CSci 402 - Operating Systems
Midterm Exam (DEN Section)
Spring 2025

(10:00:00am - 10:40:00am, Wednesday, Mar 26)

Instructor: Bill Cheng

Teaching Assistant: Zhuojin Li

(This exam is open book and open notes.
Remember what you have promised when you signed your
Academic Integrity Honor Code Pledge.)

Time: 40 minutes

_____-
Name (please print)

Total: 36 points

_____-
Signature

Instructions

1. This is the first page of your exam. The previous page is a title page and does not have a page number. Since this is a take-home exam, no need to sign above since you won't submit this file.
2. Read problem descriptions carefully. You may not receive any credit if you answer the wrong question. Furthermore, if a problem says *"in N words or less"*, use that as a hint that N words or less are expected in the answer (your answer can be longer if you want). Please note that points may get *deducted* if you put in wrong stuff in your answer.
3. If a question doesn't say `weenix`, please do not give `weenix`-specific answers.
4. Write answers to all problems in the **answers text file**.
5. For non-multiple-choice and non-fill-in-the blank questions, please show all work (if applicable and appropriate). If you cannot finish a problem, your written work may help us to give you partial credit. We may not give full credit for answers only (i.e., for answers that do not show any work). Grading can only be based on what you wrote and cannot be based on what's on your mind when you wrote your answers.
6. Please do *not* just draw pictures to answer questions (unless you are specifically asked to draw pictures). Pictures will not be considered for grading unless they are clearly explained with words, equations, and/or formulas. It's very difficult to draw pictures in a text file and you are not permitted to submit additional files other than the answers text file.
7. For problems that have multiple parts, please clearly *label* which part you are providing answers for.
8. Please ignore minor spelling and grammatical errors. They do not make an answer invalid or incorrect.
9. During the exam, please only ask questions to *clarify* problems. Questions such as "would it be okay if I answer it this way" will not be answered (unless it can be answered to the whole class). Also, you are suppose to know the definitions and abbreviations/acronyms of *all technical terms*. We cannot "clarify" them for you. We also will **not** answer any clarification-type question for multiple choice problems since that would often give answers away.
10. Unless otherwise specified and stated explicitly, multiple choice questions have one or more correct answers. You will get points for selecting correct ones and you will lose points for selecting wrong ones.
11. When we grade your exam, we must assume that you wrote what you meant and you meant what you wrote. So, please write your answers accordingly.

(Q1) (2 points) Which of the following statements are correct about **kernel mutexes** in **weenix**?

(1) **weenix** uses mutexes to ensure that only one kernel thread can be accessing a particular I/O device at a time
(2) since the **weenix** kernel is non-preemptive, kernel mutexes must be implemented to synchronize accesses to shared data structures such as the run queue
(3) since there is only one CPU in **weenix**, there is no reason for having kernel mutexes
(4) **weenix** uses mutexes for mutual exclusion whenever a kernel thread needs to update any kernel linked list that another kernel thread may be using
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q2) (2 points) Why is it that when a user thread performs a system call, the corresponding kernel thread cannot use the user-space stack of the user thread and the user thread's stack pointer?

(1) the user thread stack may be full
(2) the user thread code may not be using a compatible stack register
(3) the user thread may be dead already
(4) the user thread's stack pointer may be pointing to a bad memory location
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q3) (2 points) What are the reasons why a thread (with its code written in C) cannot **completely delete itself**?

(1) a thread cannot switch to itself
(2) a thread cannot be in user space and in kernel space simultaneously
(3) a thread cannot be running inside two functions simultaneously
(4) a thread cannot have two kernel stacks
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q4) (2 points) Which of the following statements are correct about a newly created thread?

(1) it will be in the "runnable" state
(2) if you never put the newly created thread into the run queue, it will never run
(3) it will copy the state from its parent thread
(4) its state will be uninitialized (i.e., in a random state)
(5) it will be in the same state as the process it's in

Answer (just give numbers): _____

(Q5) (2 points) In programing with POSIX threads, if your main thread wants to **wait** for all the other threads in the process to die before the main thread itself dies, what must the main thread do to accomplish this objective **properly**?

(1) call **pthread_wait()** to wait for each of these other threads to die
(2) call **pthread_cond_wait()** to wait for all these threads to die
(3) call **pthread_join()** to join with each of these other threads
(4) call **pthread_cancel()** on each of these other threads
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q6) (2 points) On Unix/Linux systems, **sequential I/O** on regular files is done by calling **read/write()** system calls. Which of the following statements are true about Unix/Linux **block I/O** on regular files?

(1) to perform block I/O, you need to first open a file in the block I/O mode, then use read() and write() system calls to perform block I/O
(2) using the write() system call can also be considered as doing block I/O if the last argument of write() is the size of a disk block
(3) to perform block I/O, you need to first map a file (or part of a file) into your address space
(4) block I/O is available to both kernel and user processes
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q7) (2 points) Which of the following statements are correct about **signals** and **interrupts**?

