CSci 402 - Operating Systems

Midterm Exam (DEN Section)

Spring 2021

(10:00:00am - 10:40:00am, Wednesday, March 24)

Instructor: Bill Cheng

Teaching Assistant: Ben Yan

(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) What is true about **interrupts** and **Unix signals**?

(1) an interrupt is delivered to a user program while a signal is delivered to the kernel
(2) a signal can be blocked/disabled while an interrupt cannot be blocked/disabled
(3) a user program can specify what interrupt service routine to use for a given interrupt
(4) an interrupt is delivered to the kernel while a signal is delivered to a user program
(5) a user program can specify what function to call when a signal is delivered

Answer (just give numbers): ______________________________

(Q2) (2 points) Suppose that you have already implemented the following **guarded command** using a POSIX mutex **m** and a POSIX condition variable **c** where the calling thread can be blocked on a call to **pthread_cond_wait(&c,&m)**.

```
when (guard) [
  my_atomic_func();
]
```

Also suppose that you have implemented a function call **modify_guard()** that would **modify a variable** in the above `guard`. What is the proper way (using **pthread function calls**) to invoke the **modify_guard()** function to work coherently with the above guarded command? Your answer must include a call to **modify_guard()**. Please assume that all the variables have been properly initialized.

(Q3) (2 points) Which of the following statements are correct about the scheduler in **weenix**?

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

Answer (just give numbers): ______________________________

(Q4) (2 points) What is the difference between a **hard link** and a **soft/symbolic link** in Unix?

(1) a hard link can link to a file in another file system while a soft link cannot
(2) a hard link increases reference count in a **file object** while a soft link does not
(3) a hard link cannot be added to an existing directory while a soft link can
(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): ____________________

(Q5) (2 points) After a process has entered the **zombie** state, how would it exit the **zombie** state?

(1) when its parent process calls **exit()**
(2) when all it's child processes have died
(3) when the wall clock advanced to the next day
(4) when the kernel perform garbage collection
(5) none of the above is a correct answer

Answer (just give numbers): ____________________

(Q6) (2 points) What are the **general mechanism** for the operating system to **provide access** to something in the **extended address space** for a user process while not giving it direct access to any kernel data structure?

(1) use an address of the kernel data structure so that when the user process tries to dereference it, it will trap into the kernel
(2) use an index into a kernel array
(3) use a handle
(4) use an encryption key
(5) use a hash value for a kernel hash table

Answer (just give numbers): ____________________

(Q7) (2 points) The **file descriptor table** for a process is maintained inside the kernel. What security problem would occur if such a table and **open file context** information is maintained in **user space**?

(1) a user program will be able to write to a specific file that it has read-only access
(2) a user program will be able to execute a specific file that it has only read access
(3) a user program will be able to execute a specific file that it has read+write access
(4) a user program will be able to read a specific file that it does not have access
(5) none of the above is a correct answer

Answer (just give numbers): ______________________________

(Q8) (2 points) Which of the following systems do **not** have Unix as one of its direct ancestor(s)?

(1)	Solaris	(3)	Windows 10	(5)	Mac OS X
(2)	Mac iOS	(4)	Android		

Answer (just give numbers): ______________________________

(Q9) (2 points) On Unix/Linux systems, **sequential I/O** on files is done by calling **read/write()** system calls. Which of the following statements are true about Unix/Linux **block I/O** on files?

(1) block I/O is done by mapping files into address space
(2) block I/O is done by calling **lseek()** and then **read/write()**
(3) block I/O is not available to user processes
(4) using **read/write()** is also considered block I/O
(5) none of the above is a correct answer

Answer (just give numbers): ______________________________

(Q10) (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 can typically happen?

(1) higher priority interrupt will be blocked until the current interrupt handler yields the processor
(2) nothing, because when an interrupt handler is running, all other interrupts are blocked/disabled
(3) the higher priority interrupt will use the same stack as the current interrupt handler
(4) the higher priority interrupt will execute with a newly allocated kernel stack
(5) this is a violation and will cause a kernel panic

Answer (just give numbers): ______________________________

(Q11) (2 points) Which of the following is part of an actual Unix **upcall**?

(1) keyboard interrupt to terminate a user process
(2) a system call is invoked by a user process
(3) trap into the kernel
(4) the kernel invokes code that's part of a user program
(5) timer interrupt to switch from one user process to another

Answer (just give numbers): ______________________________

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

(1) a thread cannot have two stacks
(2) a thread cannot delete its own stack
(3) a thread cannot be in user space and in kernel space simultaneously
(4) a thread cannot be running inside two functions simultaneously
(5) none of the above is a correct answer

Answer (just give numbers): ______________________________

(Q13) (2 points) If **foobar** is a valid program name, you type **foobar** in a command shell and wait for it to finish executing, what **system calls** will get called by the **command shell**?

(1) **fork()**
(2) **exit()**
(3) **one of the exec system calls**
(4) **open()**
(5) **wait()**

Answer (just give numbers): ______________________________

(Q14) (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> will cause the default SIGINT handler to execute
(2) SIGINT will be delivered the next time **sigwait()** is called
(3) your program will crash
(4) the 2nd <Cntrl+C> becomes pending
(5) the 2nd <Cntrl+C> is lost

Answer (just give numbers): ______________________

(Q15) (2 points) What does **async-signal safety** mean?

(1) one should always use synchronous signals
(2) never use a global variable in a signal handler
(3) do as little as possible in a signal handler
(4) make sure that an asynchronous signal cannot corrupt data structures
(5) make your code safe when working with asynchronous signals

Answer (just give numbers): ______________________

(Q16) (2 points) Which statements are correct about **copy-on-write** (as compared to not implementing copy-on-write in the kernel)?

(1) without copy-on-write, **execl()** may take longer
(2) copy-on-write often increases physical memory usage
(3) without copy-on-write, **fork()** may take longer
(4) copy-on-write occurs every time a user process writes to virtual memory
(5) none of the above is a correct answer

Answer (just give numbers): ____________________

(Q17) (2 points) What are the types of **isolation** that a general-purpose operating system must provide?

(1) isolate user processes from each other
(2) isolate the OS kernel from user processes
(3) isolate file system from process management in the OS kernel
(4) isolate threads within the same process from each other
(5) isolate kernel device drivers from the OS kernel

Answer (just give numbers): ____________________

(Q18) (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 X is in directory Y, the file size of X is stored in the directory file for Y
(2) a directory file contains a mapping from strings to integers
(3) if file X is in directory Y, the inode number of file X is stored in the directory file for Y
(4) a directory file contains a mapping from file names to disk addresses
(5) none of the above is a correct answer

Answer (just give numbers): ____________________