EI_Meghalohar_49398

PAGE:
DATE: / /

Sunbeam Institute of Information Technology, Pune

Embedded Operating System Rapid Fire Sheet

Rapid Fire Questions

Question 1

What is OS ? What are its important functions ?

Operating System
- Interface between end user and computer hardware.
- Interface between Programs and computer hardware.
- Control program that controls execution of all other programs.
- Resource manager / allocator that manage all hardware resources.
- Bootable CD / DVD = Core OS + Applications + Utilities
- Core OS = Kernel
  It performs all basic functions of OS.

OS Functions
Process management
CPU scheduling
Memory management
File & IO management
Hardware abstraction
Networking
Security & protection
User interfacing

El_ Megha Lohar - 49398

Question 2

What is system call ? How it is executed ?

System call

- System calls provide the interface between a running program and the operating system.
- Typically written in a high-level language (C or C++).
- Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use.
  * Win 32 API for Windows.
  * POSIX API for POSIX-based systems
- Defination of System call

Functions exposed by the kernel so that user programs can access kernel functionalities, are called as "System calls".

e.g.

Process Management : create process, exit process, communication, synchronization etc.

- System calls are specific to the OS.

UNIX => 64 syscalls e.g. fork()

Linux => 300 + syscalls e.g. fork(), clone()...

Windows => 3000 + syscalls e.g. CreateProcess()...

System call execution.

Device driver is a program that gives instructions to device controller & also handles intrs. from that device.

El_ Megha Lohar - 49398

PAGE:
DATE: / /

System call execution:

