CSci 402 - Operating Systems
Midterm Exam (AM Section)
Spring 2021

(10:00:00am - 10:40:00am, Thursday, March 25)

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) Let's say that your **umask** is set to **056**. (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?

(Q2) (2 points) Assuming that you only have one processor, which of the following statements are correct about **interrupts**, **traps**, and **Unix signals**?

(1) a trap is caused by the currently running program
(2) if a program is not currently running, it is not possible for it to cause a trap
(3) a user program can specify what function to call when a signal is delivered for most signal types
(4) an interrupt is most likely caused by the currently running program
(5) a signal can be blocked/disabled while an interrupt cannot be blocked/disabled

Answer (just give numbers): _____

(Q3) (2 points) Let's say that you have a uniprocessor and a user thread (thread X) is executing when a hardware interrupt occurs. Which of the following statements are true?

(1) in some OS, the kernel would allocate a new kernel stack to be used by the interrupt handler
(2) in some OS, the interrupt handler will use the kernel stack of thread X
(3) in some OS, the interrupt becomes pending until thread X traps into the kernel, then the interrupt handler will be invoked using the kernel stack for thread X
(4) in some OS, a special kernel stack is shared by all interrupt handlers
(5) in some OS, the interrupt handler will run in user mode and will use the user-space stack of thread X

Answer (just give numbers): _____

(Q4) (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) a writer thread may never get to write even if there are no reader thread present
(2) two reader threads may block each other out for a long time
(3) the code can deadlock
(4) a writer thread may block forever if there are always reader threads present
(5) the code has a serious memory corruption bug

Answer (just give numbers): _____

(Q5) (2 points) Which of the following statements are correct about a newly created thread?

(1) it will copy the state from its parent thread
(2) it will be in the same state as the process it's in
(3) it will be in a sleeping state
(4) its state will be uninitialized (i.e., in a random state)
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q6) (2 points) What are the operations that are **locked** together in an **atomic operation** by **pthread_cond_wait(cv,m)** (where **cv** is a POSIX condition variable and **m** is a POSIX mutex)?

(1) move thread to the cv queue
(2) unlock the mutex
(3) lock the mutex
(4) broadcast the condition
(5) signal the condition

Answer (just give numbers): _____

(Q7) (2 points) What are the ways a thread can **self terminate**?

(1) calls **pthread_exit()**
(2) return from its first procedure
(3) send itself a SIGTERM signal
(4) join with itself
(5) calls **pthread_test_cancel()**

Answer (just give numbers): _____

(Q8) (2 points) What are the main tasks of a **linker**?

(1) maintain reference count for file system hard links
(2) perform relocation
(3) compile source code into relocatable object files
(4) allocate and manage the heap space
(5) resolve addresses for symbols

Answer (just give numbers): _____

(Q9) (2 points) The code below is a solution to the **barrier problem** (to synchronize n threads) implemented using a POSIX mutex **m** and a POSIX condition variable (**BarrierQueue**).

```
int count = 0;
void barrier() {
  pthread_mutex_lock(&m);
  if (++count < n) {
    while (count < n)
      pthread_cond_wait(&BarrierQueue, &m);
  } else {
    /* release all n-1 blocked threads */
    pthread_cond_broadcast(&BarrierQueue);
    count = 0;
  }
  pthread_mutex_unlock(&m);
}
```

Assuming that all the variables have been properly initialized, which statements below are correct about the above code?

(1) the n^{th} thread causes other threads to leave the barrier but it can get stuck at the barrier
(2) when the n^{th} thread wakes up all the other threads, not all of them may exit the barrier
(3) the barrier may collapse unexpectedly
(4) some thread may leave the barrier before the n^{th} thread arrives
(5) spontaneous return of **pthread_cond_wait()** can break the code

Answer (just give numbers): _____

(Q10) (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 may not have a corresponding stack register
(2) the user thread stack frames may be upside-down
(3) the user thread may be dead already
(4) the user thread stack may be running out of space
(5) the user thread stack pointer may point to a bad memory address

Answer (just give numbers): _____

(Q11) (2 points) In the code below, **setitimer()** was used to trigger SIGALRM to be delivered and **pause()** was used to wait for SIGALRM to occur.

```
struct itimerval timerval;
... /* setup timerval to timeout in 10ms */
sigset(SIGALRM, DoSomethingInteresting);
setitimer(ITIMER_REAL, &timerval, 0);
pause();
```

Assuming the code has no compile-time error, under what condition would the above code **freeze**?

(1) SIGALRM is delivered immediately before **pause()** is called
(2) the **DoSomethingInteresting** function does not exist
(3) SIGALRM is generated immediately after **pause()** is called
(4) the thread calling **pause()** may self-terminate
(5) the code can never freeze

Answer (just give numbers): _____

(Q12) (2 points) For an **open file**, what **context information** is stored in a **file object** in the kernel's system file table?

(1) reference count
(2) access mode
(3) file type
(4) file ownership
(5) file size

Answer (just give numbers): _____

(Q13) (2 points) What are the ways a **user thread** would **trap** into the kernel?

(1) deliver a signal
(2) call **write()**
(3) dereference a NULL pointer
(4) use up its time-slice
(5) open a file for reading

Answer (just give numbers): _____

(Q14) (2 points) Under what **general condition** would using mutex to access shared data by concurrent threads be 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 some threads just want to read the shared data
(3) when the execution order of threads is mostly predictable
(4) when access pattern is random and the probability of accessing the same part of shared data simultaneously is small
(5) when one thread is the parent thread of another thread

Answer (just give numbers): _____

(Q15) (2 points) Which of the following statements are correct about a Unix **command shell**?

(1) the commmad shell can have at most one child process
(2) the command shell knows how to perform I/O redirection
(3) since the command shell can run another program in the background, it must be multi-threaded
(4) the command shell knows how to launch another program
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q16) (2 points) Which of the following statements are correct?

(1) PIO devices are considered more "intelligent" than DMA devices
(2) DMA devices needs to be told what operation to perform
(3) DMA devices can perform writes to physical memory
(4) PIO (Programmed I/O) devices can only work after the CPU downloads a "program" into it
(5) PIO devices can only read from physical memory but cannot write to physical memory

Answer (just give numbers): _____

(Q17) (2 points) Why must a process enter a **zombie** state immediately after it terminates?

(1) because its process ID needs to be recycled immediately
(2) because its parent process hasn't retrieve this process' return/exit code
(3) because operating system can run faster this way
(4) because its children processes may not have died
(5) because a process cannot delete its own stack

Answer (just give numbers): _____

(Q18) (2 points) What are the operations that are **locked** together in an **atomic operation** by **sigsuspend(set)** (where **set** specifies a set of signals)?

(1) if a signal specified in **set** is to be delivered, it calls the corresponding signal handler
(2) it suspends all other threads in the same process
(3) it unblocks the signals specified in **set**
(4) it blocks all signals
(5) it waits for the signals specified in **set**

Answer (just give numbers): _____