Internal Assessment Test 1

Solution

Sub:	Operating System - OS					Sub Code:	BCS303	Branch:	CSE
Date:	05/11/2024	Duration:	90 mins	Max Marks:	50	Sem / Sec:	3 A,B,C		

1. A

Distinguish between the following terms:
1. Multiprogramming and Multitasking.
2. Multiprocessor systems and Clustered systems (5M)

Solution

1. **Multiprogramming and Multitasking. (2.5M)**

Multiprogramming	Multitasking
Several programs are kept in memory so the CPU always has one to execute.	Several tasks or processes are executed seemingly at the same time by rapidly switching between them.
The main goal is to **increase CPU utilization**.	The main goal is to **increase user interaction and responsiveness**.
CPU switches to another job when the current job is waiting for I/O.	CPU switches between tasks so that **each gets a small time slice**, giving the illusion of parallel execution.
Typically used in **batch processing systems** where there is little or no user interaction.	Used in **time-sharing systems** that allow **multiple users** or tasks to interact with the system simultaneously.
Example: Running multiple background jobs like compilation, printing, and data processing.	Example: Typing a document while listening to music and browsing the internet.

2. Multiprocessor systems and Clustered systems (2.5M)

Multiprocessor systems	Clustered systems
Consist of **multiple processors within a single computer system** sharing the same memory and bus.	Consist of **two or more independent computers (nodes)** connected through a high-speed network.
Processors work **together in tight coupling** (share memory and I/O).	Systems work **together in loose coupling** (each has its own memory and storage).
All processors run under **a single operating system**.	Each node may run **its own operating system**, but they work together for a common task.
Used to **increase computing speed and performance** within a single system.	Used to provide **high availability, load balancing, and reliability** across multiple systems.
Failure of one processor usually affects the whole system.	Failure of one node does **not affect the others**; other nodes can take over (fault tolerance).

1.B Define Operating Systems. Explain dual mode of operating systems with a neat diagram. (5M)

(Definition 1M + Explanation 2M + Diagram 2M)

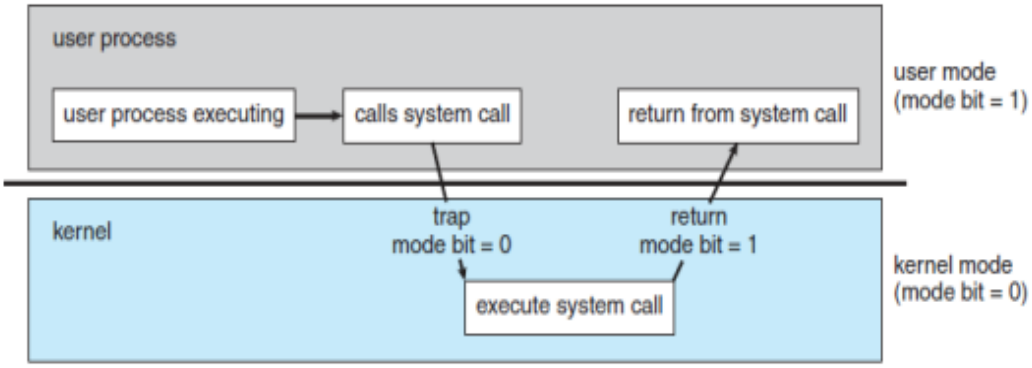
Solution: An operating system is system software that acts as an intermediary between a user of a computer and the computer hardware.

The system can be assumed to work in two separate modes of operation:
1. User mode
2. Kernel mode

When the computer system is executing a user application, the system is in user mode. When a user application requests a service from the operating system (via a system call), the transition from user to kernel mode takes place.

At system boot time, the hardware starts in kernel mode. The operating system is

then loaded and starts user applications in user mode. Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode (that is, changes the mode bit from 1 to 0). Thus, whenever the operating system gains control of the computer, it is in kernel mode.



