Golam Kibria Tuhin(it-18015)

Q-1.

1(a) What is an Operating System ? What are main goal of Operating System ?

Answer:

Operating System : Operating System can be view as resource allocator . A computer system has many resources that may be required to solve a problem CPU time m memory space ,file-storage space and so on . It is the duty of the operating system to allocate these resources.

Goals of OS

- To execute programs.
- To solve user problems easily.
- To make the computer easy to use.

1(b) What does operating system do ? Explain why operating System can be viewed as resource allocator ?

Answer:

Operating system is a framework for all other software run in the computer. It provides some set of services and deals with devices in some organized way. These services could be for instance:
- maintaining process priorities
- maintaining file access
- queuing requests for exclusive resources

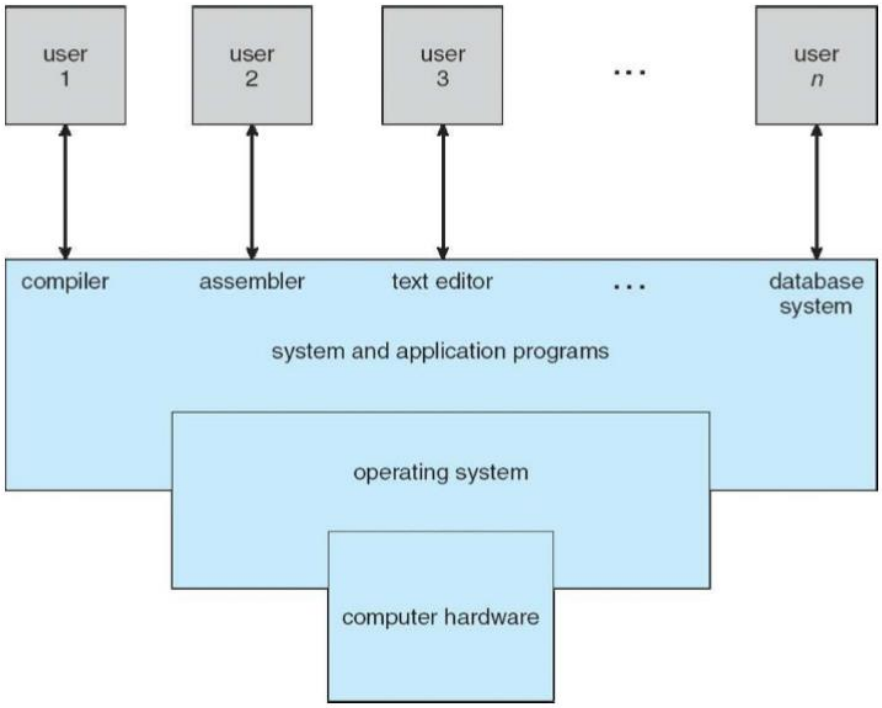
and many many more. It sets, executes and keeps some rules according to which other programs deal with each other and deal with hardware.

Operating System can be viewed as resource allocator:

A computer system has many resources - hardware & software that may be required to solve a problem, like CPU time, memory space, file-storage space, I/O devices & so on. The OS acts as a manager for these resources so it is viewed as a resource allocator.

1(c) What are System components of an operating System ?

Answer:



Q-2.

2(a) What is VMM ? Why we used emulation and virtialization ?

Answer:

VMM:

A Virtual Machine Monitor (VMM) is a software program that enables the creation, management and governance of virtual machines (VM) and

manages the operation of a virtualized environment on top of a physical host machine.

VMM is also known as Virtual Machine Manager and Hypervisor. However, the provided architectural implementation and services differ by vendor product.

Emulation used when source CPU type different from target type ..

- Generally slowest method.
- When computer language not compiled to native code – Interpretation .

Virtualization – OS natively compiled for CPU, running guest OSes also natively compiled

- Consider VMware running WinXP guests, each running applications, all on native WinXP host OS .

- VMM (virtual machine Manager) provides virtualization services.

2(b) Define multiprocessor system? What are the three advantages and disadvantages of multiprocesspr system?

Answer:

Multiprocessor : A Multiprocessor is a computer system with two or more central processing units (CPUs) share full access to a common RAM. The main objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching.

Advantages :

- Execute programs more quickly.
- Can have increase reliabilitu.

- It can save money by not dupliacating power supplies housing and peripherals.

Disadvantages :

- They are more complex in hardware.
- They are more complex in software.
- Large main memory required.

2(c)Describe storage management . Write some common function of interrupts .

