- The moment process , process load the brand new program from the hard disk into its process at a space. And start executing the new program.
- No new pid generated.

atexit : It registers the function with kernel. In the normal process terminal, it executes the function in the reverse order.

Atexit is used to execute the function of pointer.

Only function and array have this ability.

Atexit function: Function calling the registered function , which are previously registered for a function process , and executes in the reverse order of the function.

_exit(0) function : It is not calling the functions which are registered with atexit function.

NI: Nice value using which we can increase or decrease the prior process.

WCHAN: It is the address at kernel where process is executing.

STIME: Where the process is starting.

Pts/0: Where the ps -elf is started.

1. Block parent: All child is still executing.
2. If a child terminated: wait () will immediately exit status of terminated.
3. Error: when there is no child processed.

--------------------*----------*----------*----------*----------*

FORK: To maintain /create a process allocates lots of resources and very expensive in terms of CPU usage, memory usage and I/O usage.

Creating maintain and destroying threads far cheaper will compare to process.

Threads are called as lightweight processes and talks about very few processes.

Thread is a parallel context of execution. Set of instructions that are executed to perform a specific task.

Multi-threading*:* Multiple parallel context of execution. Sets of different instructions are executing concurrently. Each thread is performing a specific task.

Resources that thread needs:

1) **Thread resource:**
   - Own stack regain.
   -Thread specific date
   - own registers
   -Can scheduling poliling and priority
   - own signals sigblack & sigmask

<u>Linux thread model/one-to-one thread model:</u>

1. User level thread
2. Kernel level thread

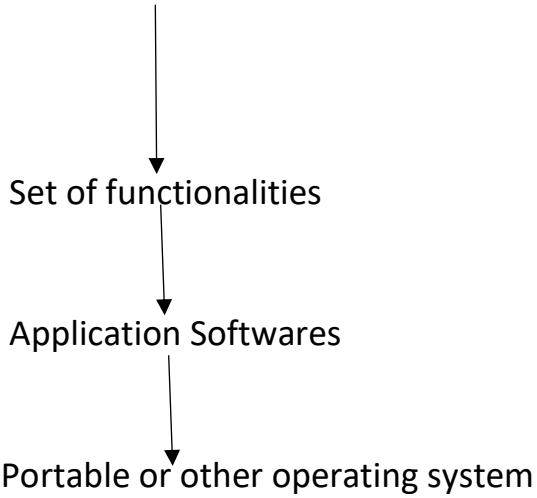
ADVANTAGES OF THREADS:

1. Threads do not carry their own at a space.
2. Communication between your process and thread happens by default.
3. Communication is very fast.

DISADVANTAGES OF THREADS:

1. Atleast one thread is causing a *bug* then entire thread and applications is crashed.

POSIX : Portable operating System Interface - Are



Posix maintains its own library

POSIX thread creation –

On success it executes thread task function.

On Failure it returns Non zero value.

Two ERROR codes :

1. EAGAIN-→When you don't have sufficient to create a thread.
2. EPERM-→When don't have enough permissions.

pthread: Returns to call the thread ID, pthread id is the same value that is store to the point pthread_t in pthread function.

Pthread term :

1. Main() return
2. Pthread_exit→For term calling
3. Int pthread_cancel→ for calling thread id

Using pthread_cancel you can terminate using pthread_self()

Pthread_exit will terminate the current task and allows the pending task to get execute.

A same thread task function can be shared by two different threads

Pthread attributes: It defines the behavior of your threads once your thread created how it should behave.

5 Different categories:

1. Stack management
2. Pthread schding policy and priority
3. Thread specific data
4. Thread synchronization data
5. Thread signal management

Whenever you want to change default attributes of thread :

1. Declare pthread attribute object (pthread_attr obj)
2. Initialize attribute object (Pthread_attr_init (: thread-attr-obj *attr)
3. Call appropriate thread att. Functions

4. To create a thread
5. Destroy thread(pthread_attr_destroy)

- No two threads can point to same thread stack
  Pthread_attr_getstack(pthread_attr *attr, void *attr, int size)

  Pthread_attr_setstack(

  Step-1 Declaration of semaphore

PROCESS OPERATION ON SEMAPHORE:

Types of semaphore:

1. **Binary Semaphore –** A Semaphore whose token value changes b/w 0-1 & 1-0 during process is called binary semaphore.
2. **Counting Semaphore –** A Semaphore whose token point value is greater than 1 is called as counting semaphore.

PTHREAD AND SPINLOCK –

Context Switching: Switching of CPU from One executing process code to another task.

Deadlock situation: Neither of the process will move further. Each process is waiting for other process to release the lock.

Advantage of dynamic initialization:
During the runtime you can change the process of MUTEX.

Static initialization: Mutex attributes cannot be changed throughout the program, they are fixed clear program.

Pthread detch call will gave the calling thread ID into detachable state.

Pthread condition variables(w.r.t. mutex)
pthread_cond_t

pthread_cond_wait(pthread_cond_t *ptr,pthread_mutex_t, *int)

pthread_cond_wait() –
pthread is acquiring a first lock and waiting for an event to happen then calling pthread condition wait call. The moment pthread condition call thread is unlocking mutex lock automatically and places itself in the wait key condition variable
The moment thread executes pthread_cond_wait call.

