CS474 Operating System

Project #1 (Process Creation)

September 14, 2022

Project Objectives:

The purpose of this project is to introduce students to the concept of process creation.

Total points Available: 100

Due: Sep 14 at 11:59 pm

Project Description:

Your first programming assignment is to do the Collatz conjecture in C with the following changes. You will fork two processes to print their respective sequence for the Collatz conjecture. The first will produce the sequence which is indicated by the number on the command line and the second process produces the sequence from the command line number plus 4. Please print the child (1 or 2) with each number output and be sure the forked processes can run concurrently.

Here is the original Collatz conjecture (problem 3.21 in the textbook):

The Collatz conjecture concerns what happens when we take any positive integer n and apply the following algorithm:

$$n = \begin{cases} n/2, & \text{if n is even} \\ 3 \times n + 1, & \text{if n is odd} \end{cases}$$

The conjecture states that when this algorithm is continually applied, all positive integers will eventually reach 1. For example, if n = 35, the sequence is

35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1

Write a C program using the fork() system call that generates this sequence in the child process. The starting number will be provided from the command line. For example, if 8 is passed as a parameter on the command line, the child process will output 8, 4, 2, 1. Because the parent and child processes have their own copies of the data, it will be necessary for the child to output the sequence. Have the parent invoke the wait() call to wait for the child process to complete before exiting the program. Perform necessary error checking to ensure that a positive integer is passed on the command line.

Sample output:

From Child 2 init n=15, From Child 2 n=46, From Child 2 n=23, From Child 2 n=70, From Child 2 n=35, From Child 2 n=106, From Child 2 n=53, From Child 2 n=160, From Child 2 n=80, From Child 2 n=40, From Child 2 n=20, From Child 2 n=10, From Child 2 n=5, From Child 2 n=16, From Child 2 n=8, From Child 2 n=4, From Child 2 n=2, From Child 2 n=1,

One done!

From Child 1 init n=11, From Child 1 n=34, From Child 1 n=17, From Child 1 n=52, From Child 1 n=26, From Child 1 n=13, From Child 1 n=40, From Child 1 n=20, From Child 1 n=10, From Child 1 n=5, From Child 1 n=16, From Child 1 n=8, From Child 1 n=4, From Child 1 n=2, From Child 1 n=1,

Children Complete

The number entered on the command line must be greater than zero and less than 40. Please put the function code in your file.

You will need to use stdlib.h if you want to use atoi to translate a character string into an integer. Use sprintf to put values into strings. You will need to do wait twice so that the main program finishes after the children (no cascading termination). You will need to use argc and argv to get command line arguments. **Also, observe whether the processes always finish in the order in which they are forked.**

Submitting your assignment

- All files need to be submitted via Canvas including the code and report.
- All files should be zipped together.
- There should be a readme file explaining in detail the exact steps to be taken to compile and execute the code files and the title page
- Testing of this work should be done only on the Computer Lab machines. Please make sure these machines are not locked up due to your code. The execution for grading purposes will be done on the lab machines. (**Be extremely careful that a child process does not itself fork a process or you can fill the process table and lock up the machine.**)
- In case of any code errors, partial credit may be offered based on the code and documentation.
- A report that presents the performance evaluation of your solution.
    - The report should be properly formatted (an academic format style, such as ACM or IEEE being preferred) and contain quantitative data along with your analysis of these data.

Late Submission Policy

Late work will be not accepted. You have 6 fund/slack days. Use them wisely.

Development Environment

You may write your program using any available editor Nano, Emacs, Vi or whatever editor you are most comfortable with, BUT, it must compile with gcc and be executable on one of the Computer Lab machines.

Compiling a program: gcc -o helloworld helloworld.c
Execute the program: ./helloworld

Enter your account credentials by typing your username and password into the login screen if at a physical lab PC, via PuTTY if at a remote Windows PC, or via *ssh yourusername@hostname.cs.nmsu.edu.*

You may go to https://intranet.cs.nmsu.edu/wp/cog/remote-access-to-the-cs-domain/ for detail host list.

To login to these machines remotely, download PUTTY (for Windows, Linux users skip this step) by going to: https://www.chiark.greenend.org.uk/~sgtatham/putty/

Then after the download, execute PUTTY. Their hostnames are *hostname.cs.nmsu.edu*. Then enter login name and password.

Hints:

Build your project in an incremental fashion. Attempt to meet each objective before moving on to the next.

Some useful UNIX commands:

If you are not familiar with Linux here is some useful information. pwd tells you the directory you are in, cd - changes directories, mkdir creates a new directory. The compiler is gcc.