Solution

Internal Assessment Test 1 – May 2021

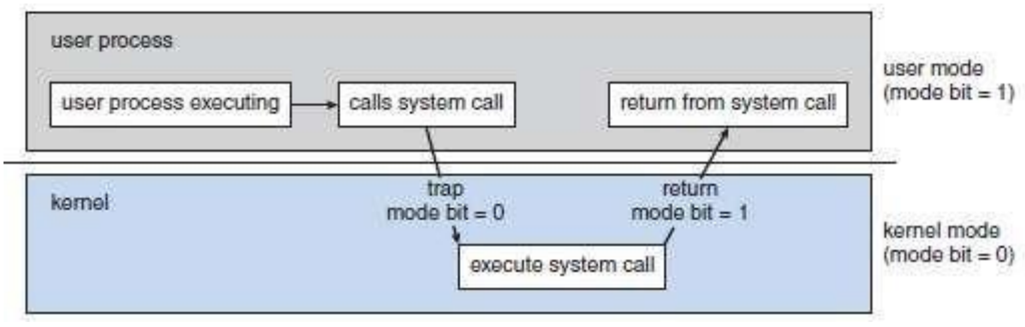
Sub:	Operating Systems							**Code:**	**18CS43**
Date:	**20-5-2021**	Duration:	90mins	Max Marks:	50	**Sem:**	IV	**Branch:**	ISE

SECTION A

DESCRIPTIVE QUESTIONS – 30 marks

Note: Answer Any Five Questions (5*6 = 30)

1. **Why do you think dual mode operations of an Operating System is important? Explain.**
   - Two modes of operation 1) User mode and 2)Kernel mode
   - A mode bit is a bit added to the hardware of the computer to indicate the current mode: i.e. kernel (0) or user (1)



Transition from user to kernel mode

   - Working principle:
     1. At system boot time, the hardware starts in kernel-mode.
     2. The OS is then loaded and starts user applications in user-mode.
     3. Whenever a trap or interrupt occurs, the hardware switches from user-mode to kernel-mode (that is, changes the state of the mode bit to 0).
     4. The system always switches to user-mode (by setting the mode bit to 1) before passing control to a user-program.
   - Dual mode protects
     → OS from errant users and
     → errant users from one another.
   - Privileged instruction is executed only in kernel-mode.
   - If an attempt is made to execute a privileged instruction in user-mode, the hardware treats it as illegal and traps it to the OS.
   - A system calls are called by user-program to ask the OS to perform the tasks on behalf of the user program.

2. **Justify the statement "Most users' view of the operating system is defined by system programs, not the actual system calls."**

   System Programs
   - They provide a convenient environment for program development and execution. (System programs also known as system utilities).
   - They can be divided into these categories:

- Six categories of system-programs:
    1) **File Management:** These programs manipulate files i.e. create, delete, copy, and rename files.

2) **Status Information:**
    - Some programs ask the system for
        → date (or time)
        → amount of memory (or disk space) or
        → no. of users.
    - This information is then printed to the terminal (or output-device or file).

3) **File Modification**
    - Text editors can be used to create and modify the content of files stored on disk.

4) **Programming Language Support**
    - Compilers, assemblers, and interpreters for common programming-languages (such as C, C++) are provided to the user.

5) **Program Loading & Execution**
    - The system may provide
        → absolute loaders
        → relocatable loaders
        → linkage editors and
        → overlay loaders.
    - Debugging-systems are also needed.

6) **Communications**
    - These programs are used for creating virtual connections between
        → processes
        → users and
        → computer-systems.
    - They allow users to
        → browse web-pages
        → send email or
        → log-in remotely.

- Most OSs are supplied with programs that
    → solve common problems or
    → perform common operations. Such programs include
        → web-browsers
        → word-processors
        → spreadsheets and
        → games.

    These programs are known as application programs**.**

System Calls

- These provide an interface to the OS services.
- These are available as routines written in C and C++.
- The programmers design programs according to an API. (API=application programming interface).
- The API
    → defines a set of functions that are available to the programmer (Figure 1.15).
    → includes the parameters passed to functions and the return values.
- The functions that make up an API invoke the actual system-calls on behalf of the programmer.
- Benefits of API:
    1) Program portability.
    2) Actual system-calls are more detailed (and difficult) to work with than the API available to the programmer.

