OPERATING SYSTEMS

2K19/CO/319
RITIK SINGH

ASSIGNMENT - I

Q1 System Call : It is the programmatic way in which a computer program requests a service from the kernel of the operating system on which it is executed.

→ Various types of system calls are :

- Process control → Eg fork(), wait(), exec(), exit() end, abort, load, execute, create process, terminate process, get process attributes, set process attributes, wait for time, wait for event, signal event, allocate and free memory.

- File management : open(), read(), write(), close()
  Functions → create file, delete file, open file, close file, read, write, reposition, get file attributes, set file attributes.

- Device Management : open(), read(), write(), close()
  Functions → create file, delete file, open file, close file, read, write, reposition, get file attributes, set file attributes, logically attach or detach devices.

- Information maintenance : getpid(), alarm(), sleep()
  get time or date, set time or date, get system data, get system order, get process, file or device attributes, set process, file, or device attributes

vijeta

- Communications : pipe(), shmget(), mmap()
  Fuctions - create, delete communication, send receive messages, transfer status information attach or detach remote devices.

Q2 System programs are programs designed to provide a platform to other software. They provide a convenient environment for program development and execution.

They are divided into the following categories :

- File management : These programs create, delete, copy, rename print, dump, list and generally manipulate files and directories.
- Status Information : Some programs simply ask the system for the date, time, amount of available memory or disk space, number of users, or similar status information
- File modification : Several text editors may be available to create and modify the content of files stored on disk or other storage devices.
- Programming - language support : Compilers, assemblers, debuggers and interpreters for common programming languages are often provided to the user with the OS.
- Program Loading and Execution : Once a program is assembled or compiled it must be loaded into memory to be executed. The system may provide absolute loaders relocatable loaders, linkage, editors, and overlay loaders.

- **Communications:** These programs provide the mechanism for creating virtual connections among processes, users and computer systems.

Q3 Structure of UNIX

(the users)