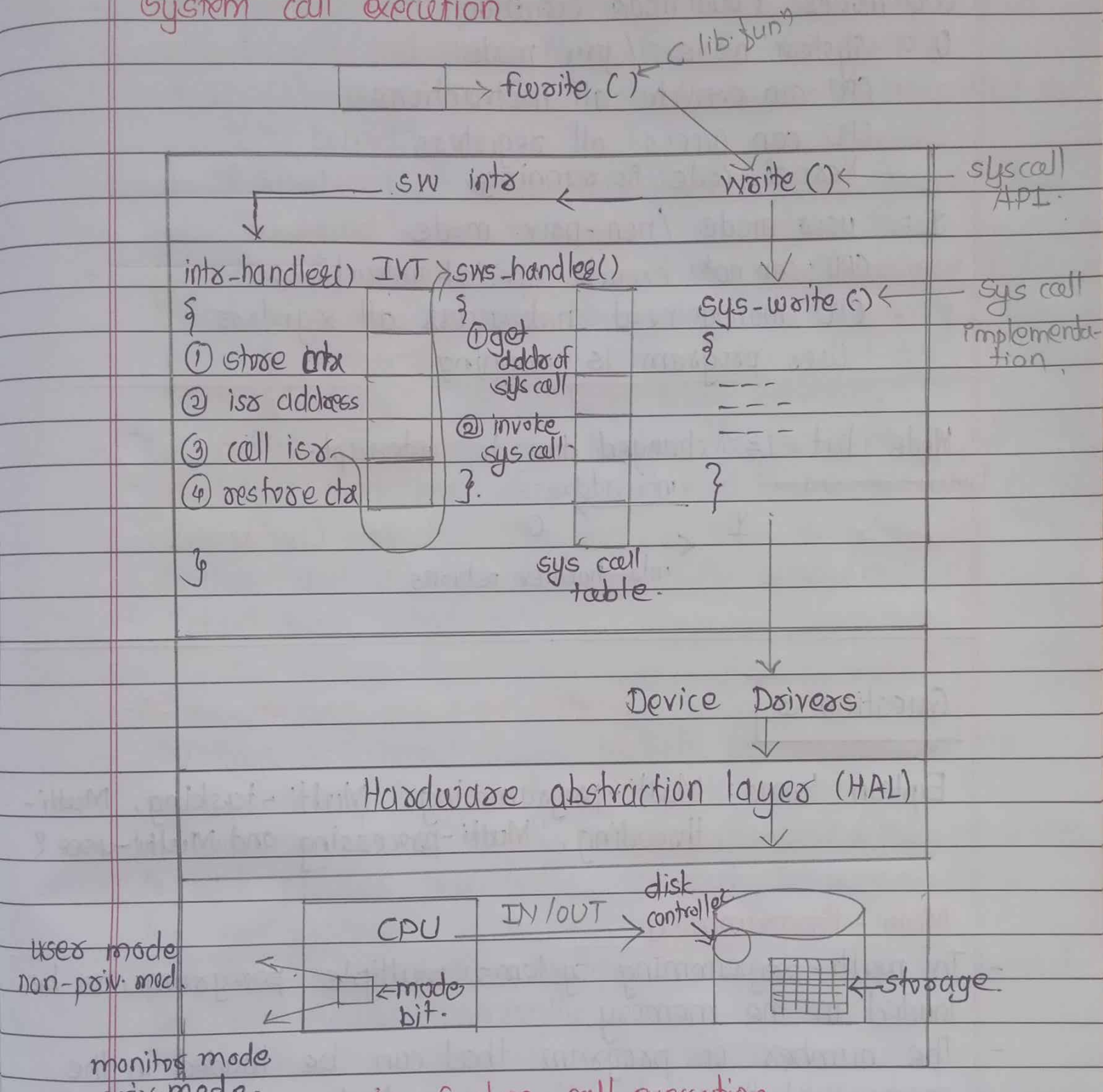


fig. System call execution

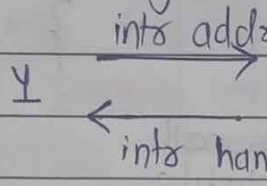
PAGE:
DATE: / /

CPU modes (Dual mode execution)

0 : System mode / priv mode.
CPU can execute all instructions.
CPU can access all registers.
Kernal code is running.

1 : user mode / non-priv. mode
CPU can not execute special instructions
CPU management not access all registers.
User program is running.

Mode bit is changed due to interrupt.



Question 3

Explain terms: Multi-programming, Multi-tasking, Multi-threading, Multi-processing and Multi-user?

Multi-Programming

- In multi-programming systems, multiple program can be loaded in the memory.
- The number of program that can be loaded in the memory at the same time, is called as "degree of multi-programming".
- In these systems, if one of the process is performing IO, CPU can continue execution of another program. This will increase CPU utilization.

PAGE:
DATE: / /

- Each process will spend some time for CPU computation (CPU burst) and some time for IO (IO burst).
  * If CPU burst > IO burst, then process is called as "CPU bound".
  * If IO burst > CPU burst, then process is called as "IO bound".
- To efficiently utilize CPU, a good mix of CPU bound & IO bound processes should be loaded into memory. This task is performed by an unit of OS called as "Job scheduler" OR "Long term scheduler".
- In multiple programs are loaded into the RAM by job schedwer, then one of process need to be executed (dispatched) on the CPU. This selection is done by another unit of OS called as "CPU schedulee" OR "short term scheduler".

Multi-tasking or time sharing.

- CPU time is shared among multiple processes in the main memory is called as "multi-tasking".
- In such system, a small amount of CPU time is given to each process repeatedly, so that response time for any process < 1 sec.
- With this mechanism, multiple tasks (ready for execution) can execute concurrently.
- There are two types of multi-tasking:
- Process based multitasking: Multiple independent processes are executing concurrently. Processes running on multiple processors called as "multi-processing".
- Thread based multi-tasking or multi-threading ⇒ Multiple parts/ functions in a process are executing concurrently

PAGE:
DATE: / /

Multi-users

- Multiple users can execute multiple tasks concurrently on the same systems, e.g. IBM 360, UNIX ; Windows Servers, etc.
- Each user can access system via different terminal.
- There are many UNIX commands to track users and terminals.
  * tty, who, who am i, whoami, w.

Question 4

Explain Linux kernel design (monolithic or modular)

Monolithic Kernel

1) Multiple kernel source files are compiled into single kernel binary image. Such kernels are "mono-lithic" kernels.

2) Since all functionalities present in single binary image, execution is faster.

3) If any functionality fails at runtime, entire kernel may crash.

4) Any modification in any component of OS, needs recompilation of the entire OS.

5) Examples : BSD Unix, Windows (ntoskrnl.exe), linux (vmlinuz), etc.

PAGE:
DATE: / /

Modular Kernel :-

1) Dynamically loadable modules. (e.g. dll/.so files) are loaded into calling process at runtime.

2) In modular systems, kernel has minimal functionalities are rest of the functionalities are implemented as dynamically loadble modules.

3) These modules get loaded into the kernel whenever they are called.

4) As single kernel process is running, no need of IPC for the execution and thus improves performance of the system.

5) Example : Windows, Linux, etc.

