CSci 402 - Operating Systems

Midterm Exam (DEN Section)

Fall 2022

(10:00:00am - 10:40:00am, Wednesday, Oct 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) Let say that X, Y, Z are regular Unix user processes and X's parent is Y and Y's parent is Z. When process Y dies, what would happen?

(1) the OS kernel will terminate process X because its parent is now dead
(2) the init process becomes process X's new parent
(3) if process X is dead already, process X will become parentless
(4) process Z becomes process X's new parent
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q2) (2 points) In the most common OS implementation, when a user thread makes a system call, it simply becomes a kernel thread. In such an implementation, how is this kernel thread different from the original user thread?

(1) they access different part of the physical memory
(2) they use different CPU registers to point to their respective text segments
(3) they use different CPU registers to point to their respective stack segments
(4) their text segments are in different virtual address ranges, their stack segments are also in different virtual address ranges
(5) CPU modes are different

Answer (just give numbers): _____

(Q3) (2 points) Which of the following statements are correct about signals and interrups?

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

Answer (just give numbers): _____

(Q4) (2 points) Let's say that **foobar** is a valid program name for a very simple program. You type "`foobar &`" in a command shell to run **foobar** in the **background**. What **system calls** must the **command shell** make before **foobar** finished running?

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

Answer (just give numbers): _____

(Q5) (2 points) The **file descriptor table** for a process is maintained inside the kernel. What security problems could occur if such a table and **open file context** information is maintained and accessible in **user space** and would pass pointers to these data structures into the kernel when making file-related system calls?

(1) a user program will be able to execute a specific file to which it has only read+write access
(2) a user program will be able to read from a specific file to which it has no access
(3) a user program will be able to execute a specific file to which it has only read access
(4) a user program will be able to write to a specific file to which it has only read access
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q6) (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 threads that access shared data rarely make system calls
(2) when one thread is the parent thread of another thread
(3) when the execution order of threads is mostly predictable
(4) when some threads only want to write to the shared data and promise not to read the shared data
(5) when different threads would access different parts of shared data most of the time and only access the same parts of shared data occasionally

Answer (just give numbers): _____

(Q7) (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) some threads may leave the barrier before the n^{th} thread arrives
(2) the n^{th} thread can get stuck at the barrier after it wakes up other threads
(3) the only problem with the above code is that spontaneous return of **pthread_cond_wait()** can break the code
(4) the "barrier" may disappear unexpectedly
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q8) (2 points) Which of the following statements are correct about kernel mutexes in **weenix**?

(1) **weenix** uses mutexes for mutual exclusion whenever a kernel thread needs to update any kernel linked list that another thread may use
(2) since there is only one CPU in **weenix**, there is no reason for having kernel mutexes
(3) since the **weenix** kernel is preemptive, kernel mutexes must be implemented
(4) there is only one real thread running in the **weenix** kernel, so there is really no need for kernel mutexes
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q9) (2 points) Let's say that your **umask** is set to **0450**. (a) What file permissions will you get (in octal) if use an editor to create the **warmup1.c** file? (b) What file permissions will you get (in octal) if use a compiler to create the **warmup1** executable?

(Q10) (2 points) In the code below, **setitimer()** is used to generate SIGALRM when the alarm goes off, **pause()** is used to wait for SIGALRM to occur, and the SIGALRM handler is simple and does not do anything that can cause any problem.

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

Answer (just give numbers): _____

(Q11) (2 points) What does **async-signal safety** mean?

(1) do as little as possible in a signal handler
(2) one should always use synchronous signals
(3) never use a global variable in a signal handler
(4) make sure that your code that handles an asynchronous signal is "safe", e.g., cannot corrupt data structures shared with the rest of your program
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q12) (2 points) What are the types of **isolation** that a general-purpose operating system must provide?

(1) isolate user processes from each other
(2) isolate kernel device drivers from the OS kernel
(3) isolate the OS kernel from user processes
(4) isolate threads within the same process from each other
(5) isolate file system from process management in the OS kernel

Answer (just give numbers): _____

(Q13) (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 can map a file (or part of a file) into your address space
(2) block I/O is only available to kernel processes and not available to user processes
(3) using read/write() system calls can also be considered as doing block I/O
(4) block I/O is performed by first calling lseek() and then read/write()
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q14) (2 points) Which of the following are **not** the purpose of a **process**?

(1) initializes the stack space when a new thread is created
(2) keeps track of threads that belongs to the process
(3) maintains an address space
(4) keep track of files that have been opened and closed by the running program
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q15) (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 the buddy system runs out of memory, it's your bug
(2) without QEMU, you can still run weenix
(3) if the run queue is empty, it's your bug
(4) if you get a kernel panic, it's your bug
(5) if the kernel deadlocks when the grader is grading your work, it's your bug that causes the deadlock

Answer (just give numbers): _____

(Q16) (2 points) An object file (i.e., a **.o** file) is divided into sections that correspond to memory segments. **Within each section**, what types of information are kept there?

(1) instructions for recompilation
(2) names of library files that contain the actual data for undefined symbols in that memory segment
(3) owner and group owner information of the memory segment
(4) order in which global variables must be initialized
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q17) (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   2   11
5   4    6
2   2    7
4   4    3
```

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).

(Q18) (2 points) In a multi-threaded process, if a **detached** POSIX thread calls **pthread_exit()**, it cannot delete its user-space stack. Which of the following statements are correct about when and/or where the stack of this **detached** thread get freed up (i.e., destroyed)?

(1) only the kernel can perform such a task
(2) such a task can only be performed only when a thread in the process calls **exit()**
(3) such a task can only be performed inside a signal handler
(4) a "reaper" process can be used to perform such a task
(5) none of the above is a correct answer

Answer (just give numbers): _____