Pthread_cond_signal() –
Pthread condition signal function will read the variable argument and if any thread waiting on the condition will send a wakeup signal to thread.

Pthread_cond_broadcast(Pthd cond*ptr) – If a thread executes broadcast function reads the condition argument and wake up all the threads which are sleeping on condition variable.

Pthread_once: The purpose of pthread code is to implement an initialization code.

Pthread library provides a macro *PTHREAD_ONCE_INIT* setting this object to some initial value.
PTHREAD_ONCE function when executes checks for the first argument whether the argument is initialized or not If it is initialized
If not initialized will never schedule the function.

- Pthread_once schedule and initialize a function of type which takes no argument and return nothing.
- Pthread_once(once, void *fun())

PTHREAD SCHEDULE ATTRIBUTE: When a thread creates a thread a newly created thread will inherit newly schding property of parent thread**,** there is a one member called as inherit scheduler member That specifies us whether a thread is inheriting a policies and priority from its parent or managing on its own.

-Inherit Schedule member

→ **Pthread_inherit_sched**

→ **Pthread_explicit_sched**

If thread is having a inherit member as Pthread_INHERIT_SCHED So from the parent thread its getting its policy and priority

Or if member is the thread_explicit_sched it is having policy and priority of its own.

Pthread_attr_getinherit_sched

Pthread_attr_setinherit_sched

Policy

- **Default:** If the policy is default all the threads will have same policy same CPU time slice sharing. Which one runs depends on which one run first in the round robin execution.
- **Round robin –** Each and Every thread will have equal priority and execute in circular order. Each thread will use all resources for the same amount of time.
- **FIFO (First in First Out) -**threads will have the fixed priority ranging from 1 to 99.
    - Linux threads will have default schdling policy called as SCHD_OTHER
    - For default schdling policy, priority is dynamic that can be changed by your system based on behavior of the thread.
    - A thread with priority 1 in schd FIFO policy will execute first when compare to default policy thread.

Pthread_getschedparam(Thd id, int Policy, Struct_Sched_param)

Prempting schdling kernel forcefully transferring the CPU from the process without requesting the process.

- Process never leave CPU on an interruption; it will only leave when the process is finished.

Linux Employees complete fair scheduler: Each and every process will get proportions of the CPU based on factor called as **Nice value**.

- Nice value gives weightage to the process wrt the CPU slice time.

i.e., Nice value used for prioritizing the process means to increase or decrease the priority of the process.

- If Nice value is same for all process each and every process will get a time slice of **1/n*CPU** slice time (n = No. of process)

Nice Value Table:

Nice Value	Priority	CPU slice Time
-20	Highest	85 ms
0	default	10 ms
+19	lowest	58 ms

Even the lowest priority process will get a CPU slice time.

taskset: It is a command which assign a CPU code to the process. It's a utility to comes by default with Linux distribution.

& - It is a special background symbol , forces the following command to execute in the background.

Scheduler Algorithms

Round Robin :

- Each and Every process will get equal proportions of CPU slice time in circular order, if a process freeze the given slice time ,the process moves to the next process . Each and every process use CPU and resources for the same amount of time.
- No issue of priority
- Implements star wisher free process

Process Context kernel context Interrupt context

At any given point of time kernel will be executing either in process context or in a context process or in a interrupt context.

Kernel executing a piece of software on behalf of a process which had initiated the system call is called as a **system process context**.

Kernel context: kernel executing a piece of software (kernel service) on behalf of another kernel service.

Each and Every device on a system architecture is connected to a system pick controller through a physical wire called as interrupt request lines.

Interrupt context: kernel executing a piece of software on behalf of an interrupt.

Linux memory management unit:

Memory manipulation calls –

1. **memset** – when process executes memset function jumps to the address location provided by pointer argument and start set the data with the given constant C for n no. of bytes for given data.
2. **memchr** – The moment process executes memchr function jumps to address location provided by pointer variable and start scanning for the given constant C.

- According to this example, str is the direct pointer that is point into start address there ret point into interior of the point address.

3. **memcmp**- The moment process executes memcmp memory function its jumps to two address locations and start comparing the addresses byte-by-byte until it gets unmatching data returns as a +1, -1,0.

4. **memmove**-The moment process executes memmove operation copies the data from source buffer to destination buffer for given n no. of bytes. Memmove function finally returns destination address.

- If destination already have data (Data gets Overwritten)
- Slow
- Reliable
- Generate guaranteed data
- In case of memory overlapping (source and destination when get overlapping) memove provides the guaranteed solution that is why operation is slow.

5. **memcpy**-
    - Fast
    - Not reliable
    - There is no temporary buffer

alloca: The moment process executes alloca memory call allocates memory from stack segment (heap segment is filled it takes memory from stack segment) and returns memory.

Advantages:

- Minimum memory wastage
- Operation is faster when compared to malloc operation
- It doesn't maintain pool of fixed block sizes and hence no memory fragmentation
- Need not to call free memory call, the memory is automatically deallocated or removed when function terminates or ends.

