Computer Science 338 - Project 1

Process Trees

due: 9:55 AM, Tuesday, September 26, 2000

This project involves some programming with Unix processes and pthreads. The goal is not to produce a useful program (as will become obvious) but rather to get more practice programming and debugging with multithreaded code, and to see the differences in using threads and full-fledged processes.

Your task is to design and implement a program which takes an integer argument n from the command line and creates a binary tree of processes or threads of depth n. When the tree is created, each process or thread displays an integer which corresponds to its numbering in a breadth-first traversal of the binary tree, its process and thread ID, its position in the tree relative to its parent (left child or right child), and the parent's process and thread ID. After printing this information, it sleeps for one second, and continues on to the next level of the tree. Each process or thread waits for its children to terminate, and when a child terminates, the parent prints the child's process and thread ID.

Your program should build such a tree using using the `fork()` and `wait()` system calls and with threads, specifically *pthreads*. The output I'm looking for can be found by running my version of the program. FreeBSD/i386 binaries are in `/home/faculty/terescoj/shared/cs338`. You may write one program which builds either a process or a thread tree based on a command-line flag, or you may have two separate programs.

What to submit

You may work individually or in groups of two for this project. Your submission should be by electronic mail. Use the *shar* utility to create a single archive which includes the following:

1. Your documented source code
2. A `Makefile` which builds your executable(s) with the command *make*
3. A brief `README` file which explains the contents of each file submitted,

instructions on how to build the executable(s) (which should be "type make"), and instructions on how to run your program(s).

4. A text file with the answers to the discussion questions below

You may work on any unix machine, but it is your responsibility to make sure the submitted code and `Makefile` works on the CSLab public FreeBSD systems, and that the submitted shar file can be extracted properly (try mailing it to yourself and building your project in a new directory before submitting). All files should be clearly marked with your name, or the names of all group members, if it is a group submission.

Discussion Questions

1. What is the largest tree depth which you can run successfully with each version of the program (without getting error returns from `fork()` or `pthread_create()`)? Why? Try this on both the FreeBSD and Solaris platforms.

2. Run the thread version of your code on each available platform which supports threads. I will run my version on several other Unix systems not available in our lab and post the output on the class web page. What do the return values of the `getpid()` calls for the threads tell you about the thread implementation on each platform? Which platforms are using user-level threads and which are using kernel-level threads?

Grading

Your grade on this project will be based on correctness of the program, documentation (both comments in the source code and the `README` file), and your answers to the discussion questions.