Midterm Exam
CS 111, Principles of Operating Systems
Fall 2017

Name: William Chou

Student ID Number: 304764446

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 a fixed partition of a given processes' virtual address space. A page frame is a fixed partition of the physical address space, with the same size as pages. VM management systems map pages into page frames, and the MMU, upon receiving a memory access at a virtual address in one of the calling processes' pages, will translate the virtual address into a physical address by obtaining the virtual page number, translating it into a physical page number (page frame), and going to the same offset within the page frame. (in the page)

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

ABIs (Application binary interfaces) are already compiled versions of the source code. This is very important for convenient application software distribution because the users will not have to compile the source themselves (or know how to compile it). ABIs also provide some level of abstraction and interface for which user applications can run and interact safely and correctly with the operating system and given software and hardware specifications.

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

Information hiding is a good property in an operating system interface because it hides some lower-level implementation details of the OS. This offers better protection of the OS/kernel code, which is very important because the OS must trust that its code is safe and isolated in order to ensure proper OS functionality. It also allows applications to use OS functionalities without having to know or deal with implementation details, which allows the OS to carry out the desired function for the application in a safe and controlled manner.

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

The OS must save the process state of the process being switched out of. This includes the process' stack, data segments, registers, and program counter. This state must be saved into the process control block before a context switch into a different process (w/ different stacks, data, registers, PC/PS) so that the OS can resume execution of the process with correct stack, data, registers, and PC/PS when this process is next switched into.

5. What is the purpose of a trap table?

The trap table contains the memory locations of trap-handler code segments for the OS to run in events of traps. When the CPU traps into the OS, the OS can refer to the trap table, which is also indexed by numbers (e.g. syscall) that each handle a diffrent exception, to find the correct corresponding trap-handler code to run. The trap table is only known to the hardware to ensure that normal applications cannot access the traphandler code segments.

6. What is a race condition?

A race condition occurs when the correctness of a program depends on the order of execution of the program. It can be a problem when multiple threads of a program are all accessing shared resources, and are being interrupted during the execution of the critical section (where the thread accesses shared resources). This can cause the shared resource to not be updated as expected, which can lead to incorrect results by the program.

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

Blocking is a problem for user-mode threads because the kernel is unaware of user-mode threads. In the OS' perspective, there is one process running (OS does not know it has user-level threads). Thus, when blocking occurs in a process' user-mode thread, the OS blocks the whole process, as it is unaware that other user-mode threads exist and can be running while one of the threads is blocked. This is not a problem for kernel-mode thread, because the OS is aware of kernel-mode threads. Thus, if a kernel-mode thread is blocked the OS knows to let another thread run, or let another thread run on another core.

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

STCF requires preemption because it dictates that whenever a new job is introduced to the scheduler, the scheduler will choose the job that has the shortest-time-to-completion to run. It will do this by preempting (hand control back to OS) when a new job is introduced, evaluating the time to completion for each job, and choosing the one with shortest time to completion. For example, if a long-running job is in the middle of execution, and a very short job is introduced, the scheduler will preempt, find that the new job has shorter time-to-completion, and will switch to running the new job.

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

When a Unix-system forks a process, the contents of the new forked process are a copy of the original calling process. Thus, when a fork is followed with an exec, the memory needed for the executable called by exec needs to be loaded into the new process. This means that the stack, registers, data, and program counter need to be replaced with data required for the executable to run.

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

A paging memory management system will suffer from internal fragmentation.. This is because pages are fixed-size partitions of memory, so when the paging memory management system allocates memory in a way such that one of the allocated pages is not fully occupied, there is internal fragmentation, where some memory in the fixed-size pages is not used/wasted. However, the fragmentation is not as bad in paging systems. This is because page sizes are usually chosen as small partitions, so a typical allocation will occupy many full pages, and only the last page may have internal fragmentation less than or equal to the page size.