EECE.4810/EECE.5730: Operating Systems

Spring 2017

Homework 1

Due **3:15 PM**, **Wednesday, 2/1/17**

Notes:

- While typed solutions are preferred, handwritten solutions are acceptable.
- Any electronic submission must be in a single file. Archive files will not be accepted.
  - As noted in the syllabus, you will lose 10 points if you fail to follow this rule.
- Electronic submissions should be e-mailed to Dr. Geiger at Michael_Geiger@uml.edu.
- EECE.4810 students must complete problems 1-4, for a total of 50 points.
- EECE.5730 students must complete problems 1-6, for a total of 75 points.

1. (10 points)

a. (3 points) Briefly describe the characteristics of zombie and orphan processes.

b. (3 points) When a process becomes a zombie, what information about that process is still maintained in the operating system? Why?

c. (4 points) Are there conditions under which you would want a process to be orphaned? If so, give an example of a good situation in which a process should be orphaned.

2. (15 points) List the five states in which a process can exist, describe why a process would transition into that state, and describe why a process might transition out of that state.

3. (10 points) For the program below, assume the parent process has process ID (PID) 4810, and the child process has PID 5730. If the `getpid()` function returns the PID of the currently executing process, what will the program print?

```
int main() {
   pid_t pid, pid1;

   pid = fork();
   if (pid == 0) {
      pid1 = getpid();
      printf("child: pid = %d\n", pid);
      printf("child: pid1 = %d\n", pid1);
   }
   else if (pid > 0) {  // Parent process
      pid1 = getpid();
      printf("parent: pid = %d\n", pid);
      printf("parent: pid1 = %d\n", pid1);
      wait(NULL);
   }
   return 0;
}
```

4. (15 points) Briefly describe each of the components of a process control block and explain why the operating system needs to track each of these pieces of information.

5. (15 points, *EECE.5730 only*) Including the initial parent process, how many separate processes are created by the program shown below? Draw a process tree showing all of the processes created to help explain your answer.

```
int main() {
   for (int i = 0; i < 4; i++)
      fork();

   return 0;
}
```

6. (10 points, *EECE.5730 only*) What will the program below print?

```
int nums[5] = {0,1,2,3,4};

int main() {
   int i;
   pid_t pid;

   pid = fork();

   if (pid == 0) {
      for (i = 0; i < 5; i++) {
         nums[i] *= -I;
         printf("CHILD: %d\n", nums[i]);
      }
   }
   else if (pid > 0) {
      wait(NULL);
      for (i = 0; i < 5; i++)
         printf("PARENT: %d\n", nums[i]);
   }
}
```