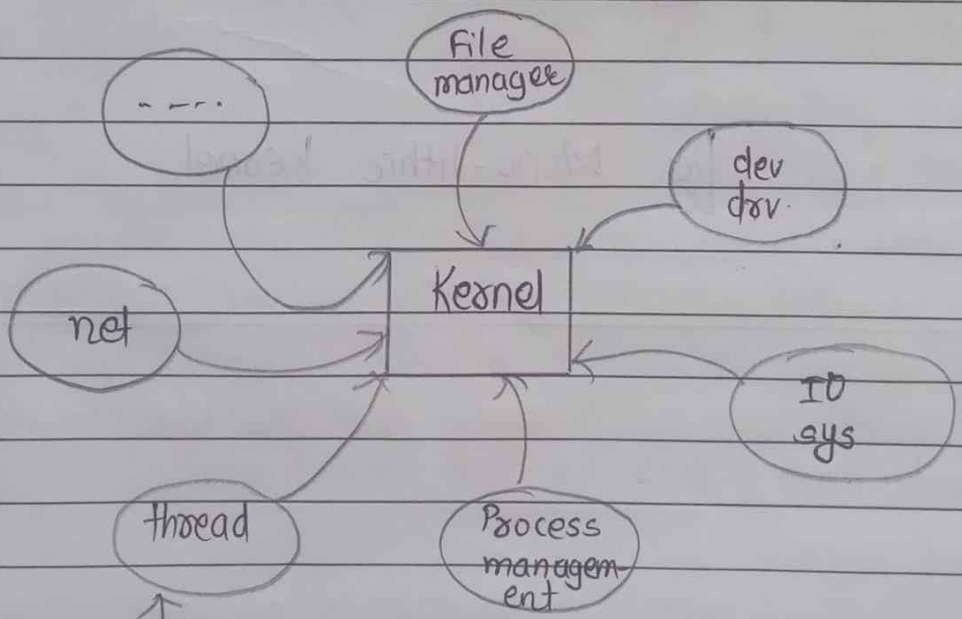


fig. modular kernel.

PAGE:
DATE: / /

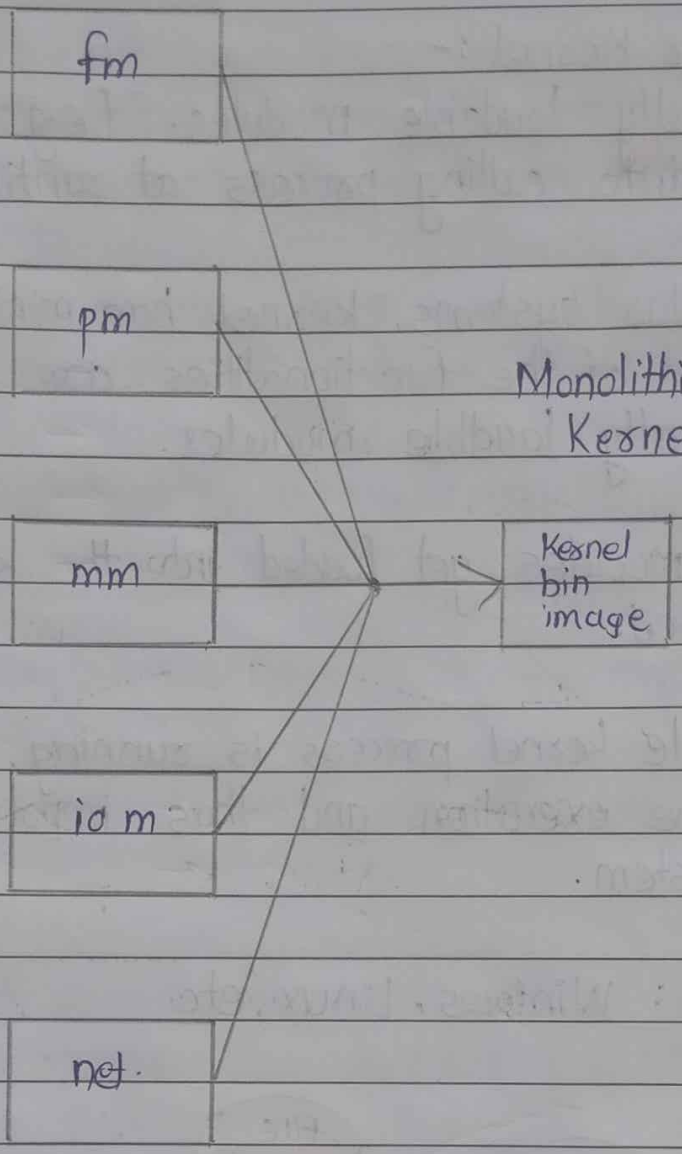


fig. Mono-lithic kernel.

PAGE:
DATE: / /

Question 5

Explain process life cycle.

