Holocraft Sticker

AP/20A 095398

Barcode Sticker Paste here

Examination	MITWPU Endterm December 2022
Day & Date	Monday, 26th December 2022
Subject	CET2003B – Operating Systems
Paper No.	CET2003B | Sec | –
Medium of Answer	English
Seat No. : In Figure & in Words	1 0 3 2 2 1 0 8 8 9
	One zero three two two
	One zero eight eight eight
Signature of Candidate	
Signature of Jr. Supervisor	
Center No.	1 0 3

Holocraft Sticker

MIT-WPU
॥ विद्यानिधिर्ददाति ॥

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Instruction to Candidate :
1. Candidate has to confirm seat number, subject and centre number printed on Bar code and write it on attendance sheet.
2. Paste Bar Code in prescribed space.
3. Do not write anything on bar code sticker, otherwise it will be treated as unfair means.

Supplements Attached

Main Answer Booklet	No. of Supplement(s)	Total
1	+	=

Specific remarks of ACoE / Dean

Total Marks	Total Marks in Words	Sign

Q.No.	Evaluer					Moderator					Revaluer				
	A	B	C	D	Total	A	B	C	D	Total	A	B	C	D	Total
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
Total															

Q.No.

Operating Systems

Q. 1. The operating system is a computer is a complex piece of software that interacts with the hardware for us.

→ Naturally, it has evolved over the years and people have found several ways to create and interact with this software. These methods are what define the architecture of the OS.

→ Broadly speaking: It is of 3 types

1. Layered
2. Microkernel
3. Monolithic

(*) The Monolithic Architecture:

→ This is one of the primitive and simple OS architectures

→ Everything important and key to hardware communication is part of the kernel itself

→ It is difficult to program and manage

→ Only the user applications and utilities run in the user space. Rest of the services like system processes, IPC, modules, system libraries, I/O buffers, device drivers etc run in kernel mode.

Q.No.

(*)