(1) all signals are supposed to be delivered to user processes
(2) all signals are generated by hardware devices and are supposed to be delivered to the kernel
(3) all interrupts are supposed to be delivered to the kernel
(4) all signals are generated by user processes and can be delivered to another user process without using a system call
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q8) (2 points) Assuming that you only have one processor, which of the following statements are correct about **interrupts**, **traps**, and **Unix signals**?

(1) a signal can be blocked/masked while an interrupt cannot be blocked/masked
(2) an interrupt is most likely caused by the currently running thread
(3) even when a process is in the zombie state, its thread can still cause a trap
(4) a trap can only be caused by the currently running thread
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q9) (2 points) Which statements are correct about **copy-on-write**?

(1) never writes to a shared and writable memory page, only writes to a copy of such a page
(2) a private page is read-only until its written into for the first time, then it becomes a writable page
(3) a shared page can never be written into more than once
(4) every time a thread writes to a private page, the page gets copied and the write goes into the new copy of the page
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q10) (2 points) Let's say that your **umask** is set to **0471**. (a) What file permissions will you get (in octal) if you use an editor to create the **warmup1.c** file? (b) What file permissions will you get (in octal) if you use a compiler to create the **warmup1** executable? Please note that if your answer is not in octal, you will not get any credit.

(Q11) (2 points) Let's say that you have an infinitely fast and accurate computer and you run your warmup2 on it with the following commandline: "`./warmup2 -r 1000 -t g0.txt`" and the content of `g0.txt` is as follows:

```
4
3   3   5
2   2   9
3   4   2
3   1   2
```

How many **milliseconds** into the simulation will packet `p4` (i.e., the 4th packet) leaves the system? Please just give an integer value answer (no partial credit for this problem).

(Q12) (2 points) Which of the following is part of the action that must be taken during a successful Unix **upcall** to invoke a signal handler?

(1) signal handler makes a system call to figure out the reason for the upcall
(2) an upcall must be initiated from code inside a hardware interrupt service routine that's servicing a hardware device interrupt
(3) an interrupt service routine makes a system call
(4) trapping into the kernel to terminate a user process
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q13) (2 points) Which of the following statements are true about **Unix devices**?

(1) it's possible to have a device that can be accesses using only a minor device number and it has no associated major device number
(2) a device's major device number identifies a device driver
(3) minor device number is rarely used in the OS
(4) a device's major device number indicates which process is currently using the device
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q14) (2 points) Let kernel process C be the only child process of a regular (i.e., PID > 2) kernel process P in **weenix**. Which of the following statements are correct about **p_wait** (whose type is **ktqueue_t**) in the process control block in **weenix**? (Please note that since we are doing one thread per process in **weenix**, the words "process" and "thread" are considered interchangeable for this problem.)

(1) if process C is sleeping in a queue and process P calls **do_waitpid()**, process C will be moved into process P's **p_wait** queue
(2) if process C is sleeping in a queue and process P calls **do_waitpid()**, process P will sleep on its own **p_wait** queue
(3) when process C dies, it will add itself to the **p_wait** queue of the INIT process
(4) if process C is sleeping in a queue and process P calls **do_waitpid()**, process P will sleep on process C's **p_wait** queue
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q15) (2 points) Let's say that **foobar** is a valid program name for a very simple program. You type "`foobar &`" in a Unix command shell to run **foobar** in the **background**. What **system calls** must the **command shell** make before **foobar** finished running?

(1) wait()
(2) one of the "exec" system calls
(3) exit()
(4) close()
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q16) (2 points) Which of the following statements are correct about a **Unix file system**?

(1) in the Unix file system tree hierarchy, it's possible that a node in the tree has more than one "parent" node
(2) symbolic links to the same file cannot be contained in multiple directories
(3) a directory can have more than one "parent" directory
(4) hard links to the same file can be contained in multiple directories
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q17) (2 points) Let's say that a thread (thread X) is executing in an x86 CPU when a hardware interrupt (unrelated to thread X) occurs. Let's further assume that the interrupt handler is implemented using the most common implementation (especially regarding the way kernel stack is chosen to be used by the interrupt handler). The interrupt handler must execute till completion before thread X is resumed even though the interrupt has nothing to do with thread X. What bad thing can happen if we resume thread X before the interrupt handler finishes?

(1) if thread X is a user thread, there is nothing thread X can to to mess up the interrupt handler
(2) if thread X is a user thread, it can make a system call and corrupt the stack the interrupt handler is using
(3) if thread X is a kernel thread, it can make a function call and corrupt the stack the interrupt handler is using
(4) if thread X is a user thread, it can cause a deadlock in user space and never return the CPU to the interrupt handler
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q18) (2 points) For the x86 processor, the **switch()** function is depicted below:

```
void switch(thread_t *next_thread) {
  CurrentThread->SP = SP;
  CurrentThread = next_thread;
  SP = CurrentThread->SP;
}
```

In what way is this **switch()** function different from a regular C function (please note that a system call is a regular C function)?

(1) unlike most other functions, the thread that calls this function can fall asleep inside this function and wake up at a later time
(2) this function does not modify the content of the ESP register during its execution
(3) this function does not modify the content of the EBP register during its execution
(4) it would appear to an observer of the system that the thread that calls this function is not the thread that returns from this function
(5) none of the above is a correct answer

Answer (just give numbers): _____