USN □□□□□□□□□□

CELEBRATING 25 YEARS
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test I – July 2022

Sub:	**Operating Systems**				Sub Code:	**18CS43**	Branch:	**CSE**
Date:	**09/07/22**	Duration:	**90 minutes**	Max Marks:	**50**	Sem / Sec:	**IV / A, B, C**	**OBE**

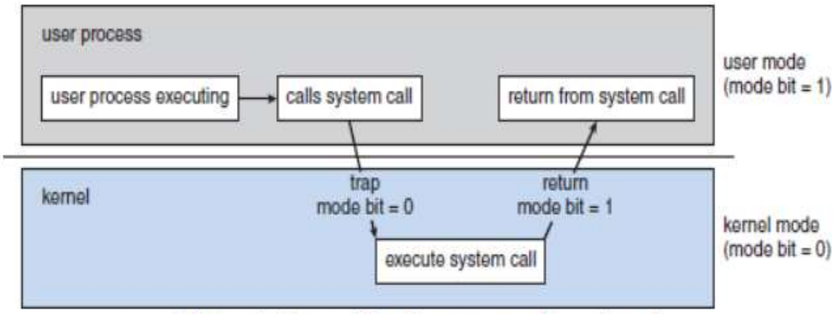
<table>
<tr><th colspan="3"><u>Answer any FIVE FULL Questions</u></th><th>MARKS</th><th>CO</th><th>RBT</th></tr>
<tr><td rowspan="2">1</td><td>a</td><td>How the privileged instructions are invoked during the execution of normal instructions.</td><td>[5]</td><td>1</td><td>L2</td></tr>
<tr><td>b</td><td>Is it possible to have more than one OS in a computer system? If so, explain with the help of architecture.</td><td>[5]</td><td>1</td><td>L2</td></tr>
<tr><td rowspan="2">2</td><td>a</td><td>Explain in detail about the various services provided by the operating system.</td><td>[5]</td><td>1</td><td>L2</td></tr>
<tr><td>b</td><td>What is the programming interface that provides interaction between user mode and kernel mode? Explain the types with examples.</td><td>[5]</td><td>1</td><td>L2</td></tr>
<tr><td rowspan="2">3</td><td>a</td><td>Elucidate the layered approach of OS structure with a supporting diagram</td><td>[5]</td><td>1</td><td>L2</td></tr>
<tr><td>b</td><td>Write short notes on the following.
a. Multi Programming and Multi-Tasking
b. Booting and Context Switch
c. CPU Scheduling and Job Scheduling</td><td>[5]</td><td>1</td><td>L2</td></tr>
<tr><td rowspan="2">4</td><td>a</td><td>In which format OS represents the Process. What are the contents of it? Briefly explain the same.</td><td>[5]</td><td>2</td><td>L2</td></tr>
<tr><td>b</td><td>What is the difference between Program and Process? Explain how the transition between the states happening.</td><td>[5]</td><td>2</td><td>L2</td></tr>
<tr><td rowspan="2">5</td><td>a</td><td>A compiler may produce assembly code, which is consumed by an assembler. Considering there is a fixed buffer, analyze how the memory will be shared between compiler and assembler by writing the logic for both processes.</td><td>[5]</td><td>2</td><td>L3</td></tr>
<tr><td>b</td><td>What is a thread? List out the benefits of multithreading? Explain the various multithreading models.</td><td>[5]</td><td>2</td><td>L2</td></tr>
<tr><td rowspan="2">6</td><td>a</td><td>Compute the Average Waiting Time and Average Turn Around Time using Preemptive and Non Preemptive Priority scheduling
<table>
<tr><th>PERSON</th><th>ARRIVAL TIME</th><th>BURST TIME</th><th>PRIORITY</th></tr>
<tr><td>Person 1</td><td>0</td><td>6</td><td>VIP</td></tr>
<tr><td>Person 2</td><td>3</td><td>5</td><td>P</td></tr>
<tr><td>Person 3</td><td>3</td><td>3</td><td>VVIP</td></tr>
<tr><td>Person 4</td><td>5</td><td>5</td><td>IP</td></tr>
</table>
*VVIP - Very Very Important Person.</td><td>[5]</td><td>2</td><td>L3</td></tr>
<tr><td>b</td><td>Consider the following snapshot of processes.
<table>
<tr><th>JOBS</th><th>ARRIVAL TIME</th><th>BURST TIME</th></tr>
<tr><td>J1</td><td>0</td><td>6</td></tr>
<tr><td>J2</td><td>2</td><td>3</td></tr>
<tr><td>J3</td><td>4</td><td>3</td></tr>
<tr><td>J4</td><td>5</td><td>5</td></tr>
</table>
Calculate the Average Waiting Time and Average Turn Around Time if it is scheduled by
a) A person who counts the shortest remaining time among all the jobs.
b) A Paani Poori Vendor with 2 seconds for each</td><td>[5]</td><td>2</td><td>L3</td></tr>
</table>