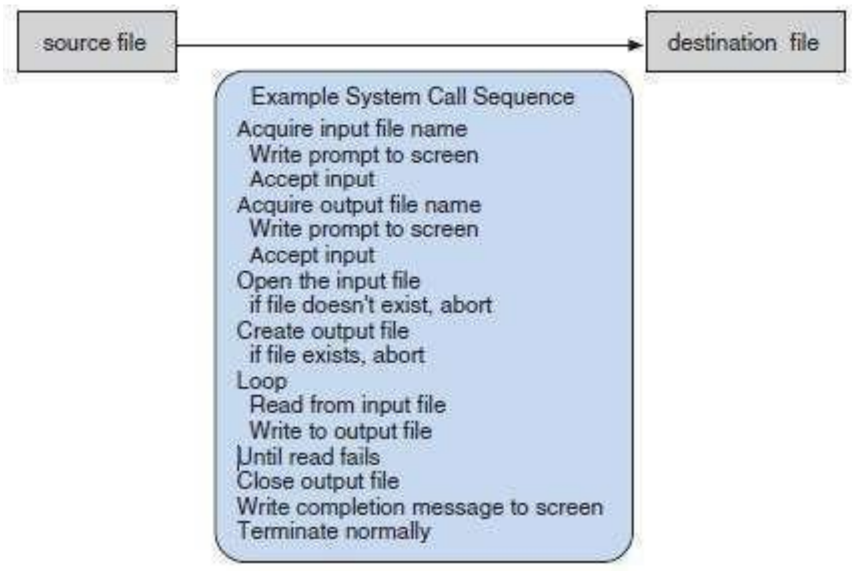
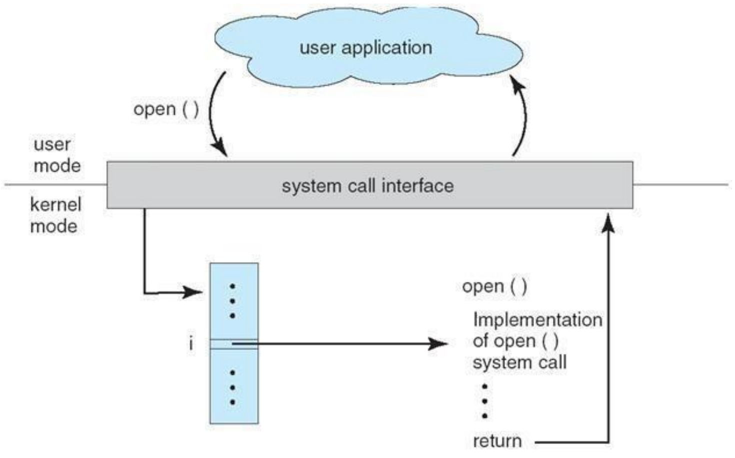


Figure 1.15 Example of how system calls are used.

- The following figure shows the relationship between an API, the system-call interface and the operating system, by illustrating how the OS handles a user application invoking the open() systemcall:



- Three general methods are used to pass parameters to the OS:
  1) via registers.
  2) Using a table in memory & the address is passed as a parameter in a register (Figure 1.16).
  3) The use of a stack is also possible where parameters are pushed onto a stack and popped off the stack by the OS.

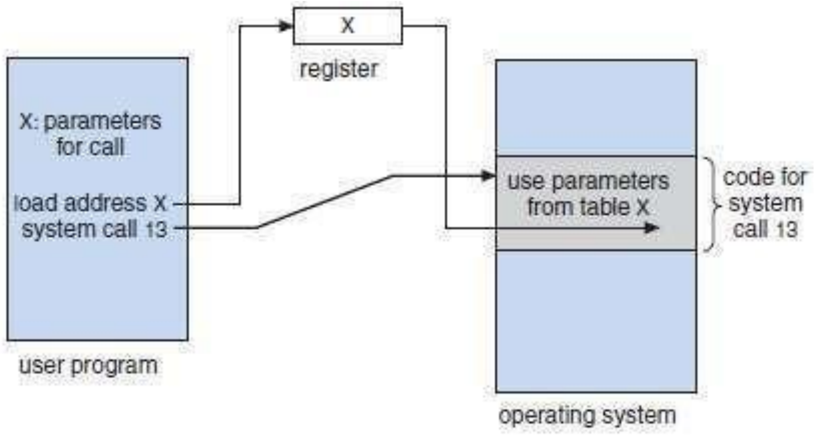


Figure 1.16 Passing of parameters as a table.