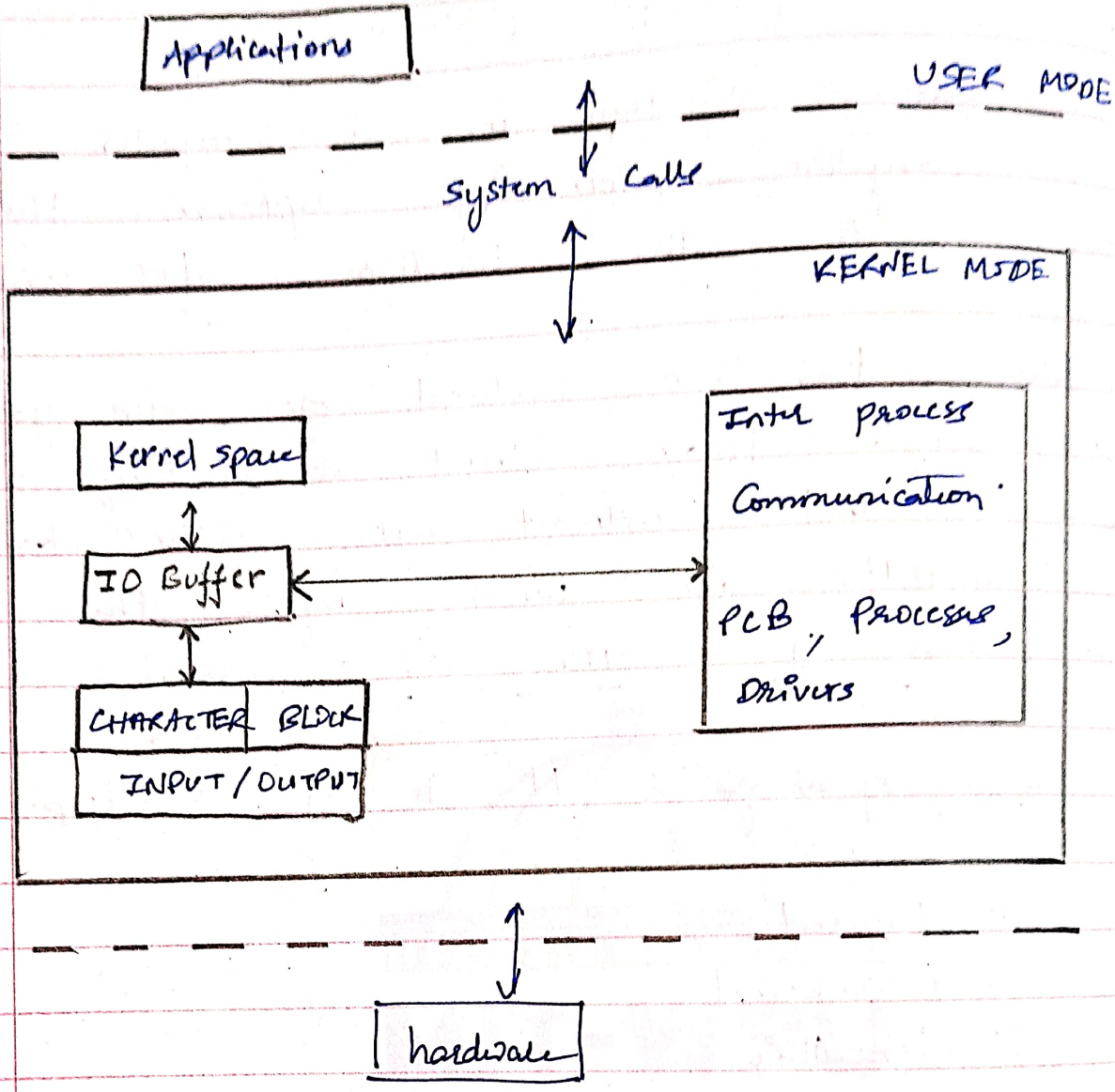


figure: Monolithic Kernel Architecture

(*)

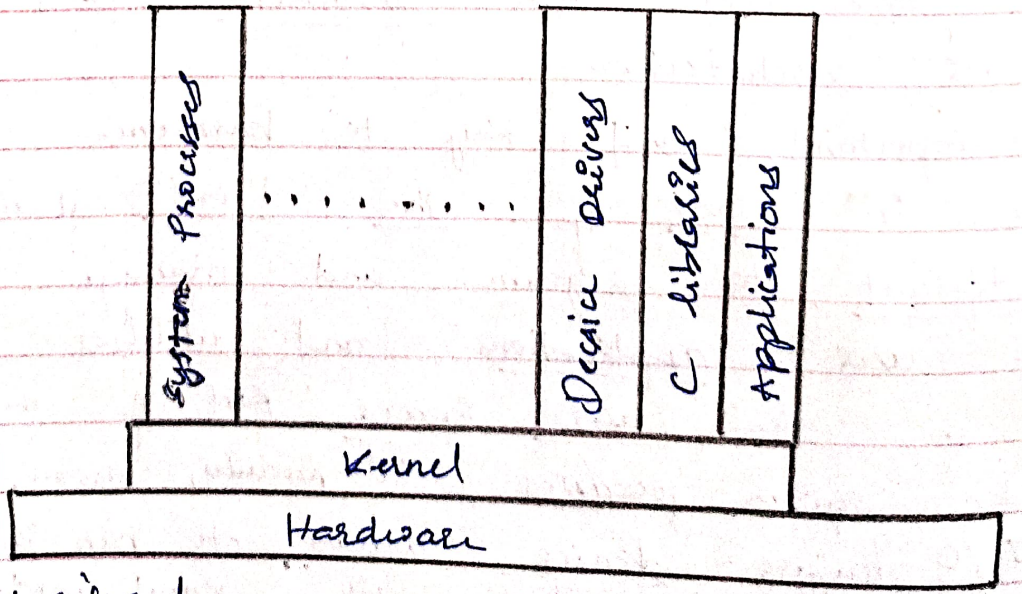


fig: Microkernel Architecture

Q.No.

Microkernel :

→ Only the base interfaces with the hardware is built into the kernel.
→ Most other tasks are performed in user space
→ Extremely Modular
→ Supports more devices due to its easy modular structure
→ eg: Linux, MacOS

Q.2. Inter - Process Communication

→ It is an essential function of the operating system that ensures safe sharing of data, and resources between several concurrently running processes in the OS.

→ A process is simply a program in execution. Several such programs may need to communicate with each other due to reliance on shared resources or functionality. This is achieved safely by using pipes.

→ In C, pipes are defined in <sys/types.h> and <unistd.h> header files.

Q.No.

