CSci 402 - Operating Systems

Alternate Midterm Exam

Summer 2021

(7:00:00pm - 7:40:00pm, Tuesday, July 6)

Instructor: Bill Cheng

Teaching Assistant: (N/A)

(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 is accessing a particular device at a time
(2) there is only one thread running in the **weenix** kernel, so there is really no need for kernel mutexes
(3) since the **weenix** kernel is preemptive, kernel mutexes must be implemented
(4) since the **weenix** is meant to run on a uniprocessor, there is no need for kernel mutexes
(5) **weenix** uses mutexes whenever a kernel thread needs to update a kernel linked list data structure

Answer (just give numbers): _____

(Q2) (2 points) What does **async-signal safety** mean?

(1) make sure that your code that handles an asynchronous signal cannot corrupt shared data structures
(2) one should always use synchronous signals
(3) make your code safe when working with asynchronous signals
(4) never use a global variable in a signal handler
(5) do as little as possible in a signal handler

Answer (just give numbers): _____

(Q3) (2 points) Which of the following statements are correct?

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

Answer (just give numbers): _____

(Q4) (2 points) Let's say that your **umask** is set to **0256**. (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?

(Q5) (2 points) In the code below, **setitimer()** was used to generate SIGALRM, **pause()** was used to wait for SIGALRM to occur, and the SIGALRM handler is simple and does not do anything destructive.

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

Answer (just give numbers): _____

(Q6) (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 running at a time
(2) the command shell knows how to launch another program as its child process
(3) since the command shell can run another program in the background, it must be multi-threaded
(4) the command shell knows how to perform I/O redirection for its child processes
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q7) (2 points) Let's say that you are executing a C function named **foobar()** on an x86 processor and your thread is executing code somewhere in the middle of **foobar()** (no interrupt is happening). Under what condition (your answers will be logically AND'ed together) would the base/frame pointer (i.e., ebp) of the x86 processor **not** pointing at **foobar**'s stack frame at this time?

(1) if **foobar()** returns **void**
(2) if **foobar()** does not call another function
(3) if **foobar()** is not a recursive function
(4) if **foobar()** does not have function arguments
(5) if **foobar()** does not have local variables

Answer (just give numbers): _____

(Q8) (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   4   10000
5000   2    8000
4000   1    3000
1000   3    5000
```

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).

(Q9) (2 points) In a multi-threaded process, if a **detached** thread calls **pthread_exit()**, it cannot delete its thread control block (TCB) and its stack. Which of the following statements are correct about when and/or where the TCB and the stack of this **detached** thread get cleaned up (i.e., deleted)?

(1) only the kernel can perform such a task
(2) such a task can only be performed inside a signal handler
(3) a reaper thread can be used to perform such a task
(4) such a task can only be performed only when the process dies
(5) another thread in the same process can perform such a task when it's convenient

Answer (just give numbers): _____

(Q10) (2 points) Which of the following statements are correct about a newly created thread?

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

Answer (just give numbers): _____

(Q11) (2 points) For an **open file** in Unix, what **context information** is stored in a **file object** in the kernel's system file table?

(1) user ID and group ID of file
(2) the size of the file (number of bytes)
(3) file type
(4) access mode to remember how the file was opened
(5) cursor position

Answer (just give numbers): _____

(Q12) (2 points) In a multi-threaded process, what are the ways a thread can **self terminate** without killing the process it's in?

(1) send itself a SIGTERM signal
(2) return from its first procedure
(3) calls **pthread_exit()**
(4) join with itself
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q13) (2 points) What was involved in POSIX's **solution** to provide **thread safety** for accessing the global variable **errno**?

(1) make accessing **errno** trap into the kernel so that the kernel can provide thread safety
(2) use thread-specific **errno** stored in thread-specific storage inside thread control block
(3) generate a software interrupt when **errno** is accessed so that the kernel can provide thread safety
(4) generate a segmentation fault when **errno** is accessed so that the kernel can provide thread safety
(5) define **errno** to be a macro/function call that takes threadID as an argument

Answer (just give numbers): _____

(Q14) (2 points) Which statements are correct about **copy-on-write**?

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

Answer (just give numbers): _____

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

(1) a system call is invoked by a user process
(2) keyboard interrupt to terminate a user process
(3) trap into the kernel
(4) the kernel invokes code that's part of a user program
(5) context switching from privileged mode to user mode

Answer (just give numbers): _____

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

(1) a directory can contain multiple subdirectories
(2) a symbolic link to the same file can be contained in multiple directories
(3) a symbolic link to the same file **cannot** be contained in multiple directories
(4) a Unix file system is a strict tree hierarchy
(5) a directory can be contained in multiple (parent) directories

Answer (just give numbers): _____

(Q17) (2 points) What are the **general mechanisms** an operating system would use to give a user process access to something in the **extended address space** while not giving it direct access to any kernel data structure?

(1) use a hash value for a kernel hash table
(2) use a semiphore
(3) use an index into a kernel array
(4) use a handle
(5) use an address of the kernel data structure so that when the user process tries to dereference it, it will trap into the kernel

Answer (just give numbers): _____

(Q18) (2 points) The code below is a suggested "solution" to the **barrier synchronization 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 code won't work because spontaneous return of **pthread_cond_wait()** can break the code
(2) the n^{th} thread can get stuck at the barrier after it wakes up other threads
(3) the barrier may collapse unexpectedly
(4) when the n^{th} thread wakes up all the other threads, not all of them may leave the barrier
(5) some thread may leave the barrier before the n^{th} thread arrives

Answer (just give numbers): _____