3. What is a **Process state**? With a neat diagram explain all the **possible transitions in the states** that can take place for a process.

   Process State:

   - As a process executes, it changes state.
   - Each process may be in one of the following states (Figure 1.24):
     - ▭ **New**: The process is being created.
     - ▭ **Running**: Instructions are being executed.
     - ▭ **Waiting:** The process is waiting for some event to occur (such as I/0 completions).
     - ▭ **Ready**: The process is waiting to be assigned to a processor.
     - ▭ **Terminated**: The process has finished execution.
   - Only one process can be running on any processor at any instant.

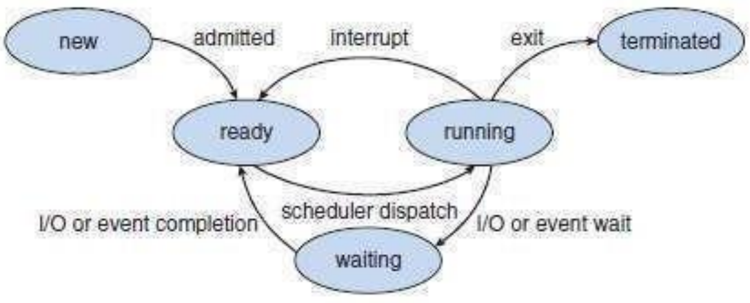


Diagram of process state

4. With a neat diagram, explain the components of a **Process Control Block (PCB)**.

   Process Control Block:

   - In OS, each process is represented by a PCB (Process Control Block).

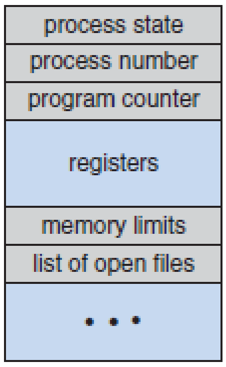
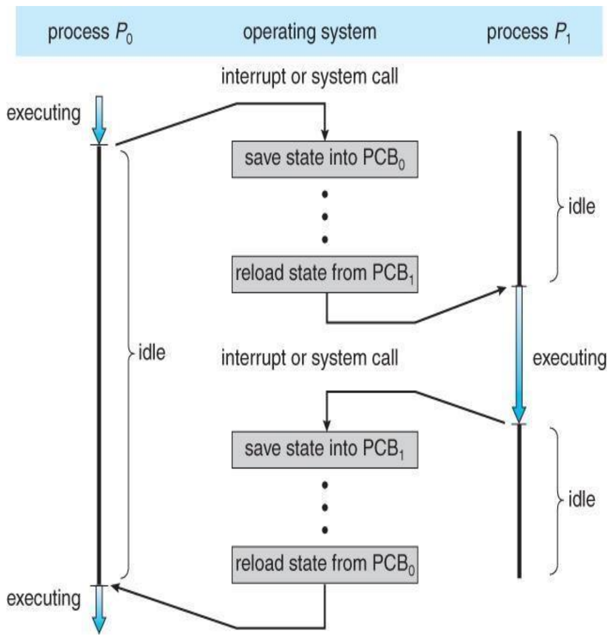


Figure 1.25 Process control block (PCB)

   - PCB contains following information about the process (Figure 1.25):
     - ▫ **Program Counter -** This indicates the address of the next instruction to be

executed for the process.

- **CPU Registers -** These include
  → accumulators (AX)
  → index registers (SI, DI)
  → stack pointers (SP) and
  → general-purpose registers (BX, CX, DX).
  →Along with Program Counter, the state information of a process must be saved when an interrupt occurs, to allow the process to be continued afterward. The following figure shows the scenario:



- **CPU Scheduling Information -** This includes
  → priority of process
  → pointers to scheduling-queues and
  → scheduling-parameters.

5. Explain the advantages of a virtual machines over a non-virtual machine with a neat sketch.

- Main idea:
  To abstract hardware of a single computer into several different execution environments.
- An OS creates the illusion that a process has
  → own processor &
  → own (virtual) memory.
- The virtual-machine provides
  → an interface that is identical to the underlying hardware (Figure 1.22).

→ a (virtual) copy of the underlying computer to each process.