Process life Cycle

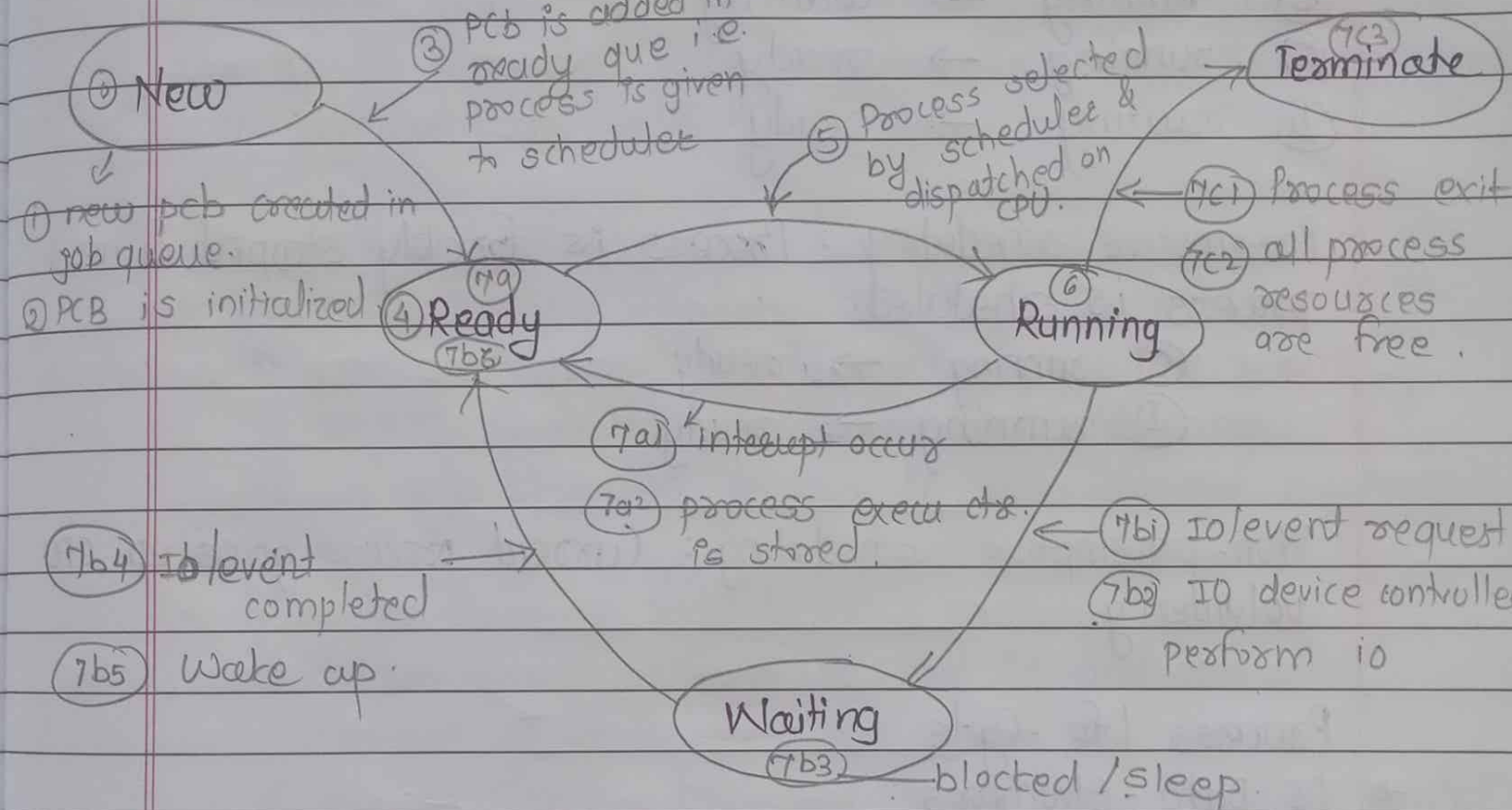


fig. Process life cycle

① Job queue / Process table / process list
- master list of all running processes.
- UNIX - Array of PCB (u area)
- Linux - Doubly linked list of PCB (task_struct)

② Ready queue / Run queue → Linux
- PCBs of all processes ready for execution on CPU.
CPU scheduler always pick next process to be executed from the ready queue.
- on SMP, one runque for each CPU.

PAGE:
DATE: / /

EI_ Megha Lohar_ 49398

③ Waiting queue
- Process waiting / blocked for IO or event.
- Wait que is for each IO device & sync. object.

CPU scheduler is executed in following situations.
(A) running → terminated
(B) running → waiting
(C) running → ready
(D) waiting → ready

Preemptive scheduling : Process is forcibly stopped & new process is scheduled.
(C) running → ready
(D) running → ready.

Non-preemptive scheduling : Current process gives up CPU voluntarily.

Process Life-Cycle.

* OS Data-Structures :
- Job queue / Process table :
PCBs of all processes in the system are maintained here
- Ready queue :
PCBs of all processes ready for the CPU execution & kept here.
- Waiting queue :
Each IO device is associated with its waiting queue & processes waiting for that IO device will be kept in that queue.