Swapping Process: When RAM is running out of memory and OS wants to launch a new application into the RAM then OS looks for the inactive processes from the

RAM and push them into the Swap partition of your storage device (Hard disk) and is called as **Swap out process**.

When the newly applications have done their job kernel will get back the processes from swap partition to RAM and is called as **Swap in process**.

The entire process is called swapping.

Memory locks: There could be a login program and we don't want kernel to swap out my login process from RAM then we can apply memory locks.

Types of memory locks:

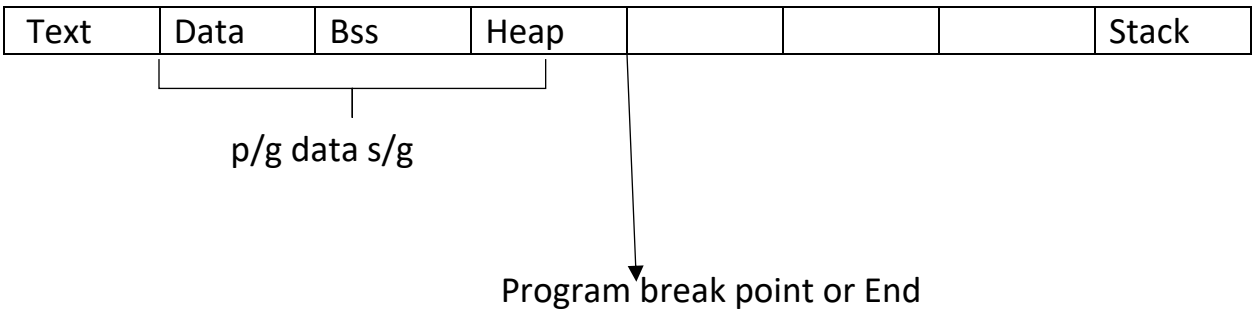
1. int mlock (void *addr, Size)
2. Int mlockall(int flags)

E NOMEM – The error returned by mlock when trying to apply a memory lock more than permutated limit will return an error.

E PERM – When No privileges for your process, a return error called E PERM

E AGAIN – If it is failed to apply a memory lock on the specified address.

MCL_CURRENT - current PAS is locked.



Address of Program

Void * sbrk (init nbytes)

int brk (void * end addr)

- sbrk and brk are used to manage memory data segments
- sbrk takes a value by which creates a new program break point and brk function takes a desired address to generate a new program breakpoint.

Void mymemcopy (char *source, char *dest., Size n bytes)

Problem with user space and kernel space

If an application is making repetitive IO request a much of CPU time spend in submitting IO operation?

M-map is a POSIX memory member function that maps a given kernel file region or a device region or some random kernel memory into the process dataspace.

void * mmap (void *addr, size_t bytes, int protection, int flags, Fd/-1, off_t)

void* addr→ This tells where to map data

size_t bytes→No. of bytes we have to map

int protection→ - PROT_READ (Program may be read)

- PROT_WRITE (Program may be write)

- PROT_EXEC (Program may be Execute)

int flags →

- MAP_LOCKED (It will apply MLOCK to shared memory region)
- MAP_SHAREED (Shared memory will not mapped between 'n' no. of process)
- MAP_PRIVATE (It create a private copy and changes will not reflect to other processes which are mapping the shared region)
- MAP_ANONYMOUS (It doesn't belong to any file)

It is recommended to use zero as address argument, zero indicates asking kernel to map at free process at-a-space.

Fd →

LINUX MMU

1. **High level MMU** → Independent of process architecture
2. **Low level MMU** → Dependent on process architecture

When the kernel booting starts, Low level MMU also starts and creates and initialize lots of memory data structure

Low Level MMU converting entire kernel memory into struct pages.

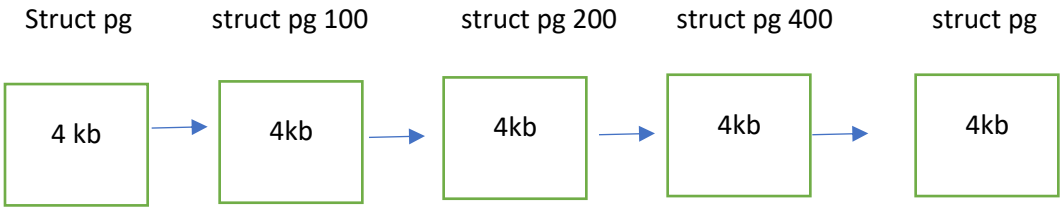
If reference to the PTE is lost kernel will never know where the process pages are in the kernel memory.

PTE identifies these particular pages belongs to 'x' particular process.

Inside the kernel there is a dynamic data structure called as **Page Frame Relation table.**

Processor will take the page no. and look for a matching frame number in frame relation table and now fetches the frame number for a Page no. Now takes the frame number and adds to OFFSET leading to the physical address of integer variable ‘x’.

For each and every IO request operation, processor must perform a logical to physical address translation. This operation is called Virtual memory address translation .



Virtual memory: A section of memory created temporarily on swap partition of the storage devices. Linux uses RAM and the virtual memory and virtual addresses are assigned to the processes.

