CSci 402 - Operating Systems

Midterm Exam (TT Section)

Spring 2024

[9:30:00am-10:10:00am), Thursday, March 21

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's say that your **umask** is set to **0475**. (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.

(Q2) (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 C function (please note that a system call is a C function)?

(1) unlike most other functions, the thread that calls this function can fall asleep inside this function and wake up at a later time
(2) it would appear to an abserver of the system that the thread that calls this function is not the thread that returns from this function
(3) this function does not modify the content of the EBP register during its execution
(4) this function does not modify the content of the ESP register during its execution
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q3) (2 points) When static linking is used, which of the following are the responsibilities of a **linker**?

(1) determines global variable addresses
(2) determines the address of the main() function
(3) figure out the number of bytes a data structure would take up
(4) compile source code files into relocatable object files
(5) determines how large the heap memory segment needs to be

Answer (just give numbers): _____

(Q4) (2 points) Assuming that you always call functions with valid arguments, which of the following are actions a **user thread** can take to always **cause a trap** into the kernel?

(1) call switch()
(2) call malloc()
(3) call wait()
(4) call free()
(5) call open()

Answer (just give numbers): _____

(Q5) (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 correct about a Unix directory?

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

Answer (just give numbers): _____

(Q6) (2 points) Which of the following statements are correct about a newly created thread?

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

Answer (just give numbers): _____

(Q7) (2 points) The code below is the body of a function called barrier_sync(), which was suggested as a "solution" to the **barrier synchronization problem** (to synchronize `n` threads) and it's 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 if it's used to synchronize first n threads that call barrier_sync()?

(1) the n^{th} thread can get stuck at the barrier after it wakes up other threads
(2) some threads may leave the barrier before the n^{th} thread arrives
(3) the first thread that enteres the barrier will be the first thread that leaves the barrier
(4) the above code doesn't work because only the n^{th} thread will leave the barrier while the other $n-1$ threads will get stuck at the barrier
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q8) (2 points) Which of the following statements are correct about a Unix **pipe**?

(1) a pipe is not exactly like a file; you can use **lseek()** to move the cursor position to the beginning of a file, but you may not be able to use **lseek()** to move the cursor position to the beginning of a pipe
(2) a pipe has two ends, a user program decides which end is for reading and which end is for writing
(3) a pipe can be used by one user thread to send characters to another user thread in the same process
(4) a pipe object lives in the user space
(5) a pipe is like a file and you can use **lseek()** to move the cursor position to the beginning of a the pipe just like you can move it to the beginning of a file

Answer (just give numbers): _____

(Q9) (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
2    4    5
3    6    9
2    4    2
5    1    2
```

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).

(Q10) (2 points) What does **async-signal safety** mean?

(1) make sure that your code that handles an asynchronous signal won't cause deadlocks
(2) do as little as possible inside a signal handler
(3) never use a global variable inside a signal handler
(4) one should always deal with signals synchronously
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q11) (2 points) Comparing cancellation in the **weenix** kernel and pthreads cancellation, which of the following statements are correct?

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

Answer (just give numbers): _____

(Q12) (2 points) Let say that X, Y, Z are regular Unix user processes and Z's parent is Y and Y's parent is X. When process Y dies, what would happen?

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

Answer (just give numbers): _____

(Q13) (2 points) Which of the following statements are true about **Unix devices**?

(1) a device's major device number indicates which process is currently using the device
(2) a device's major device number must be smaller than the device's minor device number
(3) it's possible to have a device that can be accesses only using a minor device number and it has no associated major device number
(4) a device's major device number identifies a device driver
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q14) (2 points) Which of the following statements are correct about a **Unix file system**?

(1) hard links to the same file can be contained in multiple directories
(2) a Unix file system is a strict tree hierarchy, i.e., except for the root node, every node in the tree has exactly one "parent" node
(3) symbolic links to the same file cannot be contained in multiple directories
(4) a directory can have more than one "parent" directory
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q15) (2 points) What does **thread safety** mean?

(1) make library code safe to run under multi-threading
(2) you should never use a global variable inside a signal handler
(3) must use a safe scheduler to schedule threads
(4) when you write multi-threaded code, you must make sure that threads would never deadlock
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q16) (2 points) Let's say that your thread is executing C code somewhere in the middle of a C function named foobar() on an x86 processor. If you know that the base/frame pointer (i.e., EBP) of the x86 processor is **not** pointing anywhere within foobar()'s stack frame, what conclusion can you make?

(1) foobar() does not access uninitialized global variables
(2) foobar() has uninitialized local variables
(3) foobar() does not have local variables or all its local variables are unused
(4) foobar() does not make function calls
(5) foobar() does not have function arguments or all its function arguments are unused

Answer (just give numbers): _____

(Q17) (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) block I/O is only available to kernel processes and not available to user processes
(2) using the write() system call can also be considered as doing block I/O
(3) block I/O is available to both kernel and user processes
(4) to perform block I/O, you need to first map a file (or part of a file) into your address space
(5) block I/O is performed by first calling lseek() and then call either the read() system call or the write() system call

Answer (just give numbers): _____

(Q18) (2 points) What is the difference between a **hard link** and a **soft/symbolic link** in Unix?

(1) when you add a hard link to a file, the kernel increases reference count in the corresponding **file object** while it doesn't do that for a soft link
(2) a hard link cannot be added to point/link to a directory while a soft link can
(3) a hard link is a file while a soft link is just a inode reference
(4) when you add a soft link to a file, the kernel increases reference count in the corresponding **file object** while it doesn't do that for a hard link
(5) on systems with multiple file systems, a hard link can link to a file in another file system while a soft link cannot

Answer (just give numbers): _____