CS 111 Midterm Exam

Alden Jung Yeon Kim Perrine

TOTAL POINTS

90 / 100

QUESTION 1

1 Pages and page frames 10 / 10

✓ - 0 pts **Correct**

- 10 pts No answer.
- 2 pts Did not explain that pages get placed into page frame.
- 1 pts Said page is mapped to page frame
- 1 pts No mention of location of page & page frame in system
- 4 pts Incorrect Reasoning
- 3 pts Not accurate enough
- 1 pts No mention of virtual address
- 5 pts Page frame contains exactly one page.
- 2 pts Did not mention that page and page frame are the same size.

QUESTION 2

2 ABIs 6 / 10

- 0 pts Correct
- 10 pts No answer
- 1 pts Did not mention operating system

✓ - 4 pts **Said applications don't need to worry about hardware differences**

- 2 pts Off-topic
- 3 pts Incorrect reasoning
- 2 pts Did not mention software distribution
- 3 pts Did not mention hardware and OS
- 2 pts Left out hardware

QUESTION 3

3 Information hiding 4 / 10

- 0 pts Correct
- 10 pts No answer.

✓ - 6 pts **Primary benefit is to avoid bugs arising from improper dependencies among modules.**

- 3 pts Major element of benefit is that it allows changes in module implementation.
- 7 pts Really about hiding details of OS modules' implementation, not about concealing processes' address spaces from each other.
- 2 pts Nothing to do with open vs. closed source.
- 9 pts Primarily an issue involving abstraction.

QUESTION 4

4 Context switches 10 / 10

✓ - 0 pts **Correct**

- 2 pts Not mentioning general registers
- 2 pts Not mentioning PC
- 2 pts Not mentioning Stack ptr
- 1 pts Not mentioning PSW
- 2 pts No discussion of memory mapping data
- 2 pts No need to explicitly save data, since it's already sitting in memory.
- 3 pts Generally nothing goes to disk on a context switch.
- 1 pts What about memory needs to be saved?
- 2 pts Much of this stuff need not be saved, since it's already in memory. OS just needs to be sure it can be found again when process is switched back in.
- 2 pts The PCB is an OS data structure that exists as long as the process is around, so it need not be saved on a context switch.
- 1 pts File size has nothing to do with a context switch.
- 1 pts I have no idea what the flag you're talking about is.
- 1 pts "state of the process" is vague.
- 2 pts Caches aren't saved.
- 2 pts No need to update a file descriptor during a context switch.

QUESTION 5

5 Trap tables 10 / 10

✓ - 0 pts **Correct**

- 10 pts No answer
- 3 pts Answer incomplete, should mention trap table is used to specify what code to run when trap occurs.
- 8 pts Wrong answer.
- 3 pts Answer incomplete.
- 2 pts User process has no thing to do with trap?

QUESTION 6

6 Race conditions 10 / 10

✓ - 0 pts **Correct**

- 10 pts No answer
- 5 pts Answer incomplete.
- 8 pts Answer incorrect.
- 2 pts Missing some details.

QUESTION 7

7 Blocking and threads 10 / 10

✓ - 0 pts **Correct**

- 10 pts No answer or Wrong anwser
- 5 pts Missing: User-mode threads block other threads of the same process.
- 5 pts Missing: Kernel-mode threads do not block other threads of the same process, as other threads can be scheduled to run on the same or another core.

QUESTION 8

8 STCF 10 / 10

✓ - 0 pts **Correct**

- 10 pts No answer or Wrong answer
- 5 pts Missing: interrupt the running one OR switch to the newly-added shorter ones.
- 8 pts Missing mention of new processes that might have shorter time to completion.

QUESTION 9

9 Fork and exec 10 / 10

✓ - 0 pts **Correct**

- 10 pts No answer
- 4 pts Not mentioning code replacement.
- 3 pts Not mentioning stack replacement.
- 3 pts Not mentioning heap replacement.
- 9 pts The question was about what happens after the exec, not the fork.
- 8 pts What resources are replaced by the exec?
- 7 pts Stack and code are changed by exec.
- 5 pts Fork/exec work with processes, not threads.
- 2 pts Even any data written after fork gets replaced by exec.
- 2 pts The old stack is totally overwritten.
- 10 pts Totally wrong. Nothing to do with multithreading and multicore.
- 6 pts So, what resources are replaced?