Free command provides amount of available virtual memory and VM used by system.

v command is very similar to vmstat -s

cat -current virtual memory report

vmstat is for complete virtual memory statistics reporter.

- b in vmstat is an uninterruptable command.
- si – Swap In, so- Swap Out
- bi – blocks received from blocked device
- bo – blocks sent to blocked device
- in – no. of interrupts per second
- cs – no. of context per second

Library: It is a group of pre-compiled object code.

1. **Static Library** – They are one which are statically linked to program executable file at compile run time.
2. **Dynamic library** – They are one which are dynamically linked to program executable file at run time.

Linux OS static library extension is '. a'

ar is linux command line tool called as archive that is used to create static library.

rcs – replace create symbol

PIC: Position Independent Code

Dynamic file has extension .so and its is also called as shared library.

Dynamic library creation uses a flag called as PIC (Position independent code)

It should be relocatable that means the instructions of dynamic library should be ready to get load into any memory dataspace.

- An executable file needs the address of a dynamic library. The dynamic lib. Should merge with process merge in such a way that it part of executable program. Later if another executable file requires a dynamic library, the instructions of the library should merge with memory of the executable program.
- -shared used to generate dynamic library.

There is a tool called LTD. It is a command line tool. It provides the dependency requires for executing.

PROCESSOR:

It all depends upon the processor how it is looking at memory. Based on that OS will form policy to manage memory. Processor will look at memory at two different ways →

1. REAL MODE
2. PROTECTED MODE

Once the system is power ON during the execution of BIOS code And BOOTLOADER code Processor is in REAL mode, Looking memory as array of bytes.

Once the kernel bootup start process shifted to protected mode and looking memory as set of blocks (Linked list of Pages) and this process is called as processor initialization.

Kernel OS creating some illusion and making your CPU to look memory as set of blocks.

GDB (GNU DEBUGGER TOOL)

- It is an Open-source software
- It is a command line tool that works with executable file that are produced by your compilation process
- It is a powerful debugging tool that supports – C, C++.JAVA, PTYHON, PASCAL, PHOTON

GDB list – 10 lines of code will be displayed

GDB use break command to stop the program execution at our line of choice

Print – It will print the variables value.

If we observe real time application program most of the times bugs happens to be from user's point of view. Example: Accidently dereferencing a NULL pointer, uninitialized pointers, dereferencing a Pointer which is freedom.

Accessing a memory beyond lower bound region and beyond upper bound region.

When a process created, process dataspace is allocated. Program is expected to be used only the allocated Region. Whenever program makes a memory violation, the segmentation fault occurs.

CPU will get the logical addresses and will map the logical address to the physical addresses with the help of kernel page frame relation table.

In case logical address are invalid memory addresses and CPU giving invalid addresses to kernel frame page relation table and kernel fails to get respective frame number for the invalid addresses.

Processors throw page fault error to the kernel. Now kernel will check for the process segmented area and then send a signal called **SIGSEGV** signal to the respective process. On reception of signal, the process will abort the operation with segmentation fault.

Info locals: Will give you all the variables. It will display all the value of local variables.

ptype 'variable name'-

Attribute align: It specifies please provide minimum alignment for my structure variable.

GDB fails to track down heap memory violation.

Electric fence: It is a Library (Not any debugging tool) which has got its own implementation of malloc or calloc memory allocation.

Using electric fence when requesting dynamic memory allocations using malloc and calloc will allocate for specified amount of memory.

Standard C library malloc and calloc allocating more no. of bytes than actually requested for.

Electric fence is configured to report heap memory violations either for upper bound region or lower bound region but Not both at a time.

(**Extern int EF_PROTECT_BELOW=1** i.e., you are configuring lower bound memory violation)

VALGRIND : It is a runtime tool use to trap down heap memory violation is also called as heap memory profiling tool. It is a stand run debugger used at runtime.

When the process makes use of standard C library malloc and calloc, valgrind reports why this process has failed.

You can use valgrind with or without symbol information.

x buff printing ASCII value present at the particular location.

x/s buff will print the string until you reach the NULL character.

Instead of recompiling the code changes can be made from GDB command to modify the value of existing variable with help of GDB command called set.

Makefile: It is called as Program building tool in Linux and Linux OS. Makefile is a set of commands similar to terminal commands, but the difference is makefile will have an organized text instruction.

Makefile will have variables, single targets, or multiple targets.

Three different Makefiles:

1. **Kernel build MakeFile -** Entire Linux kernel OS is build into a makefile called as kernel build makefile is located at */lib/modules/<kernel version>/build/Makefile*
2. **Processor Architecture Makefile -**Used to fill processor makefile
3. **Subsystem level Makefile-** Used to fill subsystem makefile

Rules for making Makefile:

- Makefile should start with uppercase 'M' without any extension
- All the source files, makefile should be in current directory
- Makefile start with target name and colon
- Next instruction should start with <TABSPACE>

Make Tool:

- Make is a Linux utility tool used to generate application executables.
- Make tool by-default comes with Linux distribution.
- Make will reduce workload of compilation.
- Make tool will execute the Makefile of current directory and jumps to Makefile and starts executing the instructions of Makefile.
- GCC is indirectly called by Makefile in turn develops execution applications.
- The primary objective of Make tool is to break down large project source code into smaller pieces and assess the smaller piece of code whether they require Recompilation or not.