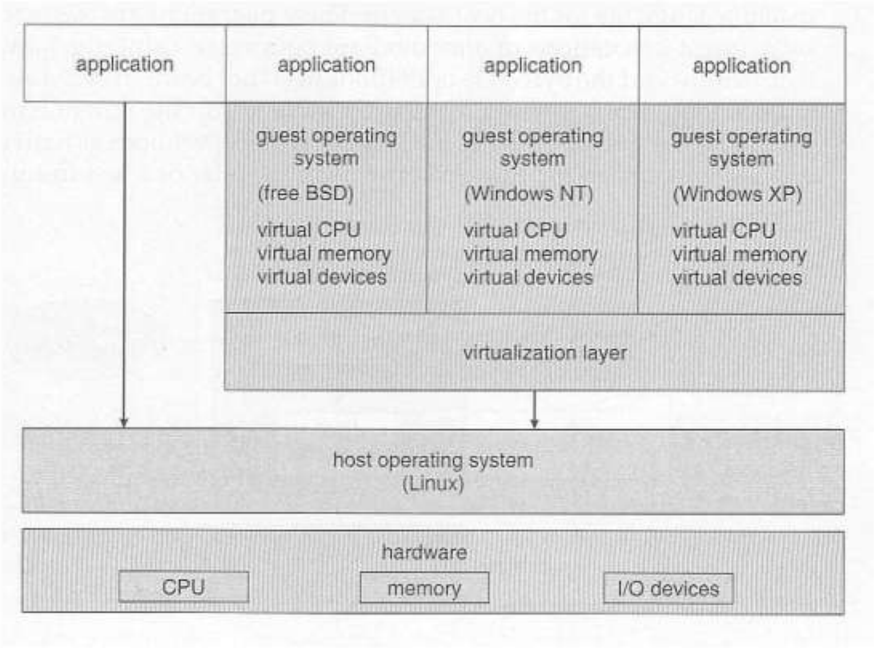
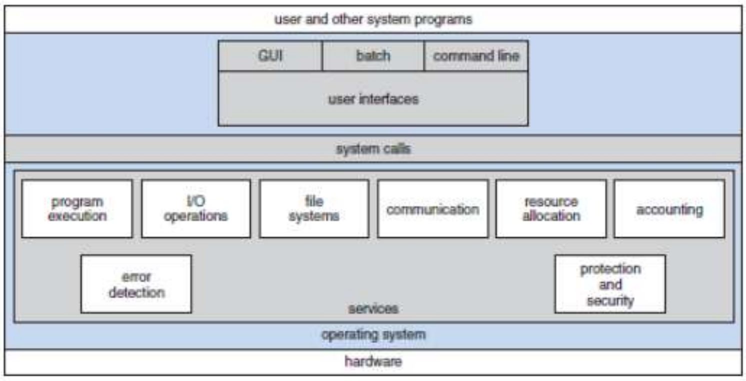
CI CCI HoD

