# CSci 402 - Operating Systems

# Midterm Exam (DEN Section)

# Fall 2021

*(10:00:00am - 10:40:00am, Wednesday, Oct 27)*

Instructor: Bill Cheng

Teaching Assistant: Ben Yan

*( 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) The code below is a solution to the **readers-writers problem**.

```
reader( ) {
  when (writers == 0) [
    readers++;
  ]
  /* read */
  [readers--;]
}
```

```
writer( ) {
  when ((writers == 0) &&
      (readers == 0)) [
    writers++;
  ]
  /* write */
  [writers--;]
}
```

It has a major problem. What can be said about this major problem?

(1) two reader threads may block each other out for a long time
(2) a writer thread may block forever if reader threads keep arriving and there are always reader threads present
(3) the code has a serious memory corruption bug
(4) some threads may **starve**, i.e., never get to run even when there are not deadlocks
(5) a writer thread may never get to write even if there are no reader thread present

Answer (just give numbers): _____

(Q2) (2 points) What are the consequences and implications of the phrase "the kernel is very very powerful and can do anything it wants" as far as our entire Kernel Assignment 1 is concerned?

(1) if you get a kernel panic, it's your bug
(2) if the buddy system runs out of memory, it's your bug
(3) if the run queue is empty, it's your bug
(4) without QEMU, you can still run weenix
(5) if the kernel deadlocks, it's your bug

Answer (just give numbers): _____

(Q3) (2 points) Let's say that you have a uniprocessor and the kernel is currently servicing an interrupt when a **higher priority interrupt** occurs. What would typically happen?

(1) to execute its interrupt service routine, the higher priority interrupt must not use the kernel stack of kernel thread that was interrupted by a lower priority interrupt
(2) nothing, because when an interrupt handler is running, all other interrupts are blocked/disabled
(3) the higher priority interrupt will execute with a newly allocated kernel stack
(4) higher priority interrupt will be blocked until the current interrupt handler yields the processor voluntarily
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q4) (2 points) Which statements are correct about what **device independence** may mean when you are **designing or implementing** an OS?

(1) no need to redesign the kernel in order to support a new device made by a new device manufacturer
(2) you don't have to recompile the kernel in order to support some new devices
(3) a device must be able to run in any OS
(4) two devices must not share the same code in the kernel
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q5) (2 points) Under what **general condition** would using only mutex to access shared data by concurrent threads an overkill because it's too restrictive and inefficient?

(1) when different threads would access different parts of shared data most of the time and only access the same parts of shared data occasionally
(2) when the execution order of threads is mostly predictable
(3) when some threads only want to write to the shared data and promise not to read the shared data
(4) when one thread is the parent thread of another thread
(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) using read/write() system calls can also be considered as doing block I/O
(2) to perform block I/O, you must map a file into your address space
(3) block I/O is performed by first calling lseek() and then read/write()
(4) block I/O is only available to kernel processes and not available to user processes
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q7) (2 points) Let's say that your thread is executing C code somewhere in the middle of a C function named **foobar()** on an x86 processor. If you know that the base/frame pointer (i.e., ebp) of the x86 processor is **not** pointing anywhere within **foobar**'s stack frame, what conclusion can you make?

(1) the function return type of **foobar()** is **void**
(2) **foobar()** is a recursive function
(3) **foobar()** does not have local variables
(4) **foobar()** does not make function calls
(5) **foobar()** does not have function arguments

Answer (just give numbers): _____

(Q8) (2 points) Let kernel process C be the child process of 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 of P in **weenix**? (Please note that since we are doing one thread per process in **weenix**, the words "process" and "thread" are considered interchangeable here.)

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

Answer (just give numbers): _____

(Q9) (2 points) What is true about **interrupts** and **Unix signals**?

(1) conceptually, an interrupt is delivered to a user program while a signal is delivered to the kernel
(2) conceptually, an interrupt is delivered to the kernel while a signal is delivered to a user program
(3) a signal can be blocked/disabled while an interrupt cannot be blocked/disabled
(4) a user program can specify what interrupt service routine to use to handle a given interrupt
(5) for all signal types, a user program can catch that signal by specifying a user space signal handler

Answer (just give numbers): _____

(Q10) (2 points) Assuming that thread X calls **pthread_cond_wait()** to wait for a specified condition, what bad thing can happen if **pthread_cond_wait(cv,m)** is **not atomic** (i.e., pthread library did not implemented it correctly with respect to atomicity)? Please assume that everything else is done perfectly.

(1) thread X may execute critical section code when it does not have the mutex locked
(2) thread X may become uncancellable
(3) a SIGWAIT signal can get generated when another thread signals or broadcasts the condition
(4) thread X may miss a "wake up call" when another thread signals or broadcasts the condition)
(5) thread X may cause a kernel panic when it calls pthread_cond_wait()

Answer (just give numbers): _____

(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 1 -t g0.txt`" and the content of `g0.txt` is as follows:

```
4
2000    2    8000
4000    1    4000
1000    3    2000
1000    2    4000
```

How many seconds into the simulation will packet `p3` (i.e., the 3rd packet) leaves the system? Please just give an integer value answer (no partial credit for this problem).

(Q12) (2 points) Which of the following operations are **locked** together in **one atomic operation** by **sigsuspend(set)** (where **set** specifies a set of signals)?

(1) it unblocks the signals specified in **set**
(2) it waits for the signals specified in **set**
(3) it suspends all other threads in the same process
(4) it blocks all signals
(5) it calls all signal handlers specified in **set**

Answer (just give numbers): _____

(Q13) (2 points) Which of the following statements are correct about a Unix **pipe**?

(1) a pipe is implemented as a file object inside the kernel
(2) a pipe has two ends, a user program decides which end is for reading and which end is for writing
(3) a pipe can be used by one user thread to send characters to another user thread in the same process
(4) a pipe object lives in the user space
(5) a pipe can be used by one user thread to send characters to the thread itself

Answer (just give numbers): _____

(Q14) (2 points) Let's say that your **umask** is set to **0340**. (1) What file permissions will you get (in octal) if use a compiler to create the **warmup1** executable? (2) What file permissions will you get (in octal) if use an editor to create the **warmup1.c** file?

(Q15) (2 points) Which of the following is part of the action that must be taken during a Unix **upcall**?

(1) keyboard interrupt to terminate a user process
(2) user process makes a system call to figure out the reason for the upcall
(3) context switching from privileged mode to user mode
(4) the kernel invokes user space code
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q16) (2 points) If your CPU is currently executing code in user mode, which of the following will **not** cause the CPU to go into the privileged mode immediately?

(1) user program executes an instruction to divide an integer by zero
(2) user program calls **free()**
(3) user program executes a software interrupt machine instruction
(4) hardware interrupt
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q17) (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;
}
```

If thread A calls **switch()** to switch to thread B, which of the following statements are correct?

(1) thread B restores its frame pointer from thread B's thread control block
(2) thread A saves its stack pointer in thread A's stack
(3) thread A saves its frame pointer in thread A's stack
(4) thread A saves its stack pointer in thread B's thread control block
(5) thread A saves its stack pointer in thread A's thread control block

Answer (just give numbers): _____

(Q18) (2 points) Every device in Unix is identified by a major device number and a minor device number. Which of the following statements are true about these device numbers?

(1) major device number indicated which process is currently using the device
(2) minor device number is rarely used in the OS
(3) major device number must be smaller than the minor device number
(4) minor device number identifies a device driver
(5) none of the above is a correct answer

Answer (just give numbers): _____