Answer:

Storage management usually refers to the management of Computer data storage, which includes memory management. It can also refer to specific methods or products for storage management, such as the following

- ADSTAR Distributed Storage Manager
- Automatic Storage Management
- Hierarchical storage management
- IBM Tivoli Storage Manager
- OpenView Storage Area Manager
- Storage Management Initiative - Specification
- Storage Resource Manager
- Storage Resource Management

Common functions of Interrupts :

Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines.
Interrupt architecture must save the address of the interrupted instruction.
Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.
A trap is a software-generated interrupt caused either by an error or a user request.
An operating system is interrupt driven.

Q-3.

3(a) what is process ? Defference between process and thread .

Answer:

Process : A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

Difference between Process and Thread:

S.NO	PROCESS	THREAD
1.	Process means any program is in execution.	Thread means segment of a process.
2.	Process takes more time to terminate.	Thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	Process is less efficient in term of communication.	Thread is more efficient in term of communication.
6.	Process consume more resources.	Thread consume less resources.
7.	Process is isolated.	Threads share memory.

8.	Process is called heavy weight process.	Thread is called light weight process.
9.	Process switching uses interface in operating system.	Thread switching does not require to call a operating system and cause an interrupt to the kernel.

3(b) Mention the advantage and disadvatages of using process and threads in different sector.

Answer:

Advantages :

1. The context switching time is reduced using threads. With traditional methods, it takes longer to switch the contexts between different processes even though they belong to the same OS. This helps in managing the time of the tasks.
2. While using threads, one task after the other is carried out without being instructed always. Hence concurrency is achieved for the entire process using threads.
3. Communication between threads and communication between processes is made efficient with the help of threads. This helps to manage the process without being tracking the entire process using a tracker. This reduces costs.

4. Since it is easy to do context switching, the cost is less and hence the entire process is economical to create and manage and switch the threads between the processes.

Disadvantages :

1. All the variables both local and global are shared between threads. This creates a security issue as the global variables give access to any process in the system.
2. When the entire application is dependent on threads, if a single thread breaks, the entire process is broken and blocked. Thus the application is crashed. This particularly happens when the application runs the process with a single thread. When many threads are used, the threads crash each other making the communication difficult.
3. Threads depend on the system and the process to run. It is not independent. Also, the execution of the process via threads is time-consuming. But processes cannot be run without threads.
4. Threads are not reusable and it requires more hardware than software due to application changes from the base. Threads cannot be made work without process as they do not have their own address space.

3(c) What are multithreading model ? Describe briefly with example

Answer :

Multithreading Models

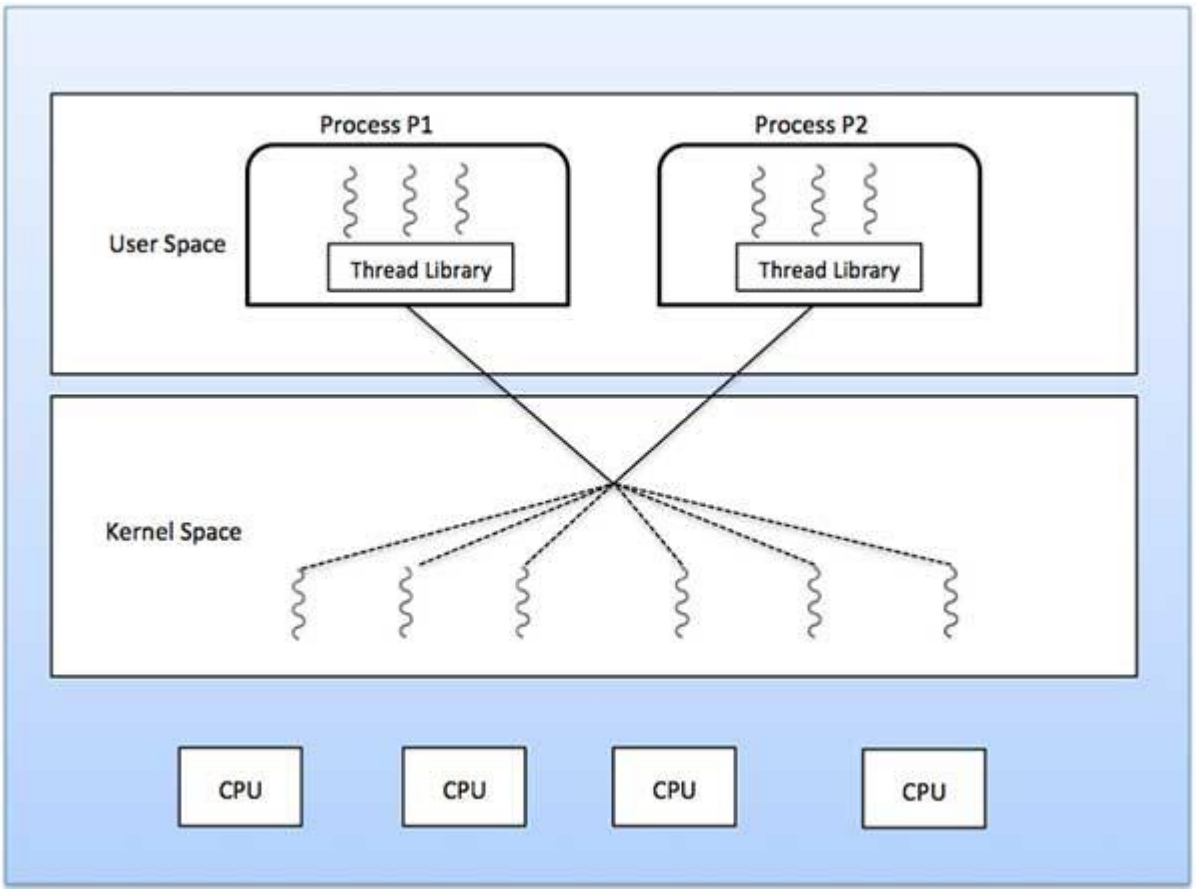
Some operating system provide a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