USN □□□□□□□□□□

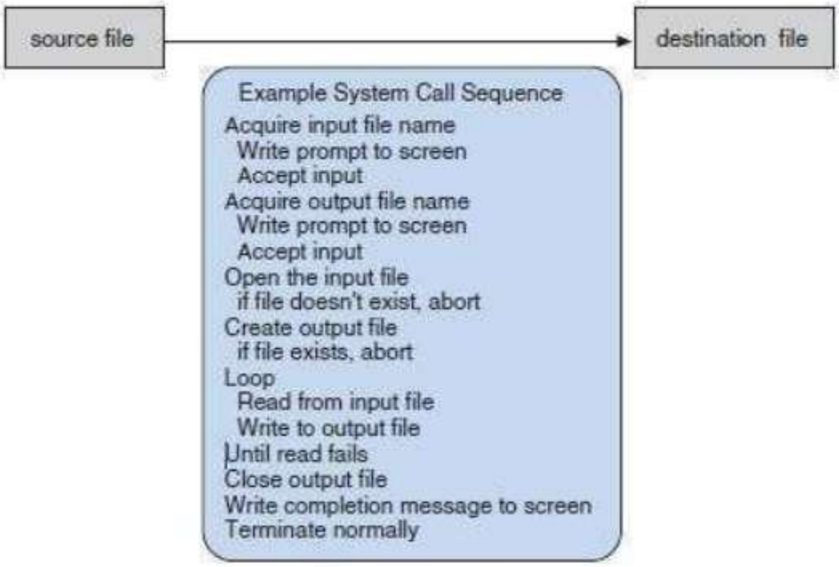
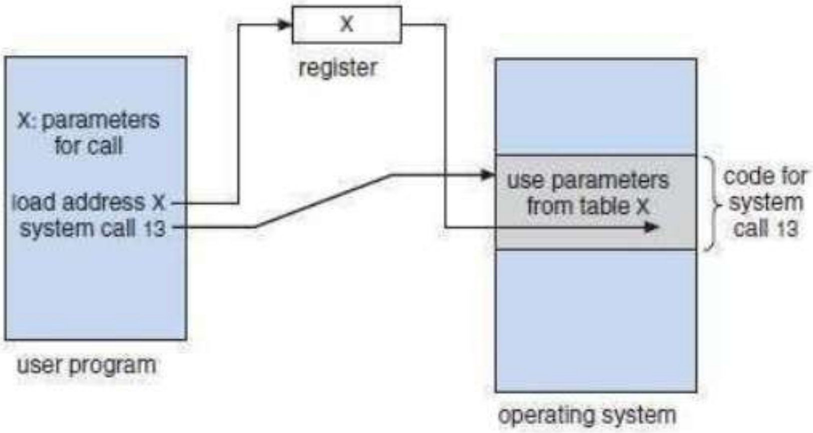
Internal Assessment Test I – July 2022

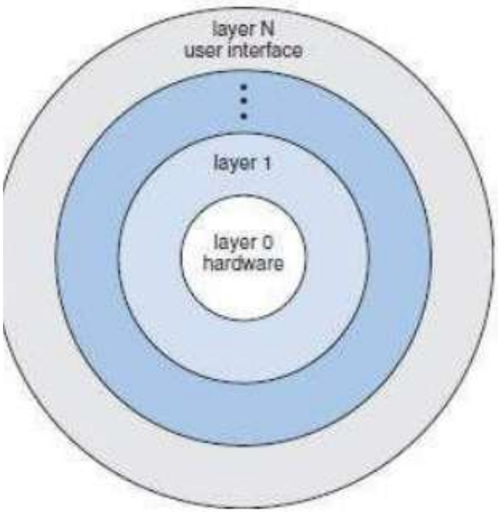
Sub:	**Operating Systems**				Sub Code:	**18CS43**	Branch:	**CSE**
Date:	**09/07/22**	Duration:	**90 minutes**	Max Marks:	**50**	Sem / Sec:	**IV / A, B, C**	**OBE**

