COSC 6360 **MIDTERM** **FEBRUARY 27, 2002**

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

1. Explain how you could use an *exokernel* to support multiple user interfaces on the same machine, say, UNIX and VMS. (5 points)

By having different libOSes exporting different user interfaces.

2. Alice manages a research project involving Bob, Carol, David, Emily and Francis. All project files are maintained on a UNIX system and all people working on the project have personal accounts belonging to the same group (let call it **`ateam`**). One morning, Alice learns that Francis is defecting to another team within the same organization. Given that Alice cannot do anything to Francis UNIX account, how she can prevent him—and him *alone*—from accessing the files she shares with the other members of her team. What steps would that involve? (10 points)

Alice must ask the system administrator to remove Francis' UNIX account from the ateam group. [Her other option is to ask the system administrator to create a new user group without Francis and change the group ownership of all the project files. The first option is the best.]

3. What is the major limitation of UNIX *pipes*? (5 points)

Pipes are only passed from parents to children. Hence we cannot use them to transmit data between unrelated processes. [Pipes are also unidirectional but that is not such a drastic limitation as bi-directional communication can be achived through pair of pipes.]

4. The UNIX method of creating a process through a **`fork()/exec()`** pair is notoriously inefficient. Why? (5 points) Describe two solutions to this problem. (2×5 points)

The inefficient step of the fork()/exec() pair is the fork() system. It copies the data segment of the parent process even though this segment will be overwritten by the exec(). Berkeley Unix has a vfork() primitive that shares the same address space between parent and child until the child process does an exec(). Mach has *copy-on-write* mechanism that only copies that pages that are modified by either the parent and the child.

5. What is an *advisory lock*? (5 points) What are their advantages and disadvantages over conventional locks? (2×5 points)

Advisory locks are only enforced for processes that request them: processes that ignore them can always get access to the data, Hence they can be easily broken by "reckless" processes, which is a clear disadvantage. The same feature is also an advantage as it eliminates the risk of deadlocks .

6. *Shared memory segments* are segments that are shared among several processes. How would you implement them in Mach? Your answer should include to the following questions:

a) Which Mach feature would you use? (5 points)

Each shared memory segment would have a separate entry in the address map of the process sharing the page.

b) Which would be the inheritance attribute of the shared memory segment? (5 point)

The inheritance attribute of the shared memory segment would be SHARED.

c) How it would be paged? (5 points)

As a separate memory object managed by the virtual memory system.

7. Comparing *partial subblocking* and *complete subblocking*, explain in a few words:

a) Which one makes the most efficient use of the TLB memory. (5 points)

Partial subblocking because it only requires one PPN per TLB entry.

b) Which one makes the most efficient use of the main memory. (5 points)

Complete subblocking because it allows the page frames containing the pages in a subblock to be placed anywhere in main memory.

8. In which sense is a Spring *door* a *capability*? (5 points)

Because each process having a door in its door table has the right to make a call to a specific entry point in a specific process.

9. Munin is said to have an *eager release* policy. Describe this policy (5 points) and explain how it differs from a *lazy release* policy. (5 points)

Munin has an eager release policy because the distributed shared memory is completely resynchronized after each release. A lazy release policy delays all distributed shared memory updates until another process issues a request

10. What are the actions to be taken by a recoverable virtual memory system when a process

a) *Aborts* a transaction? (5 points)

b) *Commits* a transaction? (5 points)

This question refers to paper that is not in the current reading list.