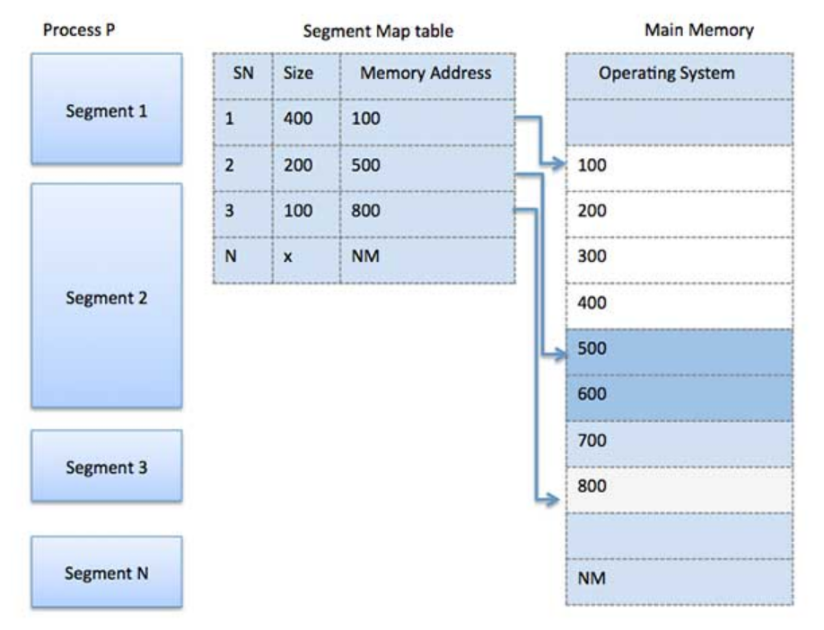
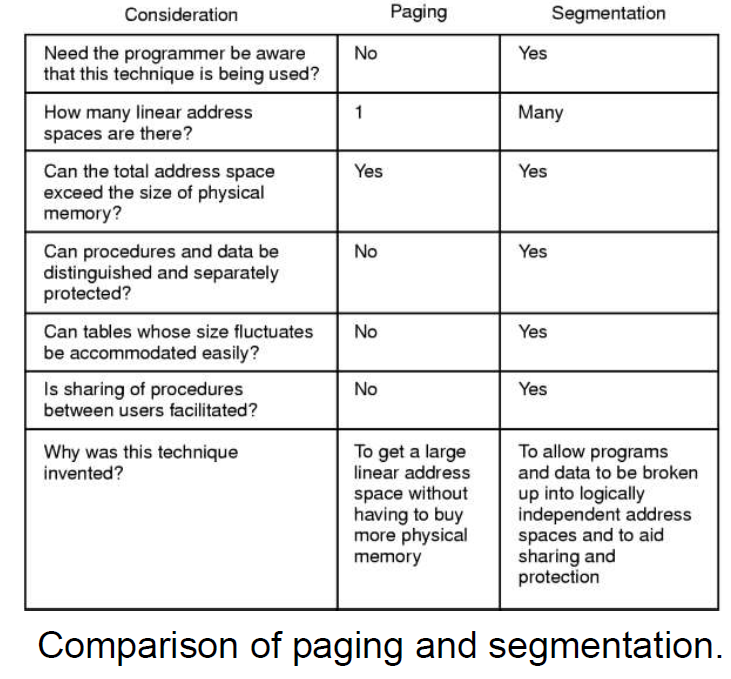
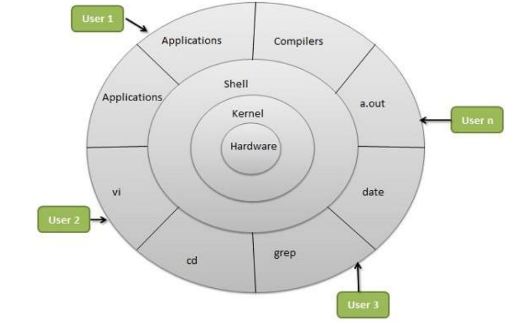
**Segmentation**

* Each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions
* Each segment is a different logical (virtual) address space of the program.
* A segmented memory allows each table to grow or shrink independently of the other tables
  + Whereas in paging, size is fixed
* OS maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory.
* For each segment, the table stores the starting address of the segment and the length of the segment.
* ****A reference to a memory location includes a value that identifies a segment and an offset.