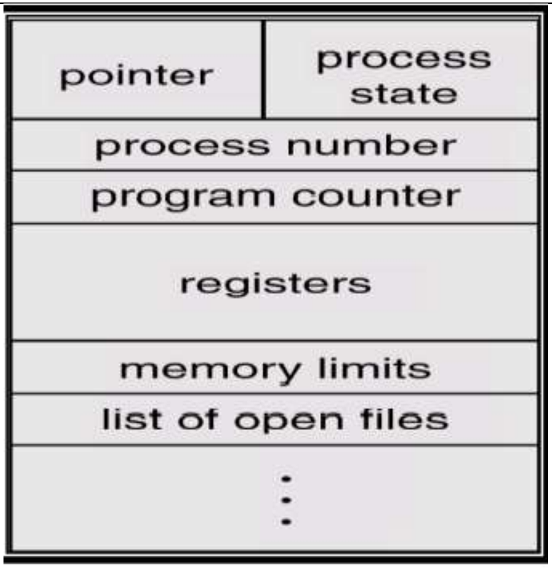
		Answer any FIVE FULL Questions	MARKS	CO	RBT
1	a	How the privileged instructions are invoked during the execution of normal instructions. We have two modes of operation (Figure 1.9): 1. User mode and 2. Kernel mode • A mode bit is a bit added to the hardware of the computer to indicate the current mode: i.e. kernel (0) or user (1)  user process user process executing calls system call return from system call user mode (mode bit = 1) kernel trap mode bit = 0 return mode bit = 1 execute system call kernel mode (mode bit = 0) Figure 1.9 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.	[5]	1	L2
	b	Is it possible to have more than one OS in a computer system? If so, explain with the help of architecture. VMware VMware is a popular commercial application that abstracts Intel 80X86 hardware into isolated virtual machines. The virtualization tool runs in the user-layer on top of the host OS. The virtual machines running in this tool believe they are running on bare hardware, but the fact is that it is running inside a user-level application VMware runs as an application on a host operating system such as Windows or Linux and allows this host system to concurrently run several different guest operating systems as independent virtual machines. In below scenario, Linux is running as the host operating system; FreeBSD, Windows NT, and Windows XP are running as guest operating systems. The virtualization layer is the heart of VMware, as it abstracts the physical hardware into isolated virtual machines running as guest operating systems. Each virtual machine has its own virtual CPU, memory, disk drives, network interfaces, and so forth.	[5]	1	L2

		 application application application application guest operating system (free BSD) virtual CPU virtual memory virtual devices guest operating system (Windows NT) virtual CPU virtual memory virtual devices guest operating system (Windows XP) virtual CPU virtual memory virtual devices virtualization layer host operating system (Linux) hardware CPU memory I/O devices			
2	a	Explain in detail about the various services provided by the operating system. Operating System Services • An OS provides an environment for the execution of programs. • It provides services to → programs and users  user and other system programs GUI batch command line user interfaces system calls program execution I/O operations file systems communication resource allocation accounting error detection protection and security services operating system hardware Figure 1.12 A view of OS services • Common functions helpful to the user are (Figure 1.12): 1. User Interface ¬ Almost all OS have a user-interface (UI). Different interfaces are: i) CLI (Command line Interface) This uses → text commands and → method for entering the text commands. ii) Batch interface ¬ Commands & directives to control those commands are entered into files, and those files are executed. iii) GUI (Graphical User Interface) ¬ The interface is a window-system with a pointing-device to → direct I/0 → choose from menus and → make selections. 2. Program Execution The system must be able to → load a program into memory and → run the program. The program must be able to end its execution, either normally or abnormally. 3. I/O Operations The OS must provide a means to do I/O operations because users cannot control I/O devices directly. For specific devices, special functions may be desired (ex: to blank CRT screen). 3. File-system Manipulation Programs need to → read & write files (or directories) → create & delete files → search for a given file and → allow or deny access to files.	[5]	1	L1

