COLBY COLLEGE
Computer Science Department

CS357 **Allen B. Downey**
Spring 1998

Assignment 3: Forking Processes

Due: Thursday 2 March

The purpose of this assignment is to get you to play around with fork and exec, and to try out a small experimental design.

1. Pick up the fork program from the following URL and examine it closely.

   `http://rocky.colby.edu/cs357/code/hw3/fork.c`

2. Write a Makefile for the program, based on the Makefile from the previous assignment. On the web page you will find a handout about `gmake`, which is the GNU version of `make`.

   `http://rocky.colby.edu/cs357/handouts/gmake.ps`

3. Compile and run the program. Make sure your output at least resembles the output in the comment.

Lots of processes

1. Right after the `#include` statements, add the line

   ```
   #define CHILDREN 2
   ```

   That defines the constant `CHILDREN` and sets it equal to 2. Note that in C, this type of definition does not end with a semi-colon.

2. Read the man pages for the `fork` and `wait` system calls. If you want to print a man page, you use the -t option, as in `man -t fork`. The result is a Postscript version of the man page, which you can redirect into a file and then print, or you can pipe it directly to lpr:

   ```
   man -t fork | lpr
   ```

3. Modify the program so that instead of creating a single child it creates as many children as the value of `CHILDREN`, such that all the children are running at the same time. Each child should print a message like, "Hello from child 2." (Or whatever number the child happens to be).

   WARNING: please be careful when you write this part of the program, to make sure that the fork command is not contained in an infinite loop. If you create a very large number of processes, you can make the machine very slow, or even crash it.

4. The parent should wait for all its children to exit before it exits. It should print a message acknowledging the death of each child.

5. What do you notice about the order of the messages from the children? What does that tell you about the scheduling policy on this machine?

Make some timings

1. Based on the code from the previous assignment, add timing statements to your code (see the cache-testing program for an example of how to do this). Estimate, as accurately as you can, how long it takes to create a new process, do nothing, and join.

Make the children live longer

1. Add some code to the children so they take a reasonable amount of time (like 1 to 10 seconds). This time should be spent doing computation, not I/O. Be creative.

2. Measure the total execution time of your program as a function of the number of children.

What to turn in

I would like a single report from each group. The report should describe each experiment you did, including, as always:

- What you set out to discover.
- How the experimental apparatus works. You can include code here, but you should include only the relevant parts of the code, and they should be explained clearly.
- How, in theory, you expect the data to look, given an idealized model of the system.
- How, in fact, the data look.
- What, if anything, we can infer from the data.
- What anomalies there are in the data (things that deviate from our theoretical expectation), and any explanation you have for them. Again, it is ok to acknowledge the existence of these anomalies even if you can't explain them.

Have fun!