RADHIKA PATWARI
18CS10062

PAGE NO
DATE 19/2/2021

Class Test - 1

Question 1:

1.1 True ; In multiprogramming, process executes till it voluntarily leaves the control of CPU. So this is same if time slice > CPU burst

1.2 True ; Core functionalities of kernel execute in supervisor mode, any malicious process can block resources if allowed to change modes.

1.3. False ; In multithreaded programming, all threads of the same process have same virtual memory address space, so commands data can be put as global variables and easily accessed by all threads without creating explicit segments

1.4. True ; In non preemptive algorithms, convoy effect can occur where process with larger CPU burst delays all other processes with shorter CPU burst → increasing avg. waiting time

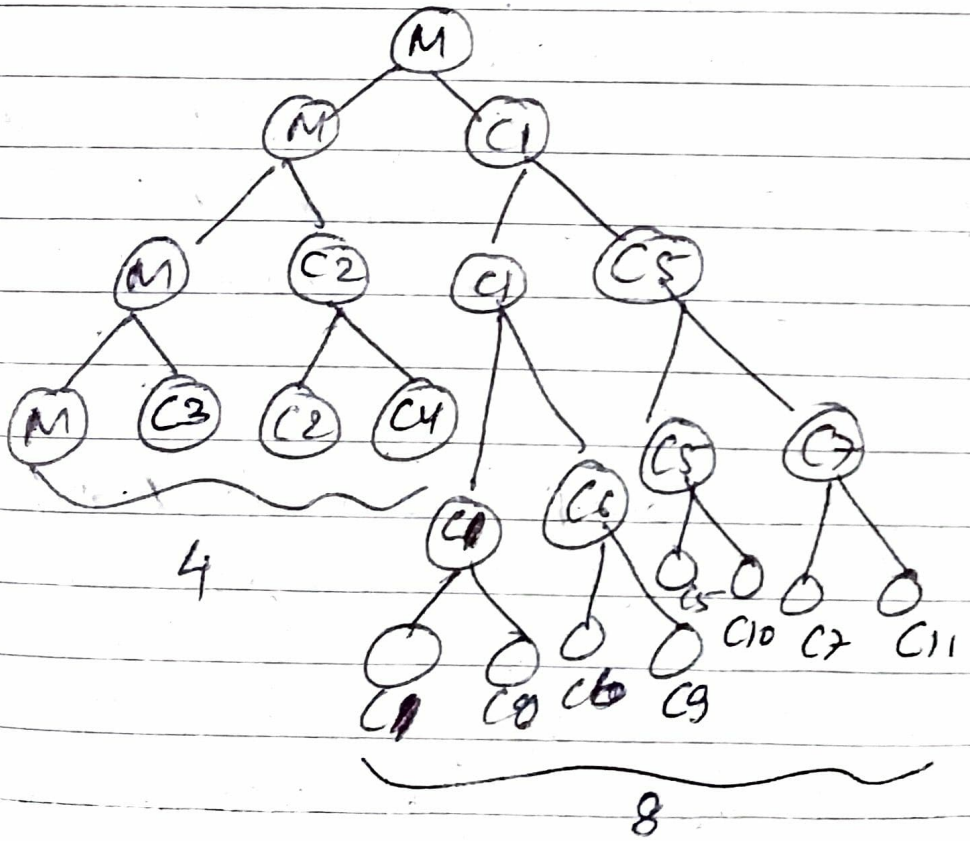
1.5. True ; It is a user-defined signal handler so that we can direct signals and perform some functionalities.

1.6. False ; local variables of a function are stored in memory stacks. All threads of the same process have their own stacks and their own local variable copy. So changes done by one thread is not visible to other threads.

1.7. True ; ~~'a' takes the default value~~

&a gives logical address of variable 'a'. Both parent and child have same virtual memory address. Their physical address are different but logical address is same. So here same answers are obtained.

1.8. False; Hello is printed <u>12 times</u>



= <u>12 process</u> = 12 Hello statements.

PAGE NO.
DATE: / /

Question 2:

2.1. (a) Changes mode from user to kernel mode using hardware mode bit to run core functionalities of kernel

(b) Checking privilege level necessary to raise the system call.

(c) Optimising time as syscalls are invoked more than hundred times per second.

2.2 No. because

(a) Unnamed pipes do not have names assigned to them. Hence other processes (except the one where pipe() is called explicitly) cannot have the file descriptors as there is no way to access the pipe.

(b) Named pipes are used as instances of file system hence they can be accessed by any process (currently it has permission to access it) using the same way in which we access files.

