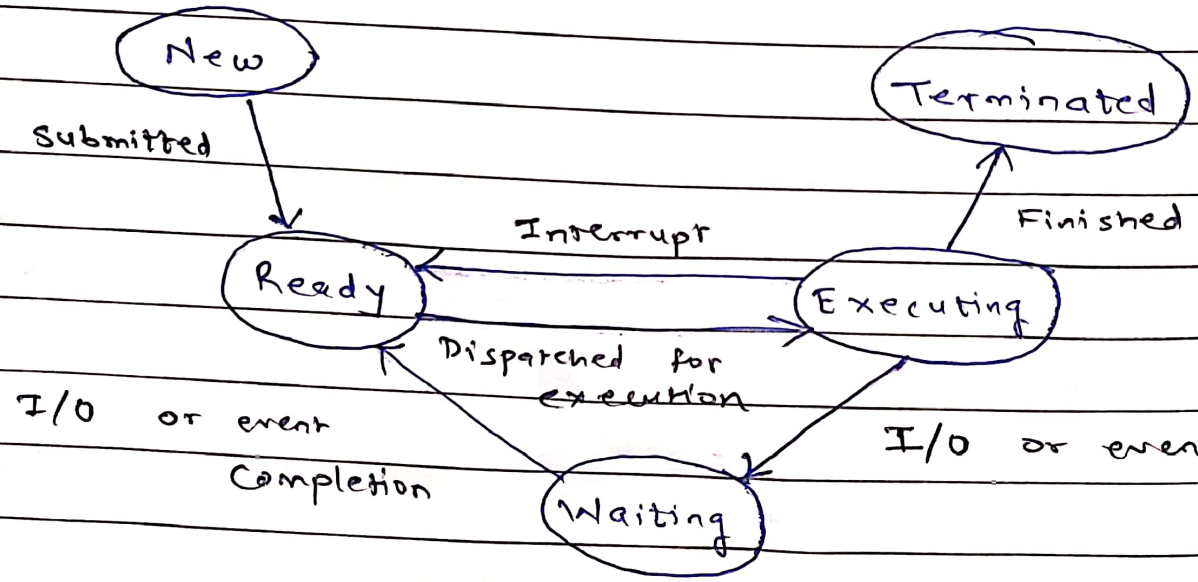
- Process is a program under execution
- Process is not as same as program code but a lot more than it.

Context Switching

- It occurs when a computer's CPU switches from one process or thread to a different process or thread
- 3 situations where context switch is necessary
  - ↳ Multitasking: CPU needs to switch processes in and out of memory so more process should run.
  - ↳ Kernel/User switch: Switching between user mode to kernel mode, it may be used
  - ↳ Interrupts: When a CPU is interrupted to return data from a disk read

Process States

Transition diagram:



① New state: The new process being created

② Ready state: A process is ready to run but it is waiting for CPU being assigned to it

③ Executing state/ Running state: A process is running if CPU is allocated to it and it is executing

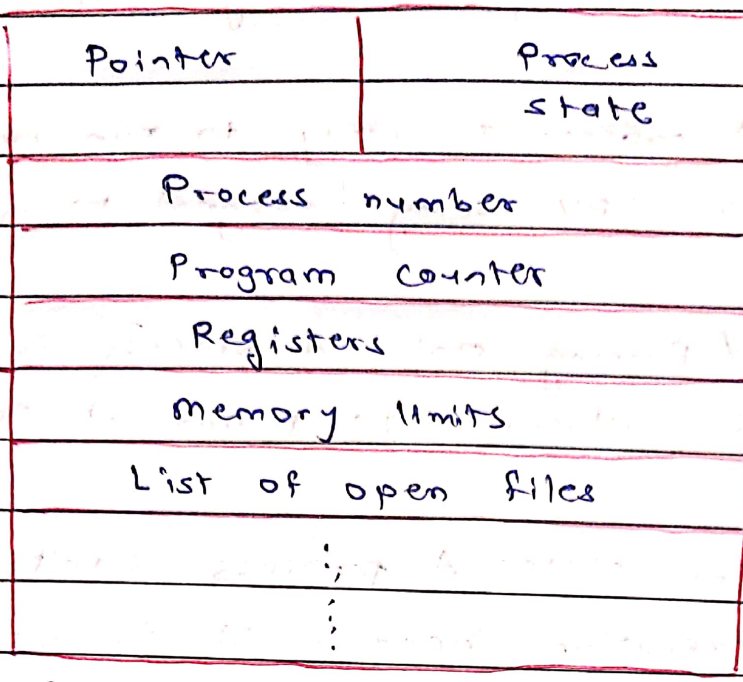
④ Waiting / blocked state: A process can't continue the execution because it is waiting for event to happen such as I/O completion.

