COSC 6360 **FIRST MIDTERM** **JULY 19, 2004**

This exam is **closed book**. You can have **one sheet** (i.e., **two pages**) of notes.
Please be specific and answer every part of every question.

1. Consider a Mach process containing, a code segment, a data segment, a mapped file and a shared memory segment. What would be the normal inheritance attributes of

(a) Its code segment? (5 points) ______ share

(b) Its data segment? (5 points) ______ copy (using copy-on-write)

(c) Its mapped file? (5 points) ______ share

(d) Its shared memory segment? (5 points) ______ share

2. What are migratory variables? (5 points) How does Munin treat them differently from ordinary shared variables? (5 points)

Migratory variables are variables that migrate among the processes accessing them: every process accessing a migratory variable will always get full read and write access even if it only requested read access.

3. How does the current Berkeley UNIX page replacement policy differ from that described by Babaoglu and Joy? (6 points) What factor motivated this change? (4 points).

A second hand, following the original clock hand at a fixed angle, was added to reclaim faster pages that have been marked not referenced

This was done because computer of the late 80's had much larger memory sizes than computer of the late 70's. As a result, the hands of their clocks had a much slower angular speeds. Adding a second hand ensured that pages that were not currently used could be reclaimed in a timelier fashion.

4. Unlike Windows, UNIX typically merges all its disk partitions into a single directory hierarchy. What is the main advantage of this approach? (5 points) Which mechanism does UNIX use to implement it? (5 points)

Since all disk partitions are included in a single directory hierarchy, programs can address all files through their path name in that hierarchy and never have to mention device names as Windows does. Hence we can move files between disk partitions without having to reinstall any program as long as we do not modify the directory as long as we do not modify the existing directory hierarchy.

The tool making this possible is the UNIX **mount**.

5. Consider a virtual memory system with a 4 KB page size and a TLB that uses *partial subblocking* with a subblocking factor of 4.

(a) What would be the best page table organization for this system? (10 points including 5 point for a detailed drawing of a page table entry)

I would use a page table entry with three entries

i) The virtual page number of the first virtual page in the subblock;

ii) The physical page number of the first page frame in the subblock;

iii) A pointer to the next page table entry.

The four valid bits and the four dirty bits for the four pages in the subblock would be stored in the unused portion of the virtual page number of the first virtual page of the subblock.

Virtual Page Number	Bits
Physical Page Number	
Next Pointer	

(b) Assuming that the kernel of this system uses a single 512-kilobyte superpage, how would your virtual memory system handle this superpage? (5 points)

I would use an ad-hoc implementation for that single superpage.

(c) How would your virtual memory system handle 64KB superpages? (5 points)

Since each superpage would consist of 16 pages, it would occupy four page table entries.

6. What is the function of the UNIX set user-ID bit? (5 points) Give one example of its practical use. (5 points) and one example of how it can be misused (5 points).

The set user-ID bit specifies that a given file should be executed with the rights of its owner rather than with the right of the user executing it. This feature can be used to let users execute programs accessing data whose access must be controlled. Big security problems will occur if a file with the set user-ID on can be modified by anyone but its owner because the other users could replace the file by their own version of the shell.

7. What makes a ***fork()*** system call so costly? (5 points) Which steps are taken by (a) Berkeley UNIX (5 points) and (b) Mach (5 points) to make forks less costly?

The UNIX fork its costly because it has to copy the contents of the parent address space into the child address space. Berkeley Unix has a **vfork()** primitive that shares the same address space between parent and child until the child process does an **exec()**. Mach uses **copy-on-write**.