Midterm Exam
CS 111, Principles of Operating Systems
Fall 2017

Name: Caleb Chau

Student ID Number: 204805602

This is a closed book, closed note test. Answer all questions.

Each question should be answered in 2-5 sentences. DO NOT simply write everything you remember about the topic of the question. Answer the question that was asked. Extraneous information not related to the answer to the question will not improve your grade and may make it difficult to determine if the pertinent part of your answer is correct. Confine your answers to the space directly below each question. Only text in this space will be graded. No question requires a longer answer than the space provided.

1. In a system using modern virtual memory techniques, what is the relationship between a page and a page frame?

A page frame is a fixed size of memory equal to the page size of the OS, and the page is the data that is associated with a particular page frame. The page frame is merely a container for a page.

2. Why are operating system ABIs of importance for convenient application software distribution?

OS ABIs define the interface for compiled programs to work with particular ISAs. It is far more convenient for users to just download the binary for a particular program instead of figure out how to compile it in order to run it, so it is thus important to distribute different binaries for your application to be compatible with the common ISAs that your users are running on. To do this, you must know the ABIs for the ISAs so that your distributed application can run on a variety of platforms without requiring your users to specifically compile your application for their platform.

3. Why is information hiding a good property in an operating system interface?

By hiding complexity with abstractions and having a standard interface for interacting with the OS, applications don't need to worry about the nasty details of the physical devices executing their instructions and can just focus on using the simpler OS services to implement their program's functionality. This eases the job of programmers by providing abstractions to work with physical devices and allowing programmers to build on top of these abstractions to create user-friendly applications.

4. When an operating system performs a context switch between processes, what information must the OS save?

The OS must save the registers, PC/PS of a process to its kernel stack when performing a context switch. That way, when it is time for the process to run again, the OS can restore its state through its registers and use the saved PC to resume execution where it left off.

5. What is the purpose of a trap table?

The trap table is used to determine what trap handler to run based on the number of the exception that occurred, which serves as an index into the trap table.

6. What is a race condition?

A race condition is when the order of execution of a particular set of instructions is extremely important, and any deviation from the order that it is run produces non-deterministic output. If a thread is interrupted mid way through executing an order-sensitive sequence of instructions and another thread begins executing the same sequence of instructions, then we have a race condition and have no way of predicting the output.

7. Why is blocking a problem for user-mode threads? Why isn't it a problem for kernel-mode threads?

If a user-mode thread blocks, the OS can only see that the process as a whole blocked, and thus will not schedule any of the other possibly runnable threads to run. This isn't a problem for kernel-mode threads because the kernel is aware that it is running multiple threads, and if it sees one of its threads block, it can just schedule another of its threads to run.

8. Why does Shortest Time-To-Completion First (STCF) scheduling require preemption?

In order to know whether an incoming process will take the shortest time to complete, the OS needs to be able to see into the future. Otherwise, it has no way of knowing how long a particular job will take and thus cannot feasibly schedule the shortest job first. Running the shortest ~~[illegible]~~ time to completion first requires knowing the times to completion ahead of time, which is not possible for the OS, and so it must use preemption to do its best to perform similarly.

9. When a Unix-system follows a fork with an exec, what resources of the forked process are replaced?

The code, data, registers, and stack of the forked process are replaced when a Unix-system follows a fork with an exec. It is almost equivalent to creating a process from scratch, except a child process can perform some setup for exec-ing the new process. Also, since the child's data is set to copy-on-write to avoid having to copy all the data from the parent to the child if possible, if the child doesn't write to its data, the exec just needs to replace the stack and registers of the forked process.

10. What form of fragmentation do we still suffer if we use a paging memory management system? For a segmented paging system, how much fragmentation per segment do we see?

We still suffer from internal fragmentation if we use a paging memory management system, although it is very slight and only occurs in the last part of the last page allocated for a particular set of contiguous pages. For a segmented paging system, we also suffer from internal fragmentation since we are allocating a fixed-size partition to a particular request for memory. However, the fragmentation would be worse since a process could ask for 1 KB of memory and receive 4 KB of memory, assuming the size of the fixed partition is 4 KB. This wastes 3 KB a memory, and if the process asks for another 1 KB of memory, the OS must allocate another 4 KB of memory to the process, thus wasting a total of 6 KB of memory.