Target:

Target is a Filename which is generated by Makefile program. A target can be an executable application or target can be an Object file or it may be an action such as Clean.

appln: main.o calc_mean.o calc_sub.o Target dependencies (Sub target)

In this Makefile we are having variable and multiple targets, here we make object as target . If I want to recompile any individual file this make will reduce workload of recompiling all files. Make will only compile modified file.

System(fork, Excel, wait)

int system(Const char* command)

ERROR CASE: →error child not created

→error for little argument

The return system system call will talk about the status of the shell. When you are getting '0' shell is not available as shell is busy in executing command.

When NULL given as a command, the return value is 1 i.e., nonzero value that indicates shell is available.

atomic_t :

Whenever declare a variable of type atomic_t Linux kernel provides a set of functions to safeguard the declared integer variable.

atomic_t is a datatype declaring atomic variables. To initialize atomic_t u = ATOMIC_INIT(0);

atomic_t variable ensures safety of integer variable on concurrent access.

Inter Process Communication:

Example- HTTP client Process connecting to http server

- POP (Post office Protocol) client process connecting to POP server

Each node in the network is identify by a unique 32-bit IP address and is also used for connecting and communicate and 16-bit port address is used to identify process in particular node

LINUX has provided various inter communication techniques-

1. Pipe-
    - It is a serial communication device that permits unidirectional data transfer.
    - Can be used b/w parent and child process.
    - When a pipe system is called from the process, a pipe gets from the kernel space
    - Pipe also returns 2 descriptors fd0, fd1.. fd0 is always associated with read end of pipe and fd1 is always associated with right end pipe.
    - When parent process calling fork child is created
    - Parent copying the file scripters to the child process (Child inherit a pipe file from parent). Thus, the pipe is limited to parent and child process.
    - After a write operation parent flushes, the data with which data immediately reflected on read end.

2. **FIFO**

$ mkfifo(const char *pathname, mode_t modes)

Header files required are:

- <sys/type.h>
- <sys/stat.h>

The moment process executes mkfifo function a FIFO file gets created and makes a entry into LINUX kernel OS file system. A FIFO file is a special file which gets all the attributes any process can open it for reading and writing like any other normal regular file.

Ensure that FIFO is open from both the ends simultaneously by reader and writer.

If a process is opening FIFO for reading operation has to wait for other side process to open FIFO for right side operation and vice versa.

A FIFO can have multiple readers and writers and supports connection-oriented communication.

FIFO is used for simple client server communication protocol.

3. **Message Queue-** Each message queue has allocated with a structure struct msqid_dS

msgget – The moment process calls the msgget system call, a new message queue is created in the kernel space with a unique message queue ID. For newly created message queue a structure called msqid_ds structure with default initial values.

Key - Whenever a process is requesting resources such as message queue, shared memory, semaphores. OS needs a key of type key_t which is supposed to be integer defined in header file <sys/stat.h>

Int flag – It talks about a flag called as IPC_CREATE returns message queue ID for new message queue created, returns a message queue ID for Existing message queue of same key.

msgsnd(int msgid_ds, const void* ptr, size_t n bytes, int flags)

The moment process executes msgsnd function pointer pointing the message that is sent to the message queue whose ID provided by message queue ID. On success message is send.

msgrcv(int msgid, const void* ptr, size_t nbytes, long type, int flags)

It will receive the message from message queue of given type and copy into the second argument pointed by pointer.

msgctl(int msgid, int cmd, struct msgid_ds, *buf)

4. **Shared memory –**

- It allows two or more process to access a given region of shared memory
- Shared memory is fastest IPC communication technique.
- Need not to jump from one memory location to other memory location for data read and write operation b/w reader and writer process or also in client and server process.
- Shared memory can be used b/w server and client's machines only trick is synchronizing access of given shared region with semaphore techniques.
- Each and every shared memory have the shmid structure

int shmget(key_t key, size_t nbytes , int flag)

void *shmat(int shmid, void *addr, int flags)

when a process executes shmat system call will take the given shared memory region and map into free process dataspace (2nd argument provided ZERO) and we have flag SHM_RDONLY

shmdt(void *addr) → Returned by shmat ()

Shmctl(shmid, int cmd, struct shmid_ds, *buf)

LINUX KERNEL ARCHITECTURE:

System call are kernel space services not user space services. They are meant to invoke kernel services. Each and Every system call has to pass through system call interface and then invoke a appropriate kernel service.

Driver from kernel Point of view –

Device Driver is a piece of software that communicates with hardware, manages with hardware, and brings functionality of the end device with the user.

In LINUX OS device drivers recides in the kernel space because only in the kernel space, driver has privilege to talk to the hardware.

Driver has two interfaces –

1. Interface b/w application and driver i.e., OS specific.
2. Interface driver b/w hardware

5. Semaphore

It's an IPC communication technique. It deals with array of semaphores, Array of semaphore is a bit complex issue but in the large application software, process needs to work on lots of resources needs more protection than having an array of semaphore is a big advantage.

Semget(key_t key, int no. of sem, int sem flag)

On successful execution creates a semaphore and return semaphore id, on failure returns -1.

