COSC 6360 FIRST MIDTERM FEBRUARY 18, 2004

THIS EXAM IS CLOSED BOOK. YOU CAN HAVE ONE SHEET (I.E., TWO PAGES) OF NOTES.
TO GET FULL CREDIT, YOU MUST ANSWER EVERY PART OF EVERY QUESTION.

1. Which feature of Mach allows it to implement more efficiently the UNIX **fork()**? (5 points) How is their solution better than the Berkeley UNIX **vfork()**? (5 points)

The feature is called _ copy-on-write _____

It is better than the UNIX vfork() because it is more _ general _____
(vfork() only works when the fork() is quickly followed by an exec())

2. 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 *four* seconds, what are the minimum and a maximum times a page is guaranteed stay in memory after having been referenced for the last time? (2×5 points)

At least _ 45/360 of a rotation = 0.5 _____ seconds

and at most _ 405/360 of a rotation = 4.5 _____ seconds.

3. 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 unit. These transfer units happen to be pages in Munin. False sharing results in continuous exchanges of the whole 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.

4. What is the main advantage of *mapped files*? (5 points) Why would they be very hard to implement on a conventional virtual memory system? (5 points) Which features of the Mach virtual memory system make their implementation feasible and **why**? (2×5 points)

Mapped files reduce the number of context switches required to implement I/O operations because it allows user processes to access mapped files through library calls instead of through system calls.

Mapped files are very hard to implement on a conventional virtual memory system because these systems treat all pages in the data segments of their processes in the same fashion.

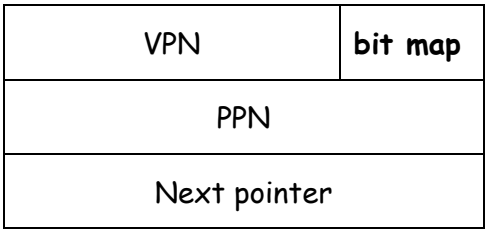
The two features of Mach that allow the implementation of mapped files are:

1. External pagers: since they allow mapped file pages to be paged from the file system rather than from the swap area of the virtual memory system.
2. Inheritance: since it allows mapped file pages to be inherited in shared mode.

5. Explain the function and the purpose of the set user-ID bit in UNIX. (10 points)

Any file that has its set user ID flag set will execute with the rights of the file owner rather than with the rights of the user running it. This allows users to execute programs accessing data whose access needs to be controlled

6. Consider a 64-bit system with page size of 8 kilobytes. Describe a page table organization that would interact in an effective fashion with a TLB using *partial subblocking* with a subblocking factor of 8. (10 points if your answer includes a correct diagram). How would your page table organization handle both *64-kilobyte and 512-kilobyte superpages*? *(*(2×5 points)



Every page table entry will occupy 3x8=24 bytes and map eight contiguous page tables into eight contiguous physical pages.

Since each 64-kilobyte superpage contains exactly eight pages, each superpage will be mapped into a single page table entry.

There are two good solutions for handling 512-kilobyte superpages: one could either have eight separate page table entries per superpage or use an "ad hoc" mechanism .

7. What makes the choice of an optimal block size so difficult in a file system? (5 points) How is FFS solving the problem? (5 points)

Selecting the optimal block size for a file system is a difficult task because selecting a small block size will require more I/O operations to access the same amount of data while large block sizes cause internal fragmentation.

Berkeley UNIX requires the block size to be at least 4 kilobytes but fight internal fragmentation by allowing file blocks to be partitioned into 2, 4, or 8 smaller fragments.