		5. Communications In some situations, one process needs to communicate with another process. Communications may be implemented via 1. Shared memory or 2. Message passing In message passing, packets of information are moved between processes by OS. 6. Error Detection Errors may occur in → CPU & memory-hardware (ex: power failure) → I/O devices (ex: lack of paper in the printer) and → user program (ex: arithmetic overflow) For each type of error, OS should take appropriate action to ensure correct & consistent computing Common functions for efficient operation of the system are: 1. Resource Allocation When multiple users are logged on the system at the same time, resources must be allocated to each of them. The OS manages different types of resources. Some resources (say CPU cycles) may have special allocation code. Other resources (say I/O devices) may have general request & release code. 2. Accounting - We want to keep track of which users use how many resources and which kinds of resources. This record keeping may be used for accounting (so that users can be billed) or gathering usage-statistics. 3. Protection When several separate processes execute concurrently, it should not be possible for one process to interfere with the others or with the OS itself. Protection involves ensuring that all access to resources is controlled. Security starts with each user having authenticated to the system by means of a password			
	b	What is the programming interface that provides interaction between user mode and kernel mode? Explain the types with examples. 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 . → 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. • 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 . 3. The use of a stack is also possible where parameters are pushed onto a stack and popped off the stack by the OS.	[5]	1	L2

		 Example System Call Sequence Acquire input file name Write prompt to screen Accept input Acquire output file name Write prompt to screen Accept input Open the input file if file doesn't exist, abort Create output file if file exists, abort Loop Read from input file Write to output file Until read fails Close output file Write completion message to screen Terminate normally Figure 1.15 Example of how system calls are used.  X register X: parameters for call load address X system call 13 user program use parameters from table X code for system call 13 operating system Figure 1.16 Passing of parameters as a table.			
3	a	Explain the layered approach of OS structure with a supporting diagram Layered Approach • The OS is divided into a number of layers. • Each layer is built on the top of another layer. • The bottom layer is the hardware. The highest is the user interface (Figure 1.19). • A layer is an implementation of an abstract-object. i.e. The object is made up of data and operations that can manipulate the data. • The layer consists of a set of routines that can be invoked by higher-layers. • Higher-layer → does not need to know how lower-layer operations are implemented → needs to know only what lower-layer operations do. Advantage: 1. Simplicity of construction and debugging • Disadvantages: 1. Less efficient than other types. 2. Appropriately defining the various layers.('.' a layer can use only lower-layers, careful planning is necessary).	[5]	1	L1

		 Figure 1.19 A layered OS			
	b	Write short notes on the following. a. Multi Programming and Multi-Tasking Multi Programming A System can be both multi programmed by having multiple programs running at the same time and multiprocessing by having more than one physical processor. The difference between multiprocessing and multi programming is that Multiprocessing is basically executing multiple processes at the same time on multiple processors, whereas multi programming is keeping several programs in main memory and executing them concurrently using a single CPU only. • Multiprocessing occurs by means of parallel processing whereas Multi programming occurs by switching from one process to other (phenomenon called as context switching). b. Booting and Context Switch Booting Booting is the process which is initiated once the computer system starts and executes small set of instructions present in the ROM which will setup system hardware by testing them and load the operating system so that computer system can carry out further tasks. Context switch An operating system uses this technique to switch a process between states to execute its functions through CPUs. It is a process of saving the context(state) of the old process(suspend) and loading it into the new process(resume). It occurs whenever the CPU switches between one process and another. Basically, the state of CPU's registers and program counter at any time represent a context. Here, the saved state of the currently executing process means to copy all live registers to PCB(Process Control Block). Moreover, after that, restore the state of the process to run or execute next, which means copying live registers' values from PCB to registers. c. CPU Scheduling and Job Scheduling	[5]	1	L1
4	a	In which format OS represents the Process. What are the contents of it? Briefly explain the same.	[5]	2	L2

