

GTU
CSE344 - System Programming

Homework2 - Report

Ahmet Tuğkan Ayhan
1901042692

In this homework, I used one of my libraries I created in **Homework1** which is **File.h**. Since I explained how it works and handles the possible errors, I won't be explaining it again to prevent report becoming too long.

As said in the instruction pdf, there are 2 different processes in this homework. One is **parentProcess** and other one is **childProcess**. I will explain every function that I created for these processes along with the file format.

Let's start with **parentProcess..**

parentProcess.c

```
1  /* Define values */
2  #define NUM_OF_POINTS 10
3  #define MAX_CHILD 10000
4  /* Global variables */
5  static int sigintFlag = 0;
6  static int readFD = 0;
7  static int writeFD = 0;
8  static int childCount = 0;
9  static int firstChild = 0;
10 static int secondChild = 0;
11 static char * inputFilePath = 0;
12 static char * outputFilePath = 0;
13 /* Parent */
14 void parentProcess(char * fileBuffer, int fileSize);
15 /* Helper Functions of Parent process */
16 void getFilePaths          (int argc, char *argv[]);
17 int  getCoordinates        (char * fileBuffer, int fileIndex, char * x_coordinates, char * y_coordinates, char * z_coordinates);
18 void createChildProcess    (int pidList[MAX_CHILD], char x_coordinates[10], char y_coordinates[10], char z_coordinates[10]);
19 void getCovarianceMatricesFromOutputFile (float covarianceMatrices[childCount][3][3]);
20 float calculateFrobeniusNorm (float covarianceMatrix[3][3]);
21 void getFrobeniusNormOfMatrices (float covarianceMatrices[childCount][3][3], float frobeniusNorm[childCount]);
22 float findTwoClosestMatrices (float frobeniusNorm[childCount]);
23 /* STDOUT dialogues */
24 void printFirstDialogue    ();
25 void printResultToConsole  (float covarianceMatrices[childCount][3][3], float closestValue);
26 void printChildInformation (int count, char x_coordinates[10], char y_coordinates[10], char z_coordinates[10]);
27 /* Signal Handling for Parent */
28 void initializeSignalHandler ();
29 void handleSIG              (int signal);
30 void terminateProgram       (char * fileBuffer, int pidList[MAX_CHILD]);
```

#define

- **NUM_OF_POINTS** : Indicates how many coordinate points there will be for every x, y and z axis.
- **MAX_CHILD** : Actually there is no limit for creating child processes. This one shows limit for pidList which stores every child's pid. This pid used for graceful exiting for every child after SIGINT interrupt.

Global Variables

- **sigintFlag** : When a SIGINT signal is sent to the parent process, this flag is set to 1, initially it is 0.
- **readFD** : Contains file descriptor value for inputFilePath. I make it global because there are many functions that use this descriptor value.

- **writeFD** : Contains file descriptor value for outputPath. This writeFD value later sent to child processes as they created. Thus, closing and opening operations only done in parent process.
- **childCount** : Stores current child count. Also used as index for pidList which stores child process id's
- **firstChild** : After child processes finish, frobenius norm values are calculated. After finding two matrices that has least difference between them, index of first matrix is stored in this global variable.
- **secondChild** : Also index of second matrix is stored in this global variable. Then this information used in covarianceMatrix[childCount][][] array to receive matrix values.
- **inputFilePath** : Inside main, getFilePaths is executed and after doing error checks, input file's path assigned to this char pointer.
- **outputFilePath** : Same as inputFilePath but this one stores command line argument which comes after "-o" indicator.