Shells and commands, compilers and interpreters, system libraries

System-call interface to the kernal

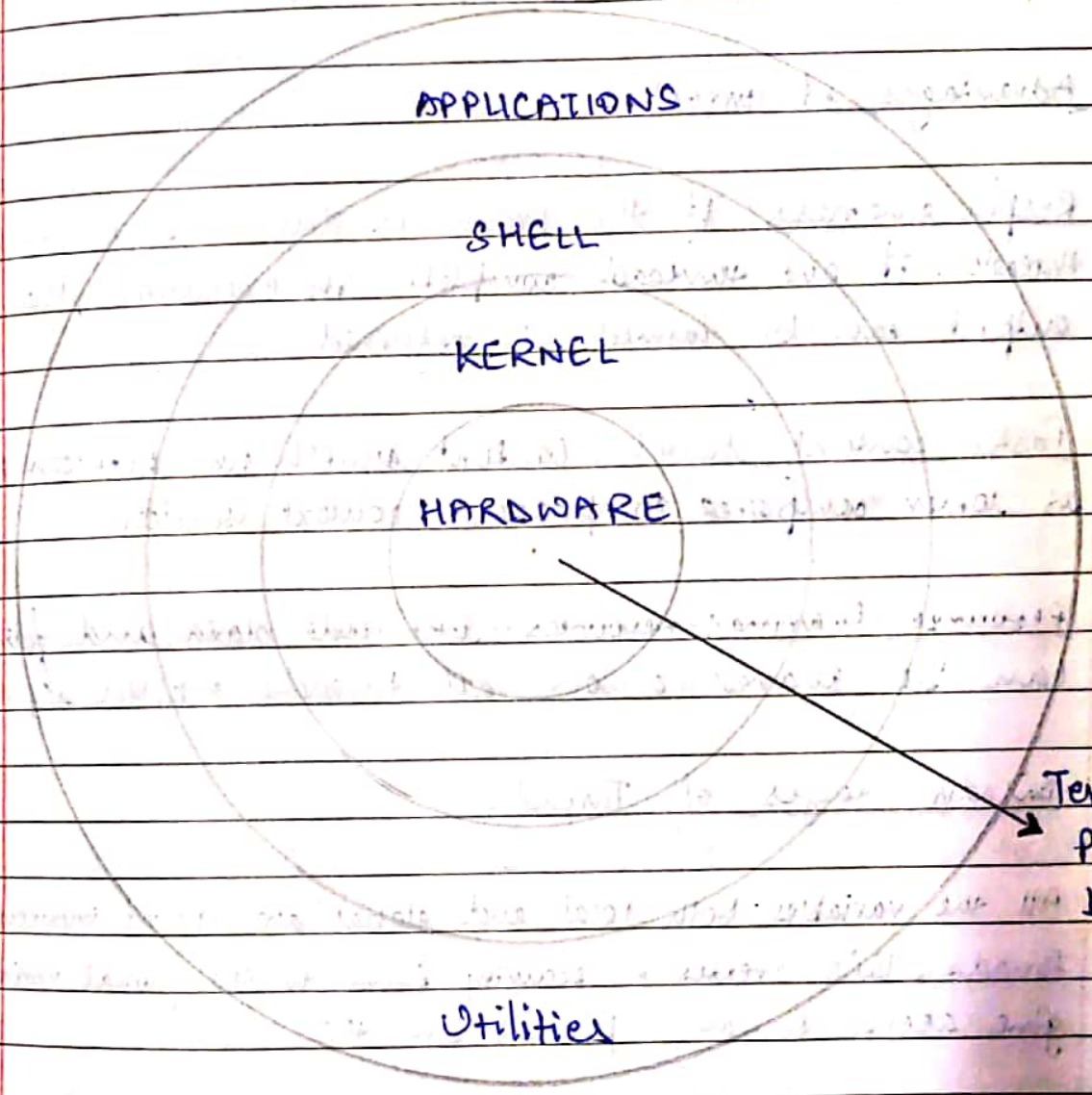
kernel

signals terminal handling	file system	CPU scheduling
character I/O system	swapping block I/O system	page replacement
terminal drivers	disk and tape drivers	demand paging
		virtual memory

Kernal interface to the hardware

terminal controllers	devices controllers	memory controllers
terminals	disks and tapes	physical memory

Structure of LINUX



Kernel is central component of an operating system that manages operations of computer and hardware. It basically manages operations of memory and CPU time. It is core component of an operating system. Kernel acts as a bridge between applications and data processing performed at hardware level using inter-process communication and system calls.

Q4 **Thread:** A thread is a path of execution within a process. A process can contain multiple threads.

Advantages of Threads

① **Responsiveness:** If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.

② **Faster context switch:** Context switch time between threads is lower compared to process context switch.

③ **Resource sharing:** Resources like code, data and files can be shared among all threads within a process.

Disadvantages of Threads

* All the variables both local and global are shared between threads. This creates a security issue as the global variables give access to any process in the system.

* Threads depend on the system and the process to run. It is not independent.

* Threads are not reusable and it requires more hardware than software due to application changes from the base.

vijeta

Advantages of Multithreading

→ Improved throughout many concurrent compute operations and I/O requests within a single process.
→ Minimized system resource usage
→ Program structure simplification
→ Better communication
→ Superior application responsiveness.

Q5 File Management commands in UNIX.

1) Files Listing: To perform files listing or to list files and directories ls command is used
eg $ls

2) Creating Files: touch command can be used to display the contents of a file.
eg $ cat filename.

3) Displaying File contents: cat command can be used to display the contents of a file.
eg $ cat filename

4) Copying a File: cp command is used to create the filename.
eg $ cp source/filename destination/

5) Moving a File: mv command could be used to move a file from source to destination.

vijeta

⑥ Renaming a file: mv command could also be used to rename files.

