CENG 328 - Operating Systems

Resitation Hour **30.03.2007**

1) You are to compare reading a file using a single-threaded file server and a multithreaded server. It takes 15 msec to get a request for work, dispatch it, and do the rest of the necessary processing, assuming that the data needed are in the block cache. If a disk operation is needed, as is the case one-third of the time, an additional 75 msec is required, during which time the thread sleeps. How many requests/sec can the server handle if it is single threaded? If it is multithreaded? (Hint: 1000 msec= 1 sec)

Answer: In the single-threaded case, the cache hits take 30 msec and cache misses take (15+75) 90 msec. The weighted average is 2/3*15+1/3*90=40. Thus the mean request takes 60 msec and the server can do 1000/40 per second. For a multithreaded server, all the waiting for the disk is overlapped, so every request takes 15 msec, and the server can handle 1000/15 requests per second.

2) What is meant by the term context switch? What might cause a context switch to occur?

Answer: Switching between processes is termed as context switch. When the CPU switches to another process, the system must save the state of the old process and load the saved state for the new process.

3) Consider the following sets of processes, with the length of the CPU-burst time is given in milliseconds. Arrival time is the time at which the process is added to the ready queue.

Processes	Burst Time	Arrival Time
P1	30	0
P2	15	0
P3	9	0
P4	12	0
P5	6	12
P6	3	18

a. Draw appropriate charts illustrating the execution of these processes using FCFS, SJF non-preemptive, and SJF preemptive.
b. Calculate the wait time of each process for each strategy. Calculate the average wait times under each strategy.

Answer:

Process	FCFS	SJF-nonpreemptive	SCF-preemptive
P1	0	45 (9+12+3+6+15)	45 (9+6+3+12+15)
P2	30 (30)	30 (9+12+3+6)	30 (9+6+3+12)
P3	45 (30+15)	0	0
P4	54 (30+15+9)	9 (9)	18 (9+6+3)
P5	54 (30+15+9+12-12)	12 (9+12+3-12)	0
P6	54 (30+15+9+12+6-18)	3 (9+12-18)	0
Average	237/6=39.5	99/6=16.5	93/6=15.5

4) Three jobs (A, B, and J) arrive to the job scheduler at time 0. Job A needs 10 seconds of CPU time, Job B needs 20 seconds, and Job C needs 30 seconds.

a. What is the average turnaround time for the jobs, assuming a shortest-job-first (SJF) scheduling policy?
b. What is the average turnaround time assuming a longest-job-first (LJF) scheduling policy?
c. Which finishes first, Job C in SJF or Job A in LJF?

Answer:

a. Turnaround time: The total time that the job spends in the system (from when it was submitted to when it completes). SJF runs A to completion, then B to completion, and finally C to completion. A finishes in 10 seconds, B finishes in 30 seconds (10+20), and C finishes in 60 seconds (10+20+30). Average turnaround time is thus (10+30+60)/3, that is 100/3 seconds.
b. LJF runs C to completion, then B to completion, and finally A to completion. C finishes in 30 seconds, B finishes in 50 seconds (30+20), and A finishes in 60 seconds (30+20+10). Average turnaround time is thus (30+50+60)/3, that is 140/3 seconds.
c. C finishes last in SJF (60 seconds) and A finishes last in LJF (60 seconds). Both finishes at the same time, so the answer is neither of them finishes first.

5) Describe the Round-Robin scheduling algorithm. Discuss what issues need to be consider good performance from this algorithm?

Answer: Round-Robin reduces the penalty that short jobs suffer with FCFS by preempting running jobs periodically. The CPU suspends the current job when the reserved quantum (time-slice) is exhausted. The job is then put at the end of the ready queue if not yet completed. The critical issue with the Round-Robin policy is the length of the quantum. If it is too short, then the CPU will be spending more time on context swithcing. Otherwise, interactive processes will suffer.

8) a. A system has two processes and three identical resources. Each process needs a maximum of two resources. Is deadlock possible?

b. Consider previous problem again, but now with **p** processes, each needing a maximum of **m** resources and a total of **r** resources available. What condition must hold to make the system deadlock free?

Answer:

a. The system is deadlock free, because, suppose that each process has one resource. There is one resource free. Either process can ask fro it and get it, in which case it can finish and release both resources. Consequently deadlock is impossible.

b. If a process has m resources, it can finish and cannot be involved in a deadlock. Therefore, the worst case is where every process has m-1 resources and needs another one. If there is one resource left over, one process can finish and release all its resources, letting the rest finish too.

9) A computer has six tape drives, with ***n*** processes competing for them. Each process may need two drives. For which value of ***n*** is the system deadlock free?

Answer: With three processes, each one can have two drives. With four processes, the distribution of drives will be (2, 2, 1, 1) allowing the first process to finish. With five processes, the distribution will be (2, 1, 1, 1, 1) which still allows the first one to finish. With six processes, each holding one tape drive and waiting another, we have a deadlock. Thus, for $n < 6$, the system is deadlock free.

10) What is starvation, give an example?

Answer: Starvation is a multitasking-related problem, where a process is perpetually denied necessary resources. Without those resources, the program can never finish its task. It occurs when one program holds resources that the other needs, but it is unwilling to give them up.