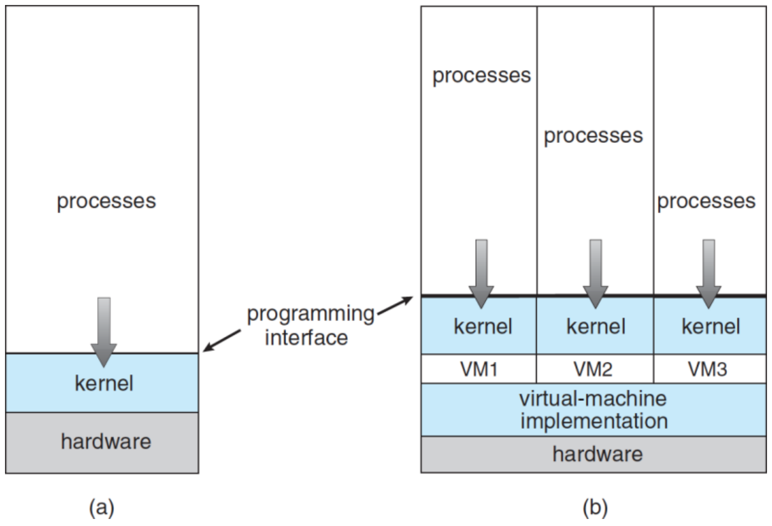


Figure 1.22 System models, (a) Nonvirtual machine, (b) Virtual machine.

- Problem: Virtual-machine software itself will need substantial disk space to provide virtual memory. Solution: provide virtual disks that are identical in all respects except size.
- Advantages:
  1) Complete protection of the various system resources.
  2) It is a perfect vehicle for OS"s R&D.
- Disadvantage:
  1) Difficult to implement due to effort required to provide an exact duplicate to underlying machine.

Implementation:

- The VM software can run in Kernel mode and VM itself can execute only in user mode.
  - Some actions need transfer from virtual user mode to virtual kernel mode. Example:
- A system call made by a program on a virtual machine in virtual user mode will transfer the

  control to virtual machine monitor in the real machine.
- Then the VM monitor changes the register contents and program counter for the VM to simulate the effect of the system call.
- It can restart the VM, noting that it is now in virtual kernel mode.

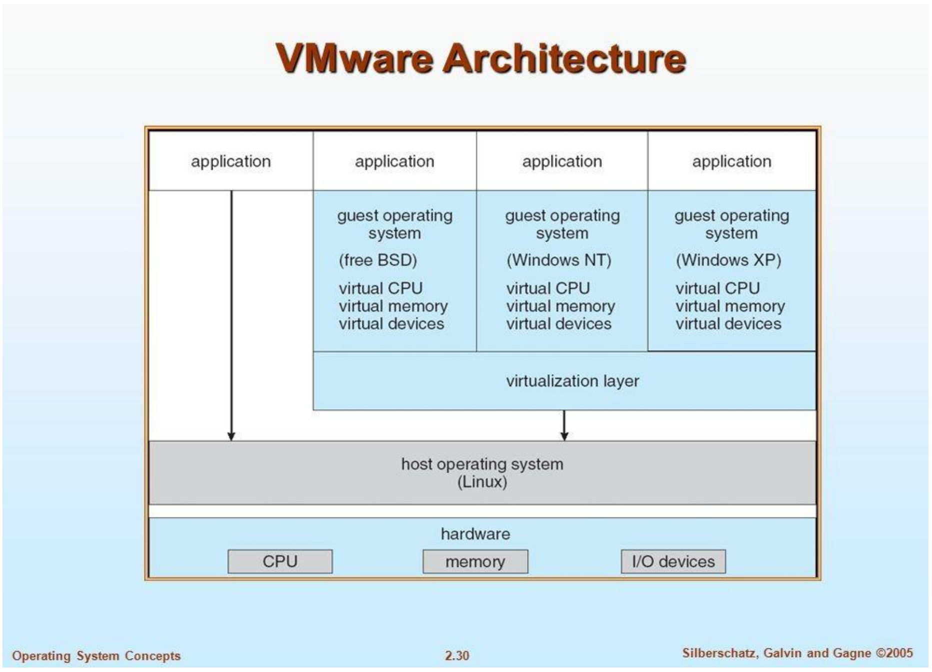
Difference:

- A real I/O might have taken 100ms, the virtual I/O might take less or more time
- The CPU is being multiprogrammed among many virtual machines, further slowing down the VMs in unpredictable ways.
- In an extreme case it may be necessary to simulate all instructions to provide true VM
- VM works for IBM machines because normal instructions for VMs can execute directly on hardware and privileged instructions must be simulated and execute more slowly.