* Process states : New, Ready, Running, Waiting, Terminated

El-Meghd Lohar - 49398

Question 6

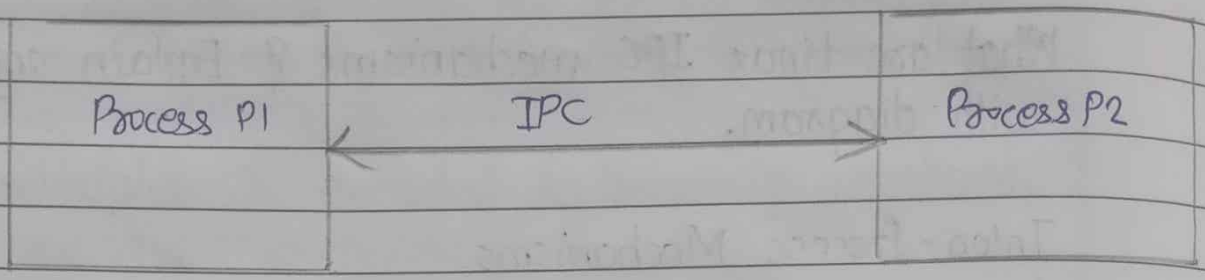
What are Linux IPC mechanisms ? Explain any three with diagram.

Inter-Process Mechanisms.
A process cannot access of memory of another process directly. OS provides IPC mechanisms so that processes can communicate with each other.

IPC models
- Shared memory model.
  1) Processes write / read from the memory region accessible to both the processes.
  2) OS only provides access to the shared memory region.
- Message passing model.
  1) Process send message to the OS and the other process receives message from the OS.
  2) This is slower compared to shared memory model.

Linux IPC mechanisms
Signals
Shared memory
Message queue
Pipe
Socket.

PAGE:
DATE: / /



Shared Memory

- Shared memory is the memory that can be simultaneously accessed by multiple processess.
- This is done so that the processes can communicate with each other

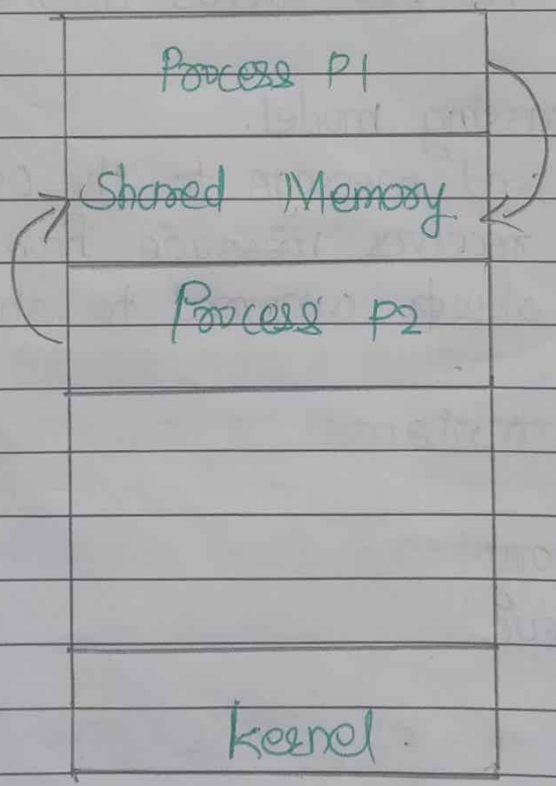


fig. Shared Memory

PAGE:
DATE: / /

Message Queue.

- Multiple processes can read & write data to the message queue without being connected to each other.
- Message are sorted in the queue until their recipient retrieven them.
- Message queues are quite useful for interprocess communication & are used by most operating system.

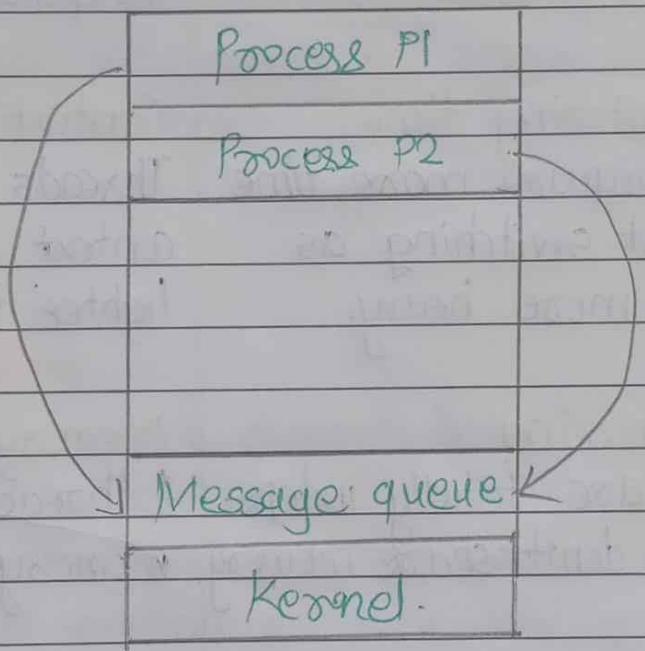


fig. Message queue.