2.3. (i) Yes ; file descriptors are copied from parent to child and points to the same file. Hence child can access the 'file.txt'

(ii) No; as child has separate copy of file descriptor, it does not depend on activity of parent after fork() is called.

PAGE NO
DATE: / /

2.4

(i) Running to Ready

→ Context switching saves the data of CPU register for the running process into its Process Control Block in memory.
→ Ready queue is a circular linked list. PCB of this process is added to this queue at the tail.
→ 'Process state' value in PCB is changed.

(ii) Waiting to Ready

→ Process Control Block of waiting process is removed from waiting queue and added to ready queue.
→ Process state value in PCB is changed from 'waiting' to 'ready' state.
→ I/O status is updated in PCB if required.

Question 3.

3.2 # of processes = 100
Scheduling overhead = 5 μs
Let time quantum be x μs.

Given, Max waiting time for any process = 1 sec.

$$1\ sec = (100-1)(5+x)$$

$$\Rightarrow \frac{1}{99} = (5+x)\ \mu s \quad \Rightarrow 1.0101 \times 10^{-2}\ sec = (5+x)\ \mu s$$

$$\Rightarrow 1.0101 \times 10^{4}\ \mu s = (5+x)\ \mu s.$$

$\Rightarrow 10101 = (5+x) \rightarrow x = 10096$ us

$= 0.010096$ seconds

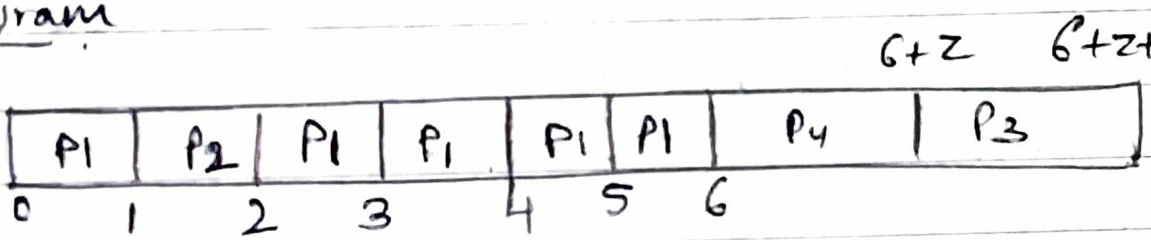
$= \underline{0.01 \text{ seconds}}$.

or

10096 microseconds.

3.3 Preemptive SRTF.

Gantt diagram



P_1	P_2	P_3	P_4
0	)	3	4
0 5	1	3	2
1 4			
3			
2			

At 3, both P1 & P3 have 3 ms left. Assuming scheduler runs P1

x y+ 6

Avg waiting time $\Rightarrow 2 = \frac{P_1 + P_2 + P_3 + P_4}{4} = \frac{1+0+P_3+P_4}{4}$

$\Rightarrow 8 = 1 + 0 + P_3 + P_4$

$\Rightarrow P_3 + P_4 = 7$

$\Rightarrow (x-3) + (y-4) = 7$

$\Rightarrow x + y = 14$

$x \in [6, 9]$

P_3 allotted at x ms

P_4 allotted at y ms

PAGE NO
DATE

$x \to P_3$ 3 6 6
$y \to P_4$ 7 9 6

$x + y = 14 \rightarrow (i)$

If P_3 is alloted at $x = 6$ ms, then $y = 6 + 3 = 9$ ms

But $6 + 9 \neq 14$ $\longrightarrow$ Not possible

If P_4 is alloted at $y = 6$, then $x = 6 + z$ ms.

Using (i), $(6 + z) + (6) = 14$

$\Rightarrow 12 + z = 14$

$\Rightarrow z = 2$ ms $\longrightarrow$ Thus P_4 is alloted at 6 ms, then P_3 is again alloted at 8 ms.

$\therefore \boxed{z = 2 \text{ ms}}$

3.1 At time 0,

	P_1	P_2	P_3
CPU burst →	10	20	30
I/O	2	4	6
computation	7	14	21
printing result	1	2	3

20
70
10

SRTF

CPU idle for 2 units.

At $t = 2$ units, P_1 is free so alloted CPU

At $t = 9$ unit, P_1 goes for I/O

P_2 alloted for 14 ms.

$t = 22$ units, P_3 alloted as P_2 goes for I/O.

44

Thus CPU idle for 2 units. $= \frac{2}{44} \times 100 = 4.55\%$