Assignment 1 – Family Simulation

Assigned: Thursday, February 3, 2011
Due: Thursday, February 17, 2011, 1pm (as instructed below)

Assignment Description

You are to simulate a family tree by using process creation and deletion system calls. Your program will be given an input file `input.txt` which will have the following line format:

<Parent 1> <Parent 2> <n> <Child 1> <Child 2> … <Child n>

This line means that **<Parent 1>** and **<Parent 2>** have **<n>** children whom are named **<Child 1>**, **<Child 2>**, and so on. The `input.txt` file will include several of such lines. Some of the children on one line will possibly be parents at other lines. The parents on first line of the input file are the eldest people in the family tree.

Your program is required to print out the family tree in indented format starting from the first line. Specifically, your program will first create a process for **<Parent 1>**. Notice that there will not be any process for **<Parent 2>**.

After this initialization, each process will perform the followings depending on whether or not they are parent or child:

Parent process:
- create a new process for each of its children
- wait until all the children die

Child process:
- check if myself is going to get married to another person
- print my name and my partner's name as: **<my name>-<partner's name>**

Assume that names of family members are represented by letters (***graduate students cannot use this assumption – see below***). The output must be properly indented so that the children will have tab characters (i.e. "`\t`") based on their generation. For instance, a second-generation child will have two tab characters in front of its printed name.

For example, for an `input.txt` file:

```
A B 3 C D X
D Y 2 M E
M F 0
C P 1 K
```

Your program must output:

```
A-B
        C-P
                K
        D-Y
                M-F
                E
        X
```

In order to make a parent process wait for its children you will have to use a system call named **`waitpid`**. For example, in C notation "**`waitpid(pid, NULL, 0);`**" makes the process wait until the process with ID **`pid`** dies. See UNIX manual pages for further details.

Graduate students only (extra credit for the others): Assume that names of family members can be a word instead of just a letter. Further, your input file `input.txt` will not include the number of children each parent pair has. Also, your program must output its parent's process ID (see the manual pages for the corresponding system call) for each family member. For example, for an `input.txt` file:

```
Adam BB Casey Donna X
Donna Yong Mary Emily
Mary Frank
Casey Paul Karen
```

Your program must output:

```
Adam(0)–BB
      Casey(839)-Paul
            Karen(857)
      Donna(839)-Yong
            Mary(906)-Frank
            Emily(906)
      X(839)
```

Submission Requirements

A **`makefile`** is required. All files in your submission will be copied to the same directory, therefore do not include any paths in your **`makefile`**. The **`makefile`** should include all dependencies that build your program. If a library is included, your **`makefile`** should also build the library.

To make this clear: ***do not hand in any binary or object code files***. All that is required is your source code, a **`makefile`**, and other necessary files as stated in the assignment description. Do test your project by copying the source code only into an empty directory and then compile it by entering the command **`make`**. Your code *must* compile without any errors or warnings and run properly under the Linux system used in the ECC lab (check their Web site for hours). You may develop and test your programs on your own UNIX machine, but it is your responsibility to ensure that they also work properly on the Linux installation of ECC, under the default ECC Lab shell. There will be a heavy penalty if they don't. The specific programming language you use is your choice, but your **`makefile`** has to be compiling your source code and producing a program that is executable in the Linux installation of ECC.

Your source code will be assessed and marked. Commenting is definitely required in your source code.

Go through the following steps to make your submission file ready to submit:

1. Go to your home directory and make a folder named with your name in format LastName_FirstName.
2. Put your source code and all other necessary files in this folder
3. In your home directory type and enter:
   `tar -cvf <`LastName_FirstName**`>.tar <`**LastName_FirstName**`>`**
4. Finally, type and enter:
   `gzip <`LastName_FirstName**`>.tar`**

These above steps will yield a new file named **`<`**LastName_FirstName**`>.tar.gz`** in your home directory. To submit your file, log in to WebCampus and select the CS 446/646 course. From the "Course Tools" panel on left go to 'Assignments" and choose "Assignment 1". Attach your submission file and submit.

If you have any problems please e-mail the instructor or the TA. ***Submissions including multiple files or sent by e-mail will NOT be evaluated.*** Do not turn in printouts in any form. Late assignments will be marked down according to the late policy published on the course Web page.

Good luck!