2.A List and explain the services provided by the OS for the user and efficient operation of the system. (5M)

Solution:

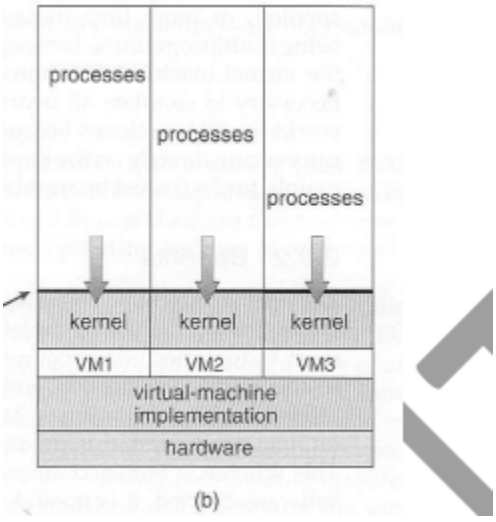
1.User Interface : Means by which users can issue commands to the system. Depending on the operating system these may be a command-line interface and a Graphical User Interface.

2.Program Execution - The OS must be able to load a program into RAM, run the program, and terminate the program, either normally or abnormally.

3. I/O Operations - The OS is responsible for transferring data to and from I/O devices, including keyboards, terminals, printers, and files.

4.Communications - Inter-process communications, IPC, either between processes running on the same processor, or between processes running on separate processors or separate machines.

5.Resource Allocation – Resources like CPU cycles, main memory, storage space, and I/O devices must be allocated to multiple users and multiple jobs at the same time.

2. B	What are microkernels? Point out their advantages. (5M) (Definition 1M + Advantages 4M)
Solution:	A **microkernel** is a minimal and compact type of operating system kernel that contains only the most essential functions needed to run the system. **Advantages of Microkernels:** 1. **Reliability and Stability:** Since most services run in user space, a fault in one service does not crash the entire system. 2. **Security:** The small size of the kernel reduces vulnerabilities and makes it easier to secure. 3. **Ease of Maintenance:** It is easier to modify or update user-level services without affecting the kernel. 4. **Portability:** Microkernels are easier to adapt to new hardware because hardware-dependent code is kept separate.
3 a	What are virtual machines? Explain with a block diagram. Point out its benefits?  A **Virtual Machine (VM)** is a software-based environment that emulates a physical computer system. It allows multiple operating systems to run

	concurrently on a single physical hardware using a **Virtual Machine Monitor (VMM)** or **Hypervisor**. The hypervisor provides an abstraction layer between hardware and software, enabling each virtual machine to operate independently with its own CPU, memory, and storage as if it were a separate physical system. **Benefits of Virtual Machines:** 1. **Isolation:** Each VM runs independently, preventing one from affecting others. 2. **Efficient Resource Utilization:** Multiple OS instances share the same hardware resources. 3. **Flexibility and Portability:** VMs can be easily created, modified, or moved across systems. 4. **Testing and Development:** Ideal for software testing without affecting the host system. 5. **Improved Security and Backup:** Snapshots allow quick recovery and safe experimentation.
3b	Describe the implementation of inter-process communication using shared memory and message passing? **Inter-Process Communication (IPC)** allows processes to exchange data and coordinate their actions. **1. Shared Memory:** ● A **common memory region** is established that can be accessed by cooperating processes. ● Once created, each process can read or write directly to this shared area. ● The OS provides system calls like `shmget()`, `shmat()`, `shmdt()` (in UNIX) for creation and attachment. ● Processes must use **synchronization mechanisms** (like semaphores or

mutexes) to avoid race conditions.

Example:
One process writes data to the shared segment, and another process reads it.

Advantages:

- Fast communication (no kernel involvement after setup).
- Efficient for large data transfer.
- **2. Message Passing:**
- Processes communicate by **sending and receiving messages** through the kernel.
- No shared memory is used; the OS manages message queues and buffers.
- System calls like `msgsnd()` and `msgrcv()` (UNIX) are used.
- Synchronization is achieved through **blocking** or **non-blocking** sends and receives.

Advantages:

- Easier to implement in distributed systems.
- Provides built-in synchronization.

