CSci 402 - Operating Systems

Midterm Exam (PM Section)

Fall 2021

(12:00:00pm - 12:40:00pm, Wednesday, Oct 27)

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) 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;
}
```

In what way is this **switch()** function different from a regular function?

(1) this function does not touch the ebp register
(2) this function references global variables
(3) this function is unusually short
(4) this function uses no local variables and that's unusual
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q2) (2 points) After a process has entered the **zombie** state, how would it exit the **zombie** state?

(1) when the kernel perform garbage collection (i.e., automatic freeing of useless dynamically allocatd objects)
(2) when all its child processes have exited their zombie states
(3) when all its threads have also entered zombie states
(4) when its parent process enters a cancellation point
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q3) (2 points) Which of the following are sure ways a **user thread** would **trap** into the kernel immediately?

(1) dereference a NULL pointer
(2) call **free()**
(3) unlock a mutex
(4) open a file for reading
(5) deliver a signal

Answer (just give numbers): _____

(Q4) (2 points) If **sigwait(set)** is implemented correctly in every way **except** that it is **not an atomic operation**, what bad things can happen (assuming that everything else is done correctly)?

(1) the thread calling **sigwait()** may become uncancellable
(2) the thread calling **sigwait()** may get terminated while other threads are not affected
(3) a signal specified in **set** may be lost
(4) a signal specified in **set** may become pending forever and never delivered
(5) **sigwait()** may never return

Answer (just give numbers): _____

(Q5) (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 semiphore
(2) use a "handle"
(3) use an index into a kernel array
(4) use a hash value for a kernel hash table
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q6) (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) order in which variables must be initialized in that segment
(2) owner and group information of that segment
(3) what symbols are undefined in that segment
(4) relation instruction
(5) names of library files that contain the actual data for undefined symbols in that segment

Answer (just give numbers): _____

(Q7) (2 points) Assuming that you only have one processor, which of the following statements are correct about **interrupts**, **traps**, and **Unix signals**?

(1) for all signal types, a user program can catch that signal by specifying a user space signal handler
(2) a signal can be blocked/disabled while an interrupt cannot be blocked/disabled
(3) an interrupt is most likely caused by the currently running program
(4) a trap can only be caused by the currently running program
(5) if a program is not currently running, it is not possible for it to cause a trap

Answer (just give numbers): _____

(Q8) (2 points) Which of the following are **not** the purpose of a **process**?

(1) keeps track of threads that belongs to the process
(2) keep track of files that have been opened and closed by the program
(3) maintains an address space
(4) under multithreading, decides which thread to run next
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q9) (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 stack frames may be upside-down
(2) the user thread code may not be using a compatible stack register
(3) the user thread stack space may have very little room left (i.e., it is running out of stack space)
(4) the user thread may be dead already
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q10) (2 points) Let's say that a user thread (thread X) is executing when an unrelated hardware interrupt occurs. Let's further assume that the interrupt handler uses the most common implementation (especially regarding the way kernel stack is chosen to be used by the interrupt handler). The interrupt handler must execute till completion before thread X is resumed even though the interrupt has nothing to do with thread X. What bad thing can happen if we resume thread X before the interrupt handler finishes?

(1) thread X can make a system call and corrupt the stack the interrupt handler is using
(2) thread X can cause a deadlock in user space and never return the CPU to the interrupt handler
(3) thread X can be running in the kernel already as a kernel thread and it can make a function call and corrupt the stack the interrupt handler is using
(4) there is nothing thread X can to to mess up the interrupt handler
(5) none of the above is a correct answer

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    4    8000
5000    2    9000
4000    1    3000
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).

(Q12) (2 points) When static linking is used, which of the following are **not** the job of a **linker**?

(1) maintain reference count for file system hard links
(2) compile source code into relocatable object files
(3) determines final addresses for global variables that need to be relocated
(4) allocate and manage the heap space
(5) figure out what to do with unresolved symbols

Answer (just give numbers): _____

(Q13) (2 points) Let's say that you use a thread to catch <Cntrl+C> using the code below (please assume that SIGINT is blocked everywhere else in the process, all the variables have been properly initialized, and you have implemented everything else correctly):

```
void *monitor() {
  int sig;
  while (1) {
    sigwait(&set, &sig);
    pthread_mutex_lock(&m);
    display(&state);
    pthread_mutex_unlock(&m);
  }
  return(0);
}
```

If the user pressed <Cntrl+C> and **sigwait()** returns. What would happen if the user pressed another <Cntrl+C> before **sigwait()** is called again in the next iteration of the **while** loop?

(1) the 2nd <Cntrl+C> becomes pending
(2) the 2nd <Cntrl+C> is lost
(3) SIGKILL will be delivered to your program and your program will be killed
(4) SIGINT will be delivered the next time **sigwait()** is called
(5) the 2nd <Cntrl+C> will cause the default SIGINT handler to get executed

Answer (just give numbers): _____

(Q14) (2 points) Let's say that your **umask** is set to **0252**. (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) In Unix, a directory is a file. Therefore, I will use the term "directory file" to refer to the **content** of a directory. Which of the following statements are **incorrect** about a Unix directory?

(1) if file Y is in directory X, the file size of Y is stored in the directory file for X
(2) if file Y is in directory X, the inode number of file Y is stored in the directory file for X
(3) a directory file contains a mapping from file names to disk addresses
(4) a directory file contains a mapping from strings to integers
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q16) (2 points) If **foobar** is a valid program name, you type **foobar** in a command shell and wait for **foobar** to finish executing, what **system calls** must be the **command shell** make before **foobar** finished running?

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

Answer (just give numbers): _____

(Q17) (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 and have it sleep there even if another thread has already signaled/broadcasted the cv
(2) signal the condition
(3) lock the mutex
(4) unlock the mutex
(5) broadcast the condition

Answer (just give numbers): _____

(Q18) (2 points) In a multi-threaded process, how can a thread **self terminate** without killing the process it's in?

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

Answer (just give numbers): _____