Practice Midterm Exam

CS252 Spring 2012

Name:_____

Question	Max.	Current
True/False	10 pts.	
Short Questions 1-4	20 pts.	
Short Questions 5-8	25 pts.	
9.	15 pts.	
10.	15 pts.	
11	15 pts.	
	Total:	

Part 1. True False Questions

Answer True/False (T/F) and give a short justification. (2 points each)

____ Shared libraries add to the length of an executable file.

____ POSIX sets a standard for the UNIX interface.

____ /etc/passwd contains the passwords of the users in UNIX

____ Race conditions are more likely to happen in multiprocessor machines.

____ If a file is removed from a directory, the blocks it uses in disk are returned immediately to the list of free blocks.

Part 2. Short questions.

2. (5 pts.) Enumerate the memory sections of a program. Also describe what is stored in each section.

3. (5 pts.) Describe the differences between a dynamic linker and a runtime linker.

4. (5 pts.) What are the advantages and disadvantages of using threads vs. using processes?

5. (5 pts) Assume that the program "hello" crashes with SEGV and generates a core file. Write down the gdb invocation and the gdb commands that you will type to show the stack-trace where the program crashed.

6. (10 pts) How many times does the following code print “Hello World!”?

```
void* printHello(void* arg) {
    if (arg == NULL) {
        pthread_t thr;
        pthread_create(&thr, NULL, printHello, (void*)1);
    } else {
        fork();
    }
    printf("Hello World!\n");
}

int main() {
    int ret = fork();
    if (ret == 0) {
        printHello(NULL);
    }
    ret = fork() && ret;
    if (ret == 0) {
         printHello(NULL);
    }
    pthread_exit();
    return 0;
}
```

7. (5 pts) Assume that the following code for removeFirst for removing items from a linked list. removeFirst will remove and return the first item in the list or -1 if the list is empty. a) What problems do you see could happen if multiple threads try to run it? b) Rewrite the code in removeFirst to solve the problems and any mutex calls if necessary.

```
struct ListNode {
  int value;
  ListNode * next;
};

struct ListNode * head = NULL;

int removeFirst (int value)
{
  if (head == NULL) {
     return -1;
  }

  ListNode * tmp = head;
  head = head->next;
  int val = tmp->value;
  free(tmp);
  return val;
}

void insertFirst(int val) {
.....
}

main() {


 // Initialize list.


}
```

New Code

8. (5 pts.) A program calls the system call read(file_descriptor, buffer, nbytes) that reads from disk. Explain step by step the sequence of events, the checks, the interrupts in user mode and kernel mode and the change of process state that happen from the time write() is called until it returns.

Part 3. Programming questions.

9. (15 pts.) Write a shell script that will take as argument a user login, then it will run forever and it will check every minute if the user has logged in or logged out. It will send e-mail to you when the user logs in or when the user logs out. Hint. Use the commands who, grep etc.

10. (15 pts.) Write a program "grepsort arg1 arg2 arg3" that implements the command "grep arg1 | sort < arg2 >> arg3". The program should not return until the command finishes. "arg1", "arg2", and "arg3" are passed as arguments to the program. Example of the usage is **"grepsort hello infile outfile"**. This command will print the entries in file **infile** that contain the string **hello** and will append the output sorted to file **outfile**. Do error checking. Notice that the output is appended to arg3.

```
int main( int argc, char ** argv)
{









}
```

11. (15 pts) Write in C++ and semaphores a class EventListener where a thread calling waitForEvent() will block until another thread calls signalEvent(). Multiple threads can call waitForEvent(). Also, if signalEvent() is called and no threads are calling waitForEvent(), the signal will be lost. **Hint**. Have a semaphore that the threads calling waitForEvent can wait for (what is the initial counter?). A thread calling waitForEvent() will wait on the semaphores until another thread calls signalEvent will post the semaphore the number of threads waiting. The counter of the semaphore is not accessible so you will need to keep your own counter.

```
class EventListener{




public:
  EventListener();
  void waitForEvent();
  signalEvent();
};

EventListener::EventListener()
{




}

void EventListener::waitForEvent()
{




}

void EventListener::signalEvent()
{




}
```