(*) Pipes act as a simple queue data structure. They are stored in primary memory as files, temporarily located in the /proc directory.

(*) They only allow reading or writing at one time.

(*)

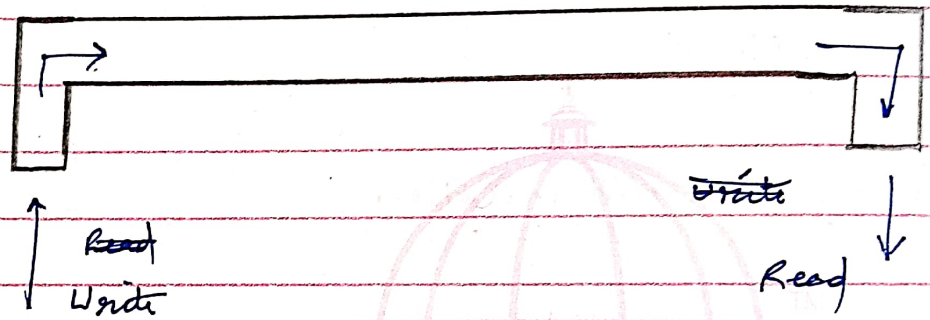
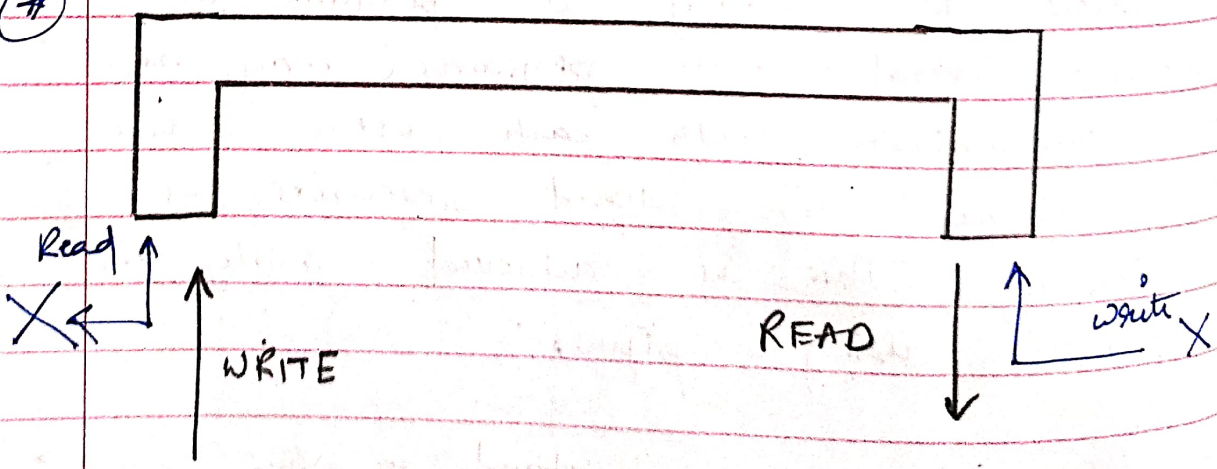


fig. → Pipe is use

(*)



Read & write Both are not allowed at the same time.

Q.No.

pipe (fildes [2]) usually requires the file descriptor of the writer and the reader. Only after the writer closes its file descriptor, can the reader read its written contents.

(*) Reading and writing together are not allowed because of the possibility of race - conditions.

(*) Race conditions arise due to improper management or miscommunication between 2 or more processes or threads accessing the same shared resource.

(*) The usual mechanisms to access shared memory is for the shared resource to be brought into main memory, and have the CPU follow instructions of read / write from processes trying to access that memory.

(*) This usually happens safely without deadlock, circular wait, or race conditions arising, if the processes co-operate with each other rather than compete.

Q.No. (3) Semaphores: Semaphores are a technique to get around the problem of safely accessing shared resources by process that are in their critical sections employing mutual exclusion. So it is a solution to the mutual exclusion problem:

The problem:

Say P_1 tries to modify a shared variable a

$a = 1$

P_1 : $a += 3$

Say P_2 : $a += (-4)$

If P_3 : read (a)

The value read by a is now dependent on which processes executes first, and how many times. Accuracy is not guaranteed. To avoid this Semaphores are used to give time to let P_3 read. This is done using wait() and signal() mechanism.

