OS Assignment

Akshay Kumar
S20170010006

1) Is it a good process to have declarations in between C program?

Ans: For compilers greater than C89 standard, it doesn't matter. All declarations are compiled at one place during compilation process. However, if you want to maintain backwards compatibility, declaring all variables at the begining of the scope would be ideal.

2) Which syscall does fopen() call?

In linux/UNIX fopen() calls the open() function.

3) Example of code in C with open(); syscall.

```
# include <stdio.h>
# include <fcntl.h>
int main() {
    int fd = open("test.txt", O_RDONLY | O_CREAT);
    printf("fd= %d \n", fd);

    return 0;
}
```

4) Does the code in ISR have privileged instructions? If so, given an example. In no, Justify.

Yes, ISRs can have privileged instructions.

Example:

In case of on I/O request / I/O ready event, the ISR associated with it must interact with the I/O device controller directly.

5) Choose two shells and explain the difference in executing similar commands.

Bash vs C-Shells

	Bash	C-Shell
Default prompt	$	%
Home dir	$Home	$home
Variable assignment	var= value	Set var = value
Environmental Variables	export var= value	set env var value
Number of arguments	$#	$# argv
aliases	alias x='y'	alias x y
Conditionals	if [$-eq 5] if	if ($i == 5) endif

6) Write a short note on POSIX.

POSIX - Portable Operating System Interface.

→ family of standards specified by IEEE for maintaining compatibility between OSes.

→ Defines the API, CLI, shells, interfaces for compatibili with UNIX and other OSes.

→ Consist of documents → (POSIX 1, POSIX 2, etc)

→ CLI scripting based on UNIX-V.

→ Also defines a standard threading API.

7) When installing OS, why is choice of 32-bit or 64-bit is given, but not of the processor?

32 or 64 bits describe the architecture of processor.
A 64-bit system can handle large amounts of random access memory (RAM) more effectively than a 32-bit system.
A 64-bit system can run 32-bit software but not vice-versa.
Hence, while installing you get the choice of either 32 or 64-bit installation, as 64-bit processors can also run 32-bit OSes, but the processor inside it is fixed, Hence we don't get choice of processors.

8) Why are buffers passed in syscalls?

Buffers are used in syscalls, either to read from or to write to them.

For example: in read(), the data from file is read and placed into the buffer, from where we can access it.

In write(), the data to be written is stored in the buffer.

9) Why is stack pointer incremented when library call returns to user?

→ Stack pointer movement depends on direction of growth of stack.

→ If stack is an upward growing one means tasks are pushed, pointer increases, and when tasks finish, pointer is decremented.

→ However in downward growing stacks, stack pointer increments when task finish/return.

→ Hence if stack pointer increases after returning from library call, it must be a downward growing one.

10) What happens in a read() call?

→ read ssize_t read(int fd, void *buf, size_t count);
It returns number of bytes read from the specified file descriptor (fd) into the buffer (buf).

→ fd → file descriptor. read starts reading the data at fd, and keeps copying it to the buffer.

→ Once "count" number of bytes have been read, Process terminates.

11) Give 5 examples of premptive scheduling, Identify state of process in each.

1) Shortest Remaining time scheduling
→ Process with shorter remaining time prempts the one with larger ~~ss~~ time.

→ Running process may be preempted by a new process with shorter run time.

2). Round Robin Scheduling
→ Each process is switched after a quantum 'q'.

→ Here running process is preempted by other process in the queue.

3) Multilevel queue scheduling.

→ Multiple queues, each queue follows own algorithm.

→ Process preempted and moved to lower queue if it

a) takes longer than quantum

b) performs I/O operations

4) Rate Monotonic scheduling.

→ Process with smaller period preempts the other processes.

→ Running process is preempted if new process is initiated with a shorter period.

5) Earliest Deadline First (EDF)

→ Process whose deadline is earlier, preempts the other process

→ The earlier the deadline, the higher the priority.

12) Given deadline and period, until what point should we draw the schedule so as to ensure no deadline misses occurs.

Deadlines are periodic functions of time.
i.e They repeat with a certain time period.

→ The same applies to the pattern of the processes too.

→ If there are 'n' processes with their own periods, the period of resultant system would be the LCM of all periods.

→ If there is no deadline miss in one period of the whole process system, we can say that there will be no deadline misses in any subsequent periods (unless new process is added)

→ Hence LCM of all periods gives us the point until which we should draw the schedule.

