Assignment – 8

#Q-1

a) What is operating system?

b) What are the services performed by OS?

c) Discuss Real-time system, Distributed system and micro-kernel OS?

d) What is system call?

#Q-2

a) What is process? Discuss different process state.

b) Discuss briefly on PCB.

c) How does process scheduling work?

d) What is thread?

#Q-3

a) What is context switching?

b) How to create process using system call?

c) How can a process be terminated? What are zombie process and orphan process?

d) Discuss different ways of process communication.

#Q-4

a) Discuss pipes in OS?

b) What is the output of the following program? How many processes are created here?

```
# include <stdio.h>
# include <unistd.h>

int main ()
{
        fork ();
        printf (" Hi\n");

        fork ();
        printf ("Hello\n");

        fork ();
        printf (" Bye\n");

        return 0;
}
```

c) What happens when fork command is used?
d) What is process tree?

#Q-5:
a) Discuss thread.
b) Discuss different models of thread.
c) What is critical section problem? And how can it be solved?
d) How do mutex lock and semaphore work?

#Q-6:
a) What is CPU scheduling? Discuss its classification.
b) Discuss different scheduling algorithm.

c) Read the following data.

Process	Burst time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

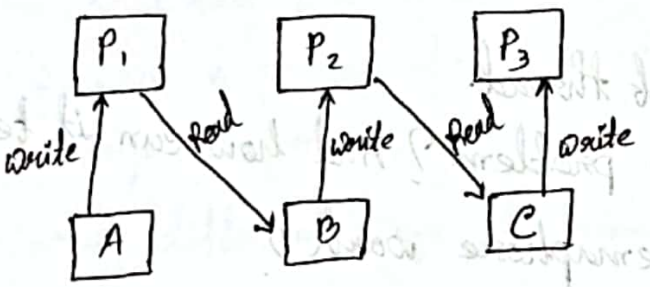
What is the average waiting time of each process for FCFS, SJF, PB and RR algorithm? You can assume the starting time as 0 and for RR the time quantam is 3 unit time. Based on your calculation, which one does work better here?

#Q-7

a) What is dead lock? Discuss deadlock using resource allocation graph.

b) Discuss different ways of handling dead lock.

c) Observe the following diagram.



Here, A, B, C are processes and P_1, P_2, P_3 are pipes. Each process wants to read or write (or both) to pipe.

What happens if all processes want to write first? How to solve this?

d) What is infinite blocking/starvation problem? When does it occur and how to solve it?

Q-8:

a) What is file and file system ~~interface~~?

b) Discuss file operation. Classify file access methods.

c) Discuss different Allocation method. Use necessary diagram.

d) Which allocation method is the best? Why? Explain it.

Answer to the question

#Q-1

a) Operating system is a system software which

→ manages computer resources (hardware, software)

→ provides an environment to run application software

b) Services performed by OS are

i) user interface
ii) Program execution
iii) I/O operation
iv) File-system manipulation
v) communication
vi) error detection
vii) resource allocation
ix) Accounting
x) Protection & security

Details – chapter-2 (2.1, page- 56,57)

c) Real time system – chapter-1 (1.11.8, Page-43)
Distributed System – chapter-1 (1.11.3, Page-37)
Micro-kernel OS – chapter-2 (2.7.3, page-81)

d) system call - chapter-2 (2.3, Page-62)

#Q-2

a) Process – chapter-3 (3.1.1, Page-106)
Process state – chapter-3 (3.1.2, Page-107)

b) PCB – chapter-3 (3.1.3, page 107, 108, 109)

c) Process schedule – chapter-3 (3.2, 3.2.1, 3.2.2, page- 112, 113, 114)

d) Thread-chapter-4 (4.1, page-162)

#Q-3:

a) context switching – chapter-3 (3.2.3, page- 114, 115)

b) using fork (). Details – chapter-3 (3.3.1, page- 116, 117, 118, 119, 120)
in linux

c) with exit() in linux (system call) or using signal (sigterm, sigkill) | Details - chapter-3 (3.3.2, page-120, 121)

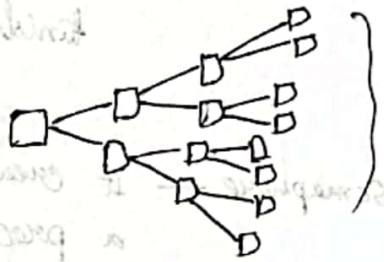
d) zombie → child process executes before wait() at parent process
Orphan → parent process terminates before child process. | Detail - chapter-3 (3.3.2, page-121 last two para.)

