What is the operating System ?

program that acts as intermediary between the user of a computer and the computer hardware

- **OS is resource allocator** : Manages all resources and Decides between conflicting requests for efficient and fair resource use
- **OS is control program :** Controls execution of programs to prevent errors and incorrect use of the computer

What is the operating system goals ?

- Execute user programs
- Make the computer system suitable to use
- Make solving user problems easier

What is the computer system component ?

- **Hardware** : provides basic computing resources , CPU , memory , I/O
- **Operating system**
- **Application program** : define the ways in which the system resources are used to solve the computing problems , word , excel
- **System program** : compiler , database system , assembler
- **Users**

What is the different between program and process ?

It is the program image that is in RAM to be execution . process It is a unit of work within the system. Program is a passive entity, process is an active entity.

What is the kernel ?

The one program running at all times on the computer

What is CPU component ?

- Register
- ALU
- Control unit

What does the RAM content ?

Instruction or data

What is the bootstrap program ?

- It's the program the loaded typically in ROM and it known as firmware

- It Initializes all aspects of system
- Loads operating system kernel and starts execution

Explain the computer system operation ?

- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Controller is response fro move the date from/to devices
- Device controller informs CPU that it has finished its operation by causing an **interrupt**

Explain the interrupt ?

- Signals the CPU to a temporary suspension of the current execution
- Interrupt transfers control to the interrupt service routine ISR, through the interrupt vector, which contains the addresses of all the service routines
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- Operation system is **interrupt driven**
- Interrupt driven by hardware .

What is the different between interrupt and trap with example ?

Trap : software-generated interrupt caused either by an error or a user request **example** : division by zero , invalid memory access

Interrupt : some time hardware-generated **example:** problem in hard disk or Power outage

What is the types of interrupt handling ?

- **polling**
- **vectored** interrupt system

What is the interrupt types ?

- **User interrupt** : user cancel the program before it finished
- **Software interrupt** : deviation by zero

- ❖ **Hardware interrupt** : power outage
- ❖ **I/O interrupt** : Device controller informs CPU that it has finished
- ❖ **Timer interrupt** : round robin algorithm

How can the operation system save the CPU state ?

storing registers and the program counter

What is the methods of I/O structure ?

- ❖ **After I/O starts, control returns to user program only upon I/O completion**
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request is outstanding at a time

- ❖ **After I/O starts, control returns to user program without waiting for I/O completion**

- **System call** : request to the operating system to allow user to wait for I/O completion
- **Device-status table :** contains entry for each I/O device indicating its type, address, and state
- Operating system access into I/O device table to determine device status and to modify table entry

What is the direct memory access ?

- ❖ Used to allow high-speed I/O devices to transmit information at close to memory speeds
- ❖ Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- ❖ Only one interrupt is generated per block, rather than the one interrupt per byte

What is the storage structure in computer system ?

- ❖ **Main memory (RAM)** : array and volatile and only large storage media that the CPU can access directly
- ❖ **Secondary storage** : extension of main memory that provides large nonvolatile storage capacity
- ❖ **Magnetic disks** : rigid metal or glass platters covered with magnetic recording material

What does disk surface contain ?

Disk surface is logically divided into **tracks**, which are subdivided into **sectors**

What is the disk controller ?

It determines the logical interaction between the device and the computer

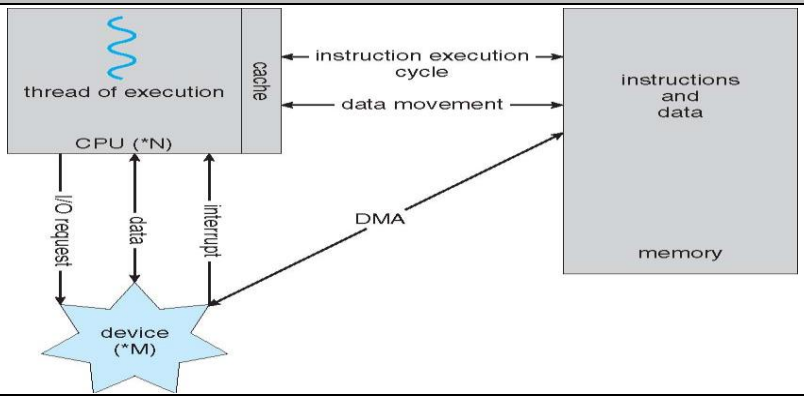
According to what the storages organized ?

- Speed
- Cost
- Volatility
- Capacity

What is the caching ?

- **copying information(not process) into faster storage system that means Information in use copied from slower to faster storage temporarily :**
  - cache memory is cache for main memory
  - main memory is *cache* for secondary storage
- **Faster storage (cache) checked first to determine if information is there**
  - If it is, information used directly from the cache (fastest)**(cache hit)**
  - If not, data copied to cache and used there (cache miss)

Summarize in figure how the computer work ?



Who manage the transition between storage device ?

- Hard ware is response to the transition between cache, RAM and register
- OS is response to transition between RAM and hard disk

Sort the storage device from small to large ?

- Register
- Cache
- RAM

- ❖ Electronic disk
- ❖ Magnetic disk
- ❖ Optical disk
- ❖ Magnetic tapes

What is the current computer system architecture ?

- ❖ Most systems use a single general-purpose processor
- ❖ Most systems have special-purpose processors like multiprocessors, multi-cores and Clustered .

What is the advantages of multiprocessors (parallel, coupled) computer ?

- ❖ **Increased throughput** : increase the rate of process finished on unit time because of each processors work in one process .
- ❖ **Economy of scale** : because of shared recourses such as I/O and RAM .
- ❖ **Increased reliability** : continue to run even if one or more processor get failure .

What is the types of multiprocessing ?

Asymmetric multiprocessing : on of the processors is master and it monitor the other processors , and if one is failure it do its task .
Symmetric multiprocessing : all the processors are the same .

What is the ready queue ?

To run the process, this process should be in ready queue , this main this process hold all of its resources .

What is the different between multiprocessor and multi-cores architecture ?

Multiprocessor : every processor in different ship .
Multi-cores : all core are in the same ship .

What is the advantages of multi-core architecture on multiprocessor ?