Functions

- **main** : Main function initializes signal handlers, if there is no error then gets file paths. After everything is ok it opens the both input and output files stores their file descriptor value with readFD and writeFD then calls parentProcess.
- **parentProcess** : Does everything that is explained inside instruction pdf. Prints dialogues, checks if there are 30 or more ASCII character left, turns them into coordinates if there are, creates the child process and sends these coordinates to child process via environment variables(char *agvp[]). Waits child processes to finish, closes files, gets covariance matrices from output file, calculates frobenius norm for each of these matrices and finds the closest two. Finally prints the result to the console.
- **getFilePaths** : Called inside main function. Checks if given arguments are correct or not. If it is then assigns **inputFilePath** and **outputFilePath** values.
- **getCoordinates** : After controlling remaining characters inside input file, it gets these characters and assigns them into x_coordinates[10], y_coordinates[10] and z_coordinates[10].
- **createChildProcess** : Executes fork-exec paradigm. Checks if forking is successful or not. If it is then calls execve function after assigning coordinate values into environment variables(agvp).
- **getCovarianceMatricesFromOutputFile** : When child processes are finished, this function reads the matrices and assigns them into covarianceMatrices[childCount][3][3] 3d array.
- **getFrobeniusNormOfMatrices** : After filling covarianceMatrices array, this function is called inside parent function and every matrix's frobenius norm values are calculated. Then these values are stored inside frobeniusNorm[childCount] array.
- **calculateFrobeniusNorm** : Receives a 3x3 float array and calculates frobenius norm for that matrix, then returns the result back to **getFrobeniusNormOfMatrices** function. It is done for every child.

- **findTwoClosestMatrices** : After every frobenius norm values are calculated, then this function is called inside parent function. This function assigns firstChild and secondChild global variables after finding closest two value.
- **printFirstDialogue** : These print functions just prints necessary informations to the console. This one prints only first line which is “Process P reading inputFile.dat”
- **printResultToConsole** : Prints closest 2 matrices and their frobenius norm difference to the console
- **printChildInformation** : Prints “Created R_* with (...) (...) ... (...)” part.
- **initializeSignalHandler** : Signals are handled with these 3 functions. This one initializes sigset_t and struct sigaction variables. sigset_t variable is used to block every signal except SIGINT until program terminates. Sigaction structure is used to indicate that SIGINT signal is handled by **handleSIG** function.
- **handleSIG** : Receives signal value and handles that signal according to that value. For this homework there is only SIGINT. If coming signal is belong to SIGINT, then switches sigintFlag from 0 to 1 and does nothing else.
- **terminateProgram** : This function called when sigintFlag is equal to 1. I made this check condition inside **parentProcess** while waiting for child processes to finish(with **wait()** function) and inside **createChildProcess** after creating child process. So there is no program termination or busy waiting inside signal handler. This function simply closes open files, frees all allocated variables and deletes the outputFile(by using **unlink** function). Also prints the current terminating process into the console.

childProcess.c

```

1  /* Define Values */
2  #define NUM_OF_POINTS 10
3  /* Global Variables */
4  static int sigintFlag = 0;
5
6  /* Main Functions */
7  void getCoordinatesFromEnvVar (char *argv[], int * x_coordinates, int * y_coordinates, int * z_coordinates);
8  void calculateCovarianceMatrix (float covarianceMatrix[3][3], int * x_coordinates, int * y_coordinates, int * z_coordinates);
9  void writeCovarianceMatrixToFile (int writeFD, float covarianceMatrix[3][3]);
10 /* Helper Functions */
11 float getMean (int numbers[NUM_OF_POINTS]);
12 float getVariance (int numbers[NUM_OF_POINTS], float mean);
13 float getCovariance (int firstNumbers[NUM_OF_POINTS], int secondNumbers[NUM_OF_POINTS], float meanOfFirst, float meanOfSecond);
14 /* Signal Handler */
15 void initializeSignalHandler ();
16 void handleSignal (int signal);
17 void terminateChild (char * buffer);

```

#define

- **NUM_OF_POINTS** : Same define rule is used inside parentProcess.c . Same usage.