⑤ Terminated state: The process has completed execution

Process Control Block

- Each process is represented in OS by PCB known as Task Control Block
- The OS groups all information that needs about a particular process into a data structure called as PCB.
- When a process is created, OS creates a corresponding PCB and released whenever the process terminates.
- Information stored in PCB includes:

i] Process name (ID)
ii] Priority
iii] State (ready, running, suspended)
iv] Hardware state (process registers and flags)
v] Scheduling information and usage statistics
vi] Memory management information
vii] I/O status (allocated devices, pending operations)
viii] File management information (open files, access rights)
ix] Accounting information



Process Control Block (PCB)

Description of PCB :

(1) Process state : The state may be new, ready, running, waiting, halted and so on.

(2) Program counter : Counter indicates the address of the next instruction to be executed.

(3) CPU Registers : Registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers and general purpose registors.

(4) CPU scheduling information : This information includes a process priority, pointers to scheduling queues.

(5) Memory Management Information : This information may include such information as the value of base and limit registers.

⑥ Accounting information: It includes the amount of CPU and real-time used, time limits, account numbers, job or process numbers and so on.

⑦ I/O status information: It includes the list of I/O devices allocated to the process, a list of open files and so on.

Operations on processes

- Process creation
- Process termination

Events of process creations

i] System initialization

ii] If running process executes process creation system call.

iii] A user request to create a new process

iv] Starting a batch job.

Process and Threads

	Process	Thread
Defn.	① Program in execution called as process. It is heavy weight process	① Thread is part of proces It is light weight proce
Context Switch.	② Context switch takes more time.	② Context switch takes l time.
Creation	③ New process creation takes more time	③ New thread creation t less time.
Termination	④ New process termination takes more time	④ New thread termination takes less time.
Execution	⑤ Each process executes the same code but has its own memory and file resources	⑤ All threads can share same set of open files, child processes.

Types of threads

↳ User Level Threads

↳ Kernel Level Threads

User Level Thread	Kernel Level Thread
① Implemented by user	① Implemented by OS.
② OS doesn't recognize user level threads	② Kernel threads are recognized by OS.
③ Implementation of User threads is easy	③ Implementation of Kernel threads is complicated.
④ Context switch time is less	④ Context switch time is more
⑤ Context switch requires no hardware support	⑤ Hardware support is needed
⑥ If one user level thread performs blocking operation then entire process will be blocked	⑥ If one kernel thread perform blocking operation then another thread can continue execution
⑦ Example: Java Thread POSIX Thread	⑦ Example: Window Solaris

Multithreading

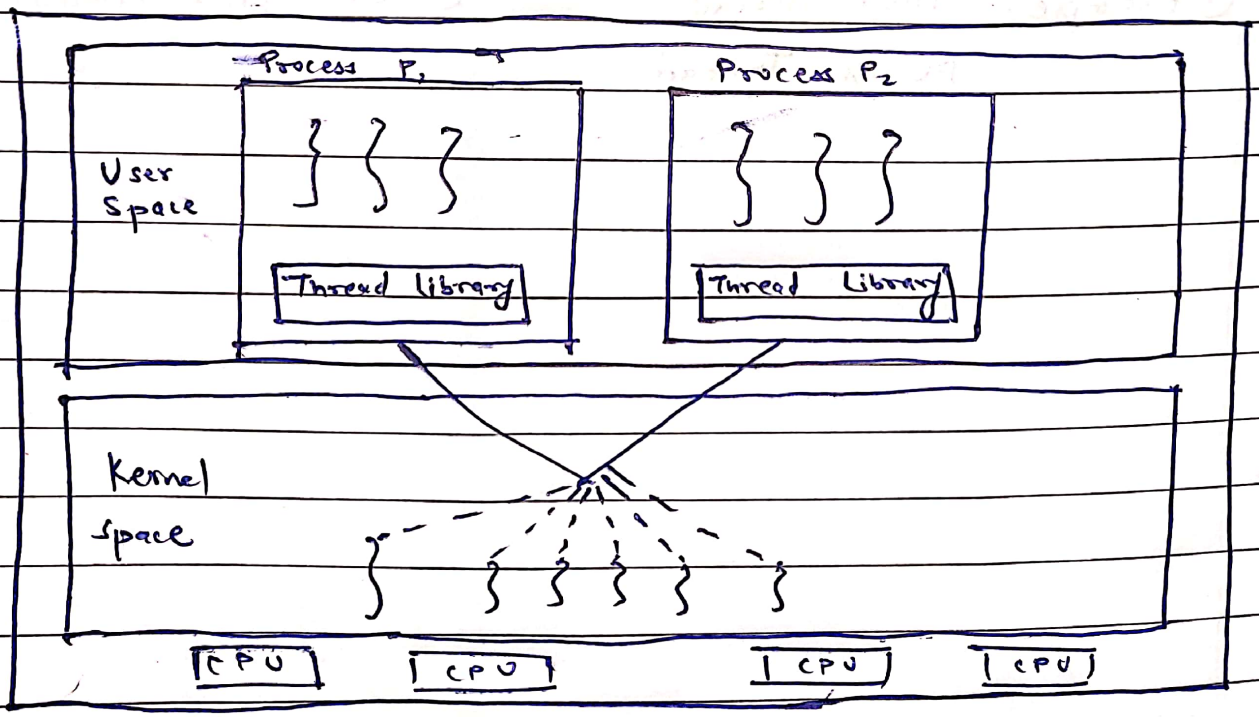
- The very basic unit which is assign to perform any task in an running process