- ❖ Faster because every core in the same ship .
- ❖ Use less power .

What is the clustered system .

- ❖ Like multiprocessor systems, but multiple systems working together .
- ❖ Usually sharing storage via a storage-area network (SAN)
- ❖ Provides a high-availability service which survives failures

What is the types of clustered ?

- **Asymmetric clustering**: has one machine in hot-standby mode
- **Symmetric clustering** : has multiple nodes running applications, monitoring each other

What is the CPU utilization ?

Keep CPU busy as possible , and it apposite of idle .

What is the multiprogramming ?

- It used for efficiency .
- Because single user cannot keep CPU busy all time .
- **Multiprogramming** organize the process so CPU always has one to execute .

What is job scheduling ?

- Because subset of total jobs in the system is kept in the memory .
- **Job scheduling** is part of OS and it used to select the process that will be loaded to memory from job pool .

What is the time sharing (multitasking) ?

It is logical extension of multiprogramming in which CPU switch jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

What is the response time ?

It's the time required to get the first response o the process (not outing)

What is the requirements of interactive computing ?

- Response time should be < 1 second .
- Each user has at least one program executing in memory **process**

What is CPU scheduling ?

- It is part of OS and it used to select a process from ready queue of the RAM to be execute .
- It used if several jobs ready to run at the same time .

What is the swapping ?

- It used if the process don’t fit in the memory
- It move the data from memory to VM in HD (swap out)

- It move the date from VM in HD to memory (swap in)

What is virtual memory ?

It is in HD and it allows execution of processes not completely in memory . if the process don't fit in the memory

What is the dual-mode operation ?

- It allow OS to protect itself and other system component .
- It is divide the mode to : user mode – kernel mode .

What mode bit ?

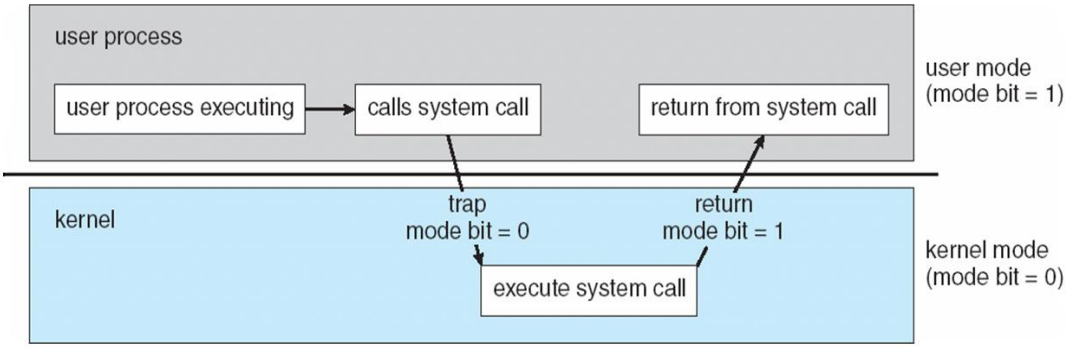
- mode bit provided by hardware .
  - Provides ability to distinguish when system is running user code or kernel code
  - Some instructions designated as **privileged**, only executable in kernel mode . (change the time)

Explain the transition between the processes ?

- Timer to prevent infinite loop
- Set interrupt after specific period
- Operating system decrements counter
- When counter zero generate an interrupt
- Set up before scheduling process to regain control or terminate program that exceeds allotted time

Explain the transition between user mode and kernel mode ?

System call changes mode to kernel, return from call resets it to user



Explain the process management ?

- Process need recourses to accomplish its task (CPU,RAM,I/O, files, data).
- Process termination requires reclaim of any reusable resources .
- Single threaded process has one counter , so the process executes

instructions sequentially, one at a time, until completion .

- Multi-threaded process has one program counter per thread

What does OS do to manage the process ?

- Creating and deleting both user and system processes : **load programs**
- Suspending and resuming processes : **interrupt**
- Providing mechanisms for process synchronization : **time sharing**
- Providing mechanisms for process communication : **data sharing**
- Providing mechanisms for deadlock handling : **cycle needing**

Explain memory management ?

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory
- Optimizing CPU utilization **(largest possible)** and computer response **(smallest possible)** to users

What does OS do to manage the memory ?

- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes (or parts thereof) and data to move into and out of memory
- Allocating and deallocating memory space as needed

Explain storage management ?

  - Abstracts physical properties of information storage to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
- **File-System management**
  - Files usually organized into directories **(folder)**
  - Access control on most systems to determine who can access what

What does OS do to manage the storage ?

- Creating and deleting files and directories **(folder)**
- Primitives to manipulate files and directories **(folder)**
- Mapping files onto secondary storage
- Backup files onto stable (non-volatile) storage media

Explain mass storage management ?

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Entire speed of computer operation hinges on disk subsystem and its algorithms

What does OS do to manage the mass storage ?

- Free-space management
- Storage allocation
- Disk scheduling

Explain performance of different storage ?

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

What is cache coherency ?

Is used to ensure that all caches of all processes has the most recent value .

What is the I/O subsystem Responsibility ?

- Memory management of I/O including
  - ✓ **buffering** (storing data temporarily while it is being transferred),
  - ✓ **caching** (storing parts of data in faster storage for performance),
  - ✓ **spooling** (the overlapping of output of one job with input of other jobs)
- General device-driver interface
- Drivers for specific hardware devices

What is the different between protection and security ?

Protection : any mechanism for controlling access of processes or users to resources defined by the OS

Security : defense of the system against internal and external attacks

What is the services that the OS provides to help the user ?

- **User interface** : Command-Line (CLI), Graphics User Interface (GUI), Batch
- **Program execution** - load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
- **I/O operations** : A running program may require I/O, which may involve a file or an I/O device
- **File-system manipulation** : programs need to read and write files and directories, create and delete them, search them .
- **Communications** : Processes may exchange information, on the same computer or between computers over a network
- **Error detection** : OS needs to be constantly aware of possible errors and correction .

what is the OS services that for ensuring the efficient operation of the system itself via resource sharing ?