- 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.

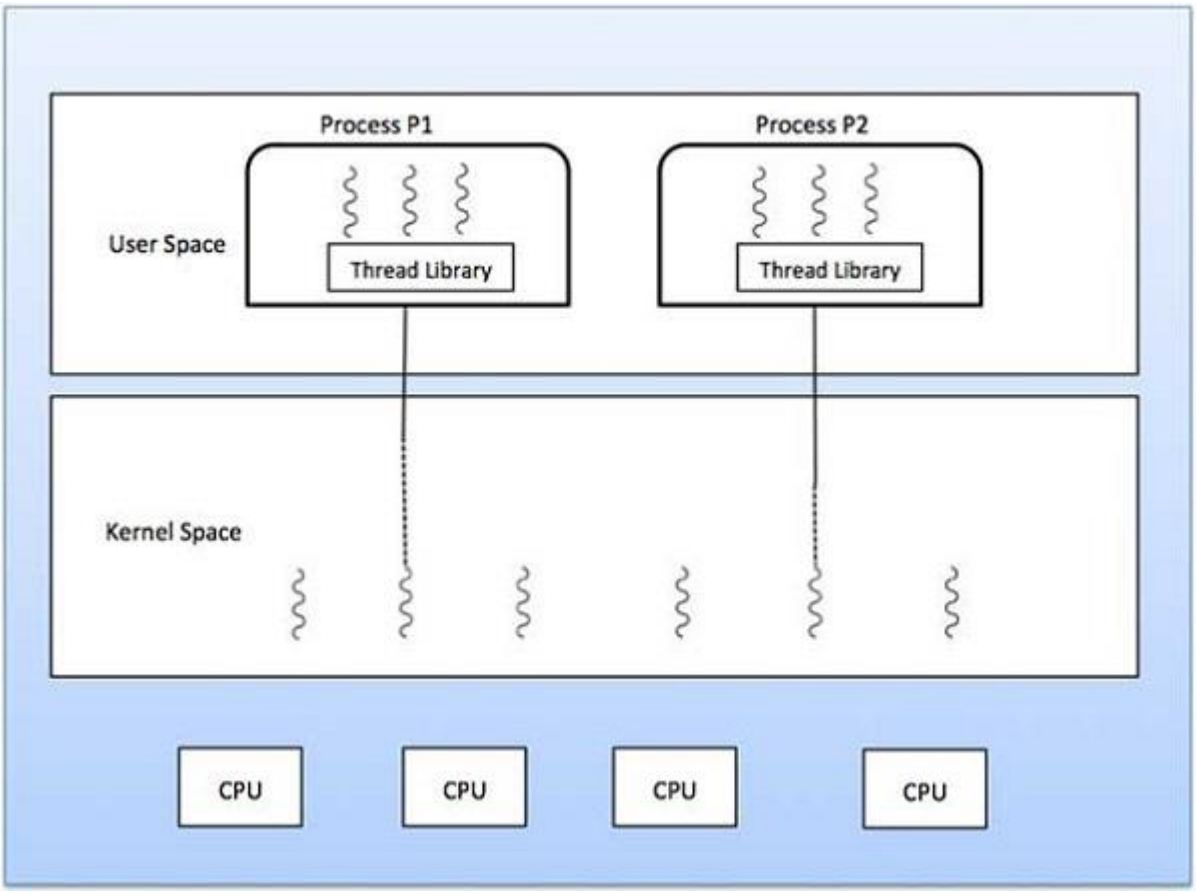
The following diagram shows the many-to-many threading model where 6 user level threads are multiplexing with 6 kernel level threads. In this model, developers can create as many user threads as necessary and the corresponding Kernel threads can run in parallel on a multiprocessor machine. This model provides the best accuracy on concurrency and when a thread performs a blocking system call, the kernel can schedule another thread for execution.



