CMSI 587

OPERATING SYSTEMS (GRADUATE LEVEL)

Spring 2006

Assignment 0321

Just a few exercises to solidify our understanding of main memory management…

Not for Submission

Read Chapters 8 and 9 in SGG; we covered Chapter 8 this week and will cover Chapter 9 next week.

For Submission 1: Exercises

Do the following exercises from SGG; submit your responses in hardcopy only. In case your edition is different from the 7th, I've added a note on the type of question involved (or even the entire question itself if it isn't too long); if you can't find the equivalent in your edition, then let me know and I can pass a copy to you.

1. SGG Exercise 8.3 — Given five memory partitions of 100, 500, 200, 300, and 600K (in that order), show how first-fit, best-fit, and worst-fit algorithms would place processes of size 212, 417, 112, and 426K (also in that order), and state the algorithm that makes the most efficient use of memory.
2. SGG Exercise 8.9 — For a page table stored in memory, (a) if a memory reference takes 200ns, how long does a paged memory reference take? (b) with TLBs, a 75% TLB hit ratio, and zero TLB access time, what is the effective memory reference time?
3. SGG Exercise 8.12 — Given a segment table with (ID, base, length) entries of (0, 219, 600), (1, 2300, 14), (2, 90, 100), (3, 1327, 580), and (4, 1952, 96), provide the physical/linear addresses for the logical addresses (0, 430), (1, 10), (2, 500), (3, 400), and (4, 112). *Tip:* You may answer this question using this week's programming assignment, but make sure that you're also comfortable with doing this by hand.

For Submission 2: Programming

Write a set of routines that simulates segmented memory management as defined by the *segment.h* interface given in class.

1. Provide a *segment.c* that implements the functions defined in *segment.h*.
2. Provide a non-interactive *segmentTest.c* test program that sets up a segment table and checks to see if your implementation returns the correct physical addresses and errors for a variety of logical addresses. No massively fancy unit testing framework this time around; just use the *assert.h* library to perform your tests.

Commit your work to your Keck lab CVS repository under the path *homework/cmsi587/hw0321*. Place source code in a *src* subdirectory; if you have any other files, such as notes, documentation, configuration, or build files, commit those at the top-level directories. As usual, make sure that your *.cvs* subdirectory (the physical location of your Keck CVS repository) is accessible to users other than yourself.