Q.No.

```
sem_t  s;
wait ()
{
    while (s<0)
    {   s--;
    }
}

signal ()
{   while (s>0)
    {   s++;
    }
}
```

wait and signal are used to block and release a variable s, that allows other processes information about when to access the shared resource.

eg.

```
Reader ()
{
    sem_wait (& mutex)
    // Read
    sem_signal (& mutex)
}
```

```
writer ()
{
sem_wait (& write)
sem_wait (& mutex)
// write (a)
sem_signal (& mutex)
sem_wait (& write)
}
```

Q.No. (4) Banker's Algorithm

	Max			Allocation			Need		
	R_0	R_1	R_2	R_0	R_1	R_2	R_0	R_1	R_2
P_0	4	1	2	1	0	2	3	1	0
P_1	1	5	1	0	3	1	1	2	0
P_2	1	2	3	1	0	2	0	2	1

Available (work)

R_0	R_1	R_2
2	2	0

Total R_0 = 2 + 1 + 1
= 4

R_1 = 3 + 2 = 5

R_2 = 2 + 1 + 2 + 0 = 5

Case - 1 : as [2, 2, 0] is available, but P_0 requires [3, 1, 0], P_0 will not be given any more resources.

Case - 2 : As [2, 2, 0] is available, P_1 needs [1, 2, 0] resources will be allocated to P_1 ;

Available would then be [1, 0, 0];

Q.No.

work
R0 R1 R2
2 2 0
1 1 0
2 5 1
2 3 0
3 5 3
0 4 3
4 5 5

After P_1 finishes execution, [4, 5, 1] will be returned to OS.

So [1, 0, 0] + [1, 5, 1]

= [2, 5, 1] as the new work ~~allocation~~ matrix.

Case : 3 : P_2 will be given resources [0, 2, 1] from available [2, 5, 1]

leaving us [2, 3, 0] in ~~atta~~ work matrix.

case : 4 : After ~~to~~ P_2 finishes execution

work = [2, 3, 0] + [1, 2, 3]
= [3, 5, 3]

Case : 5 : P_0 is now given resources

So [3, 5, 3] − [3, 1, 0]

= [0, 4, 3]

and at last [0, 4, 3] + [4, 1, 2]
= [4, 5, 5] = total Resources

Safe sequence : ~~P_1, P_0, P_2~~ $\boxed{P_1, P_2, P_0}$

Q.No. (5) To assign memory to load progs into primary memory, it has to be given partitions in order to accomodate several programs at the same time.

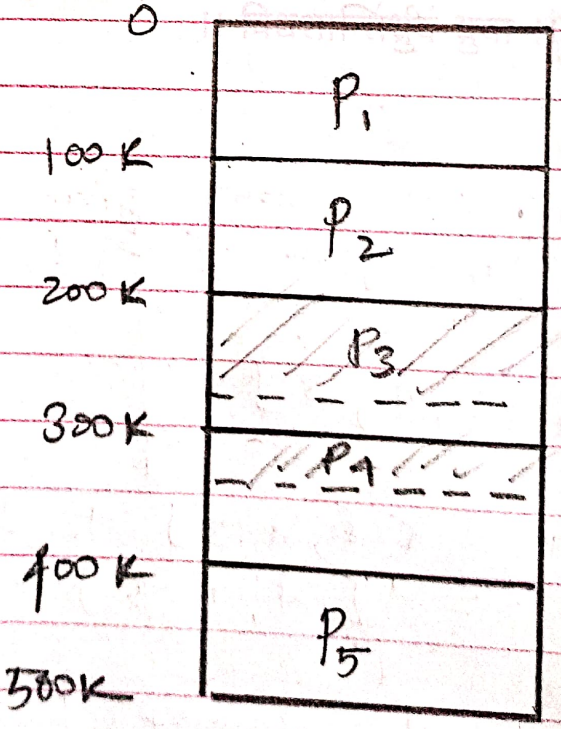
This can be done in 2 ways:

