CSci 402 - Operating Systems

Alternate Midterm Exam

Fall 2021

(7:00:00pm - 7:40:00pm, Wednesday, Oct 27)

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 the scheduling in **weenix**?

(1) **weenix** scheduler is a simple sequential (e.g., first-in-first-out) scheduler
(2) in **weenix**, the scheduler is responsible for handling cancellation and termination of kernel threads
(3) when a kernel thread in **weenix** goes to sleep, it must sleep in **some** queue
(4) **weenix** kernel uses scheduler functions such as **sched_wakeup_on()** and **sched_broadcast_on()** to give the CPU to a kernel thread
(5) none of the above is a correct answer

Answer (just give numbers): ____________________

(Q2) (2 points) Comparing cancellation in the **weenix** kernel and pthreads cancellation, which of the following statements are correct?

(1) just like pthreads, you can change **weenix** kernel thread cancellation "state" (enabled/disable) programmatically
(2) unlike pthreads, you cannot change **weenix** kernel thread cancellation "type" (asynchronous/deferred) programmatically
(3) since the **weenix** kernel is non-preemptive, it cannot have cancellation points
(4) **weenix** kernel thread cancellation "state" is always enabled and "type" is always deferred
(5) none of the above is a correct answer

Answer (just give numbers): ____________________

(Q3) (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 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) SIGALRM is generated immediately after **pause()** is called
(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) the code can never freeze

Answer (just give numbers): ____________________

(Q4) (2 points) Which statements are correct about Unix signals?

(1) when a signal is generated, if it's blocked, it is destroyed
(2) an application can ask the OS to not deliver certain signals
(3) when a signal is generated, if it's blocked, it can still be delivered sometimes
(4) by convention, a signal handler should return a value of zero if the signal was handled successfully and return a non-zero value otherwise
(5) none of the above is a correct answer

Answer (just give numbers): ______________________________

(Q5) (2 points) Which of the following statements are correct about a Unix **command shell**?

(1) the command shell knows how to launch another program as its child process
(2) the commmad shell can have at most one child process running at a time
(3) since the command shell can run another program in the background, it must be multi-threaded
(4) the command shell can redirect **stdout** for its child process to go into a file
(5) none of the above is a correct answer

Answer (just give numbers): ______________________________

(Q6) (2 points) What is the difference between a **hard link** and a **soft/symbolic link** in Unix?

(1) a hard link cannot be added to an existing directory while a soft link can
(2) when you add a hard link to a file, the kernel increases reference count in the corresponding **file object** while a soft link does not
(3) a hard link is an inode reference while a soft link is not
(4) a hard link can link to a file in another file system while a soft link cannot
(5) none of the above is a correct answer

Answer (just give numbers): ______________________________

(Q7) (2 points) Which of the following statements are correct?

(1) a signal is generated by a user process and can be delivered to another user process without using a system call
(2) a signal is a software interrupt
(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): ____________________

(Q8) (2 points) Which of the following statements are correct?

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

Answer (just give numbers): ____________________

(Q9) (2 points) What are the operations that are **locked** together in an **atomic operation** by **sigwait(set)** (where **set** specifies a set of signals)?

(1) it unblocked the signals specified in **set**
(2) if a signal specified in **set** is pending, it calls the corresponding signal handler
(3) it waits for any signal specified in **set**
(4) it clears all pending signal specified in **set**
(5) none of the above is a correct answer

Answer (just give numbers): ____________________

(Q10) (2 points) Assuming regular Unix/Linux calling convention when one C function calls another, which of the following statements are true about an **x86 stack frame**?

(1) the values of all CPU register are automatically saved in the caller's stack frame
(2) the EBP register points to something that looks like a doubly-linked-list in the stack
(3) content of the EIP register is never saved in a the stack frame
(4) content of the EAX register is never saved in a stack frame
(5) content of the ESP register is often saved in the callee's stack frame

Answer (just give numbers): ______________________________

(Q11) (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 unexpectedly, what happens to process X?

(1) the OS kernel will terminate process X
(2) even if process X is dead already, process X must still be reparented properly
(3) the init process becomes process X's new parent
(4) process X becomes parentless
(5) process Z becomes process X's new parent

Answer (just give numbers): ______________________________

(Q12) (2 points) What's the reason why it is not possible for a thread (with its code written in C) to **completely kill itself**?

(1) a thread cannot delete its own stack
(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 switch to itself
(5) a thread cannot have two stacks

Answer (just give numbers): ______________________________

(Q13) (2 points) Let's say that your **umask** is set to **0450**. (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?

(Q14) (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**?

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

Answer (just give numbers): ______________________________

(Q15) (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) when the n^{th} thread wakes up all the other threads, not all of them may leave the barrier
(2) the n^{th} thread can get stuck at the barrier after it wakes up other threads
(3) the barrier may collapse unexpectedly
(4) some thread may leave the barrier before the n^{th} thread arrives
(5) the code won't work because spontaneous return of **pthread_cond_wait()** can break the code

Answer (just give numbers): ______________________________

(Q16) (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    7000
5000   2    8000
2000   2    1000
1000   3    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).

(Q17) (2 points) Which statements are correct about **copy-on-write**?

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

Answer (just give numbers): ______________________________

(Q18) (2 points) In a multi-threaded process, if a **detached** POSIX 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 freed up (i.e., destroyed)?

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

Answer (just give numbers): ______________________________