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.

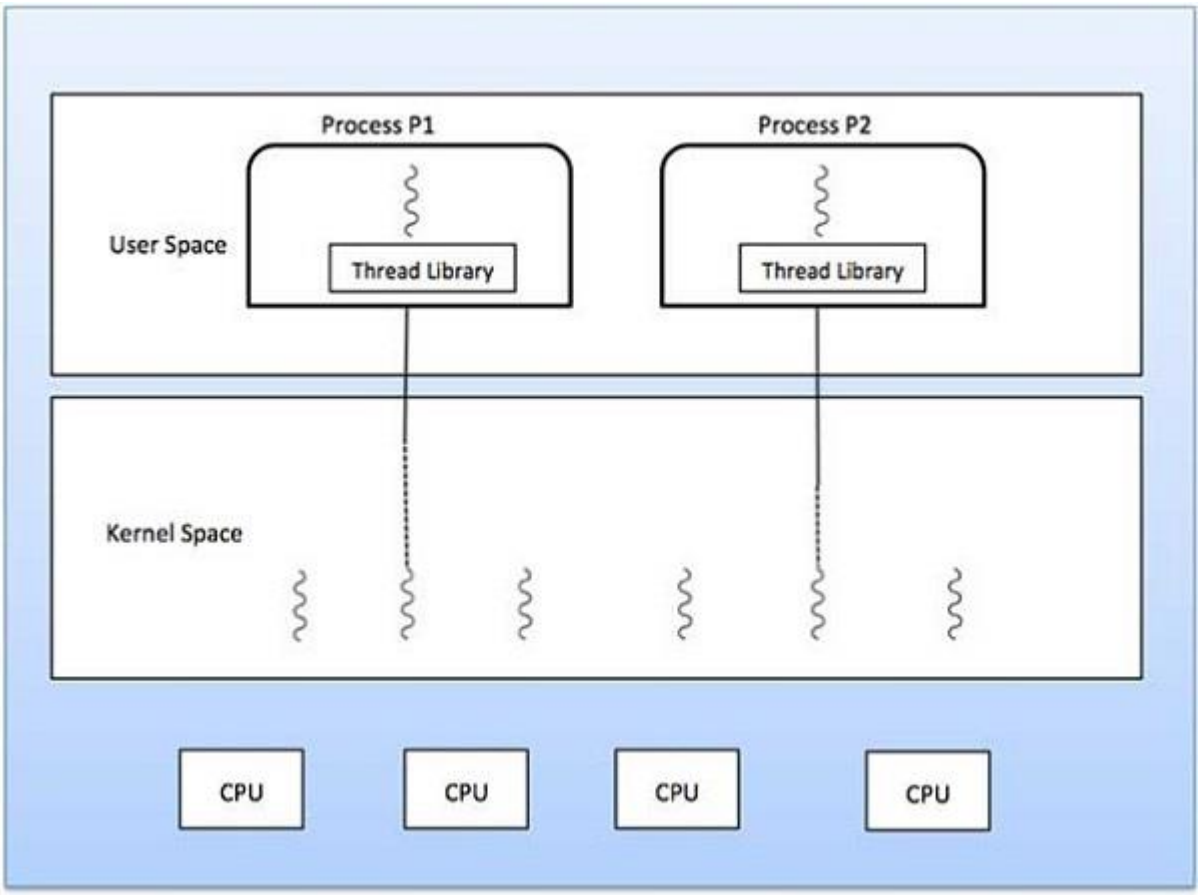
If the user-level thread libraries are implemented in the operating system in such a way that the system does not support them, then the Kernel threads use the many-to-one relationship modes.



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 in parallel on microprocessors.

Disadvantage of this model is that creating user thread requires the corresponding Kernel thread. OS/2, windows NT and windows 2000 use one to one relationship model.