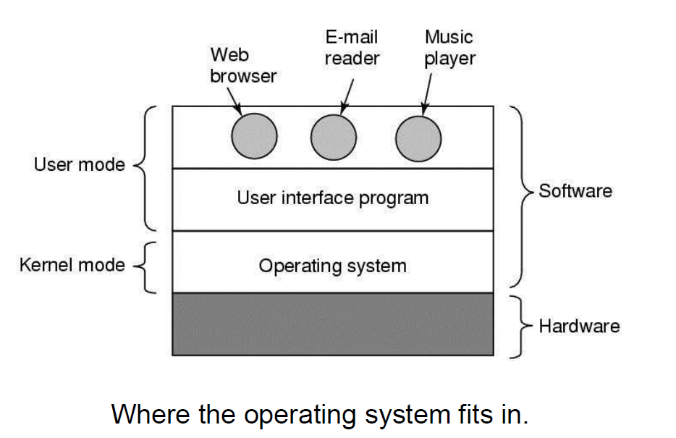
**Kernel Mode vs User Mode**

* Kernel mode – full access to all resources of the computer
  + Disable all interrupts
  + Set the time-of-day clock
  + Create new directory
  + **Interact directly with hardware, provide low level services**
* User mode – support code in System Library
  + no access to system hardware and kernel code
  + user programs/utilities – read the time-of-day clock

**Shell** – LINUX program which can be used to execute commands of the OS. It can be used to do various types of operations, call application programs, etc.

* hide complexity of kernel’s functions from users
* takes commands from the user and executes kernel’s functions

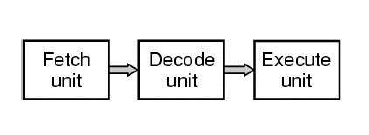
**Operating System**

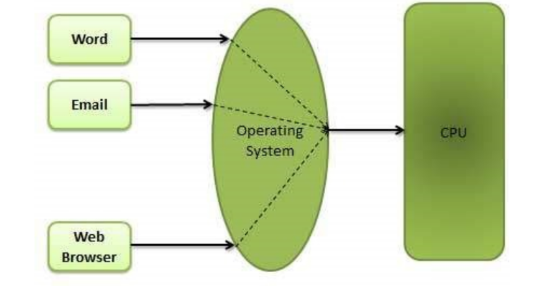
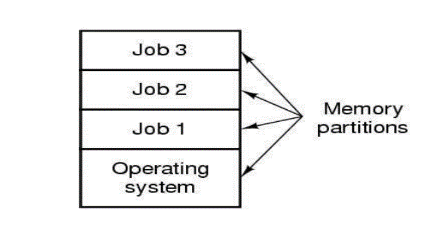
* A software to manage all components of a computer: processors, main memory, disks, printers, input/output devices.
* provides services to both the users and to the programs

in user mode, OS cannot access the other user’s node

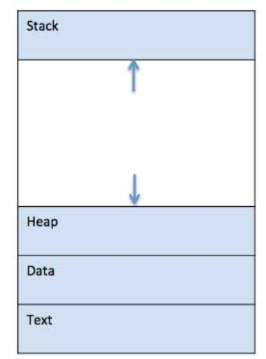
in kernel node, OS access everything but users cannot access

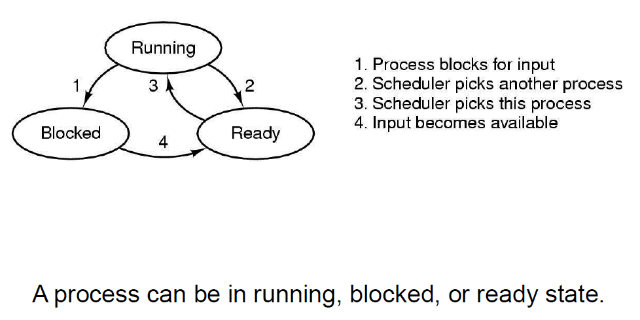
* turns ugly hardware into beautiful abstractions
* Common services: program execution, I/O operations, file system manipulation, communication, error detection, resource allocation, protection
* Major activities: loads a program into memory, executes the program, handles program’s execution, provides a mechanism for process synchronization, process communication, and deadlock handling
* CPU pipelining