~~Page~~

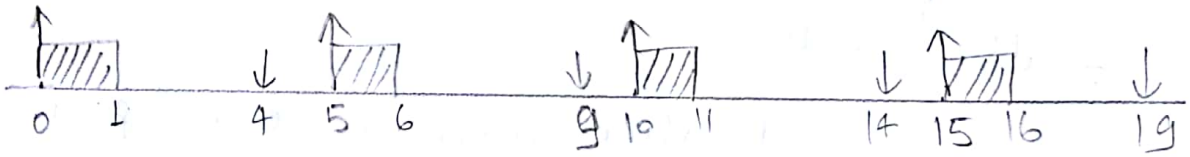
13

Problem-1

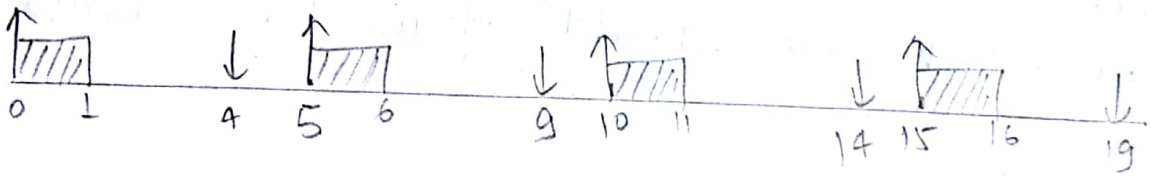
A task 'A' has a period 5ms and execution time of task 'A' is 1ms, deadline of task 'A' is 4ms. first instance of task 'A' arrives at 0ms.

↑ → instance of A
↓ → Deadline of previous instance.

RMS



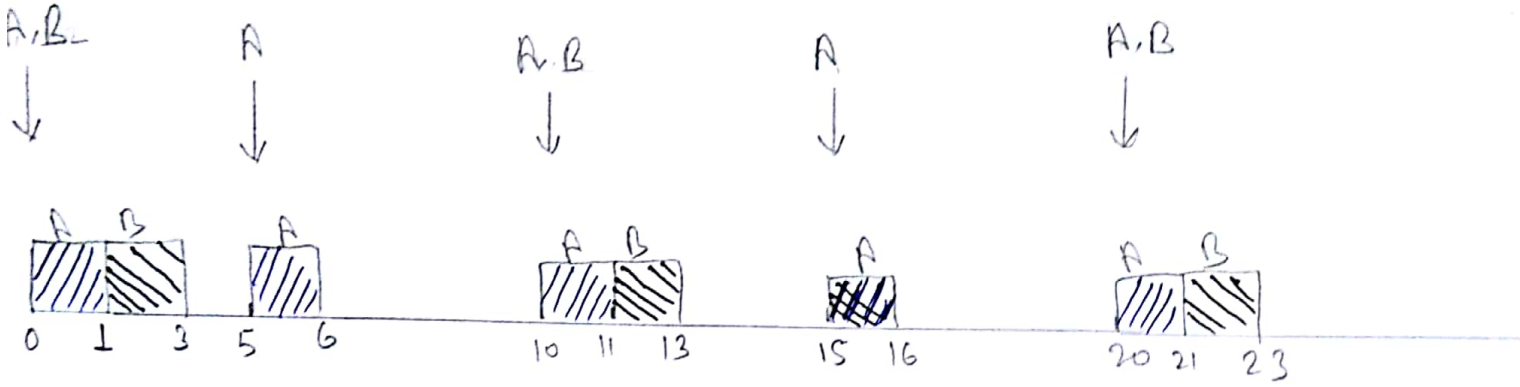
EDF



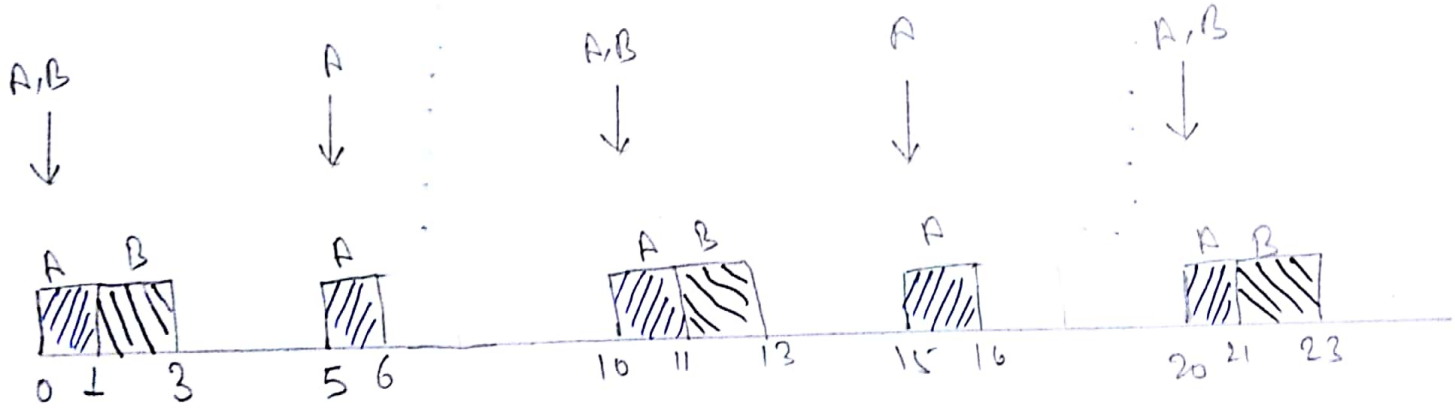
Problem. 2

Process A and B, period of A is 5, Period of B is 10 execution time of A is 1 and that of B is 2. Deadline is equal to period for both the processes.

