CIS 415 – Operating Systems  
Winter 2013

Homework Assignment 2  
Processes and Threads  
January 24th, 2013  
Assignment due: Friday, February 1st, 2013, at the beginning of the class.

This assignment’s objective is to review the concepts studied in class about processes and threads. Upon completion of the assignment, the student will have a firmer grasp of those concepts, and questions to be clarified before taking the midterm exam.

Solve the following exercises:

1. Construct a process tree similar to Figure 3.9. To obtain process information for the UNIX or Linux system, use the command `ps -aef`. Use the command `man ps` to get more information about the `ps` command. On Windows systems, you will have to use the task manager.

2. Including the initial parent process, how many processes are created by the program shown in Figure 3.28?

```c
#include <stdio.h>
#include <unistd.h>

int main()
{
    /* fork a child process */
    fork();

    /* fork another child process */
    fork();

    /* and fork another */
    fork();

    return 0;
}
```

**Figure 3.28** How many processes are created?
3. Using the program in Figure 3.29, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main()
{
    pid_t pid, pid1;

    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error occurred */
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { /* child process */
        pid1 = getpid();
        printf("child: pid = %d",pid); /* A */
        printf("child: pid1 = %d",pid1); /* B */
    }
    else { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d",pid); /* C */
        printf("parent: pid1 = %d",pid1); /* D */
        wait(NULL);
    }

    return 0;
}
```

Figure 3.29 What are the pid values?

4. Consider the RPC mechanism. Describe the undesirable circumstances that could arise from not enforcing either the “at most once” or “exactly once” semantics. Describe possible uses for a mechanism that had neither of these guarantees.