- **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
- **Accounting -** To keep track of which users use how much and what kinds of computer resources
- **Protection and security :** 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 .

Explain CLI ?

- Command Line Interface (CLI) or command interpreter allows direct command entry
  - Sometimes implemented in kernel, sometimes by systems program
  - If the latter, adding new features doesn't require shell modification

Explain GUI ?

- User-friendly desktop metaphor interface
- Usually mouse, keyboard, and monitor , Icons represent files, programs
- Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder)

What is system call ?

Programming interface to the services provided by OS .

How can system call used ?

Mostly access by programs via high-level application program interface(API) rather than direct system call use .

What is API ?

Application program interface used to specify the parameter and specification of system call .

List three example of API ?

Win32 API for windows .
POSIX API for POSIX-based system like UNIX, Linux and Mac
Java API for java virtual machine .

Explain system call implementation ?

Number associated with each system call , and system call interface(API) maintain a table indexed according to these number .

How system call used ?

System call interface (API) invokes intended system call in OS kernel and returns status of the system call and any return values .

What is the benefits of API and not in direct system call use ?

- The caller need know any thing about how the system call are implemented
- Just need to obey API and understand what OS will do as result call
- So the benefit is API hides most details of OS interface from programmer
- Other benefit is you can run any program on any machine that use the same API platform .

What is ways of passing system call parameters ?

- Simplest: pass the parameters in registers
  - In some cases, may be more parameters than registers
- Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register
- Parameters placed, or pushed, onto the stack by the program and

popped off the stack by the operating system

- Block and stack methods do not limit the number or length of parameters being passed

What is the types of system call ?

- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection

What is the system programs ?

- System programs provide a convenient environment for program development and execution
- Some of them are simply user interfaces to system calls; others are considerably more complex
- Most users' view of the operation system is defined by system programs, not the actual system calls

What is type of system programs

- **File manipulation** : Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories

- **Status information** : Some ask the system for info - date, time, amount of available memory, disk space, number of users, Others provide detailed performance, logging, and debugging information

- **File modification** : Text editors to create and modify files, Special commands to search contents of files or perform transformations of the text .

- **Programming language support** : Compilers, assemblers, debuggers and interpreters sometimes provided

- **Program loading and execution** : Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level

and machine language

- **Communications** : Provide the mechanism for creating virtual connections among processes, users, and computer systems, Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

- **Application programs**

How the operating system designed and implemented ?

- Design and Implementation of OS not "solvable", but some approaches have proven successful
- Internal structure of different Operating Systems can vary widely
- Start by defining goals and specifications
- Affected by choice of hardware, type of system

What is types of operating 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

What is the impotent principles that has be separated in operating systems?

- **Policy** : what will be done ?
- **Mechanism** : how to do it ?

Give example of separation between policy and mechanism in OS ?

In timer construction :

- **Policy** : the loop don't spend more than 5 msec .
- **Mechanism** : set timer and OS decrement the counter until it be zero then interrupt will be generated .

What is the benefit of separation between policy and mechanism ?

It allows maximum flexibility if policy decisions are to be changed later .

What is the main structures of operating systems ?

- Simple
- Layered

- Microkernel
- Modules

What is the simple structure of operating systems structures ?

- It is MS-DOS (Microsoft disk operating system)
- It is written to provide the most functionality in the available hardware.
- It isn't divided to module , so the interface and levels of operation are not separated .
- There isn’t has dual mode .

What is the layered approach ?

- Divide the operating system into number of layered .
- Each layer built on top of lower layers .
- The button layer (layer0) is the hardware .
- The highest layer is the user interface .
- With modularity layers are selected such that each uses functions (operations) and services of only lower-level layers
- **the main disadvantage:**
  - ✓ how to define the layer
  - ✓ Less efficient than other structure

What is the structure of UNIX operating system ?

- UNIX limited hardware functionality .
- The UNIX OS consists of two separated part : **System programs** and **kernel**

What is The kernel ?

It the program that running in every time on the computer , and Consists of everything below the system-call interface and above the physical hardware .

What is microkernel structure ?

- It means move as much from kernel to user space. (Win2000)
- The communication takes place between user modules using **message passing** not use shared memory.
- **Advantages** :
  - Easier to extend microkernel because its size is small.
  - Easier to port the OS to new architecture

- More reliable **because** of less code is running in kernel mode
- More secure

- **Disadvantages** :
  - Performance overhead of user space to kernel space communication
  - What part of kernel will be move.

That is the module structure ?

- Most modern operating systems implement kernel modules
- similar to layers but with more flexible and efficient.
- **Characteristic**
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel

What is the virtual machine ?

The operating system host creates the illusion that a process has its own processor and (virtual memory)

What is the debugging

finding and fixing errors, or bugs

What is types of error and where it will be store ?

- **Log files**: contains information of error caused by OS
- **Core dump file**: contain information of error caused by application
- **Crash dumb** : contain information of error caused by kernel

What does Kernighan's Law say ?

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it".

What does DTrace tool in solaris do ?

Help in debugging. by allowing live instrumentation on production systems

What is booting ?

Starting a computer by loading the kernel .

What does process include ?

- Program counter
- Stack
- Data section

List the process states ?

- **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

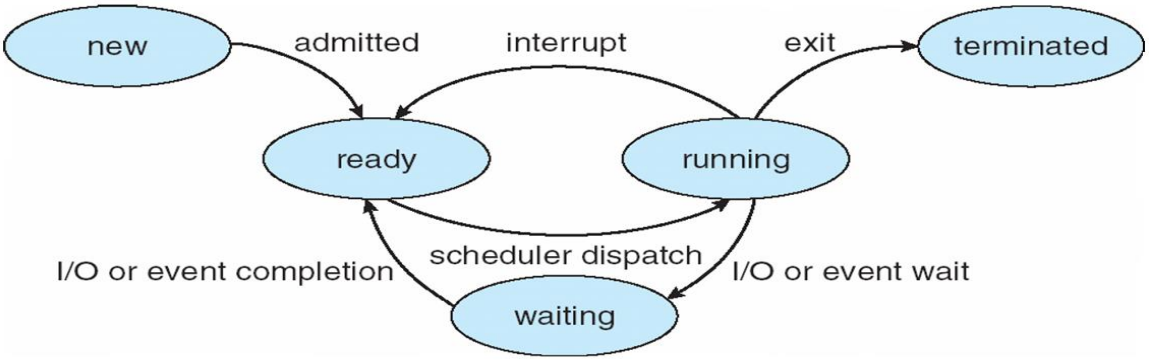
What is the information that associated with each process PCB ?

