Assignment 1 – Family Simulation

Assigned: Week of 01 February
Due: Friday, 12 February, 3:00 pm (as specified below)

Assignment Description

You are to simulate a family tree by using process creation and deletion system calls. Your program, called **`family`** will be given an **input file** 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 file will include several lines with this kind of data. Some of the children on one line may be parents on other subsequent lines. The parents on the first line of the input file are the eldest people in the family tree.

Your program must accept the data file as a command-line parameter, and then must 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 following depending on whether or not they are parent or child:

Parent process:

- a new process ***must*** be created for each of a parent's children
  - programs that do not create new processes for each parent action will receive no credit for this part
  - you do not have to create a separate process for the first parent
- wait until all that parent's children die

Child process:

- check if the present name is getting married to another person
- print the present name and the present partner's name as: **`<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 the following input 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
```

Note that all parent's names will be found after their presentation as a child in a previous family. Also note that 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, and note that there are other ways to program a wait.

Graduate student requirements (extra credit for the others): Assume that names of family members can be a word instead of just a letter. Further, your input file 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 the following input 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)
```

Evaluation Component

Once the assignments have uploaded, the Instructor will then randomly dispatch each program to students for grading. This will be implemented no later than the Sunday after the Friday due date. You must download and print a rubric, and evaluate the program code you are provided through WebCampus email by the following Friday. You will also be provided with two sets of input and output examples for the undergraduate and graduate/extra credit requirements. You must use the data provided to evaluate the operation of the program.

The expectation for this program is that of a 400-, or 600- level program. While specific requirements may not be stated, your program must handle any reasonably expected exceptions or errors appropriately, any user interface must be of high quality (although a user interface is not required for this project), any data structures or management must be of high quality and efficiency, and so on. If, at the discretion of your grader, the quality is not found, your credit may be reduced.

As in industry, you are accountable for your evaluation of the program. You must grade the program fairly and consistently, making notes for each grade given. The rubric sheet supports this format. Your grading process will also be graded and <u>your grading score may be reduced if a student successfully challenges your grading evaluation</u>.

Dates:

Program due date: 12 February
Program Distribution: no later than 13 February
Program grading due date: 19 February

Submission Requirements - Program Code Preparation

You may work with any other student or students during the **<u>design</u>** phase of your work. If you work with any other student(s), you must place their names, along with yours in the "Submission" box as fellow designers. When your design is complete and/or you feel prepared to develop the code, you **must** work alone; you may not share code with anyone. Once you create the program code, your name or personally identifiable information must not be found in any submitted file.

A **`makefile`** is required and the C programming language must be used. All files in your submission will be copied to the same directory, therefore do not include any paths in your **`makefile`**. The **`makefile`** must include all dependencies that build your program. If a library is included, your **`makefile`** must also build the library. ***It must be possible to unzip the file in a directory then compile and run it with no further modification or adaptation.*** Program packages that do not successfully accomplish this will earn very little credit on this assignment. 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. Test your project by copying your zipped file into an empty directory, unzipping it, and then compiling it by entering the command **`make`**. Your code *must* compile without any errors or warnings (**`-Wall`** is your friend) 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 Linux 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. If the program cannot be run on an ECC Linux machine, very little credit may be earned.

Your source code and running program will be assessed and graded according to a rubric that will be posted along with the other materials associated with this assignment.

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

1. Go to your home directory and make a folder with your ID Code number which you will find posted in your grade list in WebCampus.

2. Put your source code and all other necessary files in this folder; make sure your name or any personally identifiable information is removed from all submitted files

3. Locate your ID Code, posted in your grade file on WebCampus

4. In your home directory type and enter: **`tar -cvf PA1_<ID Code>.tar PA1_<ID Code>`**

5. Finally, type and enter: **`gzip PA1_<ID_Code>.tar`**

These above steps will yield a new file named **`PA1_<ID Code>.tar.gz`** in your home directory. For example, if your ID Code is 12345, then your file would be: **`PA1_12345.tar.gz`**

Submission Requirements - Upload Process

To submit your materials, log in to WebCampus and select the CS 446/646 course and go to the Programming Assignment 1 link found in the Week 3 folder. Attach your zipped file containing the code materials specified above, and submit.

You are only given one opportunity to upload your program for full credit so make sure everything is correct and acceptable before uploading. If you must upload a newer or corrected version, your credit will be reduced. You may refer to the "How to Upload Your Laboratory Assignments" found in the General Course Information folder. Mistakes made using WebCampus will cause a reduction in grade or possible loss of credit for the assignment. Make sure you do this correctly.

Submissions including multiple code files or sent by e-mail will NOT be evaluated. Also, do not turn in printouts in any form.

Assignments turned in after 3:00 pm but before 12:00 midnight will be awarded 50% of the earned credit. The system will not accept assignments after 12:00 midnight. If you have any problems, email Michael via WebCampus but note that if this happens at or near the deadline time(s), your credit may still be reduced.