Signal.

- Signals are useful in interprocess communication in a limited way.
- They are system messages that are sent from one process to another.
- Normally, signals are not used to transfer data but are used for remote commands betⁿ processes.

E1 - Megha Lohar - 49398

PAGE:
DATE: / /

Question 7

What is difference between process and thread? How can you create them in Linux program?

Process	Thread
Definition	**Definition**
A process is a program under execution i.e. an active program.	A thread is a lightweight process that can be managed independently by a scheduler.
Context switching time.	**Context switching time.**
Processes require more time for context switching as they are more heavy.	Threads require less time for context switching as they are lighter than processes.
Memory sharing	**Memory sharing**
Processes are totally independent and don't share memory	A thread may share some memory with its peer threads.
Communication	**Communication.**
Communication bet^n processes requires more time than between threads.	Communication bet^n threads requires less time than bet^n processes.
Blocked.	**Blocked.**
If a process gets blocked, remaining processes can continue execution.	If a user level thread gets blocked, all of its peer threads also get blocked.

E1_ Megha Lohar - 49398

PAGE:
DATE: / /

Process	Thread.
Time for creation Processes requires more time for creation.	Time for creation. Threads require less time for creation.
Time for termination Processes require more time for termination.	Time for termination. Threads require less time for termination.
Data and code sharing Processes have independent data & code segments.	Data and code sharing. A thread shares the data segment, code segment files, etc. with its peer thread.

Process creation.

A new process can be created by the fork() system call. The new process consists of a copy of the address space of the original process. fork() creates new process from existing process. Existing process is called the parent process and the process is created newly is called child process.

Thread creation.

The first argument is a pthread_t type address. Once the function is called successfully, the variable whose address is passed as first argument will hold the thread ID of the newly created thread. The second argument may contain certain attributes which we want the new thread to contain.

PAGE:
DATE: / /

Question 8

What is difference between semaphore and mutex? How can you create them in linux program?

Semaphore	Mutex
Mechanism It is a type of signaling mechanism.	Mechanism It is a locking mechanism.
Data type Semaphore is an integer variable.	Data type Mutex is just an object.
Modification The wait and signal operations can modify a semaphore.	Modification It is modified only by the process that may request or release a resource.
Thread You can have multiple program threads.	Thread You can have multiple program threads in mutex but not simultaneously.
Types Types of semaphore are counting semaphore and binary semaphore.	Types. Mutex has no subtypes.

El- Megha Lohar - 49398

PAGE:
DATE: / /

Semaphore	Mutex
Operation Semaphore value is modified using wait() & signal() operation.	Operation Mutex object is locked or unlocked
Resource Occupancy It is occupied if all resources are being used and the process requesting for resource performs wait() operation and blocks itself until semaphore count becomes > 1.	Resource Occupancy. In case if the object is already locked, the process requesting resources waits and is queued by the system before lock is released.

- Semaphore is initialized by sem_init() system call
  Synopsis of sem_init():

```
#include <semaphore.h>
int sem_init (sem_t *semaphore, int pshared,
              unsigned int value);
```

- Mutex is special mechanism to help create concurrency in program. Treat mutex as a lock, there are two fun^n pthread_mutex_lock (mutex): If the mutex is unlocked.

PAGE:
DATE: / /

Question 9

What is file ? Which system calls are used to manipulate files in Linux ? Write their prototypes and explain.

* File => A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic types & optical disks.

-In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator & user.

* File management is system calls are responsible for device manipulation such as creating a file, reading a file, writing into a file etc.

* File System syscalls

chdir()	symlink()
mkdir()	chmod()
link()	chown()
unlink()	

chdir() => chdir is change directory.

* chdir command is a system fun^n which is used to change the current working directory.

mkdir() => make directory.

* 'mkdir' : command in linux allows the user to create directories. This command can create multiple directories at once as well as set the permissions for the directories.

link()

* The UNIX s|m file structure allows more than one named reference to a given file, a feature called 'aliasing'.

