COSC 6360 MIDTERM MARCH 7, 2001

This exam is **closed book**. You can have **two sheets** (i.e., **four pages**) of notes.
Please answer every part of every question

1. Consider a clock policy with two hands where the second hand follows the first hand at a fixed angle of 45 degrees. Assuming that both hands complete a full rotation each *x* seconds, what are the minimum and a maximum times a page can stay in memory after having been referenced for the last time? (2×5 points)

   It will stay in memory for at least __*x*/8_____ seconds.

   It can stay in memory for at most _9*x*/8_____ seconds.

2. Describe the contents of a Unix ***directory entry***. (2×5 points)

   A UNIX directory entry contains (a) a file name and (b) the number of of the corresponding i-node (i-number).

3. Why is it a good idea to replicate *superblocks*? (10 points)

   Because losing the superblock of a disk partition means the permanent loss of all data in that partition.

4. What is *complete-subblocking*? (5 points) What is its major advantage? (5 points) Describe in some detail a page table organization that would support complete-subblocking with a blocking factor of 4? (10 points: you will lose five points if your answer does not include a diagram)

   Complete subblocking associates with each page table entry a set of contiguous virtual pages and their corresponding addresses in main memory. Its major advantage is that it saves a lot of space. For instance, a page table organization that supports complete-subblocking with a blocking factor of 4 would require 6x8 bytes per entry to map four virtual pages. A conventional page table entry would occupy 3x8 bytes and map a single virtual page.

5. Why does Spring group all threads directly or indirectly involved in a cross-domain call into a *single entity*? (10 points)

   To create a single scheduling entity while preserving thread identities. Under conventional mechanisms a remote procedure call moves the receiving thread from the waiting to the ready state. Thanks to shuttles, the receiving thread can "inherit" the CPU from the calling thread.

6. What feature of Mach does allow it to implement the Unix **`fork()`** system call more efficiently than other Unix kernels? (5 points) How? (5 points) Could you think of circumstances where this implementation could be *less efficient*? (5 points) Why? (5 points)

Mach implements fork() more efficiently than other Unix kernels thanks to copy-on-write. After a fork(), the shared data segment of the parent process is shared with the child in copy-on-write mode: no actual copying will take place unless one of the two processes writes into this shared data segment before the child does an exec(). Copy-on-write works very well because most child processes do an exec within a few instructions after the fork(). Conversely, copy-on-write does not work well whenever the child does not do an exec and either the parent or the child process modifies large portions of their respective address spaces.

7. What is *false sharing* in a distributed shared memory system? (5 points) What problem does it cause? (5 points) Which feature of Munin addresses that issue? (5 points) How? (5 points)

False sharing happens every time two distinct processors access two distinct variables that happen to be located in the same DSM transfer units. These transfer units happen to be pages in Munin. False sharing results in continuous exchanges of the transfer unit between the two processors.

Munin fights false sharing by offering a write-shared consistency protocol. Whenever a process is granted access to write-shared data, the page containing these data is marked copy-on-write. Hence the first attempt to modify the contents of the page will result in the creation of ac copy of the page modified (the twin). At release time, the DSM will perform a word by word comparison of the page and its twin, store the diff in the space used by the twin pages and notify all processes having a copy of the shared data of the update.