Q-4.

4(a)What is User goals and System goals ?

Answer :

 User goals and System goals

  User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast

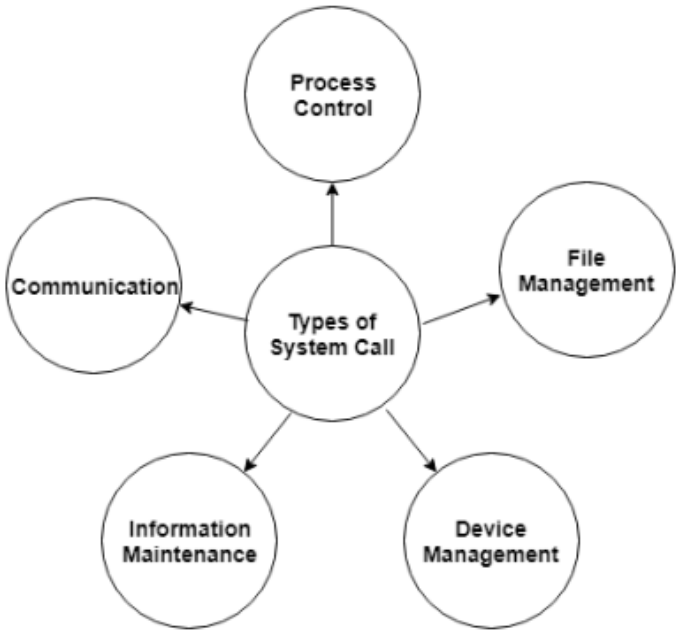
 System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

4(b) Describe the types of system calls and storage device hierarchy .

Answer :

Types of System Calls

There are mainly five types of system calls. These are explained in detail as follows −



Here are the types of system calls −

Process Control

These system calls deal with processes such as process creation, process termination etc.

File Management

These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

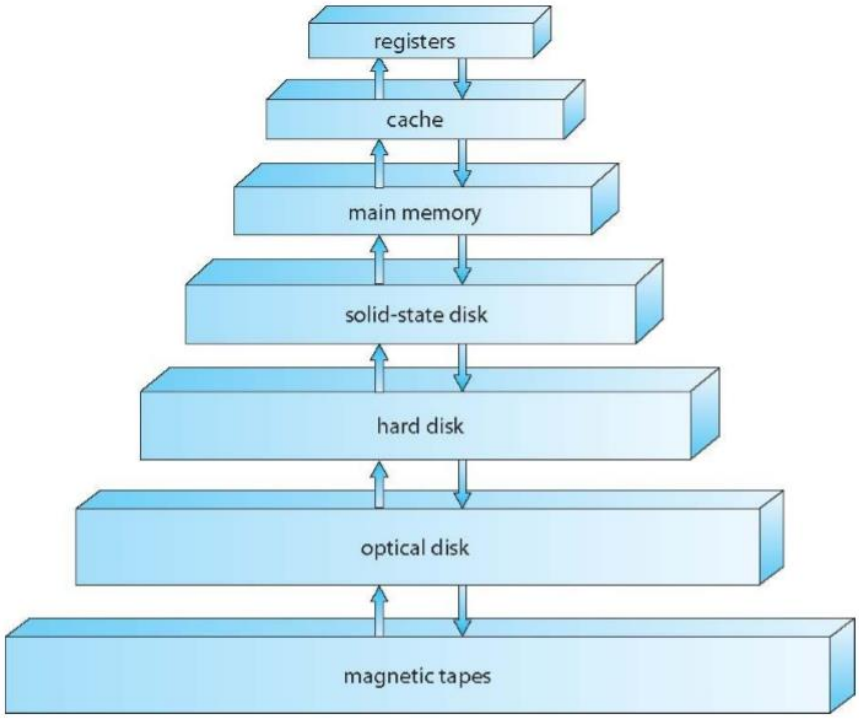
Device Management

These system calls are responsible for device manipulation such as reading from device buffers, writing into device buffers etc.

Information Maintenance

These system calls handle information and its transfer between the operating system and the user program.