$ mv filename

⑦ Deleting a file: rm command could be used to delete a file

$ rm filename

Q6

process state
process number
program counter
registers
memory limits
List of open files

PCB

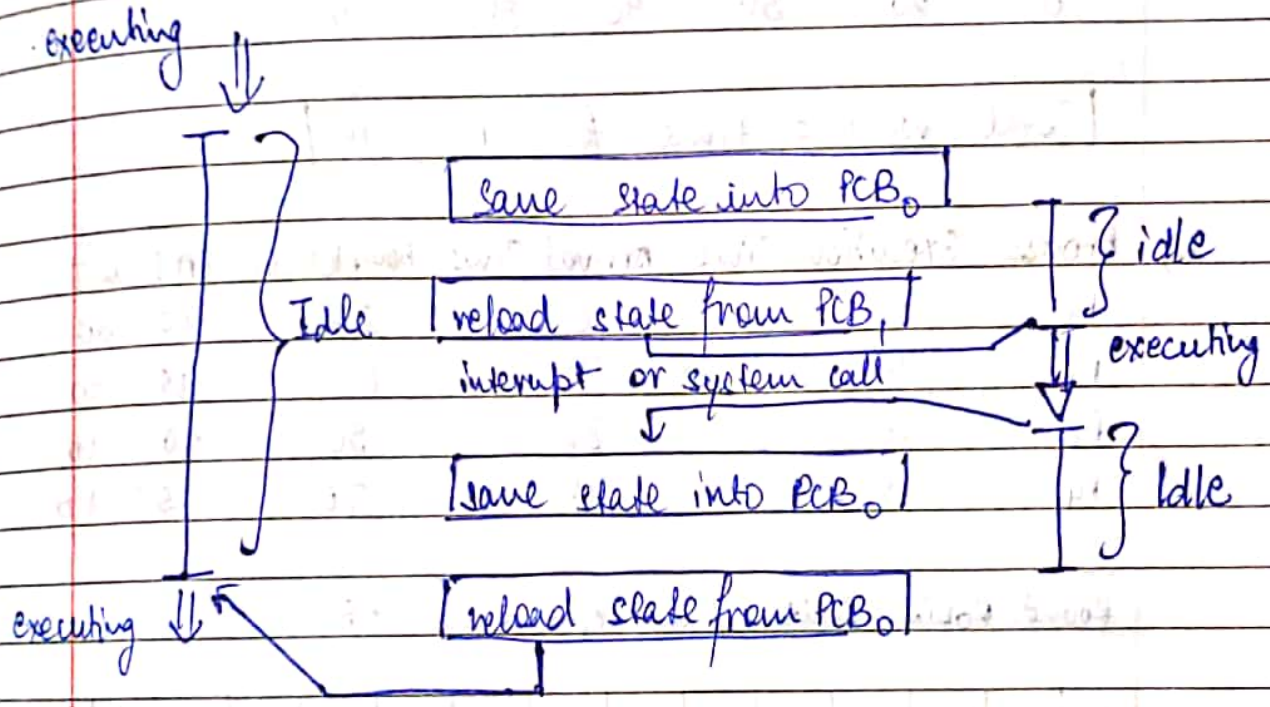
* Process state - The state may be new, ready, running, waiting or halted and so on.
* Program counter: The counter indicates the address of the next instruction to be executed in the process.
* CPU Registers: The Registers vary in number and type, depending on the computer architecture.
* CPU-scheduling information: This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

vijeta

* I/O states information : This information includes the list of I/O devices allocated to the process, a list of open files and so on.

Context Switch

When a context switch occurs the kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run.



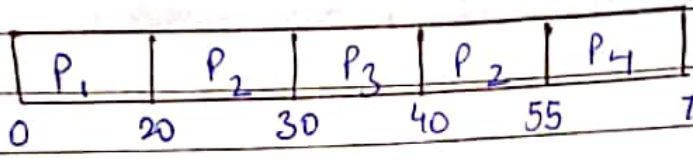
Q7 The CPU scheduling is needed to make the system efficient, fast and fair whenever the CPU becomes idle, the operating system must select one of the process in the ready queue to be executed.

