Midterm Exam
CS 111, Principles of Operating Systems
Fall 2017

Name: Arpit Jasapara

Student ID Number: 504742401

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 is the virtual memory abstraction that stores data as memory. It can either be on memory or swapped out to the disk. If it is on memory, it will be stored in a page frame, which is an open spot on the memory that the page can fit in.

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

ABIs allow the program to be executed without needing to be compiled. If we only had APIs, the source code for each executable would need to be compiled before running, which is very inconvenient, especially for people bad with technology. ABIs allow software to be distributed to all kinds of users.

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

Information hiding makes it more convenient, efficient and safer to write/run programs. The convenience comes from the fact that programmers do not have to worry abo low-level operations, memory management, or hardware, since they are represented as abstractions. The efficiency comes from the fact that programs are not fighting over information and trying to outsmart each other since the OS will handl all information access/communication. The safety and privacy comes from the fact that processes cannot see each other's virtual address spaces & other information so they cannot inadvertently (or purposely) hurt each other. The process is simply made to believe that it is the only executing process, due to information hiding.

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

The OS must save the process' entire state and virtual address space. This includes things like registers, the stack, the heap, the resources it was using, its entire virtual memory, page tables, and everything else in its virtual space.

5. What is the purpose of a trap table?

A trap table allows the OS to handle traps and exceptions in the appropriate manner. During interrupts, the OS refers to the trap table based on the signal received to perform the appropriate operation, be that simple I/O or handling an illegal exception by terminating the process. The trap table lets the OS know where the appropriate exception handlers are so the OS uses the table entry to execute the correct handler and therefore action.

6. What is a race condition?

A race condition is when the outcome of an event can change drastically based on which process/thread reached it first. This is usually associated with shared data, especially the data that has no locking mechanisms. Due to context switches and multiple processors, different threads and processes can attempt to access & modify the same data at the same time, leading to unpredictable behavior. Since we have no control over the scheduler, these race conditions usually lead to different outcomes each time and are again highly unpredictable. The data can be modified in strange ways such as one process was adding 100 over a loop, but another process jumped in and subtracted 10, so neither process gets the data as expected with potentially disastrous consequences.

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

User-mode threads are not able to control scheduling and access of the processor, so when the user-mode threads are blocked by the OS, it cannot switch to another thread that is ready to run. On the other hand, kernel-mode threads are managed by the OS, so when one thread blocks, say due to I/O, the OS knows it can simply switch over to another thread that is ready to run. For user-mode threads, the OS simply does not know that they exist so cannot switch between threads when blocking occurs. The OS treats all the user-mode threads as one so treats blocking of one as blocking of them all.

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

Preemption is needed because while you are running one task, another task may arrive that has a shorter time-to-completion. Thus, you can preempt the current task, and switch over to the new task, in accordance with STCF scheduling. This is especially true when running long background processes and an important shorter TCF task arrives. The scheduler can then preempt the background process and run the new task.

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

The majority of the resources are replaced as the desired program executable is loaded and executed. This means the registers, stack, heap, virtual memory, and the rest of the virtual address space is replaced. Inter-process communication is usually not replaced, along with file descriptors.

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?

fixed-size segments

We suffer from internal fragmentation since the memory is allocated in certain-sized pages so the data may not fill up the entire page. For fixed-size segments, we usually see anywhere up to 50% fragmentation per segment.