CSci 144 Introduction to Operating Systems
Quiz 2, October 2, 2017

1. (0.8 points)Please write a Unix/Linix code segment that creates a new process. The new process executes an executable file called "prog" and prints out "prog starts…". The parent process waits for the new process to terminate and then prints out "program finishes…".

   Refer to page 112, Figure 3.8 or page 105, Figure 3.5.

```
int child_pid = fork();

if(child_pid == 0) {
        exec("prog.exe", NULL);
        cout << "Child program starts…";
}
else {
        cout << "program finishes…";
        wait(child_pid);

        return 0;
}
```

2. (0.4 point) *Green thread* is a type of pure user-level implementation of thread, which is different from kernel-level threads. What's the *limitation* of green threads? Please briefly justify.

   Page 167.

   Limitation: The kernel is unaware of the threads in user level and therefore cannot do context switch if necessary.

   Justification: If one user-level thread makes a blocking I/O call, then kernel does not context switch to another user-level thread, leading to the blocking of the whole user process.

3. (0.4 point) What events or actions usually cause a thread to make context switch from RUNNING state to READY state? Please list two examples.

   Page 150

   Event/*Action* 1: Voluntary Yielding, e.g., sleep() statement.

   Event/*Action* 2: Involuntary Timed Interrupt, e.g., busy waiting.

4. (0.4 point) What does UNIX "open" system call do, i.e., the *procedures* it performs? Why should it be atomic?

   Refer to page 109, first sentence of the last paragraph in the section "open vs create vs stat".

   Procedure: test if the file exists, optionally create it if it does not, and then open it.

   Reason: on a multi-user system, some other user might have created the file in between the call to test for its existence, and the call to create the file. Likewise, some other use might have deleted the file between the call to create and the call to open.