Semop (int semid, struct sembuf *ptr, Size_t no. of structures used)

It is used to change the value of semaphore.

Semctl (int semid, int sem no, int cmd, union semun)

It is used to control the given semaphore operations.

- Kernel will read a flag called SEM_UNDO and will know what changes have been done to the semaphore by the process.
- If process terminated without releasing semaphore SEM_UNDO will allow to automatically release semaphore.

Int SETVAL– Its used to initialize the semaphore to a well-known value. The value is required to pass value member of 'union semun' Only then the semaphore will setup. This procedure is performed when we are using semaphore for the first time.

Examing C Source Code:

C scope- It is Linux utility tool used in software development process to examine C source code. We can examine the symbol (variable, macros, functions) of source code, to check to what values they are initialize and where they are used. Using C scope tool, you can also verify specific function calling what all other functions. And check specific function called by what all other function. We can change a string, we can find a specific file, we can check for #including header file, we can also use assignment variable option to change the variable value.

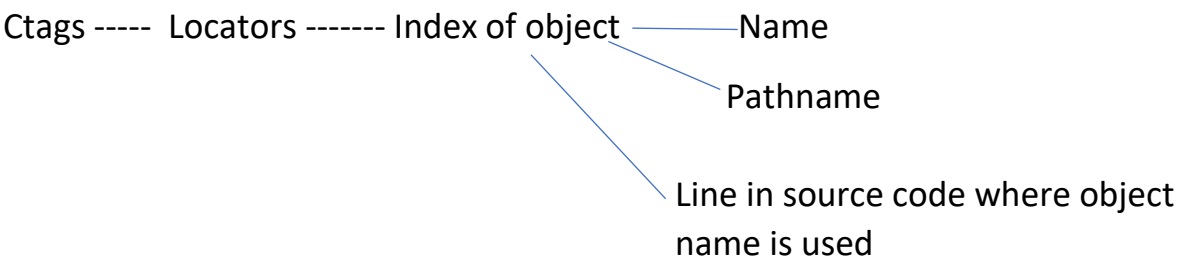
$ Ctags --list-languages → Will show all the programming languages which system supports

Ctags – Index for name found in source file & header files

Ctag is a Linux utility tool, it generates a tag file i.e., an index file for the names found in source code and header file. Index of object refers to name, path name, line where it is used as source code. Ctags provides the quick references to our source code.

Example: Find the definition of your particular function X.

Ctags uses locators and locators will locate the object and pathname of the object and line is a source code where they are used. Out of these it will generate a tag file for quick references.



SOURCE CONTROL/ VERSION CONTROL:

- Allows you to track down files progress for a period of time
- Allows to store modified or changed or newly created files.
- Source controls protects against the data lost or data damage.
- Helps in developing complex project development.
- Example- Linux is a massive project in GitHub supported by thousand competitors.

Git:

Advantage of GitHub is Disconnected Repository System.

GitHub is supported by a Disconnected Repository System

Tortoise is a Connected Repository System.

How a Git Project Starts: -

- Working →modified / Changed files
- Staging →marked or ready to move to next stage
- Repository →permanently safely stored

Git configuration variables

$ git config --global user name" "

Git config is a configuration variable used to get or set git configurations.

-- global → get information and store in your local machine.

git clone → copying a remote repository into your local machine.

LINUX OS SIGNALS:

- Signals are software interrupts They notify process about an event occurred.
- Signals are asynchronous in nature they can happen at any time.
- A CPU, kernel or any software that is running can trigger signals to the process.
- A process with enough permissions can send a signal to other process.
- A process can send signal to itself.

Terminal Generated signals- Signal generated by certain terminal keys.

LINUX signals have naming conventions they start with 3 characters SIG, Each and Every signal is defined by number provided in header file called <signal.h>

Linux signals is classified into 2 categories:

1. Std/traditional signal → 1 to 31
2. Real time signal

Hardware Exception signals-

Invalid memory reference

A signal b/w signal generation and signal delivering is known as Pending signal.

In response to signal process has to do some process

- process can run a default action
- process can ignore a signal
- process can run user-defined function or

Instead of running a default action on delivery of a process, a user can program a user-defined function and can register with a kernel. On delivery of a signal kernel should invoke a user defined function instead of default action.

Signal Handler→ user defined function

Void (*signal (int signal, void (*fun) (int))) → Installing or establishing a signal handler with kernel.

GRUB (Grand Unified Boot Loader): -

By executing in BIOS mode, CPU is in Real Mode.

Vendor of the motherboard is providing the addresses of the BIOS code.

Kernel image is divided into two parts:

1. Boot strap code
2. Vm Linuz

The first piece of code that boot strap code executes and shifts the CPU from real mode to protected mode and the process is called as process initializer.

Kernel is a first piece of software that is loaded in the RAM that executes until shutdown.

Role of kernel:

1. Will setup memory DS
2. Will setup Interrupt DS
3. Will setup Device DS
4. Will setup process stable
5. Will setup file management
6. Will setup CPU scheduler
7. Will improvise Kernel TH DS

If there are 3 boot pins, there will be 2^3 booting options.