1. fixed
2. Dynamic

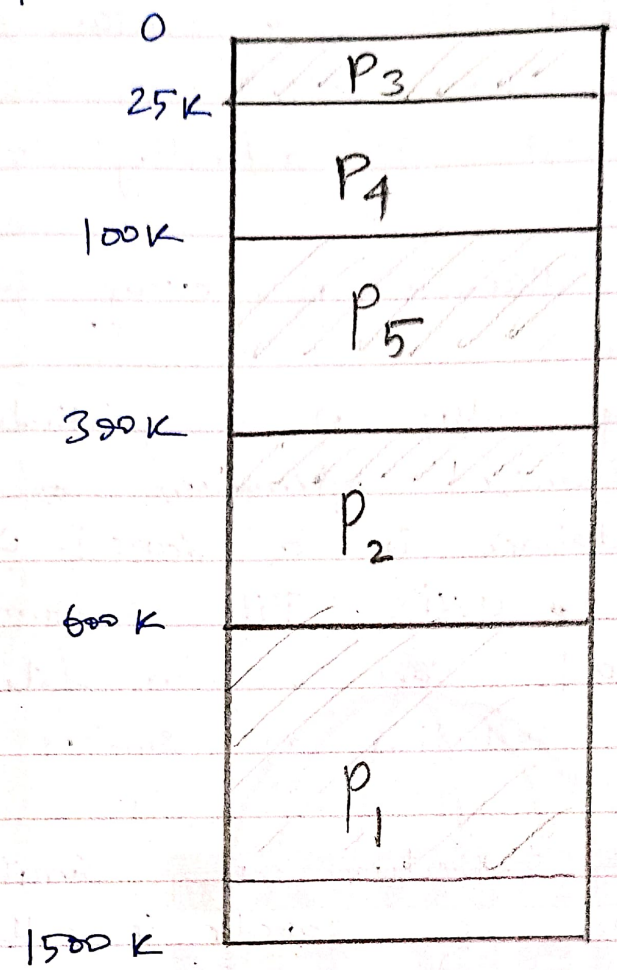
Fixed partitioning is of 2 types

1. Equal Partitions
2. Unequal Partitions.

1. Equal: Diagram.



Q.No. 2. Unequal : Diagram



Equal	Unequal
1. Less CPU overhead	More overhead
2. Takes more time as programs have to be removed and reordered	Takes less time as programs dont need to be moved so often
3. Less waste of space	Space may be wasted for smaller programs
4. Divides available memory into equal partitions	Divides into unequal partitions
5. ~~Lower~~ Process Fragmentation Higher	~~High~~ Lower Fragmentation

Q.No. 6

1. Field : A field is an entry of data. This is the portion of the file where data is actually stored.

It is an attribute of every file.

2. Record : Any entry of a particular table of organized information within a database is a record. With respect to a UNix File System, a record can be a file, or even a collection of smaller files.

3. Database : A collection of similarly associated files or records is called a database. It often contains many files, records, data fields, and sometimes even directories stored on the file system.

4. Address Information : It can refer to the pathname of each record or field within a database. With Respect to UNIX structure however, it can mean information about which frame in the page table the particular file is stored.

Q.No. 5. Access Control Information : Every file in the file system (NTFS, ext4 etc) has security parameters associated with it.

It is only accessible to those users who created it or the root user. Every file can be accessed, read, written or executed based only on its access control Information

eg. 7 7 7 file.txt

↓

111 111 111 for each group, user and root.

rwx rwx rwx

Q.No. (7) Characteristics of UNIX File System.

1. All entities, data, commands, drivers, devices and libraries are files in UNIX. — everything is a file

2. All files have access control permissions associated with them. They can be read, written or executed only by those users with permissions.

3. Unix supports ext4, exFAT, FAT, ext3, NTFS, FAT32 ~~et~~ BtrFS, ZFS etc — all kinds of file systems.

4. The usual directories in UNIX are:
   1. /boot — for kernel and GRUB bootloader menu entries.
   2. /home — for users files like desktop, downloads etc.
   3. /opt — optional programs
   4. /bin — programs essential to as commands like ls, passwd
   5. /sbin — system binaries
   6. /usr — user downloaded program files
   7. /etc — configuration files
   8. /dev — devices, buffers.

Q.No.

5. Unix files are case sensitive.

downloads ≠ Downloads.

(4) Pathnames →

Address or path of a file (leaf node) in the tree.

