CS 111 Midterm Exam

[redacted]

TOTAL POINTS

80 / 100

QUESTION 1

1 Pages and page frames 10 / 10

✓ **- 0 Correct**

- 10 No answer.

- 2 Did not explain that pages get placed into page frame.

- 1 Said page is mapped to page frame

- 1 No mention of location of page & page frame in system

- 4 Incorrect Reasoning

- 3 Not accurate enough

- 1 No mention of virtual address

- 5 Page frame contains exactly one page.

- 2 Did not mention that page and page frame are the same size.

QUESTION 2

2 ABIs 3 / 10

- 0 Correct

- 10 No answer

- 1 Did not mention operating system

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

✓ **- 2 Off-topic**

✓ **- 3 Incorrect reasoning**

- 2 Did not mention software distribution

- 3 Did not mention hardware and OS

✓ **- 2 Left out hardware**

QUESTION 3

3 Information hiding 7 / 10

- 0 Correct

- 10 No answer.

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

✓ **- 3 Major element of benefit is that it allows changes in module implementation.**

- 7 Really about hiding details of OS modules' implementation, not about concealing processes' address spaces from each other.

- 2 Nothing to do with open vs. closed source.

- 9 Primarily an issue involving abstraction.

QUESTION 4

4 Context switches 8 / 10

- 0 Correct

✓ **- 2 Not mentioning general registers**

- 2 Not mentioning PC

- 2 Not mentioning Stack ptr

- 1 Not mentioning PSW

- 2 No discussion of memory mapping data

- 2 No need to explicitly save data, since it's already sitting in memory.

- 3 Generally nothing goes to disk on a context switch.

- 1 What about memory needs to be saved?

- 2 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 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 File size has nothing to do with a context switch.

- 1 I have no idea what the flag you're talking about is.

- 1 "state of the process" is vague.

- 2 Caches aren't saved.

- 2 No need to update a file descriptor during a context switch.

QUESTION 5

5 Trap tables 10 / 10

✓ - 0 Correct

- 10 No answer

- 3 Answer incomplete, should mention trap table is used to specify what code to run when trap occurs.

- 8 Wrong answer.

- 3 Answer incomplete.

- 2 User process has no thing to do with trap?

QUESTION 6

6 Race conditions 10 / 10

✓ - 0 Correct

- 10 No answer

- 5 Answer incomplete.

- 8 Answer incorrect.

- 2 Missing some details.

QUESTION 7

7 Blocking and threads 5 / 10

- 0 Correct

- 10 No answer or Wrong anwser

- 5 Missing: User-mode threads block other threads of the same process.

✓ **- 5 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 Correct

- 10 No answer or Wrong answer

- 5 Missing: interrupt the running one OR switch to the newly-added shorter ones.

QUESTION 9

9 Fork and exec 10 / 10

✓ - 0 Correct

- 10 No answer

- 4 Not mentioning code replacement.

- 3 Not mentioning stack replacement.

- 3 Not mentioning heap replacement.

- 9 The question was about what happens after the exec, not the fork.

- 8 What resources are replaced by the exec?

- 7 Stack and code are changed by exec.

- 5 Fork/exec work with processes, not threads.

- 2 Even any data written after fork gets replaced by exec.

- 2 The old stack is totally overwritten.

- 10 Totally wrong. Nothing to do with multithreading and multicore.

- 6 So, what resources are replaced?

QUESTION 10

10 Fragmentation for memory management schemes 7 / 10

- 0 Correct

- 10 No answer

- 5 Not identifying internal fragmentation for pages.

- 5 Paged segments suffer 1/2 page fragmentation.

- 5 Fixed segments suffer 1/2 internal segment fragmentation.

- 3 On average 50%

- 2 The 1.5% was a particular example. It will be 1/2 page, on average.

- 2 Internal fragmentation has little to do with how long the system runs, unlike external.

- 2 Paging and fixed size partitions never experience external fragmentation.

- 2 The paging form of fragmentation you describe is internal fragmentation.

- 2 Paging doesn't use binary buddy. It allocates in fixed size pages.

- 4 Fixed segments are likely to waste more memory on internal fragmentation than paging, not less.

✓ **- 3 No external fragmentation with paging.**

- 5 No answer on segmented system.

- 2 Calling internal fragmentation "external".

- 1 This form of fragmentation is called "internal."

- 4 Paged segment fragmentation only occurs in the last page.

Midterm Exam
CS 111, Principles of Operating Systems
Fall 2017

Name: [redacted]

Student ID Number: [redacted]

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 some virtual memory address, while a page frame is some physical memory. A page is stored in a page frame.

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

OS ABIs are of importance for convenient application software distribution because:

- They don't need to be compiled by user, which is complicated to do
- ~~Proc~~ Provide a working environment without the need to know all the ~~details~~ implementation details of OS.

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 protects OS information from users

- it provides simplicity and convenience for users i.e. users can run programs/softwares on the OS, and don't have to worry about ~~what~~ how the OS handles them, and won't be confused ~~to~~ with ~~too~~ much information, ~~or~~ that ~~the~~ users don't need to know, in the interface

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

When an operating system performs a context switch between processes, the OS must save:

- PC/PS of the ~~running~~ ~~old~~ current program that's going to be switched out.

- Pages, not necessarily all of them, of the ~~running~~ current program, and brings in pages of the new process.

5. What is the purpose of a trap table?

When there are exceptions, interrupts, or system calls, the OS is trapped.

Each of those events has a specific identification number. The OS uses the ~~trap~~ table to find a ~~matching~~ handler for ~~the~~ a given even ~~to~~ by looking up the identification number in the trap table.

6. What is a race condition?

A race condition happens when there are threads or ~~proce~~ ~~proceesses~~ processes running in parallel and sharing memory. When 2 or more threads or processes all want to run code to on the same memory, which is critical section, and the order of execution affects the result, then there is a race between the threads/processes.

For example: counter = 0

```
Thread 1                          Thread 2
counter = counter +1;             counter = counter + 1;
```

→ Race condition

The result of counter depends on the order of execution of thread 1 and thread 2.

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

Blocking is not a problem for kernel-mode threads because kernel has the priviledge to choose which thread to block.

User-mode don't have that priviledge and all threads will be blocked, which will hurt the performance. It will then require a waitlist

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

Supposed there is a new process in the queue and it requires least time to run compared to all other processes in the queue. If the running process takes more time to finish than this new process, the scheduler will preempt to run the new process, because the scheduler always chooses the process with shortest time-to-completion.

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

If a fork calls exec, it basically runs a new program. Thus, all of the forked process' resources are replaced including virtual memory address space i.e. data segment, code segment, stack, and pages.

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 sized partition.

If we use fixed size partition system, we will suffer from both internal and external fragmentation. Not all processes use the same amount of memory, so there will be processes that don't use all the memory allocated for them, which causes internal fragmentation.

When a process finishes and releases the memory, this free space could be in between used spaces and too small for the next process to use, then the next process has to use free space elsewhere, which causes external fragmentation.

For a fixed size partition, we can see 50% fragmentation per segment on average.