- **It allow OS to control the process**
- **Process state** : new, wait, ready, terminated, run
- **Program counter** : when process stop store the instruction address
- **CPU registers** : store the process date when process stop
- **CPU scheduling information** : priority
- **Memory-management information** : limit register (how much) – base register(from what)
- **Accounting information** : which user use this process
- **I/O status information** : which data the process hold

What is the process data in the memory ?

- **Stack** : Contains the temporary data such as function program and local variables .
- **Heap** : Is memory that is dynamically allocated during the run and it efficient related
- **Data** : such as global variable
- **Text** : code

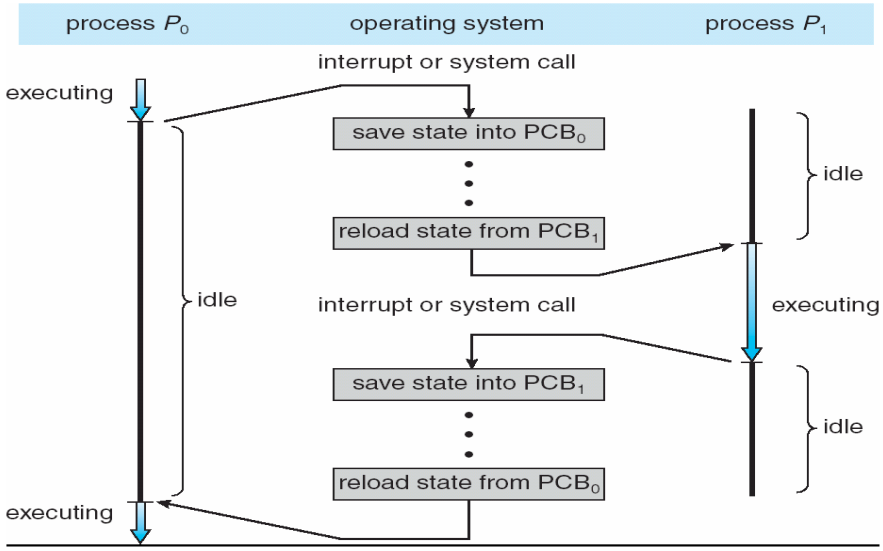
Explain the process life cycle ?



What is dispatch responsibility ?

- Save process state into PCB
- Load process state from PCB

Explain the switching between the processes ?



List the process scheduling queues ?

- **Job queue** : set of all processes in the system
- **Ready queue** : set of all processes residing in main memory, ready and waiting to execute
- **Device queues** : set of processes waiting for an I/O device

Processes migrate among the various queues

What is types of scheduler ?

- **Long-term scheduler** (or job scheduler) : selects which processes should be brought into the ready queue
- is invoked very infrequently (seconds, minutes) ⇒ (may be slow)
- It controls the *degree of multiprogramming*

- **Short-term scheduler** (or CPU scheduler) : selects which process should be executed next and allocates CPU
- is invoked very frequently (milliseconds) ⇒ (must be fast)

What is the types of process ?

- **I/O-bound process** : spends more time doing I/O than computations, many short CPU bursts
- **CPU-bound process** : spends more time doing computations; few very long CPU bursts

What is the context twitch ?

When CPU switches to another process, the system must save the state of the old(stopped) process and load the saved state for the new process via a **context switch**

What is the context of process ?

❖ **Context of a process represented in the PCB**

- Process state
- SPU register
- Memory management info

Is the context switching overhead ? why ?

Yes, because the system does no useful work while switching. And Time dependent on hardware support

Explain the process tree creation ?

Parent process create **children** processes, which, in turn create other n processes, forming a tree of processes

Via what the process identified and managed ?

via a process identifier (**pid**)

What is the types of Resource sharing between parent and children ?

- Parent and children share all resources
- Children share subset of parent's resources
- Parent and child share no resources

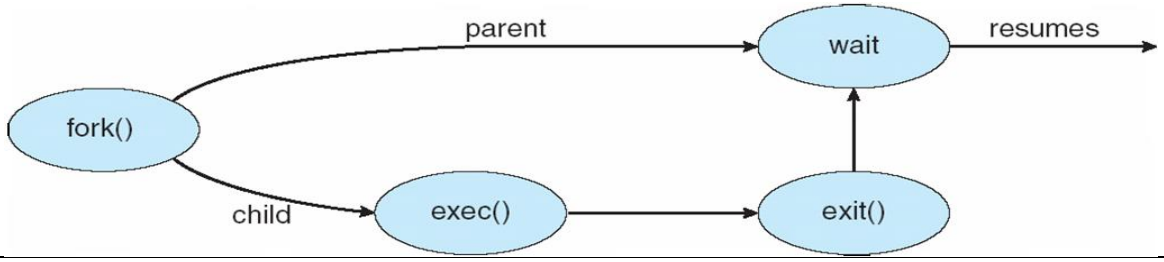
What is the types of Resource sharing between parent and children ?

- Parent and children execute concurrently (**communication**)
- Parent waits until children terminate

Give the UNIX example create the process ?

- **fork** system call creates new process
- **exec** system call used after a **fork** to replace the process' memory space with a new program

Explain the process creation in diagram ?



explain the process termination ?

- Process executes last instruction and asks the operating system to delete it (**exit**)
- Output data from child to parent (via **wait**)
- Process' resources are deallocated by operating system

why Parent terminate children processes by (**abort**) ?

- Child has exceeded allocated resources
- Task assigned to child is no longer required

What happen for children processes when the parent exiting ?

- Some operating system do not allow child to continue if its parent terminates , all children terminated - **cascading termination**
- Some operating system allow child to continue if its parent terminates

What is the reasons for cooperating processes ?