4 a

Demonstrate the process creation and process termination in Unix.?

```
#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>
```

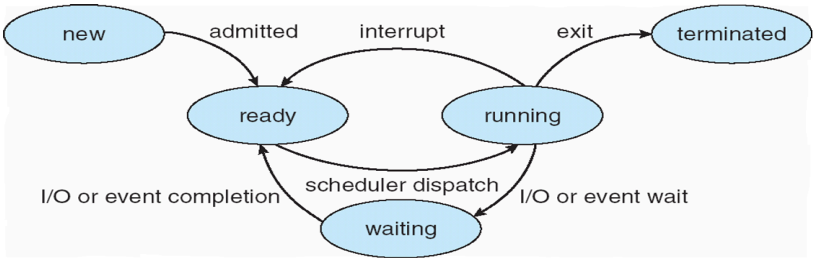
```
int main() {
    pid_t pid;
    pid = fork();

    if (pid < 0)
        printf("Fork failed\n");
    else if (pid == 0)
        printf("Child process created. PID = %d\n", getpid());
    else {
        printf("Parent process. PID = %d, Child PID = %d\n", getpid(), pid);
        wait(NULL);
        printf("Child terminated.\n");
    }
    return 0;
}
```

4b

What is a process? Illustrate with a neat diagram the different states of a process and control block.?

A **process** is a **program in execution**.

It is an active entity that includes the program code, current activity (program counter, registers), and resources such as memory, files, and I/O devices allocated by the operating system.



1. New:

The process is being **created** by the operating system.
(Example: Memory and resources are being allocated.)

2. Ready:

The process is **loaded into main memory** and waiting for CPU allocation.

3. Running:

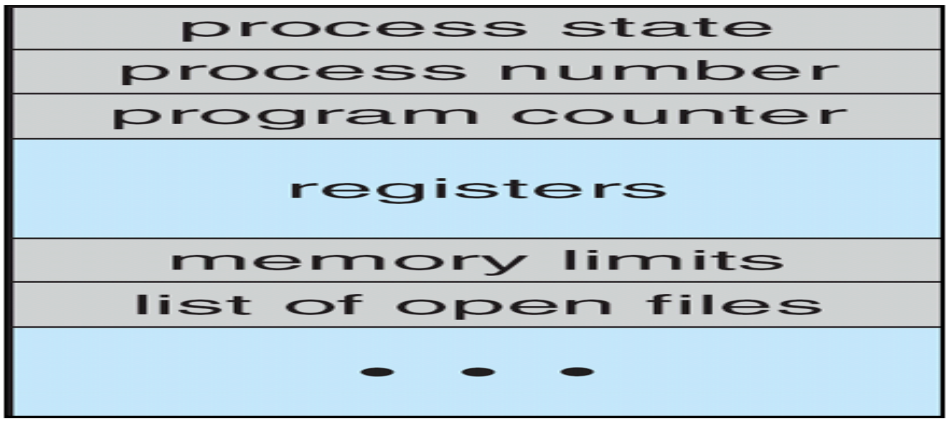
The process is **being executed** by the CPU.
(Only one process can be in running state per CPU at a time.)

4. Waiting / Blocked:

The process is **waiting for some event** to occur (like I/O completion or signal).

5. Terminated / Exit:

The process has **finished execution** and is removed from the main memory.



A **Process Control Block (PCB)** is a **data structure maintained by the Operating System** that stores all information about a process.
It acts as the **identity card** of a process — whenever a process is switched (context switching), its PCB is used to save and restore its state.

Contents of PCB:

1. **Process ID (PID):** Unique identifier for the process.
2. **Process State:** Current state (new, ready, running, waiting, or terminated).
3. **Program Counter (PC):** Address of the next instruction to execute.
4. **CPU Registers:** Contents of all CPU registers.
5. **Memory Management Information:** Base and limit registers, page tables, segment tables.
6. **Accounting Information:** CPU usage, execution time, job priority, etc.
7. **I/O Status Information:** List of I/O devices allocated, open file descriptors.

Function:

- Helps OS to **track the process**.
- Used during **context switching** to save and restore process information.