Storage device hierarchy



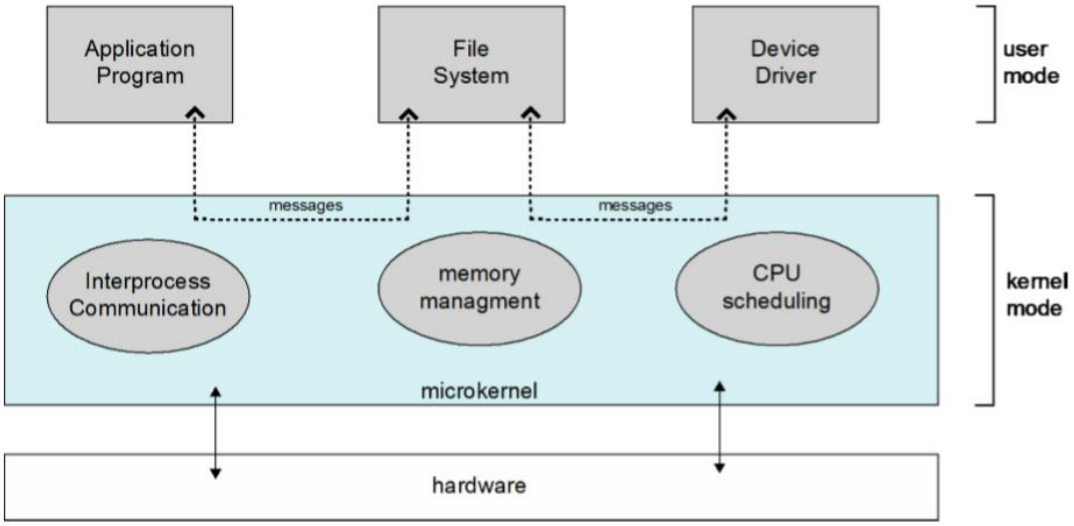
4(c) Describe Microkernel system structure and draw a diagram of Microkernel system structure .

Answer :

Microkernel system structure :

- Moves as much from the kernel into user space
- **Mach** example of **microkernel**
  - Mac OS X kernel (**Darwin**) partly based on Mach
- Communication takes place between user modules using **message passing**
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user space to kernel space communication

Diagram of Microkernel system structure :



Q-5 :

5(a).What is Process scheduler ? how to maintain schedulling queue of processes ?

Answer :

Process Scheduler

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. Process scheduling is an essential part of a Multiprogramming operating systems.

Process Scheduling Queues help you to maintain a distinct queue for each and every process states and PCBs. All the process of the same execution state are placed in the same queue. Therefore, whenever the state of a process is modified, its PCB needs to be unlinked from its existing queue, which moves back to the new state queue.

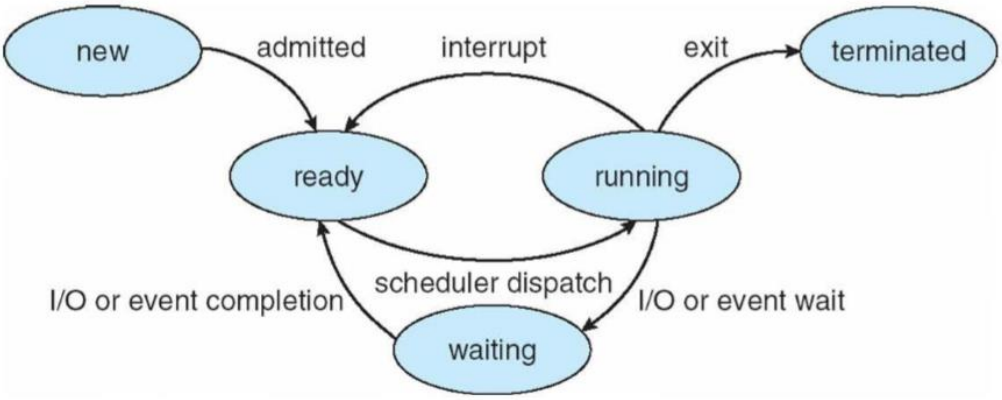
Three types of operating system queues are:

1. **Job queue** – It helps you to store all the processes in the system.
2. **Ready queue** – This type of queue helps you to set every process residing in the main memory, which is ready and waiting to execute.
3. **Device queues** – It is a process that is blocked because of the absence of an I/O device.

5(b). Write a Diagram of process state . how to execute process ?