Examples:

- VMware architecture:
  - It is a popular commercial application that abstracts Intel 80X86 hardware into isolated VMs
  - It runs as an application on a host OS such as Windows or Linux and allows the system to run concurrently several different guest OS as independent VMs.
  - The following figure the architecture:
    i. Linux is running as the host OS

ii. FreeBSD, WinNT, WinXP are running as guest OS.
iii. The virtualization is the heart of VMware, as it abstracts the physical hardware into isolated VMs running as guest OS.
iv. Each VM has its own virtual CPU, memory, disk drives, network interfaces, etc.,



6. What is **Inter Process Communication (IPC)**? Explain direct and indirect communications with respect to **message passing systems**.
   - System calls used:
     - create, delete communication connection
     - send, receive messages
     - transfer status information
     - attach or detach remote devices
   - Two models of communication.

1. Message Passing Model

- Information is exchanged through an IPC provided by OS. (IPC=inter process communication).
- Steps for communication:

i. Firstly, a connection must be opened using **open connection** system-call.
ii. Each computer has a host-name, such as an IP name. Similarly, each process has a process-name, which is translated into an equivalent identifier. The **get hostid** & **get processid** system-calls do this translation.
iii. Then, identifiers are passed to the **open** and **close** system-calls.
iv. The recipient-process must give its permission for communication to take place with an **accept connection** system-call.
(The processes that will be receiving connections are called daemons processes).
v. Daemon processes
→ execute a **wait for connection** system-call and

→ are awakened when a connection is made.
vi. Then, client & server exchange messages by **read message** and **write message** system calls.
vii. Finally, the **close connection** system-call

terminates the communication.

- Advantages:
  - i. Useful when smaller numbers of data need to be exchanged.
  - ii. It is also easier to implement than is shared memory.

2. Shared Memory Model

- Processes use map memory system-calls to gain access to regions of memory owned by other processes.
- Several processes exchange information by reading and writing data in the shared memory.
- The shared memory
  → is determined by the processes and
  → are not under the control of OS.
- The processes are also responsible for ensuring that they are not writing to the same location
  simultaneously.
- Advantage: Shared memory allows maximum speed and convenience of communication
- Disadvantage: Problems exist in the areas of protection and synchronization.