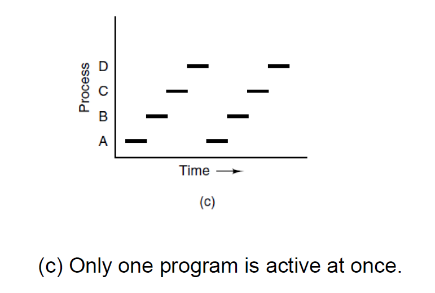


* **Batch processing**
  + A technique in which an OS collects the programs and data together in a batch before processing starts
  + OS defined a job which has predefined sequence of commands, programs and data as a single unit
  + OS keeps a number a jobs in memory and executes them without any manual information
  + Jobs are processed in the order of submission (first come first serve)
  + A new job get executed as soon as the previous job is finished
* **Multitasking**
  + Multiple jobs are executed by the CPU simultaneously by switching between them
  + Uses the concept of CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared CPU
  + A program that is loaded into memory and is executing is commonly referred to as a **PROCESS**
* **Multiprogramming**
  + When 2 or more programs reside in memory at the same time and shared the processor
  + Assumes a single shared processor
  + Increased CPU utilization by organizing jobs so that the CPU always has one to execute

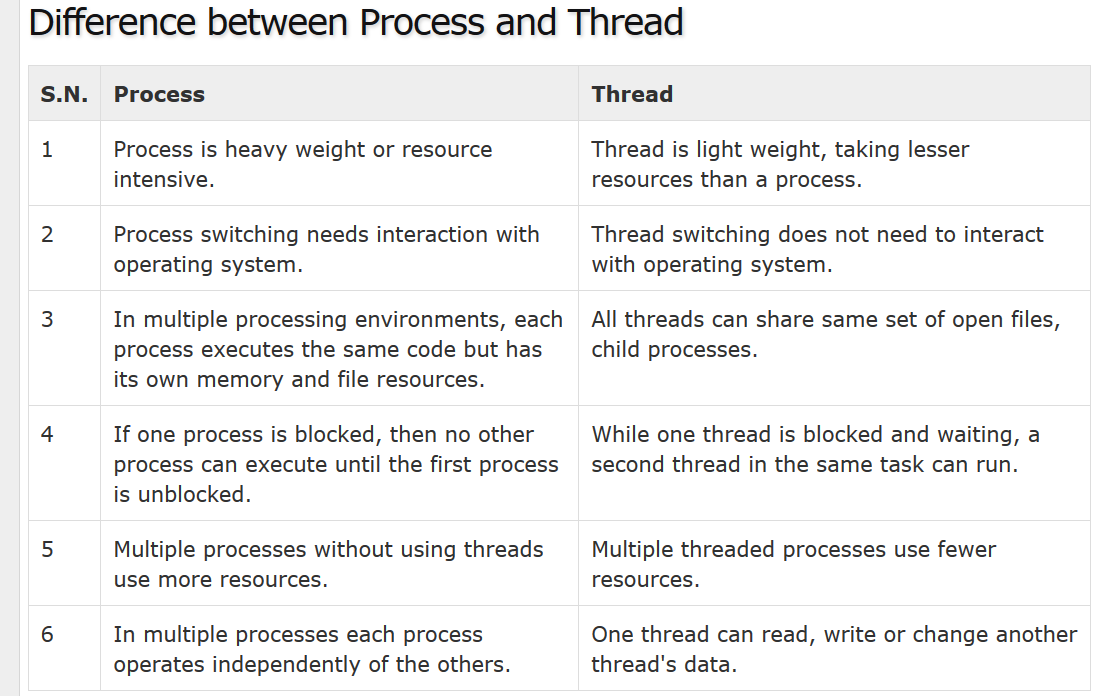
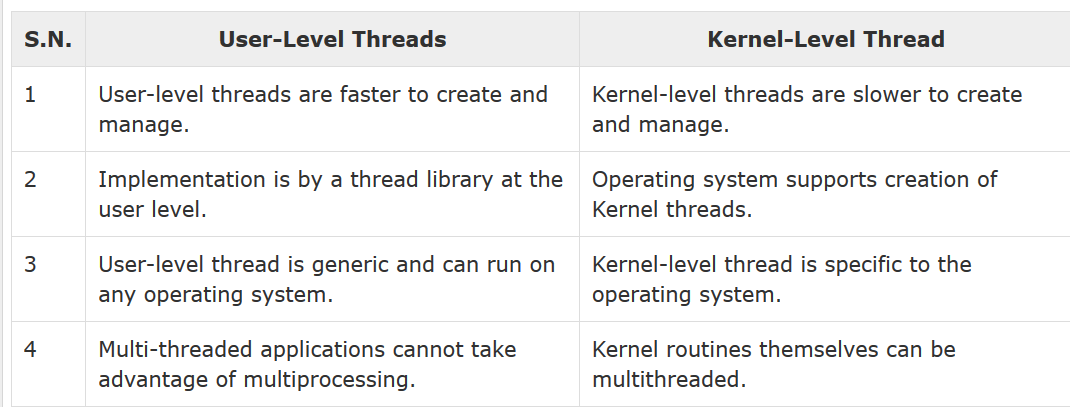
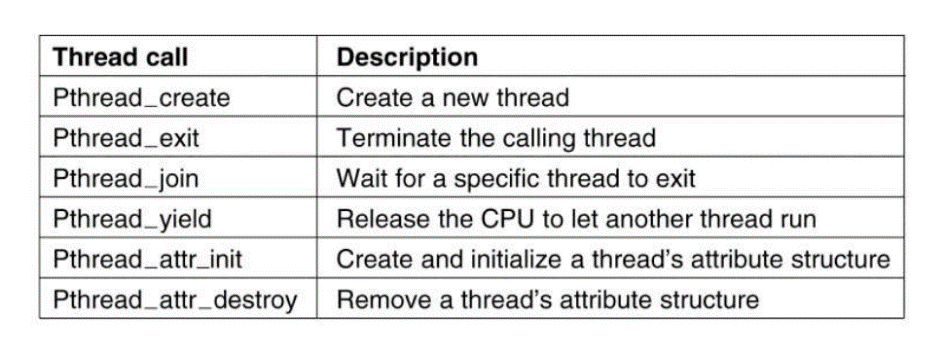
Memory layout for a multiprogramming system

* **Processes**
  + A program in execution
  + A key concept in all operating systems
  + Associated with an address space (virtual memory)
  + Also associated with set of resources
  + Process can be thought of as a container
    - Holds all information needed to run program
  + When a program is loaded into the memory and becomes a process, it can be divided into 4 sections: stack, heap, text and data
    - Stack contains the temporary data such as method/function parameters, return address and local variables
    - Heap – dynamically allocated memory to a process during its run time
    - Text – includes the current activity represented by the value of program counter and the contents of the processor’s registers
    - Data – contains the global and static variables
  + Life Cycle/States
    - START - the initial state when a process is first started/created
    - READY – the process is waiting to be assigned to a processor. Process is runnable, are waiting to have the processor allocated to them by the OS so they can run
      * May come after START
      * Or being interrupted by the scheduler to assign CPU to other process
    - RUNNING – once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its function
    - WAITING – process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting a file to become available
      * BLOCKED – unable to run until some external event happens
    - TERMINATED or EXIT – once the process finishes its execution, or being terminated by the OS, it moves to the terminated state where it waits to be removed from main memory