		 pointer process state process number program counter registers memory limits list of open files ⋮ Information associated with each process. 1. Process state: waiting, running etc.. 2. Program counter 3. CPU registers 4. CPU scheduling information: priority if any 5. Memory-management information: page table, base and limit address 6. Accounting information: amount of cpu time used, process no. 7. I/O status information: list of I/O devices allocated, list of open files			
	b	What is the difference between Program and Process? Explain how the transition between the states happening. Program-Set of instructions Process – a program in execution; process execution must progress in sequential fashion. As a process executes, it changes state • new: The process is being created . • running: Instructions are being executed. • waiting: The process is waiting for some event to occur. • ready: The process is waiting to be assigned to a CPU. • terminated: The process has finished execution	[5]	2	L2
5	a	A compiler may produce assembly code, which is consumed by an assembler. Considering there is a fixed buffer, analyze how the memory will be shared between compiler and assembler by writing the logic for both processes. Shared-Memory Systems • Communicating-processes must establish a region of shared-memory. • A shared-memory resides in address-space of the process creating the shared-memory. Other processes must attach their address-space to the shared-memory. The processes can then exchange information by reading and writing data in the shared-memory. Bounded-Buffer assumes that there is a fixed buffer-size. Advantages: 1) Allows maximum speed and convenience of communication. 2) Faster • For ex, Producer-Consumer Problem for bounded buffer Producer-process produces information that is consumed by a consumer-process. • The following variables reside in a region of memory shared by the producer and consumer processes #define BUFFER_SIZE 10 Typedef struct {…} item; Item buffer [BUFFER_SIZE]; int in = 0; int out = 0 The shared buffer is implemented as a circular array with two logical pointers: in and out. The variable in points to the next free position in the buffer and out points to the first full position in the buffer. The buffer is empty when in = out The buffer is full when ((in+1) % BUFFER_SIZE) =out. The code for producer process is as follows:	[5]	2	L3

```
 item nextProduced;
 while (true)
/* Produce an item in nextProduced
while (((in = (in + 1) % BUFFER_SIZE ) == out);
/* do nothing -- no free buffers */ buffer [in] = nextProduced;
in = (in + 1) % BUFFER SIZE;
The code for consumer process is as follows: item nextConsumed
while (true) {
while (in == out) ; // do nothing -- nothing to consume
/*remove an item from the buffer*/
nextConsumed = buffer[out];
out = (out + 1) % BUFFER_ SIZE;
/*consume the item in nextConsumed*/
}
```

b

What is a thread? List out the benefits of multithreading? Explain the various multithreading models.

[5] 2 L1

Threads: A thread is the smallest unit of processing that can be performed in an OS. In most modern operating systems, a thread exists within a process - that is, a single process may contain multiple threads.