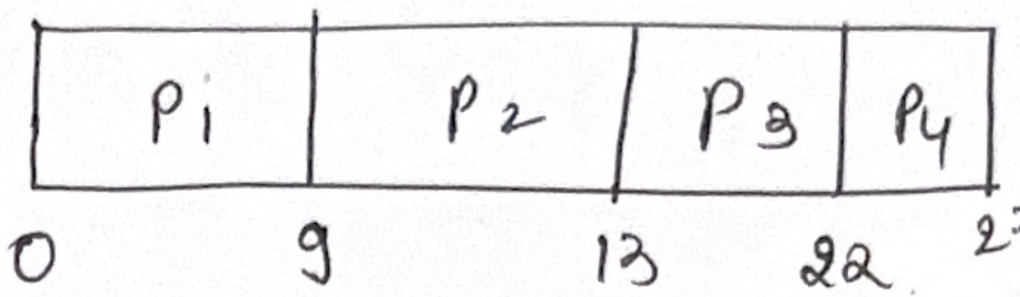
5	5	Calculate the average waiting time and the average turnaround time by drawing the Gantt chart using FCFS, SRTF, RR (q = 2 ms) and Priority algorithms. Lower priority number represents higher priority.	[10]	L3	CO2

Process	Arrival Time	Burst Time	Priority
P1	0	9	3
P2	1	4	2
P3	2	9	1
P4	3	5	4

FCFS.

Gantt chart

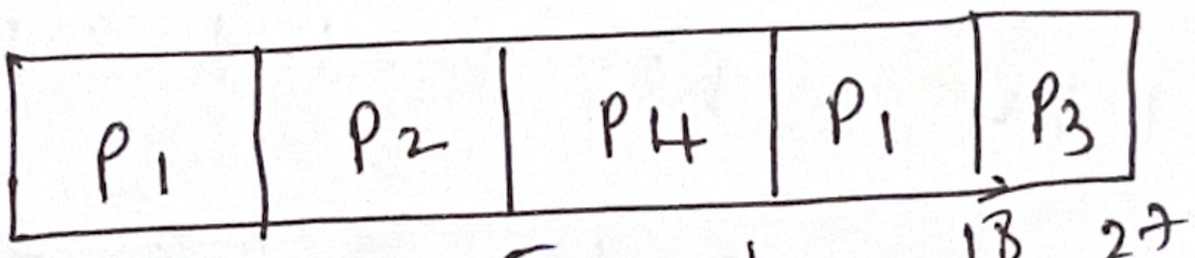
Process	A.T	B.T	C.T	TAT (AT-AT)	WT
P_1	0	9	9	9	0
P_2	1	4	13	12	8
P_3	2	9	22	20	11
P_4	3	5	27	24	19



$\therefore$ Avg WT = $(0+8+11+19)/4 = 9.50$

Avg TAT = $(9+12+20+24)/4 = 16.25$.

SRTF :-

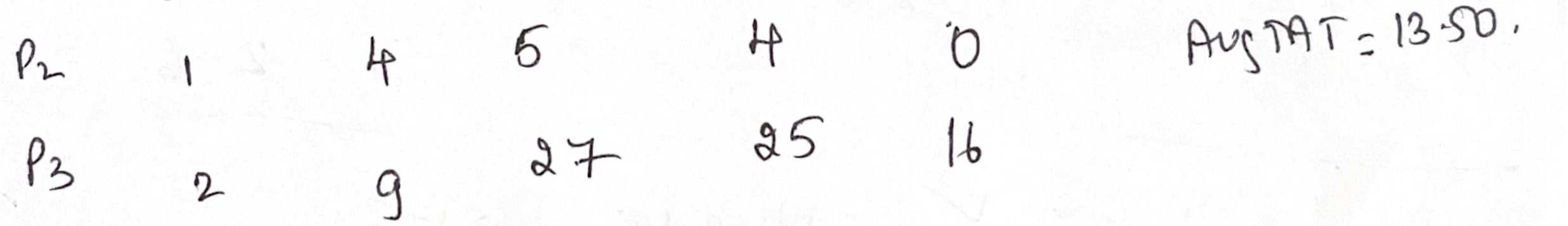


	A.T	B.T	C.T	TAT	WT
P_1	0	9	18	18	9
P_2	1	4	5	4	0
P_3	2	9	27	25	16
P_4	3	5	10.	7	2.

$\therefore$ Avg WT = 6.75

Avg TAT = 13.50.

R.R. q. = 2ms.