Global Variables

- **sigintFlag** : Same define rule is used inside parentProcess.c . Same usage

Functions

- **getCoordinatesFromEnvVar** : Since we give coordinate information via environment variables, this function, which called inside main, gets these coordinate values from command line argument `argv` and stores them inside int coordinate arrays (`int x_coordinate[10]` for example)
- **calculateCovarianceMatrix** : After storing coordinate values as int values, this function calculates covariance matrix of these coordinates and stores covariance matrix information inside given 2d array parameter `covarianceMatrix[3][3]`
- **writeCovarianceMatrixToFile** : When calculation is done, covariance matrix is written to `outputFile` (file information is not passed to the child process, only `writeFD` value is shared with `argv`).
- **getMean** : An int array is given as parameter and mean value of that array is returned as the result. It is done for every axis(x, y and z)
- **getVariance** : Same int array and mean value is given as parameter and after calculating variance result has returned. This calculation is done for every axis(x, y, z)
- **getCovariance** : Calculates covariance value of given two axis(XY, XZ and YZ).
- **initializeSignalHandler** : Same function is used inside `parentProcess.c` and works same.
- **handleSignal** : Works same as `parentProcess.c`
- **terminateChild** : Inside `childProcess`, `terminateChild` function doesn't close files, it only frees allocated variables and exits with failure.

!! After explaining every function, now I will show an example execution of the file. To imitate SIGINT signal call I placed **sleep(1)** functions inside both `parentProcess.c` and `childProcess.c` file. This can be done with also `raise` function but I thought giving SIGINT signal with `ctrl+c` is more realistic. Before submitting the homework, I will comment these sleep function calls. Inside second execution(with SIGINT interrupt) it can be seen that only one child received SIGINT call. It is because other 2 child processes already finished their job and terminated even though I put sleep function inside them.

!! Parent process first sends SIGINT signals to child processes with `kill` function call. After child processes terminate, it starts to terminate itself. Signals other then SIGINT are blocked with **sigprocmask** function at the start of `parentProcess.c` and `childProcess.c` when they call **initializeSignalHandler**.

Example Execution - Without SIGNAL Interrupt

(inputFile.dat)

```
● ● ●

1 Morbi ut suscipit sapien. Suspendisse luctus vel neque eget consequat.
2 Phasellus porttitor augue vel malesuada egestas.
3 Quisque ut efficitur ante.
4 Fusce ex mi, imperdiet in suscipit et, rutrum sodales diam.
5 Integer dapibus viverra nulla, non malesuada nisl eleifend eget.
6 Etiam viverra fermentum cursus.
7 Nunc ut magna eget turpis tristique mattis in in velit.
```

```
tgknyhn@Tugkan:/mnt/c/Users/Tugkan/Desktop/hw2$ make withArgument
gcc -g -ggdb3 -Wall childProcess.c file.c -o child && gcc -g -ggdb3 -Wall parentProcess.c file.c -o parent -lm && ./parent -i "inputFile.dat" -o "outputFile.dat"

Process P reading inputFile.dat

Created R_0 with (77,111,114),(98,105,32),(117,116,32),(115,117,115),(99,105,112),(105,116,32),(115,97,112),(105,101,110),(46,32,83),(117,115,112)
Created R_1 with (101,110,100),(105,115,115),(101,32,108),(117,99,116),(117,115,32),(118,101,108),(32,110,101),(113,117,101),(32,101,103),(101,116,32)
Created R_2 with (99,111,110),(115,101,113),(117,97,116),(46,10,80),(104,97,115),(101,108,108),(117,115,32),(112,111,114),(116,116,105),(116,111,114)
Created R_3 with (32,97,117),(103,117,101),(32,118,101),(108,32,109),(97,108,101),(115,117,97),(100,97,32),(101,103,101),(115,116,97),(115,46,10)
Created R_4 with (81,117,105),(115,113,117),(101,32,117),(116,32,101),(102,102,105),(99,105,116),(117,114,32),(97,110,116),(101,46,10),(70,117,115)
Created R_5 with (99,101,32),(101,120,32),(109,105,44),(32,105,109),(112,101,114),(100,105,101),(116,32,105),(110,32,115),(117,115,99),(105,112,105)
Created R_6 with (116,32,101),(116,44,32),(114,117,116),(114,117,109),(32,115,111),(100,97,108),(101,115,32),(100,105,97),(109,46,10),(73,110,116)
Created R_7 with (101,103,101),(114,32,100),(97,112,105),(98,117,115),(32,118,105),(118,101,114),(114,97,32),(110,117,108),(108,97,44),(32,110,111)
Created R_8 with (110,32,109),(97,108,101),(115,117,97),(100,97,32),(110,105,115),(108,32,101),(108,101,105),(102,101,110),(100,32,101),(103,101,116)
Created R_9 with (46,10,69),(116,105,97),(109,32,118),(105,118,101),(114,114,97),(32,102,101),(114,109,101),(110,116,117),(109,32,99),(117,114,115)
Created R_10 with (117,115,46),(10,78,117),(110,99,32),(117,116,32),(109,97,103),(110,97,32),(101,103,101),(116,32,116),(117,114,112),(105,115,32)

Reached EOF, collecting outputs from outputFile.dat

The closest 2 matrices are
-----
933.360046      -225.079987      -368.979980      204.689987      -165.220016      -156.360001
-225.079987      854.890015      369.139984      and      -165.220016      1198.159912      275.579987
-368.979980      369.139984      1132.040039      -156.360001      275.579987      1365.439941
-----
and their distance is: 17.915

tgknyhn@Tugkan:/mnt/c/Users/Tugkan/Desktop/hw2$
```