The operating systems using these algorithms are: solaris, windows XP, LINUX, etc.

Q8 a)

Process	Execution Time	Arrival Time	Completion	TAT	WT
P_1	20	0	20	20	0
P_2	25	15	55	40	15
P_3	10	30	40	10	0
P_4	15	45	70	25	10

Gantt Chart (pre-emptive SJF)

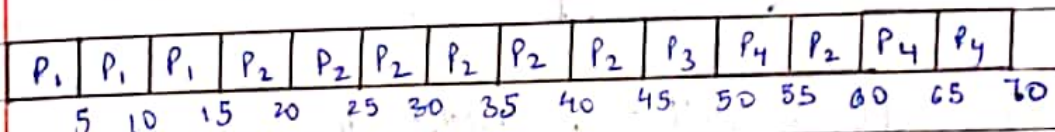


Total waiting time for $P_2 = 15$

b)

Process	Execution Time	Arrival Time	Completion	TAT	WT
P_1	20	0	25	20	00
P_2	25	15	60	45	20
P_3	10	30	50	20	10
P_4	15	45	70	25	10

Round Robin with time quantum = 5

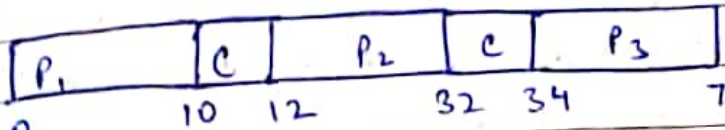


Total waiting time for $P_2 = 20$

Q9 a)

Process	Execution	Arrival	Completion	TAT	WT
P_1	10	0	10	10	0
P_2	20	2	32	30	10
P_3	30	6	74	68	38
Context Switch = 2				Avg = 36	16

Gantt Chart (Preemptive SJF)

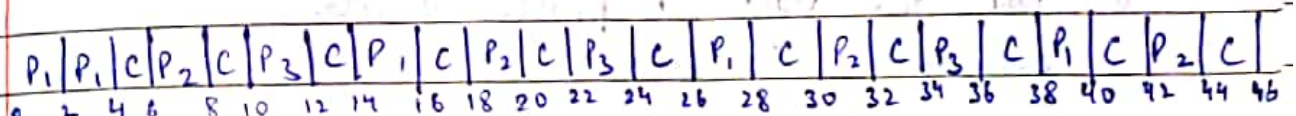


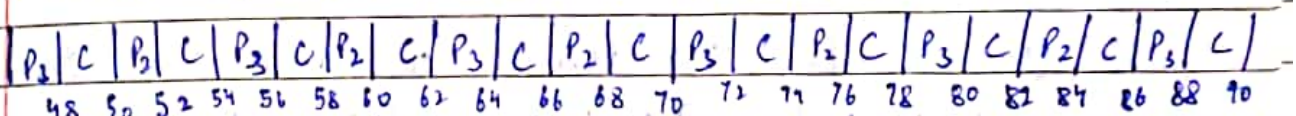
Avg. Turn Around Time = 36 time units

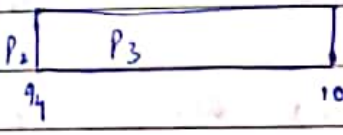
Avg. Waiting time = 16 time units

b)

Process	Execution	Arrival	Completion	TAT	WT
P_1	10	0	40	40	30
P_2	20	2	94	92	72
P_3	30	6	106	100	70
				Avg. 232/3	172/3
				= 77.3	= 57.33







9 Multiprocessor scheduling

In the multiprocessor scheduling, there are multiple CPU's which share the load that various process run simultaneously. In general, the multiprocessor scheduling is complex as compared to single processor scheduling. In multiprocessor scheduling, there are many processors and they are identical and we can

The multiple CPU's in the system are in the close communication which shares a common bus, memory and other peripheral devices. Thus these systems are tightly coupled and are used to process a bulk amount of data.

Q13 Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

Necessary conditions for deadlock occurrence.

