April 1st, 2021

CSE344 – System Programming - Homework #2 - v1
Signals and processes (no, this not an April Fool's joke)

**Objective**
Process m (m for mother) will receive as a command-line argument the path to a regular ASCII file. The file will contain at least 8 rows, where each row will contain 16 comma separated real values. Every row is assumed to represent 8 coordinates in the form of :

`x_0,y_0,x_1,y_1,…,x_7,y_7`

where `y_i = f(x_i)` is an unknown function. There are 8 rows, so there are 8 unknown functions. Your task is to estimate these 8 functions through polynomial interpolation using the Lagrange form (check its wiki page) in two rounds. In the first round you will use the first 6 coordinates of each row, and in the second round you will the first 7 coordinates of each row, leaving the last coordinate for validation.

**How**
There are 9 processes involved in this homework. The (parent) process m will receive as a command-line argument the path to a regular ASCII file. She will open that file, and then create 8 children processes (c_0, c_1, …, c_7). Each child process will read the corresponding row of the given file (c_i will read the i-th row, in order of creation), and calculate the Lagrange polynomial p of degree 5 using the 6 first coordinates of that row. Then each child process will estimate p(x_7) using that polynomial and write it to the end of the corresponding row in the file. After which, each child will let the process m know (via a signal) that it has finished its calculations.

Process m, who after creating the children has been waiting in the meanwhile for **ALL** the children processes to finish their calculations, will read the file and calculate the absolute estimation error (i.e. `|f(x_7) – p(x_7)|`) of each row, and print the average error of the 8 polynomials on screen. Then it will signal all the children that they may continue. Pay attention, m must wait for ALL the children to finish their calculations before attempting to collect the errors.

The children processes who after signalling process m, have been waiting in the meantime for process m to signal them, will once again resume computation for the second round, and this time use all 7 coordinates to calculate the Lagrange polynomial r of degree 6 and once again estimate r(x_7), and write it to the end of the corresponding row. Once they are done, the children will print each polynomial's 7 coefficients on screen and terminate.

Process m, who has been waiting again for **ALL** the children processes to finish their calculations, will once again calculate the average error of estimation, print it on screen and terminate.

**Execution example**
```
./processM pathToFile
```

**Output example (the numbers are made up)**
```
Error of polynomial of degree 5: 67.2
Error of polynomial of degree 6: 65.1
Polynomial 2: 1.1,2.2,3.1,1.6,7.6,1.1,5.1
Polynomial 1: 5.1,12.2,3.4,15.6,7.6,1.1,0.1
Polynomial 5: 0.1,2.2,3.4,5.6,7.6,1.1,0.9
Polynomial 7: 1.1,2.2,3.45,51.6,7.6,1.1,0.1
Polynomial 3: 0.1,12.2,3.4,54.6,7.6,1.1,0.1
```

```
Polynomial 4: 6.1,22.2,3.4,75.6,7.6,5.1,5.1
Polynomial 0: 1.1,12.2,13.4,15.6,7.6,1.1,8.1
Polynomial 6: 0.1,2.2,3.4,5.6,7.6,1.1,0.1
```

Use one digit after the decimal during printing.

**Evaluation**
You program will be evaluated with a different input file of the same structure. Your numerical results must be correct, and for that you need to set up the synchronization properly.

**Requirements:**
- All kinds of synchronization between process m and her children will be conducted using only SIGUSR1. `sig_suspend` is your friend. You must use it. Learn it, master it.
- You might want to use a file lock to synchronize the file access of the 8 children processes.
- Do not create additional files, not even temporary ones. All reading, writing must take place with the same input file.
- Process m must clean up after all its children synchronously, by catching the SIGCHLD signals. No zombies.
**- BUSY WAITING and SLEEPING are strictly PROHIBITED. Use signals and sig_suspend for synchronization.**

**Tips**
There are many things that can go wrong in this homework. Synchronization with signals is tricky. Remember the basic principle: **never assume which process runs when, in which order and for how long**. Any combination is possible, so plan accordingly. If you have to assume something, assume that the kernel does its best to ruin your program's flow through process scheduling, if there is a weakness synchronization-wise, it will come out.

**Rules:**
1) Compilation error: grade set to 1; if the error is resolved during the demo, then evaluation continues.
2) Compilation warning (with respect to the **-Wall** flag); -1 for every warning until 10. -20 points if there are more than 10 warnings; no chance of correction at demo.
3) No makefile: -20
4) No pdf (other formats are inadmissible) report submitted (or submitted but insufficient, e.g. 3 lines of text or no design explanation, etc): -20.
5) A report prepared via latex (pdf and tex source submitted together): +5
6) If the required command line arguments are missing/invalid, your program must print usage information and exit. Otherwise: -10
8) The program crashes/freezes and/or doesn't produce expected output with normal input: -80
9) Presence of memory leak (regardless of amount – checked with valgrind) -30
10) Zombie process (regardless of number) -30
11) Deadlock of any kind due to poor synchronization -50
12) Use of poor synchronization: busy waiting/trylock/trywait/timedwait/sleep or any other form other than those specified at homework: -80
13) Late submissions will not be accepted
14) In case of an arbitrary error, exit by printing to stderr a nicely formatted informative message. Otherwise: -10
**15) In case of CTRL-C the program (that means all 9 processes) must stop execution, return all resources to the system and exit with an information message.**

**Is my homework submission valid?**

If process m opens the file and creates the 8 children successfully, then yes.

**Submission rules:**

- Your source files, your makefile and a report; place them all in a directory with your student number as its name, and zip the directory.
- Your report must contain: how you solved this problem, your design decisions, which requirements you achieved and which you have failed.
- The report **must be in English**.
- Your makefile must only compile the program, not run it!
- Do not submit any binary executable files. The TAs will compile them on their own boxes.
- Any deviation from the submission rules will results in automatic (without content evaluation) failure of the homework.

Good luck.