CS360 – Operating Systems & Architecture I
Program 1 – Create child process

Task 1: Write a program that spawns four simultaneous children.

1. The child program(s) should execute a different program than the parent is executing (i.e. the child will need to invoke a version of the exec() system call (i.e. execvp() after the child is created from the fork() instruction. The different child processes can execute the same program but should be able to identify the different process ID for each process.

2. The parent process should print out:
   1. the parent's own total user time (the total user time might be 0 for this small program).
   2. the total accumulative user time of all children (which also might be 0).
   3. the parent's own process id (pid)
   4. the process id (pid) of the forked children

3. The parent program should create the child processes by using a loop rather than duplicating sections of code for each child. So if you needed to create many more child processes, it would require only a simple modification to a constant in your program.

4. The child process should print out
   1. the child's total user time (the total user time might be 0 for this small program).
   2. the child's own process id (pid)
   3. the parent process id (ppid)

5. It is very important that the parent and child processes generate informational messages so that you can verify, without a doubt, that all the child processes are executing concurrently.

6. Sample programs for a parent creating a child and the code to obtain the time statistics for a process are available on Blackboard in the Content > Program Examples folder.

7. You should submit your program via the *handin* utility on csa. Your parent and child programs should be documented well. As part of the handin process, be sure to compile both programs and then execute the parent program. The output of the programs (parent and forked children) is critical to providing verification that the programs work as outlined above.

Task 2: Write a program that spawns four children and writes to a common file.

1. We have discussed that a child created on a Linux system shares a number of resources with the parent,including open files.

2. The parent should open a file and write a message to it.

3. Each child should write a message to the same file.

4. The parent should close the file.

5. There should only be one open statement and one close statement executed by the parent. The children should not open or close any file.

6. Similar to the first task above, the messages that are written to the file should contain enough information to verify that the parent and each child wrote to the file.

7. The parent and child should also print informational messages to the monitor in much the same way as the first task above.

8. After you run the program, at the system prompt in *handin*, list the file contents by using the "more ” or "cat " commands.

3. You should also submit a review of your program output.

Verify that you have reviewed the output.
Verify that the output is correct or, if it is not, what is in error and how it could it be corrected.