QUESTION 10

10 Fragmentation for memory management schemes 10 / 10

✓ - 0 pts **Correct**

- 10 pts No answer
- 5 pts Not identifying internal fragmentation for pages.
- 5 pts Paged segments suffer 1/2 page fragmentation.
- 5 pts Fixed segments suffer 1/2 internal segment fragmentation.
- 3 pts On average 50%
- 2 pts The 1.5% was a particular example. It will be 1/2 page, on average.
- 2 pts Internal fragmentation has little to do with how long the system runs, unlike external.
- 2 pts Paging and fixed size partitions never experience external fragmentation.
- 2 pts The paging form of fragmentation you describe is internal fragmentation.
- 2 pts Paging doesn't use binary buddy. It allocates in fixed size pages.
- 4 pts Fixed segments are likely to waste more memory on internal fragmentation than paging, not less.
- 3 pts No external fragmentation with paging.

- **5 pts** No answer on segmented system.
- **2 pts** Calling internal fragmentation "external".
- **1 pts** This form of fragmentation is called "internal."
- **4 pts** Paged segment fragmentation only occurs in the last page.

Midterm Exam
CS 111, Principles of Operating Systems
Fall 2017

Name: Alden Perrine

Student ID Number: 204788061

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 of virtual memory is a section of memory of a specific size which can either be allocated in memory, swapped to disk, or invalid. A page for a specific process has a page table entry which defines its current state of existence, and is virtual, not real. A page frame is a section of physical memory which is the same size as a page which can hold any given page for any process. Pages are mapped to page frames to be used by running processes, and the page within the page frame can change as a result of page faults or context switches.

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

Operating system ABIs allow applications to run on client's systems without the client having to build the program with the correct API. ABIs allow application writers to write general application without worrying about what hardware is being used underneath. With the use of ABIs, applications can be compiled and distributed, and are able to run without additional set up as the proper ABIs will be used by the OS

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

Operating systems must hide information from processes to prevent any rogue process from interfering with other processes or even the operating system itself. If the operating system did not hide data related to the context of processes or their memory, another process might be able to over write the data and crash the program. Even worse, if structures within the operating system itself were not hidden, such as the trap table, an application could use any code arbitrarily and run in kernel mode to do any malicious action on the computer.

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

The operating system must save any information necessary to restore the process back to the same state after the context switch. This includes CPU registers, program status word, program counter, stack pointer, and page table for the process. Once the process is swapped back to, it can continue to run without worrying that its registers have been changed or that memory references will be invalid.

5. What is the purpose of a trap table?

The purpose of the trap table is to provide pointers to the trap handlers that the operating system will run when a specific trap occurs. The trap table is loaded at boot time with the trap numbers being indexes in the table, and the corresponding handler with each index. Once a trap occurs, the hardware uses the trap table to determine what the OS should start running to handle it.

6. What is a race condition?

A race condition is when a specific section of code must execute without a context switch occurring in the middle of it, and there are no synchronization protocols surrounding it. Race conditions typically occur in programs where a variable or object is modified by multiple threads or processes at once and multiple hardware instructions are used for the update. If a context switch occurs in the middle of the update, the state of the object may have changed upon switching back. Forcing the section of code to execute atomically is a solution to prevent race conditions.

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

User-mode threads are written at the user level, and are unable to use system resources themselves, when a user-level thread makes a system call, the entire process is blocked waiting for the call to return as the kernel acts for the process since the thread scheduler is part of the process, not the kernel. Kernel mode threads do not suffer from this problem because the kernel is in charge of scheduling, not the user. The kernel can detect when a thread is blocked and perform a context switch, so another thread can run while the first is blocked.

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

STCF requires preemption because new processes can enter the system at any moment, possibly with a shorter time to completion than the currently running process. If a long running process is currently running on the cpu and a guide process with a short runtime enters the system, the OS must be able to preempt the current process, check the runtime of the new process, and possibly schedule the new process.

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

In the forked process, when exec is called to launch a different program, the code section of memory is replaced with the code section of the new program, the heap and stack are set to their original state, and the program counter is set to the beginning of the new program. However OS resources associated with the process, such as open files in the file descriptor table, are not replaced, as the same process is being run, just with a different program.

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 partitions

Paging continues to suffer from internal fragmentation, on average half of the page size as the OS cannot allocated specifically sized chunks to fit the requirement of the process. Similarly for fixed size partitions, on average half of the partition size will be wasted as a process could use almost none of the partition, or almost all of it.