SECTION B

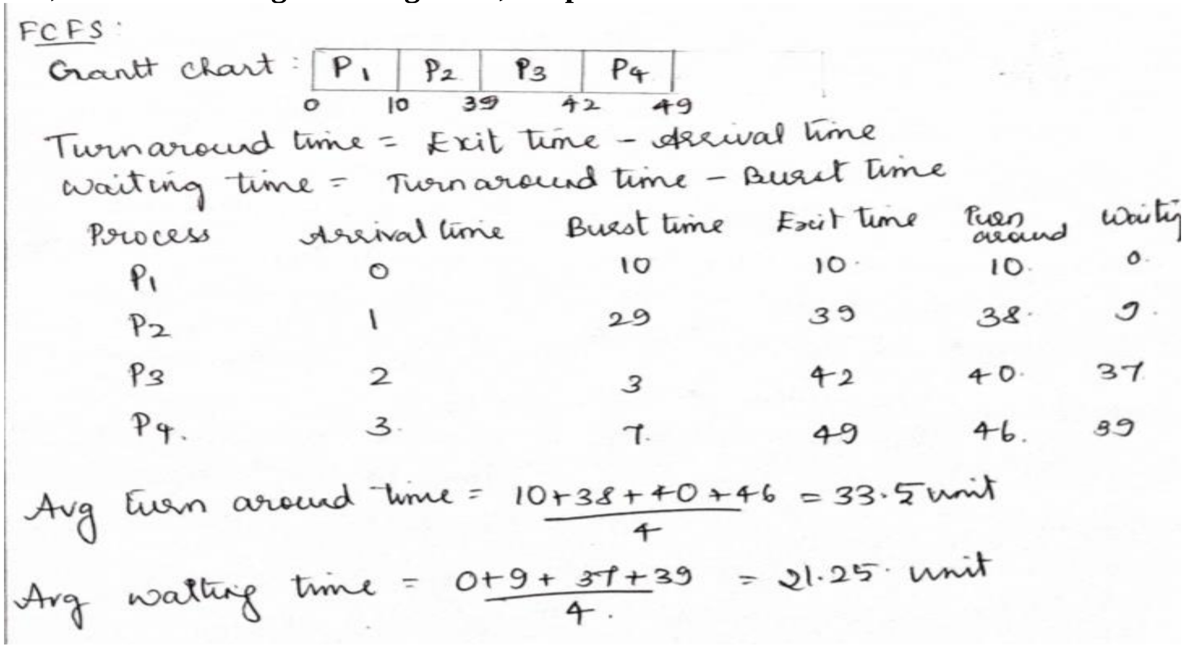
DESCRIPTIVE QUESTIONS – 20 marks

Note: Answer All the Questions (10*2 = 20)

7. Consider the following set of processes with arrival time.

Processes	Burst Time (ms)	Arrival time (ms)
P1	10	0
P2	29	1
P3	3	2
P4	7	3

Draw Gantt chart using **FCFS & SJF** CPU scheduling techniques.
Also, Calculate **average Waiting Time, Response Time & Turn Around Time** for each.



FCFS:

Gantt chart:

P1	P2	P3	P4

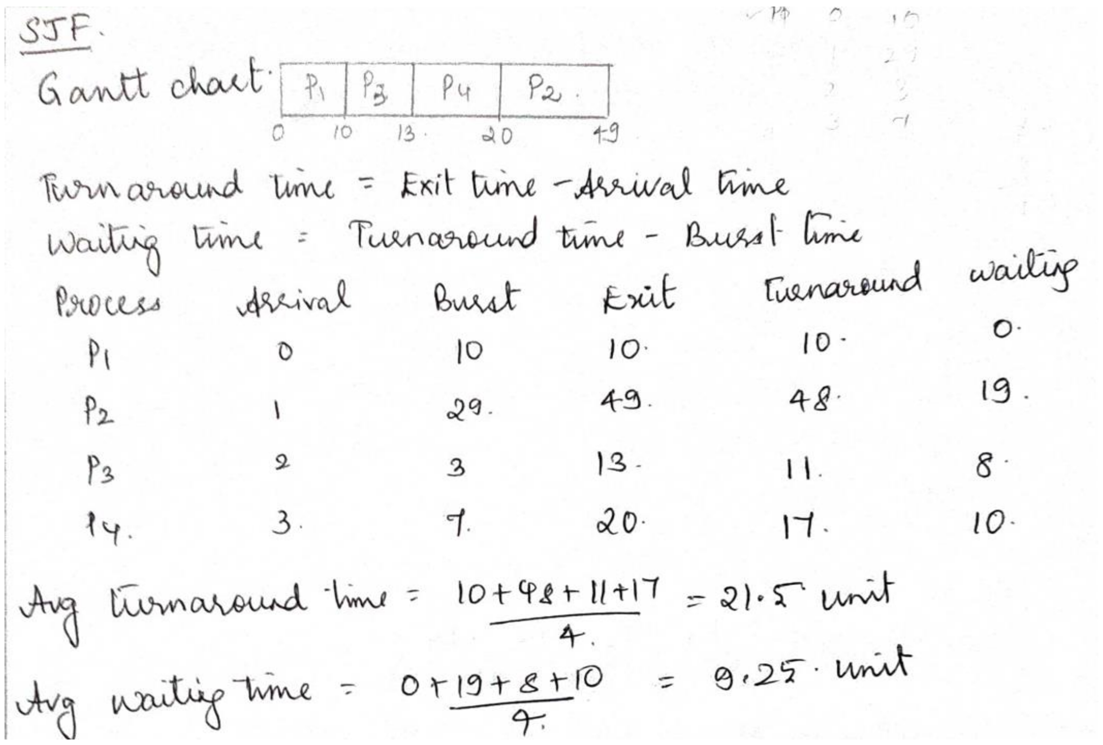
0 10 39 42 49

Turnaround time = Exit time − Arrival time
Waiting time = Turnaround time − Burst time

Process	Arrival time	Burst time	Exit time	Turn around	Waiting
P1	0	10	10	10	0
P2	1	29	39	38	9
P3	2	3	42	40	37
P4.	3.	7.	49	46.	39

Avg turn around time = $\frac{10+38+40+46}{4}$ = 33.5 unit

Avg waiting time = $\frac{0+9+37+39}{4}$ = 21.25 unit



SJF.

