# CSci 402 - Operating Systems

# Midterm Exam

# Summer 2022

*[9:30:00am-10:10:00am), Tuesday, July 5*

Instructor: Bill Cheng

Teaching Assistant: Zhuojin Li / Bowen Song

*( 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 are actions a **user thread** can take to **cause a trap** into the kernel?

(1) deliver a signal
(2) call **exit()**
(3) dereference a NULL pointer
(4) call **switch()**
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q2) (2 points) Let's say that **foobar** is a valid program name. You type **foobar** in a command shell to run **foobar** in the foreground. What **system calls** must the **command shell** make before **foobar** finished running?

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

Answer (just give numbers): _____

(Q3) (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 make function calls
(2) **foobar()** has uninitialized local variables
(3) the function return type of **foobar()** is **void**
(4) **foobar()** is a recursive function
(5) **foobar()** does not have function arguments

Answer (just give numbers): _____

(Q4) (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) if file Y is in directory X, the file size of Y is stored in the directory file for X
(2) a directory file contains a mapping from file names to disk addresses
(3) if directory Z is a subdirectory of directory X, the inode number of directory Z is stored in the directory file for X
(4) if directory Z is a subdirectory of directory X, 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): _____

(Q5) (2 points) Let's say that your **umask** is set to **0345**. (1) What file permissions will you get (in octal) if use an editor to create the **warmup1.c** file? (2) What file permissions will you get (in octal) if use a compiler to create the **warmup1** executable?

(Q6) (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 is unusually short
(2) this function references global variables
(3) unlike most other functions, the thread that calls this function can fall asleep inside this function and wake up at a later time
(4) this function does not change the content of the ebp register
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q7) (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    6
4    2    9
1    4    2
3    2    2
```

How many **milliseconds** 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).

(Q8) (2 points) What is the difference between a **hard link** and a **soft/symbolic link** in Unix?

(1) on systems with multiple file systems, a hard link cannot link to a file in another file system while a soft link can
(2) a hard link can be added to link to an existing directory while a soft link cannot
(3) 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
(4) a hard link is an inode reference while a soft link is not
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q9) (2 points) Which statements are correct about **Unix signals**?

(1) if a SIGINT handler is invoked to deliver SIGINT, SIGINT is blocked inside that SIGINT handler
(2) an application can ask the OS to not deliver certain signals
(3) when a signal is generated, if it's blocked, it is lost
(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) when a signal is generated, if it's blocked, the kernel can get it delivered by creating a new user thread

Answer (just give numbers): _____

(Q10) (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 only problem with the above code is that spontaneous return of **pthread_cond_wait()** can break the code
(3) the $n^{th}$ thread can get stuck at the barrier after it wakes up other threads
(4) the "barrier" may disappear unexpectedly
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q11) (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) make accessing **errno** trap into the kernel so that the kernel can provide thread safety
(3) use thread-specific **errno** stored inside the process control block
(4) generate a segmentation fault when **errno** is accessed so that the kernel can provide thread safety
(5) define **errno** to be a macro/function call that takes the thread identifier of the calling thread as an argument

Answer (just give numbers): _____

(Q12) (2 points) Which of the following statements are correct about a newly created thread?

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

Answer (just give numbers): _____

(Q13) (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) owner and group owner information of th memory segment
(2) names of library files that contain the actual data for undefined symbols in that memory segment
(3) order in which global variables must be initialized
(4) what symbols are undefined in that memory segment
(5) instructions for recompilation

Answer (just give numbers): _____

(Q14) (2 points) Every device in Unix is identified by a major device number and a minor device number. Which of the following statements are true about these device numbers?

(1) major device number must be smaller than the minor device number
(2) major device number indicated which process is currently using the device
(3) minor device number identifies a device driver
(4) minor device number is rarely used in the OS
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q15) (2 points) Which of the following statements are correct about kernel mutexes in **weenix**?

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

Answer (just give numbers): _____

(Q16) (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;
}
```

What may be the main issue/problem (select **only one**) with the above code?

(1) **pthread_exit()** may get called before any child thread get a chance to run
(2) deadlock would occur
(3) main thread may run in parallel with the child threads
(4) all child threads may finish before the main thread dies
(5) none of the above is a correct answer

Answer (just give numbers): _____

(Q17) (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 "handle"
(2) use a semaphore
(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): _____

(Q18) (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 use different CPU registers to point to their respective text segments
(2) they access different part of the physical memory
(3) CPU mode is different
(4) their text segments have different virtual addresses
(5) they use different CPU registers to point to their respective stack segments

Answer (just give numbers): _____