- **Information sharing**
- **Computation speedup** : in case of multiprocessor
- **Modularity** : dividing the system functionality into separate process or threads
- **Convenience** : user can work on many program concurrently

What is the IPC and what is the models of it ?

- Cooperating processes need **interprocess communication** (**IPC**)
- Two models of IPC
  - Shared memory
  - Message passing

What is the types of process in term of sharing ?

- **Independent** process cannot affect or be affected by the execution of another process
- **Cooperating** process can affect or be affected by the execution of another process

Explain cooperating between the process ?

Paradigm for cooperating processes, producer process produces information that is consumed by a consumer process

What is the types of buffer ?

- ***unbounded-buffer*** : places no practical limit on the size of the buffer
- ***bounded-buffer*** : assumes that there is a fixed buffer size

What is IPC ?

t is mechanism for processes to communicate and to synchronize their actions.

What is message passing?

Processes communicate with each other without resorting to shared variables.

What is the operations provided by IPC ?

- **send**(*message*) – message size fixed or variable
- **receive**(*message*)

How the process communicate in message passing ?

- establish a *communication link* between them
- exchange messages via send/receive

What is the types of communications in message passing ?

- **Direct Communication** : Processes must name each other explicitly
- **Indirect Communication** : Messages are directed and received from

mailboxes

Explain the direct commutation operations ?

- **Send (P, message)** : send a message to process P
- **Receive(Q, message)** : receive a message from process Q

What is the properties of communication link in direct communication ?

- Links are established automatically
- A link is associated with exactly one pair of communicating processes
- Between each pair there exists exactly one link
- The link may be unidirectional, but is usually bi-directional

Explain the indirect communication ?

- **Messages are directed and received from mailboxes (also referred to as ports)**
- Each mailbox has a unique id
- Processes can communicate only if they share a mailbox

What is the properties of communication link in indirect communication ?

- Link established only if processes share a common mailbox
- A link may be associated with many processes
- Each pair of processes may share several communication links
- Link may be unidirectional or bi-directional

What is the operations in indirect communication ?

- create a new mailbox
- send and receive messages through mailbox
- destroy a mailbox

- **send**(*A, message*) – send a message to mailbox A
- **receive**(*A, message*) – receive a message from mailbox A

where A = mailbox ID

Give three solutions for the following ?

- P_1, P_2, and P_3 share mailbox A
- P_1, sends; P_2 and P_3 receive
- Who gets the message?

- Allow a link to be associated with at most two processes
- Allow only one process at a time to execute a receive operation
- Allow the system to select arbitrarily the receiver. Sender is notified who the receiver was.

What is the benefits of Multithreaded Processes ?

- **Responsiveness** : If one thread stop the other threads will continue
- **Resource Sharing** : Shearing the RAM, code, file and data
- **Economy** : because of shearing the resources, the cost of creation process is about 30 times more than cost of create thread
- **Scalability** : in case of multicores or multiprocessor , it means tow threads of the same process can execute in the same time (**parall**)

What is challenges of multicores system ?

- Dividing activities
- Balance
- Data splitting
- Data dependency
- Testing and debugging

Explain how the multithreaded is suitable to client server architecture ?

With multithreaded if the client send request to the server to display web page content (that may be pictures, form, text, other), each item will be assigned to a thread, so if a thread of picture is stop the other parts of the web page will be displayed , but without multithreaded all web page content will not displayed .

What is the types of threads ?

- **User threads** : threads management dine by user level threads library with API **Like** (POSIX threads, Win32 threads, Java threads)
- **Kernel threads** : supported by the kernel **Like** (WindowsXP/2000, Solaris, Linux, Mac OS X)
- **The threads manage by thread libraries**

What is the model of multithreading ?

- **Many-to-one** : Many user-level threads mapped to single kernel thread
  Adv: no overhead in kernel
  Dis: if kernel thread stop the other thread will stop

- **One-to-One** : Each user-level thread maps to kernel thread
  Adv: if kernel thread stop the other thread will continue
  Dis: no overhead in kernel

- **Many-to-Many** : Allows many user level threads to be mapped to many kernel threads, and Allows the operating system to create a sufficient number of kernel threads
  Adv: no overhead in kernel , and if kernel thread stop the other thread will continue

- **Tow Level** : Similar to many-to-many, except that it allows a user thread to be **bound** to kernel thread
  Adv: allows a user thread to be **bound** to kernel thread

What is the types of threads library implementation ?

- **Exists in user space** : code and data structure for managing the thread in user space **example:** Java library threads
- **Exists in kernel space** : code and data structure for managing the thread in kernel space **example:** Win32 thread library

P thread library is example for both types (MAC-Linux)

What of Java thread created ?

- Extending Thread class
- Implementing the Runnable interface

Does **fork()** duplicate only the calling thread or all threads?

1- If **exe()** come after **fork()** immediately duplicate just the caller thread
2- If not will duplicate all threads

What is thread cancelation?

Terminating a thread before it has finished

What is general approaches to cancel thread ?

- **Asynchronous cancellation** terminates the target thread immediately
- **Deferred cancellation** allows the target thread to periodically check if it should be cancelled

What is cancelation point ?

The cancelation occur only after the target thread has checked a flag to determine either or not it should be canceled .
The thread can perform this check at a point at which it can be cancel (cancelation point in P thread).

What is CPU utilization ?

Keep CPU busy as possible

What is CPU-I/O burst cycle ?

Process execution consists of a cycle of CPU execution and I/O wait
That means the CPU either work or wait for I/O

What is CPU scheduler (short-term scheduler) ?

Selects process from the ready queue to be execute

When the CU scheduler will take place (work)?

1. Switches from running to waiting state (I/O interrupt) **nonpreemptive**
2. Switches from running to ready state (timer interrupt) **preemptive**
3. Switches from waiting to ready(higher priority process arrive to ready queue) **preemptive**
4. Terminates **nonpreemptive**

What is the preemptive?

- **nonpreemptive** : once CPU given to the process it cannot be preempted until completes its CPU burst
- **preemptive** : if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF)

What is dispatcher ?

Dispatcher module gives control of the CPU to the process selected by the short-term scheduler

What is the task of dispatcher ?

