Project 2—Process Programming Practices-**80 points**

(**Due date: 10/4/2010/Monday Midnight at Moodle**)

Your name:	Date:

==========How To Submit—Read Carefully, Pease!!==========

1. Create a directory "**project2_YourLastName**" (you must use this format for the directory name for this project; **Use Your Last Name. For example, if your** last name is Smith, you should create directory with the name of "project2_Smith"
2. Create "**project21src**", "and "**project22src**" subdirectories under "project2_YourLastName" directory.
3. When having finished your project, copy the **source files (*.java, or *.c)** to these subdirectories, respectively—you should keep this folders clean: ***only source code*** files included.
4. A "readme" file is required for the project write-up that tells how to compile/run the programs and result screenshots … keep this readme simple!

a. This "readme" must reside in the "**project2_YourLastName**" directory in the format of **.pdf,** or **.doc/docx**.

5. Compress the "**project2_YourLastName**" directory and its contents into a **zip** or **rar** file with same name.
6. Submit the compressed file to the instructor by email.
7. **Penalty** for NOT following these submission instructions (10% ~30%)

1. (**30 points**) First practice with the following program in Ubuntu (pay attention to functions "execlp", "printf"—you may need to use search engines to learn how to use these functions under Linux).

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

int main(){

pid_t pid;

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

   if (pid < 0) { /* error occurred */
         fprintf(stderr, "Fork Failed\n");
         exit(-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.\n");
            exit(0);
      }
}
```

Then edit the above program (with ***name "project21.c", and compile the file by using "cc project21.c", and then execute the run file with command "./a.out")*** to complete the following:

1. In the child process, instead of executing "ls" command, you need to execute "uname -r" command;
2. In the parent process, print out the child's PID (which is the pid number the OS has assigned to the child process);
3. Take a screenshot of the result and copy the screenshot and source code file for submission.

2. (**50 points**) Practice with the following UNIX program to learn how implement "pipe" in Ubuntu:

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

#define BUFFER_SIZE 25
#define READ_END    0
#define WRITE_END   1

int main(void)
{
  char write_msg[BUFFER_SIZE] = "Greetings";
  char read_msg[BUFFER_SIZE];
  pid_t pid;
  int fd[2];

  /** create the pipe */
  if (pipe(fd) == -1) {
        fprintf(stderr,"Pipe failed");
        return 1;
  }

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

  if (pid < 0) {
        fprintf(stderr, "Fork failed");
        return 1;
  }

  if (pid > 0) {  /* parent process */
        /* close the unused end of the pipe */
        close(fd[READ_END]);

        /* write to the pipe */
        write(fd[WRITE_END], write_msg, strlen(write_msg)+1);

        /* close the write end of the pipe */
        close(fd[WRITE_END]);
```

```
    }
    else { /* child process */
          /* close the unused end of the pipe */
          close(fd[WRITE_END]);

          /* read from the pipe */
          read(fd[READ_END], read_msg, BUFFER_SIZE);
          printf("child read %s\n",read_msg);

          /* close the write end of the pipe */
          close(fd[READ_END]);
    }

    return 0;
}
```

Then edit the above program (with name "project22.c") to complete the following:

1. Design and implement a program using pipes in which (1) one process sends the string "Greetings" to a second process; (2) the second process adds a string like "**from Beifang**" (you need to use your firstname) to the end of the string received from the first process, and (3) the second process sends this new (combined) string (i.e., "Greetings from Beifang") back to the first process. (This will require using two pipes: one for sending the original message from the first to the second process, and the other for sending the modified message from the second back to the first process.
2. Take a screenshot of the result and copy the screenshot and source code file for submission.