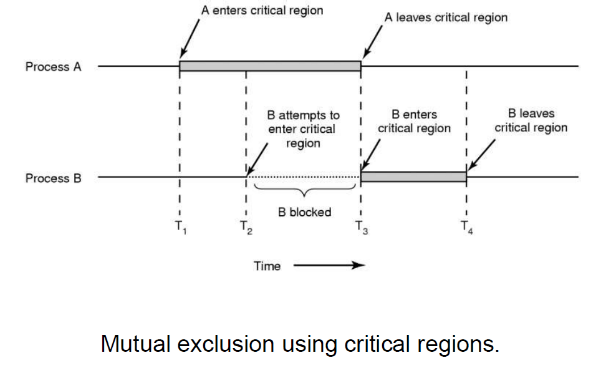


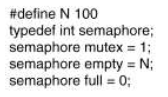
* + Process Control Block (PCB)
    - A data structure maintained by the OS for every process.
    - Keeps all the information needed to keep track of a process
      * Process state
      * Process privileges – allow or disallow access to system resources
      * Process ID – unique identification for each process
      * Pointer – a pointer to parent process
      * Program counter – a pointer to the address of the next instruction to be executed for this program
      * CPU registers - where process need to be stored for execution for running state.
      * CPU Scheduling Information - Process priority and other scheduling information which is required to schedule the process.
      * Memory management information - This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.
      * Accounting information - This includes the amount of CPU used for process execution, time limits, execution ID etc.
      * IO Status information - This includes a list of I/O devices allocated to the process.
  + Process Creation
    - System initialization
    - Execution of a process creation system call by a running process
    - A user request to create a new process
    - Initiation of a batch job
  + Process Termination
    - Normal exit (voluntary) [return 0;]
    - Error exit (voluntary)
    - Fatal error (involuntary) [divide by 0]
    - Killed by another process (involuntary)
* **Process Scheduling**
  + 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.
  + allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU
  + **First come first serve**
    - Jobs are executed on first come, first serve basis
    - Easy to understand and implement
    - Poor in performance as average wait time is high
  + **Shortest job first**
    - Best approach to minimize waiting time
    - The processer should know in advance how much time process will take.
  + **Priority Scheduling**
    - Each process is assigned a priority. Process with highest priority is to be executed first and so on.
    - Processes with same priority are executed on first come first served basis.
    - Priority can be decided based on memory requirements, time requirements or any other resource requirement.
  + Round Robin
    - Each process is provided a fix time to execute, it is called a **quantum (time slice)**.
    - Once a process is executed for a given time period, it switches to another process and executes for a given time period.

**Thread**

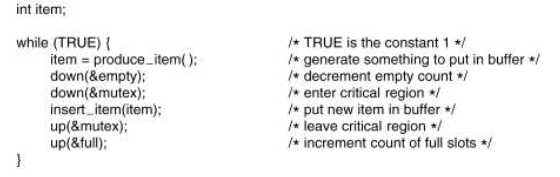
* Thread is a flow of execution through the process code, with
  + Its own program counter that keeps track of which instruction to execute next,
  + system registers which hold its current working variables
  + a stack which contains the execution history
* No connection or links between each instruction or process
* Threads provide a way to improve application performance through parallelism
* If there is only one processor for multiple threads, it splits to work in parallel
* Threads can have private global variables
* **User Mode Threads** – user managed threads
* **Kernel Mode Threads** – OS managed threads acting on kernel, an OS core
  + performs thread creation, scheduling and management
  + can be multithreading

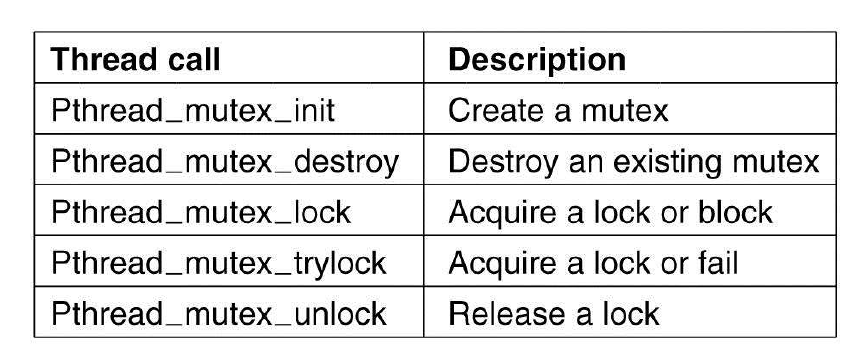
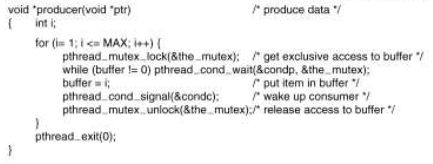
**Critical Region**