- switching context
- switching to user mode
- jumping to the proper location in the user program to restart that program

What is the dispatcher latency ?

time it takes for the dispatcher to stop one process and start another running

What is the scheduling criteria ?

- **CPU utilization** : keep the CPU as busy as possible
- **Throughput** : # of processes that complete their execution per time unit
- **Turnaround time** : amount of time to execute a particular process
- **Waiting time** : amount of time a process has been waiting in the ready queue
- **Response time** : amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

What we must keep for scheduling criteria ?

- **Max** CPU utilization
- **Max** throughput
- **Min** turnaround time
- **Min** waiting time
- **Min** response time

What is the types of scheduler algorithms ?

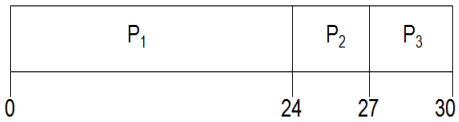
1- First come, first served (FCFS)
2- Shortest Job First (SJF)
3- Priority
4- Round robin (RR)
5- Multilevel queue

Explain FCFS scheduling ?

- **Advantages** : very simple in implementation
- **disadvantages** : when process with high burst time come first the other processes will wait for long time to start execute.

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3 The Gantt Chart for the schedule is:

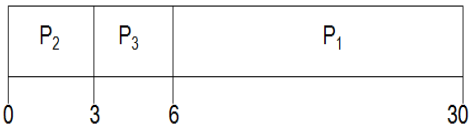


- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Suppose that the processes arrive in the order

P_2 , P_3 , P_1

- The Gantt chart for the schedule is:



- Waiting time for P_1 = 6; P_2 = 0, P_3 = 3
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case
- *Convoy effect* short process behind long process

explain SJR scheduling ?

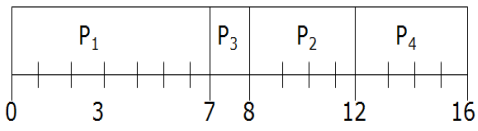
Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time

Advantages : SJF is optimal . gives minimum average waiting time for a given set of processes

Disadvantages : The difficulty is knowing the length of the next CPU request

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

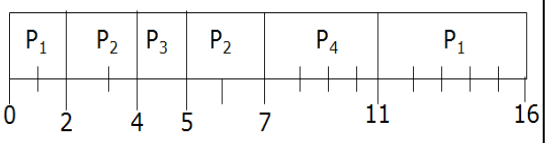
- SJF (non-preemptive)



- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

explain the previous preemptive example ?

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	17
P1	7	6	5	5	5	5	5	5	5	5	5	5	4	3	2	1	**0**
P2	---	---	4	3	2	2	1	**0**									
P3	---	---	---	---	1	**0**											
P4	---	---	---	---	---	4	4	4	3	2	1	**0**					

How to Determining Length of Next CPU Burst ?

1. t_n = actual length of n^{th} CPU burst
2. τ_{n+1} = predicted value for the next CPU burst
3. $\alpha, 0 \le \alpha \le 1$
4. Define : $\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$.

- If we assume that : **t0 = 6** and **T0 = 10**

CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Explain the priority scheduling ?

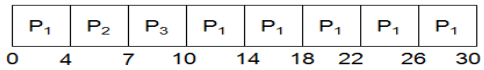
- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest)
- SJF is a priority scheduling where priority is the predicted next CPU burst time
- **Problem** ≡ **Starvation** – low priority processes may never execute
- **Solution** ≡ **Aging** –as time progresses increase the priority of the process (over head)
- **Disadvantage** : not suitable to time sharing

Explain round robin scheduling ?

- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.
- **Advantage** : not suitable to time sharing
- **Disadvantage** : higher average turnaround than SJF put better response
- **Performance**
  - *If q* large ⇒ FIFO
  - *If q* small ⇒ *overhead*

Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:



- Typically, higher average turnaround than SJF, but better *response*

Are there a relation between time quantum and the average of turnaround ?

NO

Explain multilevel scheduling ?

- Ready queue is partitioned into separate queues:
  foreground (interactive)
  background (batch)
- Each queue has its own scheduling algorithm
  - foreground – RR
  - background – FCFS

What is types of scheduling between queues in multilevel scheduling ?

- **Fixed priority scheduling**: (i.e., serve all from foreground then from

background). Possibility of starvation.

- **Time slice** : each queue gets a certain amount of CPU time which it can schedule amongst its processes; i.e., 80% to foreground in RR and 20% to background in FCFS

Explain multilevel feedback queue ?

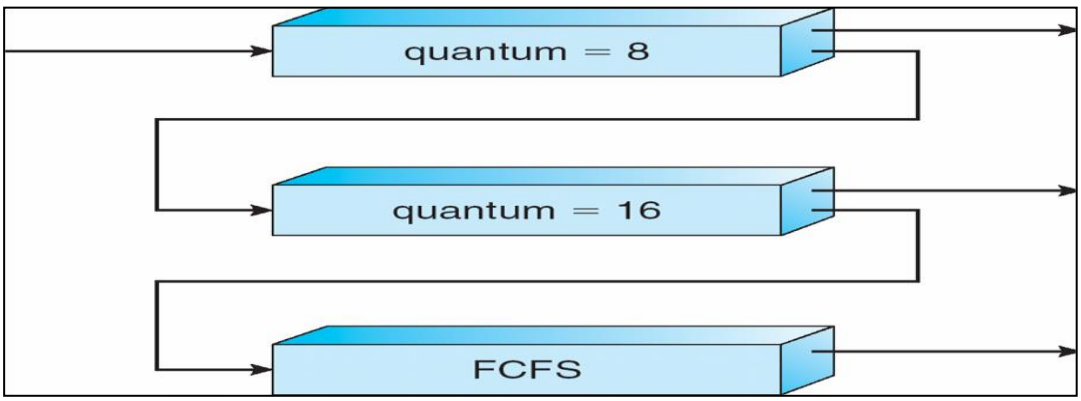
A process can move between the various queues

What is the parameters that Multilevel-feedback-queue scheduler defined by them ?

