University of Victoria
CSC 360: Operating Systems
Dr. Jianping Pan

ASSIGNMENT 2
Due: January 26th, 2018
William H. Grosset
V00820930

1. Most modern processors provide two modes of operation: user mode and kernel mode. Please answer the following questions concisely in a bullet point format.

A. What are the main differences between these two modes? [0.25]

- Higher-level tasks are executed in user mode and use the available API from the kernel to make system calls.
- A program in user mode must use a system call in order to switch into kernel mode.
- Privileged and accessible commands in kernel mode (CPU scheduling, I/O handling, direct access to all memory).

B. From the viewpoint of operating systems, why are they needed? [0.25]

- Avoid users having direct access to hardware, possibly preventing major issues.
- Given API to add a layer of abstraction and only give out necessary functionality.

C. What are the main differences between mode switch and context switch? [0.25]

- A context switch occurs in kernel mode and requires the program state to be saved and then later restored via a process control block.
- A mode switch occurs when a task needs lower-level privileges between user and kernel mode (system call).

D. What are the pros and cons of micro-kernel structures in operating systems? [0.25]

- Pros:
    - Independent modules that communicate via message passing.
    - More accessibility for system and application programs.
- Cons:
    - Overheard between kernel and user applications.

2. In the following example, assume all system and library calls always complete with no error.

```
#define OUTPUT printf("%d\n", i)

main() {
    int i=0; OUTPUT;

    if (fork()) {
        i+=2; OUTPUT;
    } else {
        i++; OUTPUT; return(0);
    }
}
```

A. Please write down all possible outputs when running this program. [1]

Output 1:	Output 2:
0	0
2	2
1	

B. Add one system call in the pseudo code to ensure that the output values are always in increasing order. [1]

```
if (fork()) {
    wait(NULL);
    i+=2; OUTPUT;
} else {
    i++; OUTPUT; return(0);
}
```

3. Processes have three major states: running, blocked (also known as waiting), and ready. For each of the following state transitions, explain whether it is feasible: if feasible, give an example; if not, give reason. [2]

A. running-to-blocked

FEASIBLE. Process running and then waiting for user input (I/O).

B. blocked-to-running

NOT FEASIBLE. A blocked process must be unblocked and have its state set back to being ready before being dispatched and executed.

C. blocked-to-ready

FEASIBLE. Process was blocked until `wait(5)` had completed, which now changes the process state back to ready.

D. ready-to-blocked

NOT FEASIBLE. A process's state must be active and running before it is blocked from another source.

E. ready-to-running

FEASIBLE. Process has been dispatched from the scheduler and can now execute it's content.

F. running-to-ready

FEASIBLE. Process was interrupted due to a scheduler program (round-robin, shortest-job) queuing a different process.