CSE344 Homework2 Report

Yeşim Yalçın-200104004094

1-) Problem Definition and My Solution Approach

The problem was to read coordinates from an ASCII file, processing the coordinates to calculate their covariance matrices in child processes and then find the closest matrices using Frobenius form. Using proper signal calls and handling child processes were also other major aims.

Command Line Argument Analysis
My first action was to analyze the command line arguments. I required exactly 5 command line arguments from the user first being the auto process name. The second and fourth had to be -o and -i (one from each) to determine the place of the input and output files. The argument provided after -o was processed as the output file and the argument provided after -i was processed as the input file.

SIGINT Handling
Afterwards, I continued with opening the files for the main process and assigning a handler for the SIGINT signal. This handler only sets a sigint flag to 1. I check this flag in various places of the program and if I see it as 1, I call the clean function to properly terminate the program.

The clean function closes all opened files, frees the childrenIds pointer, sends SIGINT signal to all alive child processes and also waits for them so that they will not be zombies. At the end deletes the output file after making sure all children are dead. I use the same function to terminate if some error happens so that no memory will be lost and the child processes will not keep running eventhough their parent is terminated.

Child Process Handling
Later on I started reading coordinates from the input file. Right after I collected 10 coordinates, I created a child process by using fork() then used execve() with the right command line arguments and environment variables. I sent the coordinates as string to the child processes. Inside the child processes, I converted them into float.

In my design, right after a child process is created, the main process keeps running until it retrieves all coordinate groups and creates all possible children processes. Moreover, the children processes can run at the same time. I suspend the main process only right before reaching the output file. Because I need to make sure all child processes are done with the output file. I achieve this by calling waitpid() function for each children. (I collect how many children I create and their pids) My first approach was to use a handler for SIGCHLD signal. However, when many children processes terminated at the same time, not all signals were caught and handled. It lead to zombie processes. Therefore, I changed my approach. I also suspend the child processes only if another child process is writing onto the output file. I achieve this by locking the output file with fcntl.

Because I call waitpid() for each children in the main process, no zombie processes are created. Also calling waitpid() right before reaching the output file does not make me lose time because I have to wait before reaching anyway.

Matrix Calculation
I divided the process into 2 parts. I firstly calculated variances of all 3 coordinate variables (x,y,z) and then calculated their corresponding covariances.
While calculating variances:
-I calculated the mean
-Substracted the mean from all x values. Found their power 2. Added all.
-Divided by the x amount.
While calculating covariances: (Ex x-y)
-I calculated the mean for each
-Substracted the means from all x, y values. Multiplied the results.
-Divided by the x amount (or y they are the same)

On the internet there were formulas using division by (amount of x)-1 as well. However divising by the exact amount was adviced by our teacher.

Result Calculation
Afterwards, I read the created matrices from the output file and calculated their Frobenius forms.
While calculating the Frobenius Form:
-I calculated the power 2 of all elements of a matrice. Added all.
-Found the square root of the addition.
By using this form, I found the closest pairs and print them out. Cleaned everything and exit.

2-) System Requirements

1-) As I have mentioned the program requires exactlt 5 command line arguments, 4 of them being inputed by the user.
2-) The program should run with a proper ASCII file.
3-) The program should be run expecting a result using a formula with exact amount division.

You can run the program with ./processP “Insert command line arguments here” command after compiling it using the makefile. The makefile includes an example run command using the provided input file. It can be used by using make run command.

3-) The Parts I Have Completed

I have completed all the requirements of the homework and tested in various ways.