(1) Absolute : irrespective to current directory

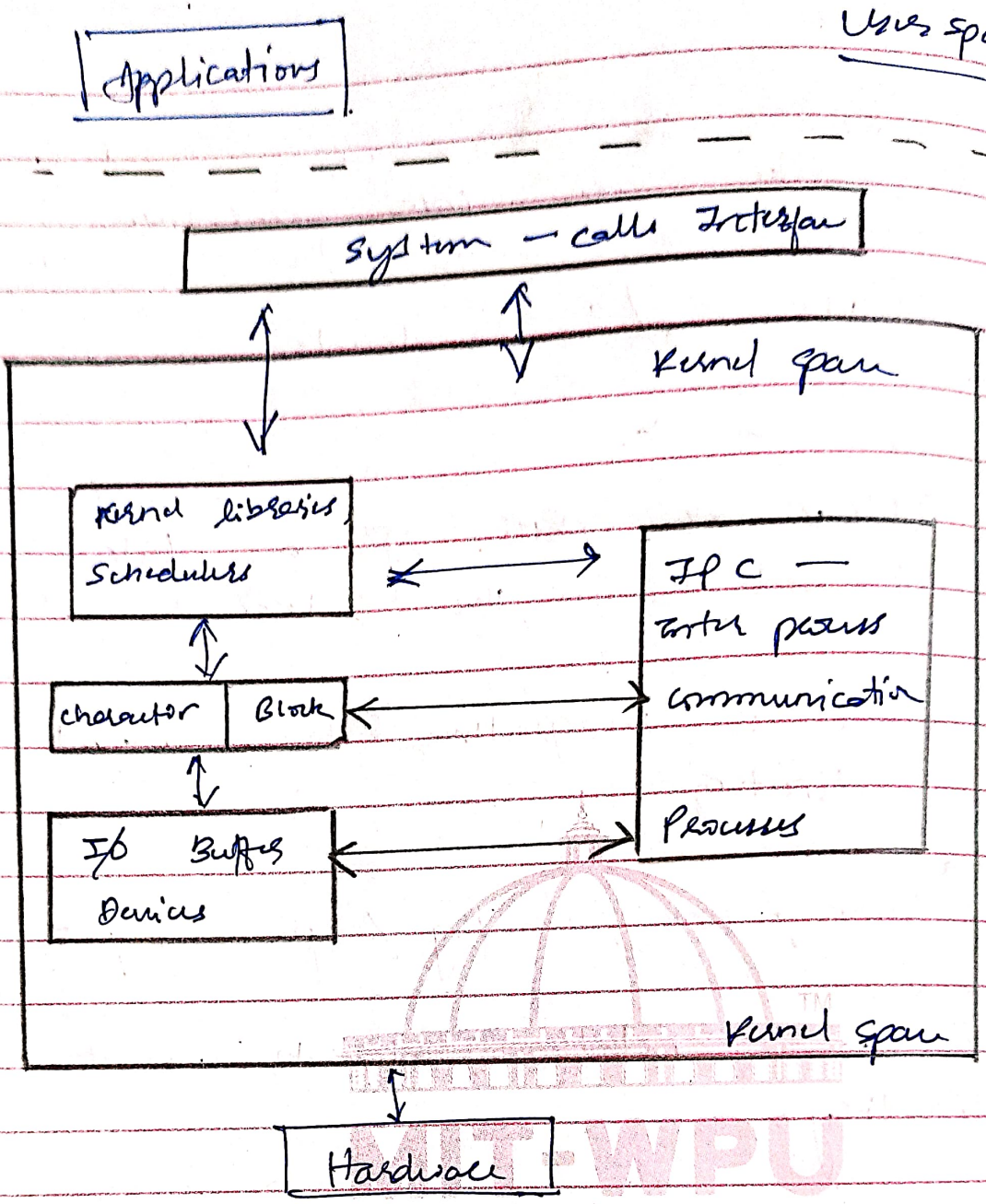
> "/home/ucs/.config/pacman/open.txt"

(2) Relative : Relative only to current directory :

eg. > ./a.out

> ./a etc.

Q.No.



UNIX Block diagram

— It consists of several blocks functio
concurrently with each other

— It is divided into user & span
and kernel span.

Q.No.

— The application layers are in user space.

— To communicate with Kernel or hardware, user programs have to perform system calls.

— The Kernel interacts with hardware using device drivers from the Kernel space

— UNIX follows a monolithic Kernel design architecture.

— System libraries in C, device drivers, I/O buffers are all parts of the Kernel space itself.

— The programs communicate within themselves using Kernel space is the process communication.