OS DAY-03:
System Arch Design of UNIX:
- file subsystem
- process control subsystem

1. Process Control Subsystem:
Q. What is a Program?
User Point of view:
A Program is a finite set of instructions written in any programming language given to the machine to do specific task.

System Point View:
Program is nothing but an executable file onto the HDD, which has got elf header, bss section, data section, rodata section, code section, symbol table etc...

- Program is a passive entity, whereas process is an active entity

Q. What is a Process?
User Point View:
- Program in execution
- Running instance of a program is called as a process
- When a program gets loaded into the RAM/Main Memory it becomes a process.

System Point of View:
- Process is nothing but program (an executable file) which is into the main memory has got one structure i.e. PCB for it into the main memory inside kernel space and program is there inside user space has got:
bss section, rodata section, code section and 2 new sections got added for a process by the kernel/OS:
stack section: it contains FAR's of called functions
heap section: dynamically allocated memory

Q. Why RAM is called as Main Memory?
Ans: For an execution of any program RAM memory is must, and hence it is also called as Main Memory.

Kernel: it is a core program/part of an OS which runs continuosly into the main memory and does basic minimal functionalities it.
- kernel gets loaded into the main memory while booting, and it resides always into the main memory till we do not shutdown the system, and hence few part/ portion of the main memory will be always occupied by the kernel.
- main memory is divided logically into two parts:
1. kernel space: portion of the main memory which is occupied by the kernel
2. user space: portion of the main memory other than kernel space

SunBeam

- When we execute a program (either by double clicking on icon of an executable file OR ./program.out + press enter):
- loader very first verifies file format, if file format mathces then it checks magic number, if file format as well as magic number both matches then only it starts an execution of that program/process gets submitted.

- When we say process gets submitted/an execution of a porgram is started => practically speaking : PCB gets created for that program/process:
upon process submission:
step-1: one structure gets created by the kernel for that process into main memory inside kernel space referred as **PCB (Process Control Block)**, PCB contains all the information which is required to complete an execution of that program.
step-2: loader removes elf header & symbol table from program, and it copies bss section, data section, rodata section & code section as it is into the main memory inside user space and 2 new sections got added for the process:
1. stack section
2. heap section

- Upon process submission, PCB for that process gets created into the main memory inside kernel space and it remains presents there till process do not gets terminated, i.e. when an execution of program is completed PCB of that process gets destroyed/removed from the main memory.

no. of PCB's into the main memory = no. of processes running in the system

- **Per process an OS creates one structure referred as PCB**. PCB is also called as PD(Process Descriptor), In Linux PCB is referred as TCB(Task Control Block).
- PCB/TCB/PD, **mainly it contains:**
1. pid – process id – unique identifier of a process for OS
2. ppid – parent's process id
3. PC: Program Counter - it contains an addr of next instruction to be executed
4. Memory Management information
5. CPU Scheduling Information like priority of a process, cpu sched algorithm etc...
6. Information about resources allocated for that process
7. Information about IO devices allocated for that process
8. Executation Context:
When the CPU is currently executing any program, information about data & instructions of that program can be kept/store temporarily into the CPU registers, collectively this information is referred as an execution context. Copy of an execution context also can be kept into the PCB of a process.

9. current state of the process: new state / ready state / running state / waiting state / terminated state.

etc.....

Sunbeam

- In Linux size of PCB is around more than 1 KB

shell => systemd – system daemon process
systemd => init process
init process – self generated process

upon process submission => PCB is there for a program into the main memory inside kernel space => **running program.**

after process submission => it may be active or inactive

active running program => if PCB is there into the main memory inside kernel space and program is also there into the main memory inside user space.

inactive running program => if PCB is there into the main memory inside kernel space but program is not there into the main memory inside user space (it can be kept temporarily into swap area).

if PCB is not there into the main memory inside kernel space => program is not running i.e. an execution of a program is finished.

- at a time multiple programs can be active, but there is limit on no. of processes can be active at a time.