RMS



EDF

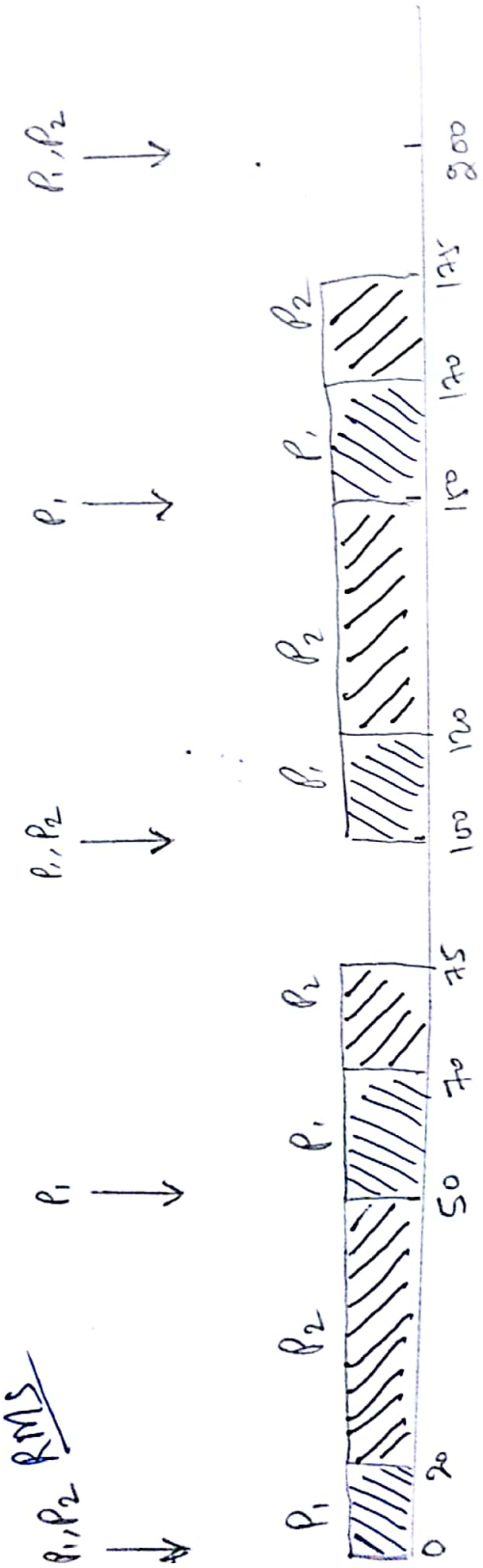


Problem 3

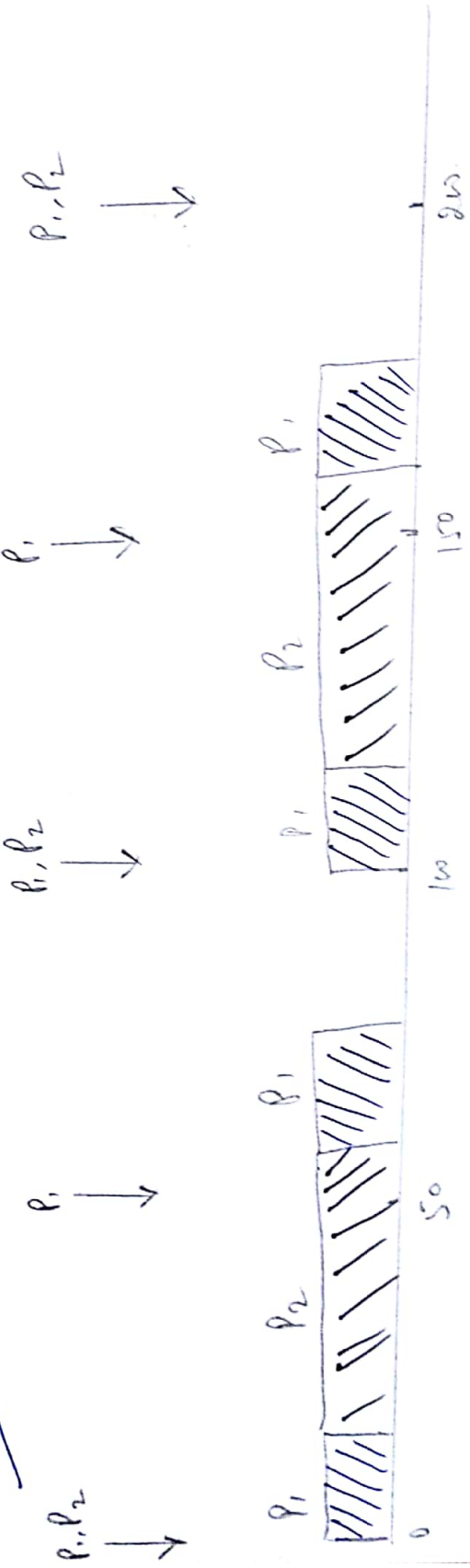
Process	Period	Execution time
P_1	50	20
P_2	100	35

Period = Deadline

RMS



EDF



Problem 4

Process	Period	Exectime
P_1	50	25
P_2	80	35

Deadline = Period