CSci 402 - Operating Systems
Midterm Exam (DEN Section)
Spring 2024

[10:00:00am-10:40:00am), Wednesday, March 20

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) Which of the following statements are correct about devices?

(1) PIO devices are considered more "intelligent" than DMA devices
(2) DMA device can only function correctly after the CPU downloads a "program" into it
(3) PIO devices can neither read from physical memory nor write into physical memory
(4) DMA devices are considered "intelligent" devices because they won't need to be told what operation to perform
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q2) (2 points) What was involved in POSIX's **solution** to provide **thread safety** for accessing the global variable **errno** when a system call returns?

(1) generate a software interrupt when **errno** is accessed so that the kernel can provide thread safety
(2) define **errno** to be a macro/function call that takes the PID of the calling process as an argument
(3) generate a segmentation fault when **errno** is accessed so that the kernel can provide thread safety
(4) use thread-specific **errno** stored inside the thread control block
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q3) (2 points) In a multi-threaded user process, if a user thread X (that's **not detached**) calls **pthread_exit()**, thread X would go into the zombie state. When can the **user space stack of thread X** get deleted?

(1) when thread X calls pthread_cancel() to cancel itself
(2) when another thread signals that it's okay for thread X to delete its stack
(3) when the process thread X is in goes into the zombie state
(4) when another thread calls pthread_cancel() to cancel thread X
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q4) (2 points) Assuming that the "`...`" in the code below represents meaningful code and the program compiles perfectly.

```
int main(int argc, char *argv[])
{
  int i;
  for (i = 0; i < 10; i++)
    pthread_create(...);
  return 0;
}
```

Which of the following are **bad things that can happen** with the above code?

(1) all child threads may finish before the main thread dies
(2) some child threads may deadlock and the process will never die
(3) cannot predict the execution order of the main thread and the child threads
(4) **pthread_exit()** may get called before any child thread get a chance to run
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q5) (2 points) Why must a user process enter a **zombie** state after it calls **exit()**?

(1) because its process ID is not ready to be recycled
(2) because user process cannot delete its own user space stack
(3) because operating system can be more secure
(4) because its parent process has not retrieve the return/exit code/status of this process
(5) because it may have child processes that are still running

Answer (just give numbers): _____

(Q6) (2 points) Let's say that you have a uniprocessor and the kernel is currently servicing an interrupt when a **higher priority interrupt** occurs. What would typically happen?

(1) the lower priority interrupt will be aborted so that the higher priority interrupt can be serviced immediately
(2) the higher priority interrupt will be lost
(3) the higher priority interrupt will become pending and will get delivered when the current interrupt handler gives up the processor voluntarily
(4) to execute its interrupt service routine (ISR), the higher priority interrupt must use a kernel stack that's different from the kernel stack used by the ISR of the lower priority interrupt
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q7) (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 some threads just want to read the shared data and promise not to modify the shared data
(3) when the execution order of threads is mostly predictable
(4) when one thread is the parent thread of another thread
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q8) (2 points) Which statements are correct about **copy-on-write**?

(1) without copy-on-write, the exit() system call may take a lot longer to execute
(2) copy-on-write occurs every time a user process writes to virtual memory
(3) without copy-on-write, pthread_mutex_lock() may take a lot longer to execute
(4) without copy-on-write, the fork() system call may take a lot longer to execute
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q9) (2 points) Which of the following statements are correct about **weenix**?

(1) **weenix** kernel uses scheduler functions such as sched_wakeup_on() and sched_broadcast_on() to give the CPU to a kernel thread
(2) when a kernel thread in **weenix** goes to sleep, it must sleep in **some** queue
(3) in **weenix**, the scheduler decides which thread gets cancelled and which thread gets terminated
(4) for your kernel 1 to work correctly, it's sufficient that the **weenix** scheduler is a simple sequential (e.g., first-in-first-out) scheduler
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q10) (2 points) Let's say that your **umask** is set to **0500**. (a) What file permissions will you get (in octal) if you use an editor to create the **warmup1.c** file? (b) What file permissions will you get (in octal) if you use a compiler to create the **warmup1** executable? Please note that if your answer is not in octal, you will not get any credit.

(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 1000 -t g0.txt`" and the content of `g0.txt` is as follows:

```
4
3   8   5
4   9   8
4   3   1
2   1   1
```

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).

(Q12) (2 points) Which of the following statements are correct about a Unix **command shell**?

(1) the command shell can redirect **stdout** for its child process to go into a file
(2) the command shell can create another child process while running a child process in the background
(3) the command shell can redirect either **stdin** or **stdout**, but it cannot redirect both **stdin** and **stdout** simultaneously
(4) the commmad shell can have at most one child process running at a time
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q13) (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 (assuming that you are running on an x86 CPU)?

(1) their text segments are in different virtual address ranges, their stack segments are also in different virtual address ranges
(2) user thread runs with interrupt enabled while kernel thread runs with interrupt disabled
(3) they use different CPU registers to point to their respective stack segment
(4) CPU modes are different
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q14) (2 points) If your CPU is currently executing code in user mode, which of the following will definitely cause the CPU to go into the privileged mode immediately?

(1) user program executes an instruction to divide an integer by zero
(2) user program calls free()
(3) user program calls malloc()
(4) user program executes a software interrupt machine instruction
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q15) (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) content of the EIP register must be saved in the callee's stack frame
(2) the values of all CPU registers are automatically saved in the callee's stack frame
(3) content of the ESP register must be saved in the callee's stack frame
(4) content of the EBP register is never saved in a the stack frame
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q16) (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) in this solution, no thread can "starve"
(2) a writer thread may never get to write even if there are no reader thread present
(3) two reader threads may block each other out for a long time
(4) a reader thread may block forever if writer threads keep arriving and there are always writer threads present
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q17) (2 points) Let **cv** be a POSIX condition variable and **m** be a POSIX mutex. When thread X calls **pthread_cond_wait(cv,m)**, several operations are **locked** together within one **atomic operation**. Which of the following operations are included within this atomic operation?

(1) broadcast the cv
(2) lock the cv
(3) thread X adds itself to mutex m's queue
(4) unlock the cv
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q18) (2 points) When **sigsuspend(set)** is called (where **set** specifies a set of signals), some operations are **locked** together within one **atomic operation**. Which of the following operations are included within this atomic operation?

(1) it blocks all signals
(2) it replaces the signal mask with the mask specified by **set**
(3) it waits for signal delivery
(4) it calls all signal handlers specified in **set**
(5) if unblocks all signals

Answer (just give numbers): _____