- Throughout an execution of any program it goes through different states and at a time it may exists only in one state, and current state of a process can be stored into its PCB.
- there are total **5 states of process:**
1. new state: upon process submission it is considered in a new state.
i.e. if PCB for any process gets created into the main memory inside kernel space state of that process is considered as a new state.

2. ready state: after process submission, if process is there into the main memory and waiting for the CPU time i.e. ready to run onto the CPU, state of that process is considered as ready state.

3. runnning state: if the CPU is currently executing any process, state of that process is considered as running state.

4. waiting state: if a process is requesting for any io device, then it goes into waiting state.
After io request completion process changes its state from waiting to ready

5. terminated state: after exit process goes into the terminated state

SunBeam

i.e. if PCB of a process is destroyed from the main memory it is considered in a terminated state.

dispatcher: it is a system program (i.e. inbuilt program of an OS) which loads program (i.e. data & instructions of a program) from the main memory onto the CPU registers.

eDAC + eKDAC = DAC
Batch => logical
Physical => each & every student

process – logical
thread – practical

- thread – smallest indivisible part of a process
- thread smallest execution unit of a process.

+ Features of an OS:
1. Multi-Programming: system in which muliple jobs/processes can be submitted at a time, i.e. at a time an execution of multiple i.e. more than one programs can be started.
- at a time multiple PCB's can be created into the system.
- degree of multi-programming – no. of processes that can be submitted into the system at a time.

2. Multi-Tasking: system in which the CPU can execute multiple processes concurrently / simultaneously (i.e. one after another).
i.e. The CPU can execute only one process/job at a time.
- the CPU executes multiple processes concurrently with such great speed, we feels/it seems that , CPU executes multiple processes at once, and hence this feature is referred as multi-tasking.

- Multi-tasking is also referred as **time-sharing** => system in which the CPU time gets shared among all running programs.

Eskon temple => arati

group of people for performing "arati"
panditji - 1 => process-1
panditji - 2 => process-2
panditji - 3 => process-3
panditji - 4 => process-4

- there are two types of multi-tasking:

SunBeam

1. process based multi-tasking => fork( ) system call

2. thread based multi-tasking => multi-threading programming with pthread library is used.

person=1 =>

3. Multi-Threading: system in which the CPU can execute multiple threads which are of either same process or are of different processes concurrently/sumiltaneously.
- the CPU executes multiple threads processes concurrently with such great speed, we feels/it seems that , CPU executes multiple threads of either same process or are of diff processes at once, and hence this feature is referred as multi-threading.

- CPU time gets shared among multiple threads of all processes.

“The CPU can execute only one thread of any one process at a time”.

4. Multi-Processor

5. Multi-User

- Drive Car:
 # DAY-01:
step-1: switch on
step-2: press cluch fully & break fully
step-3: we need to change gear from neutral to 1st
step-4: release cluch gradually till half
step-5: release break gradually
step-6: slowly increase accelrator as per requirement
.
.
.
.

#DAY-20:
step-1: switch on
step-2: press cluch fully & break fully
step-3: we need to change gear from neutral to 1st
step-4: release cluch gradually till half
step-5: release break gradually
step-6: slowly increase accelrator as per requirement
.
.

SunBeam

.
.

- we feels that we performed all steps at once => ??
responsiveness to stimuli => reaction time given by the brain to all actions is so quick, we cannot different all out actions.

- **google chrome** => process => thread based multi-tasking
new tab -1 => new thread gets created
new tab -2 => new thread gets created

- mozilla firefox => process => process based multi-tasking
new tab -1 => new diff process gets created
new tab -2 => new diff process gets created

google chrome => youtube

downloading => thread1
video streaming => thread2
audio streaming => thread3

SunBeam

- you dont have to programming by using vi editor
vs code editor

pid_t fork(void);

fork( ) sys call createa new process/child process by duplicating calling process, which do not accepts anything and it returns pid of child process to its parent and 0 into the child process.

pid is of type of **unsigned int**

typedef unsigned int pid_t;

ret = fork( );//function call

ret = return value of fork( ) is getting assigned to ret
after fork() system call execution statement, assigning return value of fork( ) to ret var

SunBeam