Answer :

Diagram of process state :



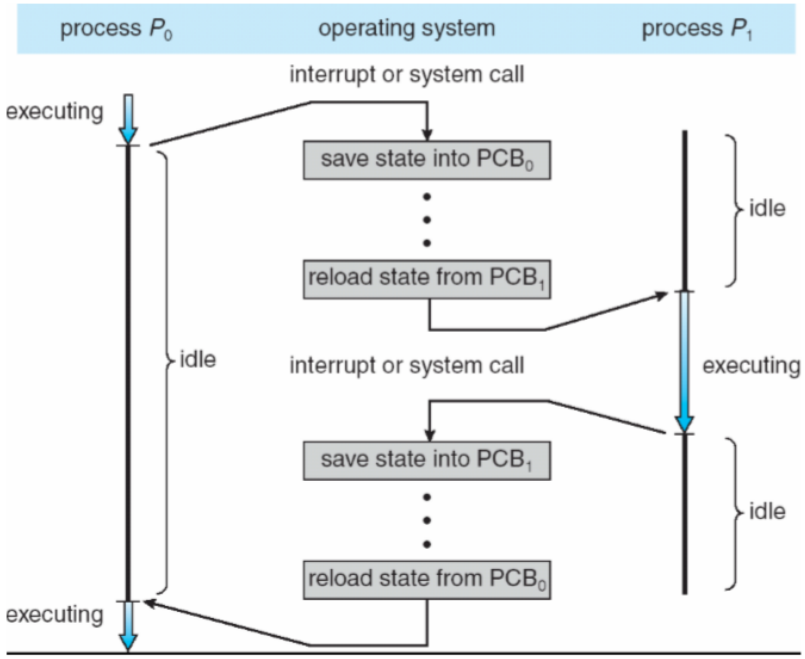
Execute process :

- As a process executes, it changes **state**
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution

5(c) How to CPU switch from process to process ?

Answer :

CPU switch from process to process :



Q-6 :

6(a). Define CPU scheduler and job scheduler . Write a diagram for process scheduling .

Answer :

- **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds) ⇒ (must be fast)
- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes) ⇒ (may be slow)
  - The long-term scheduler controls the **degree of multiprogramming**

6(b). What are operation on processes ? Write the resource sharing options and execution option and write diagrams of tree process in linux .

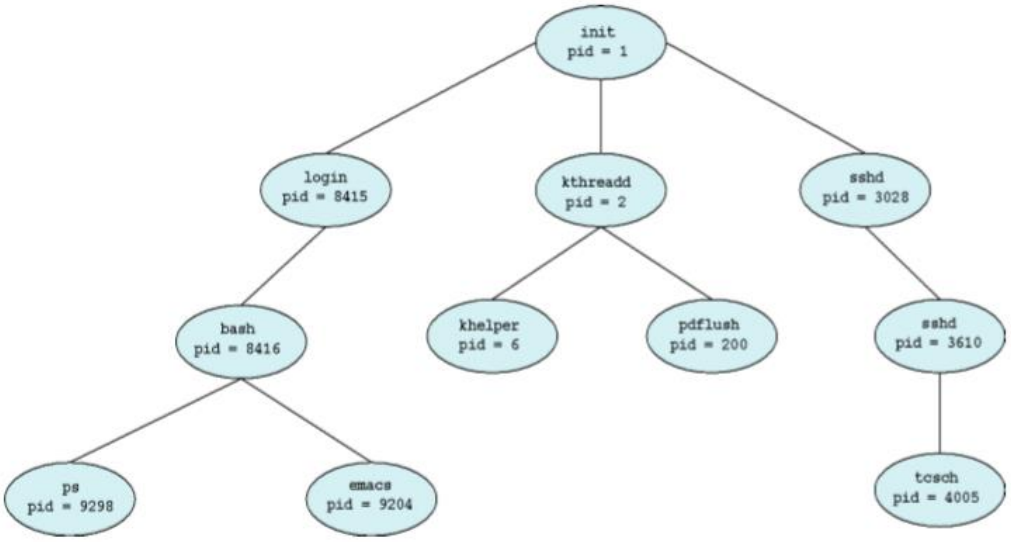
Answer :

Operation on processes :

- System must provide mechanisms for:
  - process creation,
  - process termination,
  - and so on as detailed next

- Resource sharing options
  - Parent and children share all resources
  - Children share subset of parent's resources
  - Parent and child share no resources
- Execution options
  - Parent and children execute concurrently
  - Parent waits until children terminate

Tree process in linux



6(c). Write a browser which is multiprocess and what are different type of process .

Answer :

- Google Chrome Browser is multiprocess with 3 different types of processes:
  - **Browser** process manages user interface, disk and network I/O
  - **Renderer** process renders web pages, deals with HTML, Javascript. A new renderer created for each website opened
    - Runs in **sandbox** restricting disk and network I/O, minimizing effect of security exploits
  - **Plug-in** process for each type of plug-in

Processes in the operating system can be in any of the following states:

- NEW- The process is being created.
- READY- The process is waiting to be assigned to a processor.
- 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).
- TERMINATED- The process has finished execution.

