Operating System Lab, CPE435-Lab2

Process – A process is an instance of a program. The new process is created by issuing fork system call. Each process has own area of memory, protected against modification by other processes

The fork system call creates a copy of a process that was executing. The process that executed the fork is called the parent process and the new process is called the child process. . The only difference in the returns from the fork system call is that the return value in the parent process is the process id of the newly created child process, while the return value in the child process is zero. If fork it is not successful, -1 is returned.

A process terminates by calling the exit system call. This system call never returns to the caller. When exit is called, an integer exit status is passed by the process to the kernel

- Write a program, which creates a new process. The parent process should print its id and wait for the termination of the child process. The child process should call a function that subtracts two numbers. It should then create a new child process that adds two numbers. The control should then come back to the first child process that now should multiply two numbers and then terminate. The waiting parent process should now resume and terminate the program.
- Write a program, which creates n child processes and prints their process id. The n must be even number and passed as argument, otherwise a message indicates that the number is odd and terminates the program.

HINT:

- **int fork();**

Return either 0 if it is child process or the process id if it is the parent.

- **Exit(**status**)**

. When exit is called, an integer exit status is passed by the process to the kernel.(used usually by child process to terminate and return it is status to the parent process).

- **int wait**(int *status)

A process can wait for one of its child process to finish by executing the wait system call, The value returned by wait is the process id of the child process that terminated