PAGE:
DATE: / /

Question 10

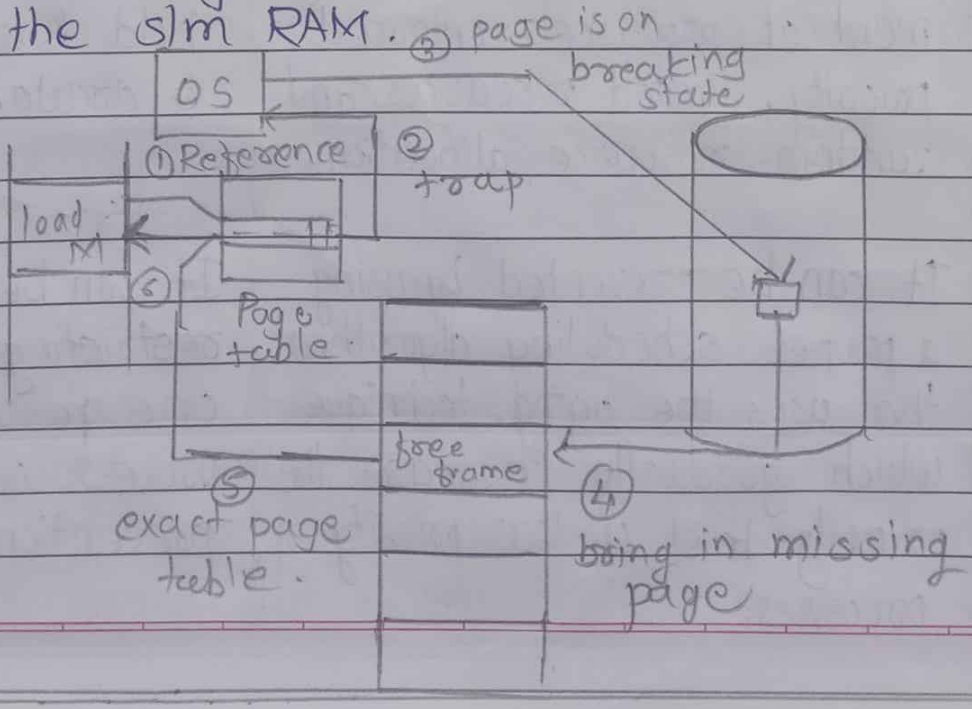
What is paging ? What is page fault and how it is handled ?

Paging

- Paging is a memory management scheme that eliminate the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non-contiguous.
- The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device & this mapping is known as paging technique.

Page fault -

- A page fault occurs when a program attempts to access a block of memory that is not stored in the physical memory or RAM.
- The fault notifies the operating system that it must locate the data in virtual memory, then transfer it from the storage devices, such as an HDD or SSD to the s/m RAM.



EL_Megha Lohar_49398

PAGE:
DATE: / /

Question 11

What is difference between starvation and deadlock?

Starvation	Deadlock
It is a condition which occurs when low priority processes are denied access to resource that are reserved for high priority processes.	It is a condition which arieses when one or more threads or processes are trying to access the same resource resulting in a deadlock.
It is triggered by a deadlock which eventually pushes a process off to an indefinite postponement period.	A set of processes are blocked and go into a waiting state, because a requested resource is being held by another waiting process.
Caused due to poor scheduling algorithm, uncontrolled management of resources, enforced priority, limited resources, and random resource allocation.	Arise when four conditions hold true: mutual exclusion, hold & wait, no preemption, & circular wait.
It can be prevented by using a proper scheduling algorithm that uses the aging technique which gradually increases the priority level of low priority processes.	It can be prevented by restricting access to only one resource by a single process or by deadlock prevention & recovery.

PAGE:
DATE: / /

Question 12

How vfork() system call works ? What is copy-on-write ?

vfork()

- Vfork() is also system call which is used to create new process. New process created by vfork() system call is called child process & process that invoked vfork() system call is called parent process.
- Code of child process is same as code of its parent process.
- Child process suspends execution of parent process until child process completes its execution as both processes share the same address space.

Copy-on-write.

* Copy on write or simply COW is a resource management technique.
* One of its main use is in the implementation of the fork system call in which it shares the virtual memory (pages) of the OS.
* In UNIX like OS, fork() system call creates a duplicate process of the parent process which is called as the child process.
* The idea behind a copy-on-write is that when a parent process creates a child process then both of these processes initially will share the same pages in memory and these shared pages will be marked as copy-on-write which means that if any of these processes will

PAGE:
DATE: / /

try to modify the shared pages then only a copy of these pages will be created and the modifications will be done on the copy of pages by that process and thus not affecting the other process.

* There will be no wastage of address space and it is the efficient way to create a process. vfork does not use copy-on-write

Question 13.

Explain OS booting. Explain Linux booting.

OS Booting

The operating system is loaded through a bootstrapping process, more succinctly known as booting. A boot loader is a program whose task is to load a bigger program, such as the operating system. When you turn on a computer, its memory is usually uninitialized. Hence, there is nothing to run.