Multi-threaded Programming

• A thread is a basic unit of CPU utilization. • It consists of

→ thread ID

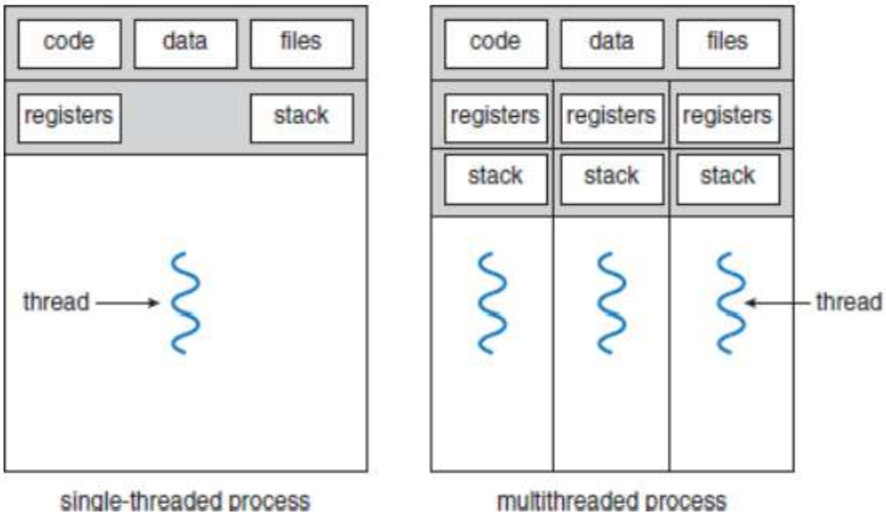
→ PC → register-set and

→a stack.

It shares with other threads belonging to the same process its code-section & data-section and other OS resources, such as open files and signals.

• A traditional (or heavy weight) process has a single thread of control.

• If a process has multiple threads of control, it can perform more than one task at a time. Such a process is called multi-threaded process (Figure 2.1).



single-threaded process multithreaded process

Motivation

• The software-packages that run on modern PCs are multithreaded. An application is implemented as a separate process with several threads of control.

Example1: A word processor may have

o → first thread for displaying graphics

o → second thread for responding to keystrokes and

o → Third thread for performing grammar checking.

Example2: A web browser may have one thread display images or text while another thread retrieves data from the network.

In some situations, a single application may be required to perform several similar tasks.

Example1: A web-server accepts client requests for web pages, images, sound, and so forth. In this case the server would create a separate thread that would listen for client requests and create another thread to service the request

Benefits

Responsiveness

• A program may be allowed to continue running even if part of it is blocked. Thus, increasing responsiveness to the user. For instance, a multithreaded web browser could still allow user interaction in one thread while an image was being loaded in another thread

Resource Sharing

• By default, threads share the memory (and resources) of the process to which they belong. Thus, an application is allowed to have several different threads of activity within the same address-space.

Economy • Allocating memory and resources for process-creation is costly. Thus, it is more economical to create and context-switch threads. For example, in Solaris, creating a process is about thirty times slower than creating a thread, and context switching is about five times slower.

Utilization of Multiprocessor Architectures • In a multiprocessor architecture, threads may be running in parallel on different processors. Thus, parallelism will be increased.

6

a

Compute the Average Waiting Time and Average Turn Around Time using Preemptive and Non Preemptive Priority scheduling

PERSON	ARRIVAL TIME	BURST TIME	PRIORITY	
Person 1	0	6	VIP	
Person 2	3	5	P	
Person 3	3	3	VVIP	
Person 4	5	5	IP	

P1	P3	P1	P4	P2	
0	3	6	9	14	19

PREMPTIVE

WT of P1= 3 WT of P2=11 WT of P3=0 WT of P4=4

Avg WT=(3+11+0+4)/4=4.5S

Avg TAT=(9+16+3+9)/4=9.25S

NONPREMPTIVE

WT of P1= 0 WT of P2=11 WT of P3=3 WT of P4=4

Avg TAT=(6+16+6+9)/4=9.25S

Avg WT=(0+11+3+4)/4=4.5s

P1	P3	P4	P2	
0	6	9	14	19

[5] 2 L3

b

Consider the following snapshot of processes.

JOBS	ARRIVAL TIME	BURST TIME
J1	0	6
J2	2	3
J3	4	3
J4	5	5

Calculate the Average Waiting Time and Average Turn Around Time if it is scheduled by

a) A person who counts the shortest remaining time among all the jobs.

J1	J2	J3	J1	J4	
0	2	5	8	12	17

Avg TAT=(12+3+4+12)/4=7.75S

Avg WT=(6+0+1+7)/4=3.5S

b) A Paani Poori Vendor with 2 seconds each

AVG WT=25/4=6.25S

AVG TAT=42/4=10.5S

J1	J2	J1	J3	J2	J4	J1	J3	J4

[5] 2 L3