Q-7 :

7(a). What is parallelism and concurrency ? Describe the types of parallelism .

Answer :

- ***Parallelism*** implies a system can perform more than one task simultaneously
- ***Concurrency*** supports more than one task making progress
  - Single processor / core, scheduler providing concurrency

- Types of parallelism
  - **Data parallelism** – distributes subsets of the same data across multiple cores, same operation on each
  - **Task parallelism** – distributing threads across cores, each thread performing unique operation

7(b). Define Amdahls law . Describe user theads and kernel threads briefly .

Answer :

Amdahls law :

- Identifies performance gains from adding additional cores to an application that has both serial and parallel components
- *S* is serial portion
- *N* processing cores

$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

- That is, if application is 75% parallel / 25% serial, moving from 1 to 2 cores results in speedup of 1.6 times
- As *N* approaches infinity, speedup approaches 1 / *S*

Serial portion of an application has disproportionate effect on performance gained by adding additional cores

- **User threads** - management done by user-level threads library
- Three primary thread libraries:
  - POSIX **Pthreads**
  - Windows threads
  - Java threads
- **Kernel threads** - Supported by the Kernel
- Examples – virtually all general purpose operating systems, including:
  - Windows
  - Solaris
  - Linux
  - Tru64 UNIX
  - Mac OS X

7(c). Describe briefly types of multithreading models .

Answer :

Multithreading models are three types . There are –

1. Many-to-One .

2. One-to-One
3. Many-to-Many

Many-to-One :

- Many user- level threads mapped to single kernel thread .
- One thread blocking czuses all to block .
- Multiple threads may not run in parallel on muticore system because only one may be in kernel time.
- Few system currently use this model.

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.
- Number of threads per process sometimes restricted due to overhead.

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.

- Solaris prior to version 9.
- Windows with the ThreadFiber pakage.

Q-8 :

8(a). Define preemptive and non-preemptive scheduling . Why do we need to consider multilevel queue scheduling ?

Answer :

Preemptive :

When a process switches from the running state to the ready state or from waiting state to ready state is called preemptive scheduling .

Non-preemptive :

When a process terminate or a process switches from running to waiting state is called non-preemptive scheduling.

It may happens that proess in the ready queue can be divided into different classes where each class has its own scheduling needs . For example common division of a foreground (interactive) process and background (btach) processes. These two classes have different

scheduling needs. For this kind of situation multilevel queue scheduling is used.

8(b). What do u mean synchronization ? Describe the types of synchronization .

Answer :

Synchronization is the precise coordination of multiple events or mechanical devices. In computing, it refers to the coordination of hardware devices, such that the data they contain or provide is made to be identical. The **synchronization** is usually done within an acceptably brief period of time.

Four types of Synchronization . There are –

1. Solaris
2. Windows
3. Linux
4. Pthreads

Solaris Synchronization :

- Implements a variety of locks to support multitasking ,multithreading and multiprocessing.
- Uses adaptive mutexes for effciency when protecting data from short code segments.
- Uses condition variables.
- Uses readers-writers locks when longer sections of code need access to data.

Windows Synchronization :

- Uses interrupt masks to protect access to global resources on uniprocessor systems.
- Uses spinlocks on multiprocessor system.
- Also provides dispatcher objects user-land which may act mutexes , semaphores, evens and timers.

Lunux Synchronization :

- Prior to kernel version 2.6 disables interrupts to implement short critical sections.
- Version 2.6 and later fully preemptive.

Pthreads Syncronization :

- Pthreads API is OS-independent
- Mutex locks
- Condition variable
- Read -write locks
- Spin locks

8(c). Describe the difference between preemptive and non-preemptive scheduling.

Answer :

Preemptive Scheduling	Non-Preemptive Scheduling
Processor can be preempted to execute a different process in the middle of execution of any current process.	Once Processor starts to execute a process it must finish it before executing the other. It cannot be paused in middle.
CPU utilization is more compared to Non-Preemptive Scheduling.	CPU utilization is less compared to Preemptive Scheduling.
Waiting time and Response time is less.	Waiting time and Response time is more.
The preemptive scheduling is prioritized. The highest priority process should always be the process that is currently utilized.	When a process enters the state of running, the state of that process is not deleted from the scheduler until it finishes its service time.
If a high priority process frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then another process with less CPU burst time may starve.
Preemptive scheduling is flexible.	Non-preemptive scheduling is rigid.
Ex:- SRTF, Priority, Round Robin, etc.	Ex:- FCFS, SJF, Priority, etc.