University of Massachusetts Dartmouth
CIS 370, Fall 2013
Lab 4 –More Process Management
10/01/2012, 10/03/2013

Objective

This lab is designed to reiterate the importance of process control and management, particularly when a parent process is responsible for creating multiple children processes. Upon completion of this lab, you should have thorough understanding of the behavior of `fork`, `wait`, and `exit` system calls.

Description

This lab consists of two programming assignments. Please submit two separate C files (one for each part of the assignment). It would behoove you to review the lecture slides on processes (from last week's lab) before you begin.

1. *Triplet Prodigies*

Write a C program called `lastnameTriplets` that accepts 3 integer parameters. The program (parent process = P) should create 3 children, C_1, C_2, and C_3, **sequentially**.

- C1 should print its PID, then calculate and print the sum of the three integers.
- C2 should print its PID, then calculate and print the product of the three integers.
- C3 should print its PID, then calculate and print the sum the squares of the three integers.

The parent should wait for all three children to complete their calculations before printing its own PID and exiting. BE CAREFUL! Only the parent process should create children.

The order of events in your program should look as follows:

1. P creates C1 and waits.
2. C1 performs sum calculation.
3. C1 exits.
4. P creates C2 and waits.
5. C2 performs product calculation.
6. C2 exits.
7. P creates C3 and waits.
8. C3 performs sum of squares calculation.
9. C3 exits.
10. P exits.

Hint: Use the function `atoi()` to convert an argument passed in through the command line into an integer.

For example:

```
atoi(argv[4]);
```

2. *Surviving Quadruplets*

A parent can obviously have more than one child process in existence at any given time. Your task is to write a C program called `lastnameQuadruplets` which will create four children. All children are required to survive until the last of them is "born." This means that the parent should *NOT* create a child, and then wait for it to be killed before creating the second child, etc. All children should print out their own PIDs and then sleep for 5 seconds. This will give the parent time to create all the children before the eldest finishes executing and exits. Only once *ALL* of the quadruplets have finished executing and are killed, should the parent then print out its own PID and exit itself.

Deadline

Section 01 – Tuesday, 10/15/13
Section 02 – Thursday, 10/17/13
(Two weeks!)

Useful Links: Linux/Unix process management

- **Chapter 5**, *Unix System Programming*, Haviland and Gray, Prentice Hall (Lab Book)
- **Chapter 3**, *Operating System Concepts*, Silberschatz, Galvin, Gagne (Course Textbook)
- http://www.advancedlinuxprogramming.com/alp-folder/alp-ch03-processes.pdf