Linux booting

Booting a Linux system involves different components & tasks. The hardware itself is initialized by the BIOS or the UEFI, which starts the kernel by means of a boot loader. After this point, the boot process is completely controlled by the operating system & handled by systemd.

E1_Megha Lohar-49398

PAGE:
DATE: / /

Question 14

How many Linux run-levels are there? Which features are enabled in each runlevel?

- There are seven runlevels exist, numbered from zero to six. After the Linux kernel has booted, the init program reads the /etc/inittab file to determine the behavior for each runlevel.

- A run level is a state of init and the whole system that defines what system services are operating.
- Run levels are identified by numbers.
- Some system administrators use run levels to define which subsystems are working.
  e.g. whether X is running, whether the network is operational, and so on.
- Whenever a LINUX system boots, firstly the init process is started which is actually responsible for running other start script which mainly involves initialization of you hardware, bringing up the network, starting the graphical interface.
- Now, the init first finds the default runlevel of the system so that it could run the start scripts corresponding to the default run level.
- A runlevel can simply be thought of as the state your system enters like if a system is in a single-user mode it will have a runlevel 1 while if the slm is in a multi-user mode it will have a runlevel 5.

El-Megha Lohar-49398

- A runlevel in other words can be defined as a preset single digit integer, for defining the operating state of your LINUX or UNIX based operating system. Each runlevel designates a different system configuration & allows access to different combination of processes.

Seven runlevels are supported in the standard. Linux Kernel
0- System halt; no activity. the system can be safely powered down.
1- Single user, rarely-used.
2- Multiple users, no NFS (network filesystem); also used rarely.
3- Multiple users, command line interface; the ~~standard~~ standard runlevel for most Linux-based server hardware.
4- User definable
5- Multiple users, GUI, the standard runlevel for most linux-based desktop systems.
6- Reboot; used when restarting the system.

By default Linux boots either to runlevel 3 or to the runlevel 5.

Question 15

Explain regex commands and wildcard characters.

Regular Expression
- Find a pattern in text files.
- Pattern is given using regex ~~com~~ wild-card characters.

El-Megha Lohar-49398

PAGE:
DATE: / /

Basic wild card characters :
$ - find at the end of line.
^ - find at the start of line.
[ ] - any single char in give range or set of chars.
[^] - any single char not in give range or set of chars
. - any single character.
* - zero or more occurrences of previous character.

Extended wild-card characters :
? - zero or one occurrence of previous character.
+ - one or more occurrence of previous character.
{n} - n occurrences of previous character.
{,n} - max n occurrences of previous character.
{m,} - min m occurrences of previous character.
{m,n} - min m & max n occurrences of previous character.
() - grouping (chars)
(|) - find one of the group of characters.

regex commands.
grep - GNU regular expression parser - Basic wild-card
egrep - Extended Grep + Basic + Extended wild-card
fgrep - Fixed Grep - No wild-card.

Command syntax.
grep "pattern" filepath
grep [options] "pattern" filepath
-c ⇒ count no. of occurrences
-v ⇒ invert the find o/p
-i ⇒ case insensitive search
-w ⇒ search whole words only
-R ⇒ search recursively in a directory
-n ⇒ show line number.

PAGE:
DATE: / /

Question 16

Which Linux command is used to kill all running instances of the same program.

⇒ - The killall command is used to kill processes by name.
- The killall command can kill multiple processes with a ~~signal~~ single command.
- By default, it will send a SIGTERM signal.

Question 17

How security is implemented in linux file systems? Tell commands related to it.

Linux File System:
- The linux security model is based on the one used on UNIX systems, and is as rigid as the UNIX security model, which is already quite robust.
- On a Linux system, every file is owned by a user & a group user.
- For easy use with commands, both access rights or modes and user groups have a code.

Commands:
ls -l
chmod.

PAGE:
DATE: / /

Question 18

How redirection and pipe in linux commands work?

Redirection

- Redirection is a feature in linux such that when executing a command, you can change the standard I/P & O/P devices.
- The basic workflow of any linux command is that it takes an input and gives an output.
- The standard input (stdin) device is the keyboard.
- The standard output (stdout) device is the screen.

O/P redirection

The '>' symbol is used for O/P (STDOUT) redirection.

I/P redirection

The '<' symbol is used for input (STDIN) redirection.

Error redirection

- Error redirection is one of the very popular features of unix/linux.
- The '2>' symbol is used for error (STDERR) redirection.

Pipe

The pipe is a command in linux that lets you use two or more commands such that output of one command serves as input to the next. In short, the O/P of

PAGE:
DATE: / /

each process directly as input to the next one like a pipeline. The symbol '|' denotes a pipe.