d) Differents way of process communication
i) shared memory (Details - chapter-3 (3.4.1, Page - 124))
ii) message passing (Details - chapter-3 (3.4.2, page-126))
iii) sockets (Details - chapter-3 (3.6.1, page-136))
iv) Pipe (Details - Chapter - 3 (3.6.3, page-142))

Q-4

a) pipe 2 types
1. named pipe
2. unnamed pipe (ordinary) | Details - chapter-3 (3.6.3, 3.6.3.1, 3.6.3.2, page-142, 145)

b) Here, "Hi" will be printed 2 times
"Hello" will be printed 4 times
"Bye" will be printed 8 times
In total 8 process will be created



8

c) fork command creates two process - 1. parent process, 2. child process. It returns a unsigned integer value. To the parent process it returns the PID of child process. To child process, it returns value 0. If any error occurs, it gives -1.

d) Process tree - A list of process. It is shown as tree form. It shows a complete list of all ~~com~~ current process in a system. In linux, we use 'pstree' to see it. 'pstree -ps' ~~shows~~ shows also the PID no.

#Q-5

a) Thread - a basic unit of CPU utilization; contains thread ID, program counter, register set, stack. 2 types → single thread, → multi thread | Details - chapter-4 (4.1, page-163, 164)

b) Different models of thread
i) Many-to-One model
ii) One-to-one model
iii) Many-to-many model
| Details - chapter-4 (4.3, 4.3.1, 4.3.2, 4.3.3, Page-169, 170, 171)

c) Critical section problem — common section of codes containing variable, files or table, for different processes. All process can access the same section of code resulting the inconsistency problem. (Details - chapter-5 (5.2, page-206))

Solution — 2 → Mutex locks (Details - chapter-5 (5.5, page-212))
→ semaphores (Details - chapter-5 (5.6, page-213, 214))

d) mutex lock — locks the critical section, before using and unlocks after finishing (Details - chapter-5 (5.5, page-212, 213))

semaphore — It creates a memory space containing an integer value. When a process accesses it it decreases value to 0 (in binary semaphore). At that time no other process can access it. After finishing it again restore values and other process can access it. (Details - chapter-5 (5.6, page-213, 214))

#Q-6:

a) CPU scheduling - In ~~sign~~ single processor CPU, one process can run at a time. So other processes have to wait. That's why the OS decide a fixed time alloted for any process. After that time, the OS takes away CPU from the previous process and gives it back to another process. This is known as CPU scheduling. (Details - chapter-6, (6.1, page-261, 262))

classification - 2 types → ~~Preemtive~~ preemptive
↳ non-preemptive (Details - chapter-6 (6.1.3, 6.1.4 page-263, 265))

b) Scheduling algorithm

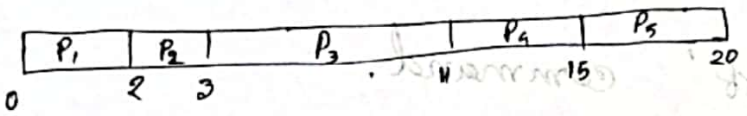
i) First-come, First-Served Scheduling, FCFS (Details - ~~6.3~~ chapter-6 (6.3.1, page-266))

ii) Shortest-Job-First Scheduling, SJF (Details - chapter-6 (6.3.2, page-267, 268))

iii) Priority Scheduling, PB (Details - chapter-6 (6.3.3, page-270, 271))

iv) Round-Robin Scheduling, RR (Details - chapter-6 (6.3.4, page-271, 272))

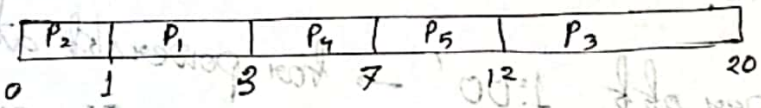
e) For FCFS.



total waiting time = (0 + 2 + 3 + 11 + 15) = ~~10~~ 31

∴ average waiting time = 31/5 = 10.333 unit

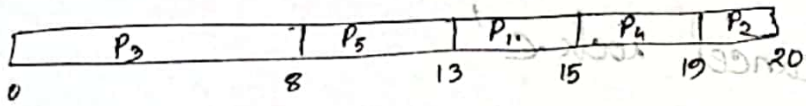
For SJF,



total waiting time = (0 + 1 + 3 + 7 + 12) = 23

∴ average waiting time = 23/5 = 4.6 unit

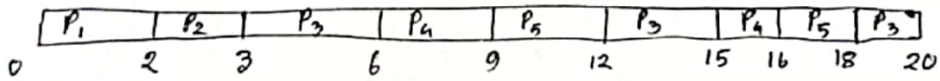
For, PB



total waiting time = (0 + 8 + 13 + 15 + 19) = 55

∴ average waiting time = 55/5 = 11 unit.