Dynamic tool Analysis of Source Code: -

GCon →

- gcc coverage tool, Open-source tool
- GCon works along with gcc.

GCon is used for analysis of your source code and check for untested part of the source code and also identifies unexecuted instructions, can also be used as profiling a tool for browsing and navigating the source code. Allows you to modify enhance the source code. Using which we can optimize the source code.

Compiling Your source code with two flags -fprofile-arcs, ftest-coverage flag will do analysis on source code file and generates a hidden code into the binary and also records how many times the code is executed.

When we are using -fprofile-arcs with the source code generating a profiling output file .gcda & .gcno

NETWORK PROGRAMMING: -

Network programming is all about writing applications program using network APIs and Network protocols. These programs communicate with program either on the same machine or programs on different machine on remote location.

Two types of computer N/W: -

1. **WAN**
    a. **CSN –**
    - A dedicated physical path is established b/w two points through several nodes.
    - A path is the connection of higher sequence of physical links b/w the nodes (Cable). If a sender wants to transfer packet, the packet goes through these physical links as rapidly as it can. Eg. Telephone communication.

    b. **PSN –**

It uses N/W protocols that divides the data into smaller packets called as Network packets , Network packets are transmitted through over the N/W in digital format. Eg. WIFI

2. LAN

LAN works on the broadcast approach without having any intermediate switches because of which data read in LAN is much greater than WAN.

Routers and switches are networking devices they are used for connecting one or more devices to other computers or other networking devices or to connect other networks.

Hub is used to connect devices in a LAN.

Bridge is a connecting device which connects two LAN's.

IP Address → IPv4, IPv6

Traffic may increase from the receiving on the receiver side, receiver may lose the packet.

Data link layer will maintain a common data rate speed.

Also does physical addressing job.

At physical link layer we have devices, routers, switches, hubs.

Ifconfig→ Used to show active network interfaces of the system.

Used to view and change network interface configuration of your system.

MTU (Maximum Transfer Unit): Maximum size of a packet in a transmission.

Txqueuelen → It is a maximum no. of packet in a transmission. It is an interface of a Linux kernel that deals with the network drivers.

Loopback address-

Each devices have a loopback address as 127.001. it is also called as local host. When we transmit the data or send the data using loopback address, data never reaches the address data is in the loop in the network card address.

Loopback address is used for testing PCPAP internal flow part.

It will help the device to send and receive the packets of its own.

Socket system calls: It supports many networks communication protocol. We can say that socket system calls are designed to support network communication protocol.

Because of same reason socket system parameters are generic in nature.

As we are using generic socket system calls, all the system call will use same socket structure as argument and take the socket structure as struct socket address.

Socket system calls also takes size as a parameter. This will identify the size of socket structure.

Socket is an end point or final point or an end link in any n/w communication.

Basically, needs two sockets, one on sender side and one on receiver side.

Socket parameter: -

1. Protocol
2. IP Address source
3. IP address destination
4. Port address source
5. Port address destination

Network application programming is all about implementing client server model programs.

HTTP client program is connecting to HTTP server program.
POP client connecting to POP mail.

Sin family → n/w protocol

Sin port → 16-bit port no. (n/w byte order)

Sin addr → 32-bit IP address (n/w byte order)

Int Socket (int family, int type, int protocol)

Int family will decide the network protocol which we want to use.

Int type will decide the type of n/w protocol.

We use a family here AF_INET

In IP protocol header the protocol value is zero.

On successful execution of a socket system call returning a small integer called a socket file descriptor. Sockets are internally nothing but special files.

Bind (int sock, struct sockaddr *source int attr)

Int listen (int sockfd, int queuelen)

Int accept (int sockfd, struct sockaddr *client, int *addrclie)

In a network network there are different types of computers which are using different CPU's And different microcontrollers. Few CPU follows big endian format and few follows little endian format to store integer byte in memory.

TCAPI specifies all the protocol headers of a n/w packet to follow n/w byte order(Its big endian format)

Every machine in the n/w should be aware of this and should convert host order to n/w byte order. Again at the receiver side while receiving incoming packet should convert n/w byte order to host byte order.

Source code will use API to convert host n/w order to match respective CPU requirement.

This API will work for both little endian and big endian machine.

Ulong Htonl (long host long)

This htonl function is taking 32-bit host order IP address and convert into 32-bit n/w byte order

Ushort (ushort host short)

Address conversion API's: -

Unsigned long Inet_addr (char *str)

"127.0.0.1" dotted IP address 32-bit n/w byte order

Difference B/W TCP & UDP: -

TCP	UDP
1. CONN. ESTABLISHMENT	1. No conn. Establishment
2. CONN. ORIENTED	2. connection less
3. SLOW	3. Fast
4. MORE RELIABLE	4. Not reliable
5. EXTENSIVE ERROR CHECKING	5. Basic error checking
6. SUPPORTS STREAMING BYTE TRANSFER	6. Support data package transmission
7. 20- 60 BYTES	7. 8 bytes
8. NO BROADCAST	8. Broadcast support
9. DATA SEQUENCING N/W PKT ARRIVE IN ORDER	9. No data sequence through appli. Manage data
10. GURRANTIES THE DATA DELIEVERY	10. No data guarantee of data delivery

11. HTTP/HTTPS/POP/FTP	11. TFTP, DHCP, DNS
12. CAN ESTABLISH 3-WAY HANDSHAKING	12. No 3-way handshaking
13. CAN CLOSE 4-WAY HANDSHAKING	13. No 4-way Handshaking
14. SERVER	14. Client

Accept system call is called by server shall accept the 1st client request on the queue and creates a new socket of the same socket type which is specified (Address family of newly created socket is also of same specified type) and returns a socket file descriptor.

