**DAY-22 (27-11)**

* **System programming:**
* If applications want to interact with hardware, it must request kernel via system calls
* System calls act as interface between os/kernel and user programs
* System calls enable user-mode applications to request kernel services like file handling, process management and network communication
* Process control: creation, termination, and management of processes
* fork () – to create process

exec () – execute another process from one process

exit () – kill or terminate (interruption)

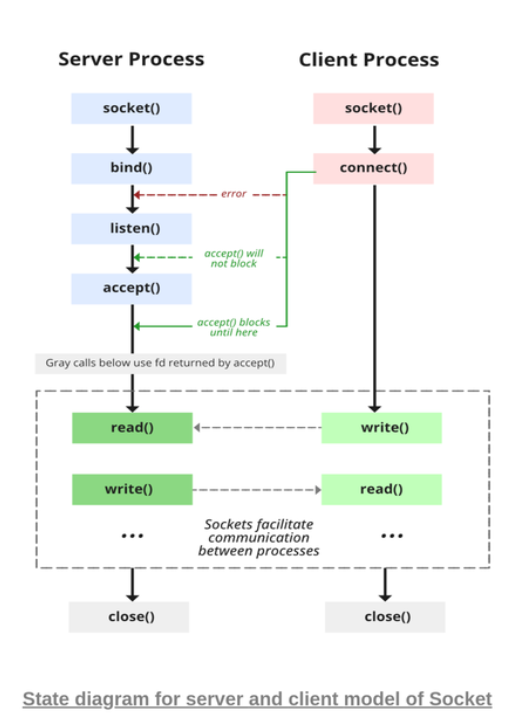
* File management: open (), read (), write (), close ()
* Device management: interacting with hardware devices
* Ioctl () –
* read () –
* information maintenance: accessing system info (getppid)
* getpid () – to know the process id
* gettimeofday () –
* communication: managing inter-process communication
* pipe () –
* socket () –
* fcntl – to know all the statics of the file (author, mode, modified information, locking and unlocking)
* strace – trace system calls made by a program
* strace ./your\_program
* process basics
* process control block
* it is DS maintained by OS to store all info about specific process
* contents of PCB: process ID, process state, program counter, registers, account information, I/O information, memory management information, scheduling information
* process – is a program in execution
* process id
* parent process
* priority
* execution context
* resource usage
* process states:
* new
* ready
* running
* waiting
* terminated
* scheduling – determines which process gets the CPU and for how long. Managed by scheduler
* types – long-term, short-term, medium-term
* in system programming we must make sure that defunct process should not be there
* fork returns child pid in parent process and fork returns zero in the child process in error it returns -1

**DAY-24 (29-11)**

* **pipes:**
* fd[0] – is meant for only reading, fd[1] – only writing
* ipcs – inter process communication table (command)
* ipcs -q: meassage queues
* ipcs -m: shared memory segments

**DAY-25 (2-12)**

* physical layer: sending raw bits of data across connection
* data layer
* **socket programming**: Socket programming is a way of connecting two nodes on a network to communicate with each other



* socket () – it will have details about IP address, port number, family of protocols
* stages of server:
* socket creation

**DAY-26 (3-12)**

* **Trees:** it is an ideal DS for representing hierarchical data
* Tree is a non-linear DS like graphs
* Connecting different nodes together is called graph
* Elements towards right of the root are greater than root and left to root are lesser than root
* Traversal
* In order: the elements are already sorted
* Post order
* Pre order
* Binary tree: no vertex has more than two children
* FAQ
* Write a C program to insert nodes in a binary tree (in order, pre-order, post-order)
* Write a C program to insert nodes in a binary search tree
* To find level of tree
* Max/min elements of a tree
* Traversal:
* Depth first search traversal (DFS):
* Preorder traversal: vertex, left, right subtree - NLR
* The first element will always be the root / middle element
* In order traversal: left, vertex, right – LNR
* It is sorted
* Post order traversal: left, right, vertex – LRN
* The first element is the last level element