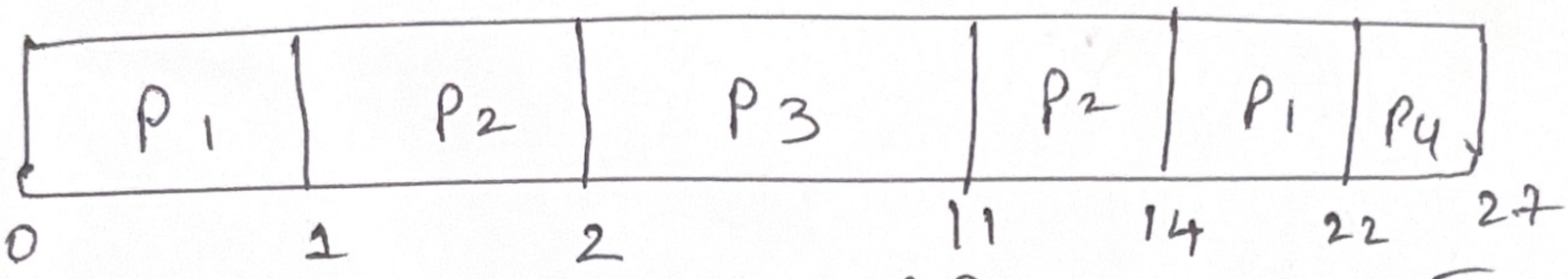
	C.T	TAT	WT
P_1	26	26	17
P_2	12	11	7
P_3	27	25	16
P_4	23	20	15

Avg WT = 13.75

Avg TAT = 20.50

1

Priority



	A.T	B.T	P	C.T	T.A.T	W.T
P_1	0	9	3	22	22	13
P_2	1	4	2	14	13	9
P_3	2	9	1	11	9	0
P_4	3	5	4	27	24	19

Avg WT = 10.25

Avg TAT = 17.00

6. Bakers

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	2	0	0	1	A 2	1	2		3	3	2	1
P_1	3	1	2	1	5	2	5	2				
P_2	2	1	0	3	2	3	1	6				
P_3	1	3	1	2	1	4	2	4				
P_4	1	4	3	2	3	6	6	5				

① For all i, make finish[i] = false.

② Work = Available = <3, 3, 2, 1>

③ Need = max - Allocation

A	B	C	D
2	2	1	1
2	1	3	1
0	2	1	3
0	1	1	2
2	2	3	3

<P0, Safe sequence.

P0 finish[0] == false ✓

Need0 <= work

<2, 2, 1, 1> <= <3, 3, 2, 1) ✓

work = work + Allocation0

= <3, 3, 2, 1> + <2, 0, 0, 1>

= <5, 3, 2, 2>

finish[0] = true.

P_1 finish[1] == false

Need 1 <= work

{2, 1, 3, 1} <= {5, 3, 2, 2}

true.

P_2 finish[2] == false.

Need 2 <= work

{0, 2, 1, 3} <= {5, 3, 2, 2} false.

P_3 finish[3] == false

Need 3 <= work

{0, 1, 1, 2} <= {5, 3, 2, 2}

work = work + allocation 3

= {5, 3, 2, 2} + {1, 3, 1, 2}

= {6, 6, 3, 4}

finish[3] = true. {P_0, P_3,

P_4 finish[4] == false

Need 4 <= work

{2, 2, 3, 3} <= {6, 6, 3, 4}

work = work + Allocation 4

= {6, 6, 3, 4} + {1, 4, 3, 2}

= {7, 10, 6, 6}

finish[4] = true. {P_0, P_3, P_4

P_1 Finish[1] == false ✓

Need 1 2 <= work

<2, 1, 3, 1> <= <7, 10, 6, 6> ✓

Work = work + Allocation 1

= <7, 10, 6, 6> + <3, 1, 2, 1>

= <10, 11, 8, 7>

Finish[1] = True. <P0, P3, P4, P1,

P_2. Finish[2] == false ✓

Need 2 <= work

<0, 2, 1, 3> <= <10, 11, 8, 7> ✓

work = work + Allocation 2

= <10, 11, 8, 7> + <2, 1, 0, 3>

= <12, 12, 8, 10>

Finish[2] = True. <P0, P3, P4, P1, P2>

Safe sequence.

6.b. If process P_2 requests <0, 1, 1, 3> resources, can it can be granted immediately?

(a) Request 2 <= Need 2.

<0, 1, 1, 3> <= <0, 2, 1, 3> ✓

(b) Request 2 <= Availab6

<0, 1, 1, 3> <= <3, 3, 2, 1> ✗

∴ P_2 request can't be granted immediately P_2 wait.