Friday, October 22, 2021 11:33 AM

Process	Arrival time	Tcpu
P1	0	6
P2	2	2
P3	5	1
P4	1	7

Calculate avg Wt, Ta using Preemptive SJF!!

Queue=P1	Queue= P4,P1	Queue=P4	Queue=P4,P1	Queue=P4			
P1 0---2	P2 preempts P1 2-----4	P1 resumes 4----5	P3 preempt P1 5---6	P1 resumes 6------9	P4 9----16		

Process	Wt	Ta=Wt + Tcpu
P1	0-0=0 4-2=2 6-5=1	3 + 6=9
P2	2-2=0	0 + 2 = 2
P3	5-5=0	0+1 = 1
P4	9-1=8	8+7=15
Avg	3+0+0+8=11 11/4=2.75	9+2+1+15=27 27/4=6.75

Context Switching =

Process once its started MAY not use CPU continuously !!!!!!

1. After timer interrupt time slice is over process returns to ready queue (RR)
2. If IO instruction is the next instruction ---CPU is not used ,process goes to wait state, then to ready state

Next time when the PROCESS resumes execution ------

It should start from the point it had left !!!

How is the kernel tracking which point the process left ???

Kernel maintains a **snapshot** of CPU registers just before the process leaves CPU

Kernel copies all the current values of the CPU registers in a data structure called as CONTEXT

Later the Kernel loads the CONTEXT into the CPU registers when the process resumes

Context Switching -----

SAVING context of outgoing process

LOADING context of incoming process

Process Management ----

Process= program in execution
Process space /address space = space allocated in the RAM that has code, data ,stack,heap
PCB = Process Control Block = info about process , one block per process
Process Life Cycle = create, ready, running, wait, terminate
What happens in create state = PID,PCB allocated, address space loaded in RAM
What is Ready State = queue in which process waits for CPU
What is running state = process uses CPU
When does process go to wait state = when next instr is IO instr process waits in wait state
From which states a process may go to ready state = created, running(preempt, interrupt) , wait state
When does the process go to terminate state = last line of code
STARVATION = process does not get CPU as higher priority processes keep coming
Turnaround Time = Wt + Tcpu (should be less)
Preemption = when higher priority process forcefully replaces the running process
MAJOR advantage of RR = Multitasking (give a **feel** of simultaneous execution)

System call = fork ()

What is a system call ? It is just a function call given to function which are in the Kernal mode

Just a function call . This function is in Kernel space and function runs in Kernel mode (privileged instructions)

System calls are different for different OS . Linux system calls are different from windows system calls !!
Your syllabus has linux system calls.

Your program is in User Space -------> system call () ---------->Kernel Space function

1. System call ---- getpid() , getppid() =

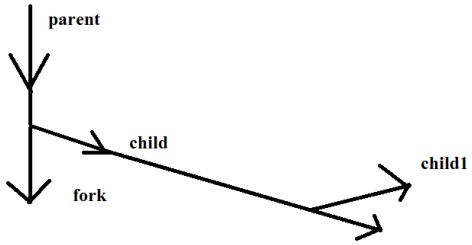
We run a C program and we display the pid allocated to the process using printf .

In Linux based OS ---- every process has a PPID = Parent Process ID

The first process has pid 0 and no ppid ==== it is the init process
The init process will spawn new CHILD processes that have more child processes

---

Linux creates a new process by a mechanism ------ FORKING



When I open a terminal = BASH program starts from INIT

INIT (pid =10)

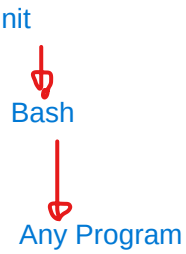
|

|

|_____BASH (pid =11, ppid =10) _____

|

|



BASH (pid= 11)

|

|

|---------- ./simple (pid = 134, ppid =11)

Simple (134)

|

|

| ----------fork() ---------- CHILD (pid =135 , ppid =134)