- number of queues
- scheduling algorithms for each queue
- method used to determine when to upgrade a process
- method used to determine when to demote a process
- method used to determine which queue a process will enter when that process needs service

Give example of multilevel feedback queue ?

- Three queues:
  - Q_0 – RR with time quantum 8 milliseconds
  - Q_1 – RR time quantum 16 milliseconds
  - Q_2 – FCFS
- Scheduling
  - A new job enters queue Q_0 which is served RR. When it gains CPU, job receives 8 milliseconds. If it does not finish in 8 milliseconds, job is moved to queue Q_1.
  - At Q_1 job is again served RR and receives 16 additional milliseconds. If it still does not complete, it is preempted and moved to queue Q_2.



Give an solution to critical section problem ?

- **Mutual Exclusion** : If process P_i is executing in its critical section, then no other processes can be executing in their critical sections **(before entering the critical section make lock in it and after finish working the critical section make unlock)**

- **Progress** : If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely.

- **Bounded Waiting** : A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted

What is the atomic instruction ?

It is the instruction that is, cannot be interrupted.

Explain Peterson's Solution ?

- **The two processes share two variables:**
  - **int turn** : indicates whose turn it is to enter the critical section.
  - **Boolean flag[2]** : array is used to indicate if a process is ready to enter the critical section. flag[i] = true implies that process P_i is ready!

```
do {
        flag[i] = TRUE;
        turn = j;
        while (flag[j] && turn == j);
                critical section
        flag[i] = FALSE;
                remainder section
  } while (TRUE);
```

What is the solution of synchronization of hardware ?

- Uniprocessors – could disable interrupts

- Currently running code would execute without preemption
- Generally too inefficient on multiprocessor systems
- Operating systems using this not broadly scalable

What is the ways of atomic instruction ?

- test memory word and set value
- Or swap contents of two memory words

Explain the solution using lock ?

```
do {
acquire lock
critical section
release lock
remainder section
} while (TRUE);
```

Explain the solution using test and set ?

```
boolean TestAndSet (boolean *target)
        {
            boolean rv = *target;
            *target = TRUE;
            return rv:
        }

    Shared boolean variable lock., initialized to false.
    Solution:
                    do {
                    while ( TestAndSet (&lock ))
                            ;   // do nothing
                            //    critical section
                    lock = FALSE;
                            //      remainder section
            } while (TRUE);
```

Explain the swap solution ?

```
void Swap (boolean *a, boolean *b)
        {
              boolean temp = *a;
              *a = *b;
```

```
        *b = temp:
    }
```

Shared Boolean variable lock initialized to FALSE; Each process has a local Boolean variable key

Solution:

```
do {
        key = TRUE;
                while ( key == TRUE)
                Swap (&lock, &key );

            //    critical section
        lock = FALSE;
                //      remainder section
            } while (TRUE);
```

Explain Bounded-waiting Mutual Exclusion with TestandSet() solution ?

```
do {
    waiting[i] = TRUE;
    key = TRUE;
    while (waiting[i] && key)
        key = TestAndSet(&lock);
    waiting[i] = FALSE;
        // critical section
    j = (i + 1) % n;
    while ((j != i) && !waiting[j])
        j = (j + 1) % n;
    if (j == i)
        lock = FALSE;
    else
        waiting[j] = FALSE;
        // remainder section
  } while (TRUE);
```

Explain semaphore ?

- It is Synchronization tool that does not require busy waiting
- Semaphore *S* – integer variable
- Two standard operations modify S: wait() and signal()
- Less complicated
- Can only be accessed via two indivisible (atomic) operations

```
wait (S) {
        while S <= 0
                ; // no-op
S--;
}
        signal (S) {
S++;
}
```

What is type semaphore ?

- Counting semaphore – integer value can range over an unrestricted domain
- Binary semaphore – integer value can range only between 0 and 1; can be simpler to implement , Also known as mutex locks

How Can implement a counting semaphore S as a binary semaphore ?

```
Semaphore mutex;    //  initialized to 1
do {
wait (mutex);
       // Critical Section
    signal (mutex);
// remainder section
} while (TRUE);
```

How can implement semaphore with out waiting ?

- With each semaphore there is an associated waiting queue. Each entry in a waiting queue has two data items:
  - value (of type integer)
  - pointer to next record in the list

- Two operations:
  - block – place the process invoking the operation on the appropriate waiting queue.

- wakeup – remove one of processes in the waiting queue and place it in the ready queue.

How will be the implement of wait() and signal() of the previous solution ?

- **Implementation of wait:**

```
wait(semaphore *S) {
        S->value--;
        if (S->value < 0) {
                add this process to S->list;
                block();
        }
}
```

- **Implementation of signal:**

```
signal(semaphore *S) {
        S->value++;
        if (S->value <= 0) {
                remove a process P from S->list;
                wakeup(P);
        }
}
```

Gave example of software solution and hardware solution ?

- **Software solution :** semaphore
- **Hardware solution :** test and set - swap

What is the deadlock ?

two or more processes are waiting indefinitely for an event that can be caused by only one of the waiting processes

What is starvation ?

indefinite blocking. A process may never be removed from the semaphore queue in which it is suspended

What is Priority Inversion ?

Scheduling problem when lower-priority process holds a lock needed by higher-priority process

What is the base and limit registers ?

- **Base register** : from what address in the memory the process start
- **Limit register** : the length of the process in the memory , that equals to end address minus start address

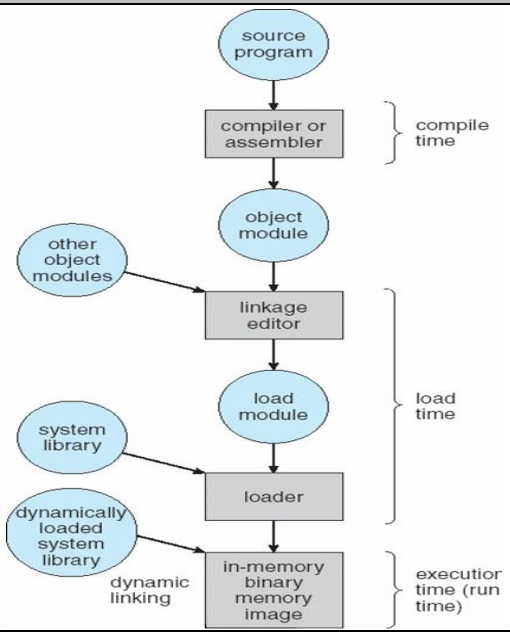
What is the address binding of instructions and data to memory addresses ?