- Some OS provide a combined user level thread and kernel level thread facility. Solaris is a good example of this combined approach
- In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not to block entire process.

↳ Many to many relationship
↳ Many to One relationship
↳ One to one relationship

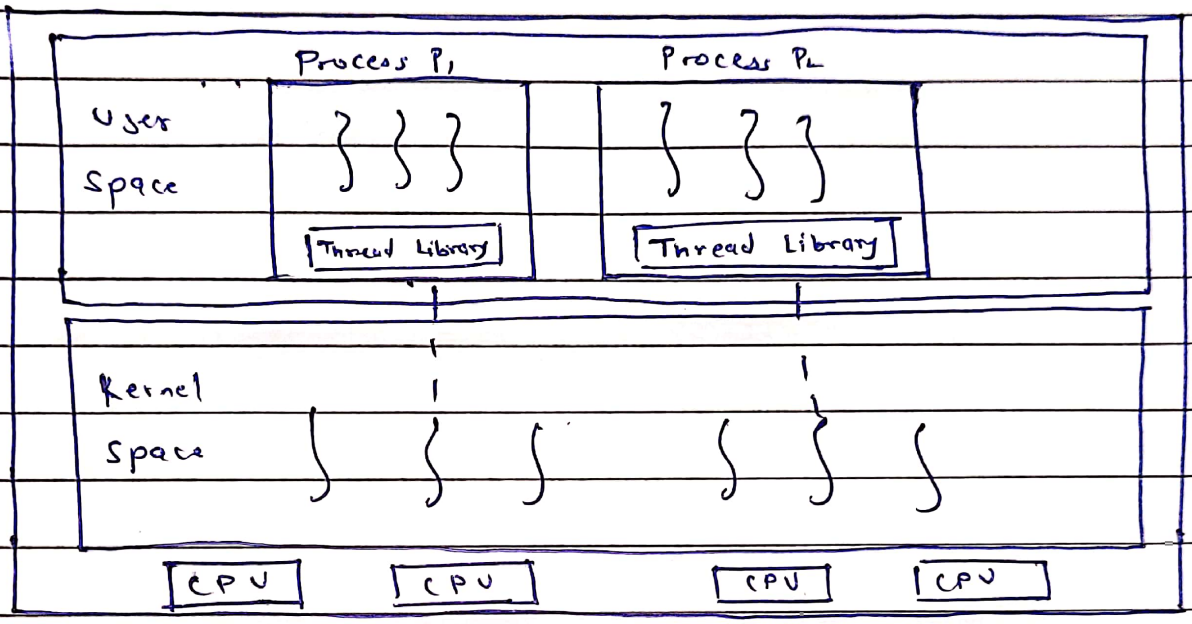
Many to many model

The many to many model multiplexes any number of user threads onto an equal or smaller number of kernel threads.



Many to One Model

Many to one model maps many user level threads to one kernel level thread. Thread management is done in user space by the thread library. When thread makes a blocking system call, the entire process will be blocked. Only one thread can access the kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.



One to One model

There is one to one relationship of user level thread to the kernel level thread. This model provides more concurrency than the many to one model. It also allows another thread to run when a thread makes a blocking system call. It supports multiple threads to execute parallely