COSC1112/1114: Operating Systems Principles

Lab 3 (week 4)

1. Create a separate process using the Unix fork system call.

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

int main(){
      pid_t pid;
      /* fork a child process */
      pid = fork();
      if (pid < 0) { /* error occurred */
            fprintf(stderr, "Fork Failed");
            return 1;
      }
      else if (pid == 0) { /* child process */
            execlp("/bin/ls","ls",NULL);
      }
      else { /* parent process */
            /* parent will wait for the child to complete */
            wait(NULL);
            printf("Child Complete");
      }
      return 0;
}
```

2. In the child process, you can use execlp to run the code you developed in the last lab. Suppose you compiled the code and generated an executable file named as “readmyfile”. Try to replace execlp("/bin/ls","ls",NULL); by execlp("./readmyfile","readmyfile",NULL).

3. Run the program below, identify the values of pid at lines A, B, C, and D and understand why (assume that the actual pids of the parent and child are 2600 and 2603, respectively.)

```
#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\n",pid); /* A */
        printf("child: pid1 = %d\n",pid1); /* B */
    }
    else { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d\n",pid); /* C */
        printf("parent: pid1 = %d\n",pid1); /* D */
        wait(NULL);
    }
    return 0;
}
```