Example Execution - With SIGINT Interrupt

```
tgknyhn@Tugkan:/mnt/c/Users/Tugkan/Desktop/hw2$ make withArgument
gcc -g -ggdb3 -Wall childProcess.c file.c -o child && gcc -g -ggdb3 -Wall parentProcess.c file.c -o parent -lm && ./parent -i "inputFile.dat" -o "outputFile.dat"

Process P reading inputFile.dat

Created R_0 with (77,111,114),(98,105,32),(117,116,32),(115,117,115),(99,105,112),(105,116,32),(115,97,112),(105,101,110),(46,32,83),(117,115,112)
Created R_1 with (101,110,100),(105,115,115),(101,32,108),(117,99,116),(117,115,32),(118,101,108),(32,110,101),(113,117,101),(32,101,103),(101,116,32)
Created R_2 with (99,111,110),(115,101,113),(117,97,116),(46,10,80),(104,97,115),(101,108,108),(117,115,32),(112,111,114),(116,116,105),(116,111,114)
^C
A child process detected SIGINT signal. Child process id : 142
Freeing allocated variables inside child process...

Parent process detected SIGINT signal.
Closing open files...
Freeing all variables...
Deleting the output file...
Terminating the program...
make: *** [Makefile:10: withArgument] Interrupt

tgknyhn@Tugkan:/mnt/c/Users/Tugkan/Desktop/hw2$
```



```
1  452.440002  423.500000  -20.460007
2  423.500000  580.849976  -51.800007
3  -20.460007  -51.800007  1298.239990
4
5  995.010071  -13.720004  -151.320007
6  -13.720004  579.639954  -227.059982
7  -151.320007  -227.059982  916.239929
8
9  420.809998  572.690002  89.090004
10 572.690002  897.409851  127.810013
11 89.090004  127.810013  625.010010
12
13 933.360046  -225.079987  -368.979980
14 -225.079987  854.890015  369.139984
15 -368.979980  369.139984  1132.040039
16
17 204.689987  -165.220016  -156.360001
18 -165.220016  1198.159912  275.579987
19 -156.360001  275.579987  1365.439941
20
21 552.089966  -170.980011  -69.559982
22 -170.980011  957.559937  -399.579987
23 -69.559982  -399.579987  1086.440063
24
25 627.650024  -352.399994  -320.000000
26 -352.399994  1079.760010  657.740051
27 -320.000000  657.740051  1531.359985
28
29 956.440063  -277.059998  -269.399994
30 -277.059998  579.640015  106.599998
31 -269.399994  106.599998  799.450012
32
33 29.409998  -0.780001  37.689995
34 -0.780001  1123.440063  -70.719994
35 37.689995  -70.719994  530.609985
36
37 868.559937  397.559967  215.500000
38 397.559967  1623.960083  231.000000
39 215.500000  231.000000  179.850006
40
41 951.559937  170.879990  -448.359955
42 170.879990  592.240051  -488.780029
43 -448.359955  -488.780029  1443.809814
```