① Mutual Exclusion: One or more than one resource are non shareable (only one process can use at a time).

② Hold and wait: A process is holding at least one resource and waiting for resource.

③ No Preemption: A resource cannot be taken from a process unless the process releases the resources.

④ Circular wait: A set of processes are waiting for each other in circular form

Methods for handling deadlock.

There are three ways to handle deadlock

1) Deadlock prevention or avoidance: The idea is to not let the system into a deadlock state.

Prevention is done by negating one of the above mentioned necessary conditions for deadlock.

Avoidance is kind of futuristic in nature. By using of avoidance, we have to make an assumption we need to ensure that all information about resources which process will need are known to us prior to execution of the process. We use Banker's algorithm in order to avoid deadlock.

2) Deadlock detection and recovery: Let deadlock occur, then do preemption to handle it once occured.

3) Ignore the problem altogether: If deadlock is very rare, then let it happen and reboot the system.

Q14 $P_1 \longrightarrow$ 2 units
$P_2 \longrightarrow$ 3 units
$P_3 \longrightarrow$ 4 units

In worst case, the number of units that each process holds = one less than its maximum demand.

So P_1 holds 1 units
P_2 holds 2 units
P_3 holds 3 units

Minimum number of units that ensures no deadlock
$= 1 + 2 + 3 + = 7$ units
7 units of R is required to ensure no deadlock.

vijeta

Q1.5 The distinction b/w kernel mode and user mode provides a rudimentary form of protection in the following manner. Certain instructions could be executed only when the CPU is in kernel mode. Similarly, hardware devices could be accessed only when the program is executing in kernel mode. Control over when interrupts could be enabled or disabled is also possible only when the CPU is in kernel mode. Consequently, the CPU has very limited capability when executing in user mode, thereby enforcing protection of critical resources.

Q1.6 The following operations need to be privileged: The set value of timer, clear memory, turn off interrupts modify entries in device-status table, access I/O device.

Q1.7 The data required by the operating system (passwords, access controls, accounting information, and so on) would have to be stored in or passed through unprotected memory and thus be accessible to unauthorized users.

Q1.10 Caches are useful when two or more components need to exchange data, and the components perform transfers at different speeds. Caches solve the transfer problem by providing a buffer of intermediate speed between the components. If the fast device finds the data it needs in the cache, it need not wait for the slower one. The data in the cache must be kept consistent with the data in the components.

vijeta

Q1.11 The client-server model firmly distinguishes the roles of the client and server under this model, the client requests services that are provided by the server. The peer-to-peer model doesn't have strict roles. In fact, all nodes in the system are considered peers and thus may act as their clients or servers or both. A node may request a service from another peer, or the node may in fact provide such a service to other peers in the system.

Q4.2 In order to calculate the speedup gain of an application that has a 60% parallel component using Amdahl's law.

$$t = \frac{1}{(S + (1-S))/N}$$

a) $N = 2$, $S = 40\% = 0.4$

$$\Rightarrow t = \frac{1}{0.4 + 0.6/2} = \frac{1}{0.7} = 1.428 \text{ times}$$

b) $N = 4$, $S = 40\% = 0.4$

$$\Rightarrow t = \frac{1}{0.4 + 0.6/4} = \frac{1}{0.55} = 1.8281 \text{ times}$$

Q4.3 It exhibits data parallelism. Each thread is performing the same task, but on different data.

vijeta

Q4.6 Because a thread is smaller than a process, thread creation typically uses fewer resources than process creation. Creating a process requires allocating a PCB, a rather large data structure. The PCB includes a memory map, a list of a user thread or a kernel thread involves allocating a small data structure to hold a register set, stack and priority.

Q4.9 When a kernel thread suffers a page fault, another kernel thread can be switched in to another kernel thread use the interleaving time in a useful manner. single threaded process on the other hand, will not be capable of performing useful work when a page fault takes place. Therefore in scenarios where a program might suffer from request page faults or has to wait for other system events, a multithreaded situation would perform better even on a single processor system.