CENG 328 - Operating Systems

Resitation Hour **(15-16).05.2006**

1) In a system with pure paging, assume we have a 48-bit address space, and a 64 KB page size.

a. How many bits of an address specify the logical page number (a.k.a. the virtual page number), and how many bits specify the offset? (Hint for part a.: 2^{16}=65536)
b. Let's say we are translating the logical address 0x000000100033; if each logical page is mapped to a physical page that is a single page number higher (i.e., logical page 10 is mapped to physical page 11, logical page 11 is mapped to physical page 12), what is the translated physical address? (Hint for part b.: 16^4=65536)

a. 64 KB (2^{16}=65536 bytes) page implies <u>16 bits</u> needed for the offset. This leaves <u>32 bits</u> for virtual page number.
b. Logical address=0x000000100033. First we have to take virtual page number and increment it by one. Because the virtual page number is the top 32 bits of the logical address, we can get it as 0x00000010. Adding one to this, we have 0x00000011. Now, by combining this with the offset, finally we get the physical address, which is <u>0x0000001100033</u>.

2) One way to use contiguous allocation of the disk and not suffer from holes is to compact the disk every time a file is removed. Since all files are contiguous, copying a requires a seek and rotational delay to read the file, followed by the transfer at full speed. Writing a file back requires the same work. Assuming a seek time of 5 msec, a rotational delay of 4 msec, a transfer rate of 8 MB/sec, and an average file size of 8 KB, how long does it take to read a file into main memory then write it back to the disk at new location? Using these numbers, how long would it take to compact half of a 16-GB disk?

It takes us 9msec to start the transfer (5msec of seek time + 4msec of rotational delay). To read a file of size 8KB (2^{13}=8192 bytes) at a transfer rate of 8MB/sec (2^{23}=8388608 bytes/sec) requires 977 msec, and by adding this to 9msec, we get a total of 9.977 msec. Writing it back again takes another 9.977 msec. So, copying a file takes 19.954 msec. To compact half of a 16-GB disk would involve copying 8-GB (8589934592 bytes) of storage, which is 2^{20} (1048576) files. At 19.954 msec per file, this takes 20.923 sec (19.954*1048576=20923285,504 msec), which is 5.8 hours. Clearly, compacting the disk after every file removal is not a great idea.

3) You are to compare reading a file using a single-threaded file server and a multithreaded server. It takes 30 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 90 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)
In the single-threaded case, the cache hits take 30 msec and cache misses take (30+90) 120 msec. The weighted average is 2/3*30+1/3*120=60. Thus the mean request takes 60 msec and the server can do 1000/60 per second. For a multithreaded server, all the waiting for the disk is overlapped, so every request takes 30 msec, and the server can handle 1000/30 requests per second.

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

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.

5) 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.

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

6) 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?

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.

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

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?

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.