Gantt chart:

P1	P3	P4	P2.

0 10 13. 20 49

Turnaround time = Exit time - Arrival time

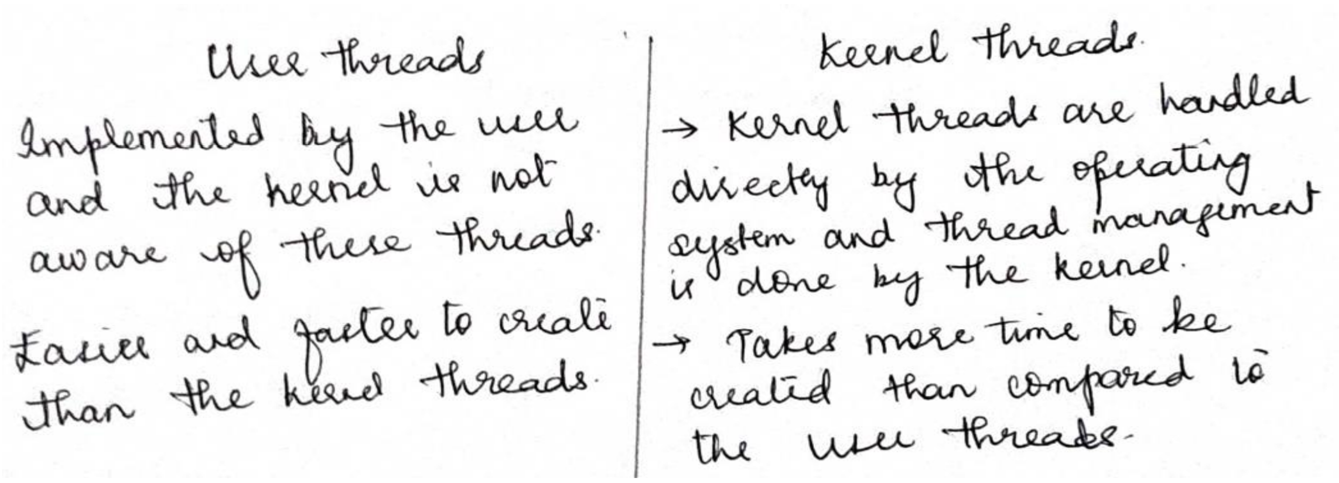
Waiting time = Turnaround time - Burst time

Process	Arrival	Burst	Exit	Turnaround	waiting
P1	0	10	10.	10.	0.
P2	1	29.	49.	48.	19.
P3	2	3	13.	11.	8.
P4.	3.	7.	20.	17.	10.

Avg turnaround time = $\frac{10+48+11+17}{4.}$ = 21.5 unit

Avg waiting time = $\frac{0+19+8+10}{4.}$ = 9.25. unit

8. Give two Comparisons for the following:
    - **a. Symmetric & Asymmetric Multiprocessor systems**
    - **b. Short-term & Long-term Scheduler**
    - **c. Processes & Threads**
    - **d. Preemptive & Non-Preemptive Scheduling**
    - e. **User threads & Kernel threads**



User threads	Kernel threads.
Implemented by the user and the kernel is not aware of these threads.	→ Kernel threads are handled directly by the operating system and thread management is done by the kernel.
Easier and faster to create than the kernel threads.	→ Takes more time to be created than compared to the user threads.

Processes	Threads.
Contains stack, heap and other data segments.	→ Does not have any heaps, data segments.
Can survive even without threads.	→ Cannot survive on its own - Thread is always associated with a process.
Long ~~Short~~ term scheduler	Short ~~Long~~ term scheduler.
Collects the process from the job pool and transfers it to the memory.	→ Transfers the process from memory and assigns a CPU to it.
Runs frequently and hence takes time to collect next process	→ Frequently selects the new processes for CPU
Job scheduler	eg: CPU scheduler.
Asymmetric multiprocessor	~~A~~symmetric multiprocessor
Master processor assigns tasks to the slave processors	→ All processors are peer processors (No concept of master and slave).
Master controls all the tasks that are being performed in the system.	→ Every processor has its own register, CPU but they share memory