Define the instruction location address

What the at three different stages that the Address binding of instructions and data to memory addresses can happen in ?

- **Compile time:** If memory location known a priori, absolute code can be generated; must recompile code if starting location changes
- **Load time:** Must generate reloadable code if memory location is not known at compile time
- **Execution time:** Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., base and limit registers)

Explain in figure Multistep Processing of a User Program ?



What is the logical address and physical address and different between them ?

- **Logical address :** generated by the CPU; also referred to as **virtual address**
- **Physical address :** address seen by the memory unit

- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme

What is memory management unit MMU ?

- Hardware device that maps virtual to physical address
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with *logical* addresses; it never sees the *real* physical addresses

What is dynamic loaded ?

- Routine is not loaded until it is called
- Better memory-space utilization; unused routine is never loaded
- Useful when large amounts of code are needed to handle infrequently occurring cases (switch in java)

What is dynamic linking ?

- Linking postponed until execution time
- Small piece of code, *stub*, used to locate the appropriate memory-resident library routine
- Stub replaces itself with the address of the routine, and executes the routine
- Operating system needed to check if routine is in processes' memory address
- Dynamic linking is particularly useful for libraries
- System also known as **shared libraries**

What is swapping ?

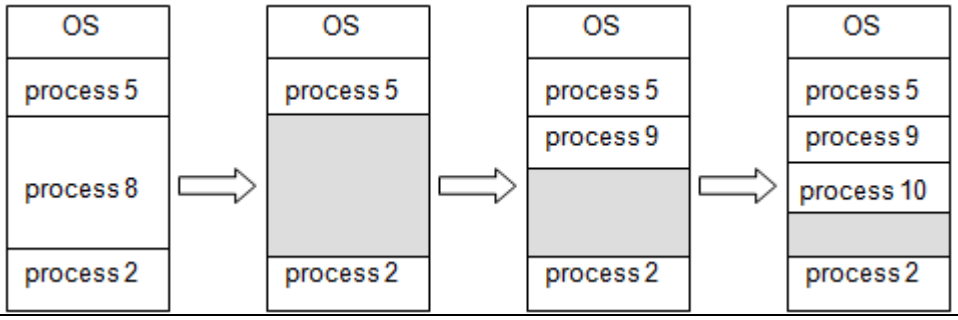
- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution
- **Backing store** : fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
- **Roll out, roll in** : swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped

What the parts of RAM ?

- **Resident operating system** : usually held in low memory or high memory depend on interrupt vector place (together)
- **User processes :** in the other side

Explain the contiguous allocation ?

- **Hole** : block of available memory; holes of various size are scattered throughout memory
- When a process arrives, it is allocated memory from a hole large enough to accommodate it
- Operating system maintains information about:
  a) allocated partitions
  b) free partitions (hole)



What is the types of satisfy a request of size *n* from a list of free holes?

- **First-fit**: Allocate the *first* hole that is big enough **(No Overhead)**
- **Best-fit**: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size **(Overhead)**
  - Produces the smallest leftover hole **(storage utilization)**
- **Worst-fit**: Allocate the *largest* hole; must also search entire list
  - Produces the largest leftover hole **(may also allocated later)**

❖ **First-fit and best-fit better than worst-fit in terms of speed and storage utilization**

What is the types of fragmentation ?

- **External Fragmentation** – total memory space exists to satisfy a request, but it is not contiguous
- **Internal Fragmentation** – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

What is the solution of external fragmentation ?

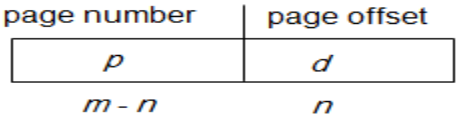
- by **compaction**
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible *only* if relocation is dynamic, and is done at execution time
- **Disadvantages** :
  - Overhead
  - I/O problem

Explain the paging ?

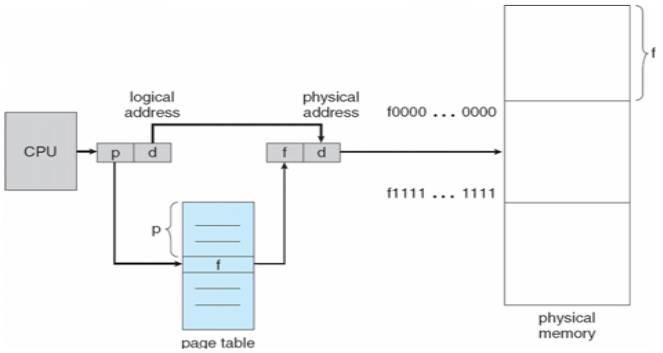
- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8,192 bytes)
- Divide logical memory into blocks of same size called **pages**
- Keep track of all free frames
- To run a program of size ***n*** pages, need to find ***n*** free frames and load program
- Set up a page table to translate logical to physical addresses
- **Disadvantage** : Internal fragmentation

How to translate between Logical to physical in paging ?

- Address generated by CPU is divided into:
  - **Page number (*p*)** : used as an index into a *page table* which contains base address of each page in physical memory
    - **Page offset (d)** : combined with base address to define the physical memory address that is sent to the memory unit



For given logical address space 2^m *and page size* 2^n



How to translate from logical address to physical address ?

Physical address = (frame number * page size) + order

What is segmentation ?

Memory-management scheme that supports user view of memory

Explain segmentation architecture ?

- Logical address consists of a two tuple:
- <segment-number, offset>,
- **Segment table** : maps two-dimensional physical addresses; each table entry has:
- **base** : contains the starting physical address where the segments reside in memory
- **limit** : specifies the length of the segment
- **Segment-table base register (STBR)** points to the segment table's location in memory
- **Segment-table length register (STLR)** indicates number of segments used by a program;
- segment number ***s*** is legal if ***s*** < **STLR**