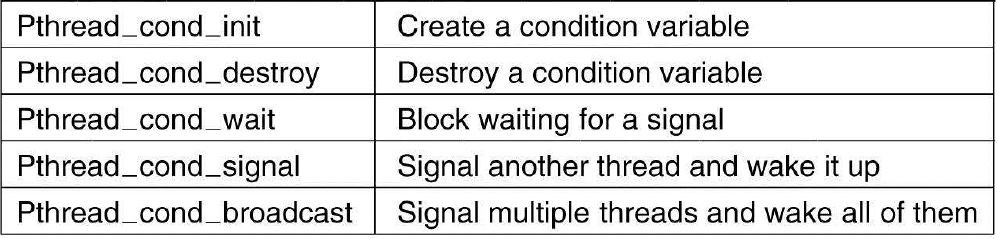
* A region of code that need to be executed continuously without interruptions until stop (i.e. uninterrupted operation)
* Avoid race conditions (two processes want to access shared memory at the same time)
  + If one thread tries to change the value of shared data at the same time as another thread tries to read the value, the result is unpredictable
* Need a lock before process critical region
* Semaphores, Mutex
* **Semaphores** 
  + An integer variable that can be modified by 2 single invisible atomic operations
    - Atomic operations: once you change value, there won’t be any interruption
  + Control who has access to the machine
  + Protect variable that shared among threads
  + A given semaphore has a predefined maximum count, and a current count

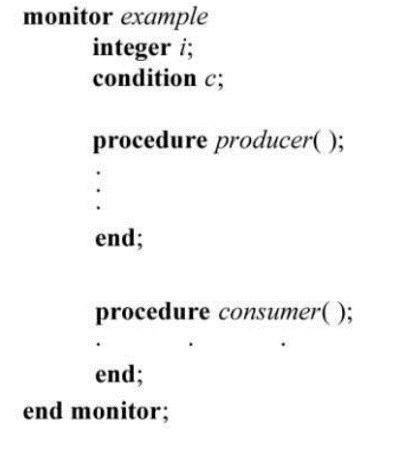


* + 2 functions: down (p) and up(v)
    - Down: take ownership with a **wait** function; decrements the semaphores
      * Blocks if result is negative without completing the operation
    - Up: release ownership with a **signal** function; increments the semaphore



* + *Any thread* can signal a semaphore, at any time, whether or not that thread has previously waited for the semaphore.
* **Mutex**
  + Mutual exclusion
  + A lockable object that can be owned by exactly one thread at a time
  + Only the thread that acquired the lock can release the lock on a mutex.
  + When the mutex is locked, any attempt to acquire the lock will fail or block, even if that attempt is done by the same thread.



* **Monitor**
  + the collection of condition variables and procedures combined together in a special kind of module or a package.
  + The processes running outside the monitor can’t access the internal variable of monitor but can call procedures of the monitor.
  + Only one process at a time can execute code inside monitors.
  + **Condition variables**
    - WAIT - Process performing wait operation on any condition variable are suspended. The suspended processes are placed in block queue of that condition variable.
    - SIGNAL - When a process performs signal operation on condition variable, one of the blocked processes is given chance.

**File System**

* collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
* a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user
* Structure:
  + Byte sequence
  + Record sequence
  + Tree
* Type – the ability of the OS to distinguish different types of file such as text files, source files and binary files, etc.
  + Executable file
  + Archive
* File Operations
  + Create
  + Delete
  + Open
  + Close
  + Read
  + Write
  + Append
  + Seek
  + Get attributes
  + Set attributes
  + Rename