For, RR,



So, P_1 waits = ~~[illegible]~~ 0 unit

P_2 waits = 2 unit

P_3 waits = $3 + (12-6) + (18-15) = 3+6+3 = 12$ unit

P_4 waits = $6 + (15-9) = 6+6 = 12$ unit.

P_5 waits = $9 + (16-12) = 9+4 = 13$ unit.

$\therefore$ total wait time = $(0+2+12+12+13) = 39$

$\therefore$ average wait time = $39/5 = 7.8$ unit

Here ~~for time~~ according to waiting time (average)

SJF < RR < FCFS < PB

So, SJF ~~is~~ works better here.

#Q-7

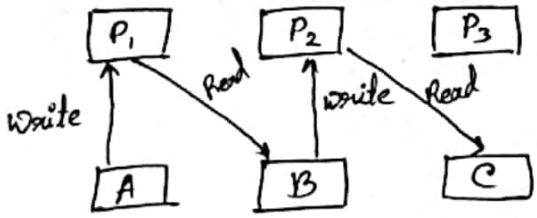
a) Dead lock – ~~a~~ When a process never finish executing. (Details - chapter-7 (page-315))

Resource allocation graph – using this, dead lock can be expressed more precisely.
If there is a cycle in the graph, dead lock occurs.
(Details - chapter-7 (7.2.2, page-319, 320, 321, 322))

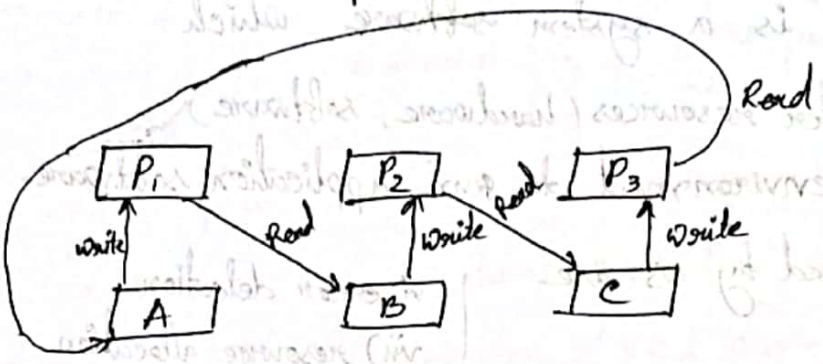
b) Different ways of handling dead locks –
i) prevent dead lock (Details - chapter-7 (7.4.1, page-323))
ii) detect dead lock and delete it (Details - chapter-7 (7.6, page-333))
iii) ignore dead lock

c) If all the process wants to write first, dead lock will appear.

Solve-1 : for process B and C, at first read operation ~~will~~ must take place and have to remove write operation to pipe-3 (P_3)



Solve-2: same as solve-1 and process C will write to pipe-3 and then process A will read to pipe-3 and in process A write operation must take place before read operation



d) Infinite blocking/starvation problem : In priority based scheduling, there may be a possibility of a process not being ~~executing~~ executed for a long time. because of low priority. This is called Infinite blocking/starvation problem. This ~~call~~ can be solved using other algorithm like- FCFS, SJF, RR.

AQ-8:

a) File:- file is a named collection of related information that is recorded on secondary storage. (Details-Chapter-11 (11.1, page-504,505))

File system:- provides the mechanism for on-line storage of and access to both data and programs of the operating system and all the users of the computer system. (Details- Chapter-11 (11.1, page-503))

b) file operation-
i) creating a file
ii) Writing a file
iii) Reading a file
iv) Repositioning a file
v) Deleting a file
vi) Truncating a file

(Details — Chapter-11 (11.2, page-506))

Access methods -
i) Sequential Access
ii) Direct Access
iii) Others

(Details- chapter-11 (11.2, 11.2.1, 11.2.2, 11.2.3, page-513, 514)

c) Allocation method – 3 types –

i) Contiguous Allocation (Chapter-12 (12.4.1, page-553))

ii) Linked Allocation (Chapter-12 (12.4.2, page-555))

iii) Indexed Allocation (Chapter-12 (12.4.3, page-557))

d) Indexed Allocation - Because – (Details - Chapter-12 (12.4.3, page-557))

i) solves external-fragmentation problem

ii) solves size-declaration problem

iii) have index block – pointing all pointer in one location

iv) have some features like – linked scheme, multi-level index, combined scheme.

v) average performance is good.

<u>Note</u>: All answer are taken and used from "Operating System Concepts" by Abraham Silberschatz. (9th edition)