Implementing Fork in server to handling a client.

Data Transfer API's:

```
Ssize_t recufrom (int sockfd,
                  Char *buf,
                  Int length,
                  Int flags,
                  Struct sockaddr *addr,
                  Int addrlen);
SSize_t Sendto (int sockfd, MSG_CONNTERM,
                  Char *buf,
                  Int length,
                  Int flags,
                  Struct sockaddr *addr,
                  Int addrlen);
```

RAW SOCKET: -

Raw socket sends raw sockets to application users now the user will get untampered final n/w packet. Now this packet is by passing all TCPIP processing. For customizing and optimizing VAN n/w.

If you want to improve the performance and security on critical traffic workloads on VAN, Raw sockets can be used.

```
Struct sockaddr_ll
{
        sl_family;
        Sl_protocol;
        Sl_pkttype;
        Sl_  type
        Sl_addr[f];
        Sl_addrlen;
}
```

Packet interface at device level: -

- PF_PACKET family used to create packet socket,
- Data transmission is from physical link layer that is below IP layer.
- Type can be SOCK_RAW SOCK_DGRAM
- When using SOCK_RAW the packet including link layer header is passed to the specific user. When using SOCK_DGRAM the packet without link layer header is passed to the user.
- ETH_P_ALL protocol value is defined by IEEE people. The value is STD ETHERNET PACKET

SetSockopt (int sockfd, int level, int option_name, char *option_value, size_t,size of option value)

Setsockop is used by the application program to change the socket behavior. We can set any option of the socket.

Implementing terminal Emulator program. While reading from multiple programs say example keyword.

If I configure a process to read from keyboard, when there is no data to read from keyboard, the process is blocked, If it has data, the process cannot read.

To overcome the issue, we may use multiple threads to read from multiple input sources.

Advanced option - We will use file descriptor has select() system call.

Select system call - Makes the programmer to read from multiple file descriptors when fd are ready for IO operations.

Int Select (int nfds, fdset *readfds, fdset *writefds, fdset *errorfds , struct timeval timeout);

ndfs tells the select system call the no. of fd's to be checked.

The first pointer of type fdset is for testing the readiness the fd's for read operations.

The second pointer of type fdset is for testing the readiness the fd's for write operations.

If any fd is having error pending will be verified by this pointer.

Timeout is the variable or object of type struct timeval structure. This timeout will make select system call to block until a fd is ready for IO operation.

Fd is set until its time expires.

Interrupted by signal.

Nfds either readfdset, writefdset, errorfdset

Open ssh client: -

$ ssh username y @ IP address of y

Y message is sent public key to X machine established? y/n

If y then X machine /home/user-→ .ssh file is created

If n then X machine → known_host

- Using SSH Client can connect to remote machines and can control the machines.
- Open ssh is an open-source software that is used to provide a secure and encrypted communication in a computer n/w using ssh protocol.
- Ssh protocol establishing connection b/w ssh client and ssh server.
- Ssh protocol used by system administrator to access a remote machine in unsecured n/w.
- Ssh is a TCP IP protocol that uses dedicated port no-22.

SCP (Secured Copy): - It uses shh for secured copy.

Ssh is used to connect two machines or the network and also to control the machine and HCP for transferring the file.

After establishing connection b/w client ssh and server ssh, ssh protocol is providing strong encryption and algorithms for ensuring the privacy and data security b/w client and server machines.

IO perm – There is only one system call from the user space that can access the data port and cmd port of the particular device.

Wireshark – It is an Open-source software tool that captures incoming and outgoing n/w data packets in real time. And the captured information is used for n/w troubleshooting and diagnosing and used for packet analysis, software development and communication protocol development.

- With Wireshark we can see the things around the n/w. And we can get the detailed packet movement information.

- Wireshark supports 100's of packets of different protocols. The only logic applied is capturing packets online and applying offline.
- It can read and write different capture file format.

- Main toolbar gives functionality of packets
- Filter toolbar is used to filter the packets.
- Status bar is used to summarize the packets.

IP Metadata: - Ip packets carry IP header (22-24 bytes + actual data)

Promiscuous mode: - We are capturing n/w data packets of other n/w devices on same n/w i.e., capturing the packets or traffic that are going through all other devices of the n/w. By default, your n/w card is listening and wanting the packets that are coming to your mac address.
Once the promiscuous mode is started allows your n/w card to listen and monitor, allows the packets of other n/w devices.

Filters: -
Captured filters- They are used for filtering the packets during a live session,
TCP.port == "Port no."
If you want to filter the packet wrt port no. or protocol. We need to inform the Wireshark before filtering. We can use IP address before filtering.