Homework - 2 CSE344
System Programming
Hamza YOĞURTCUOĞLU - 171044086

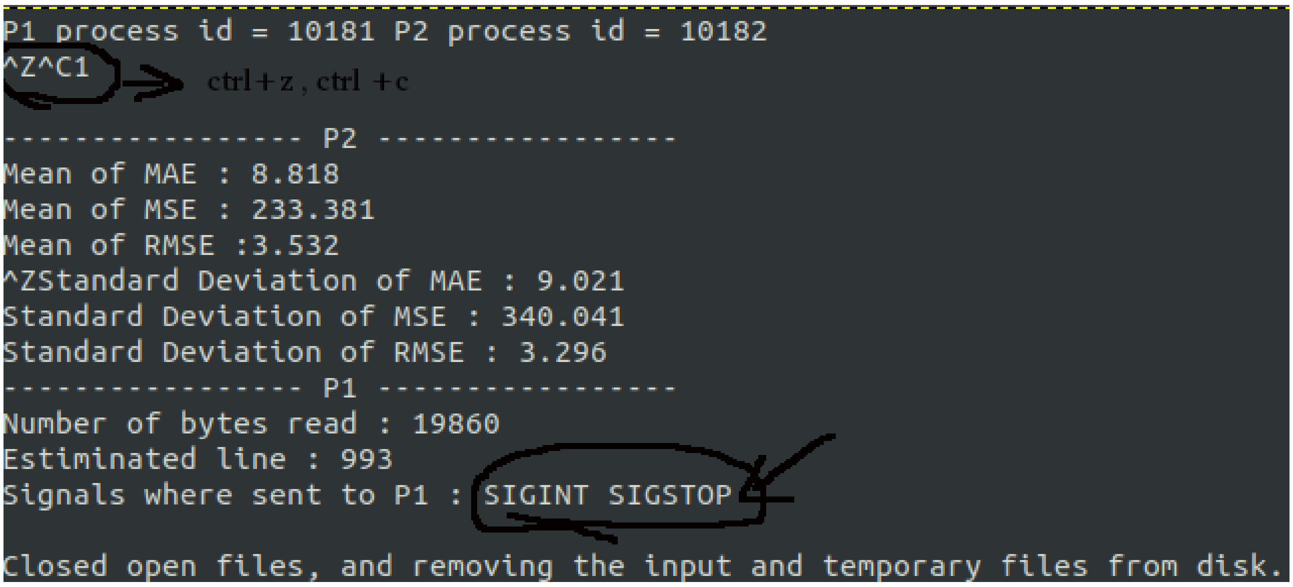
→ **main.c :** I put the **Usage** at first. I seperated command line argument with **getopt().**

Then **P1** process always received 20 bytes from **inputFile** in an while loop. Each set 2 bytes are converted to ASCII character that is converted 2D coordinate (x,y). For every 10 coordinates (i.e. every 20 bytes) it reads, it applied the **least squares method** to the corresponding 2D coordinates and calculate the line equation (ax+b) that fits them. 10 coordinates (x,y) and equation are wrote to temporary file created via **mkstemp.** But While writing to **tempfile** which is locked (flock).

→ **HOW I IGNORE SIGINT/SIGSTOP AND PRINT WHICH SIGNALS ARE COME IN CRITICAL SECTION**

When these operations are interrupted by **SIGINT/SIGSTOP.** In order to ignore these signals, **sigset_t** was created, these two signals to be blocked were added to this mask. However, signals such as **SIGSTOP and SIGKILL** cannot be blocked. That's why I ignore the **SIGTSTP** signal. Thus, while the program is in the critical section, it cannot be closed or stopped with **CTRL+Z** or **CTRL+C**. But it was necessary to print the signals coming in the critical region on the screen. I achieved this situation by the **sigpending** method.
→ **sigpending** returns the set of signals that are pending for delivery to the calling the signals which have been raised while blocked).

Sample output is as follows**[Figure 1]**. in the example **CTRL+Z and CTRL+C** are pressed while the program is running. After the program is over, the signal coming in the critical region is shown.

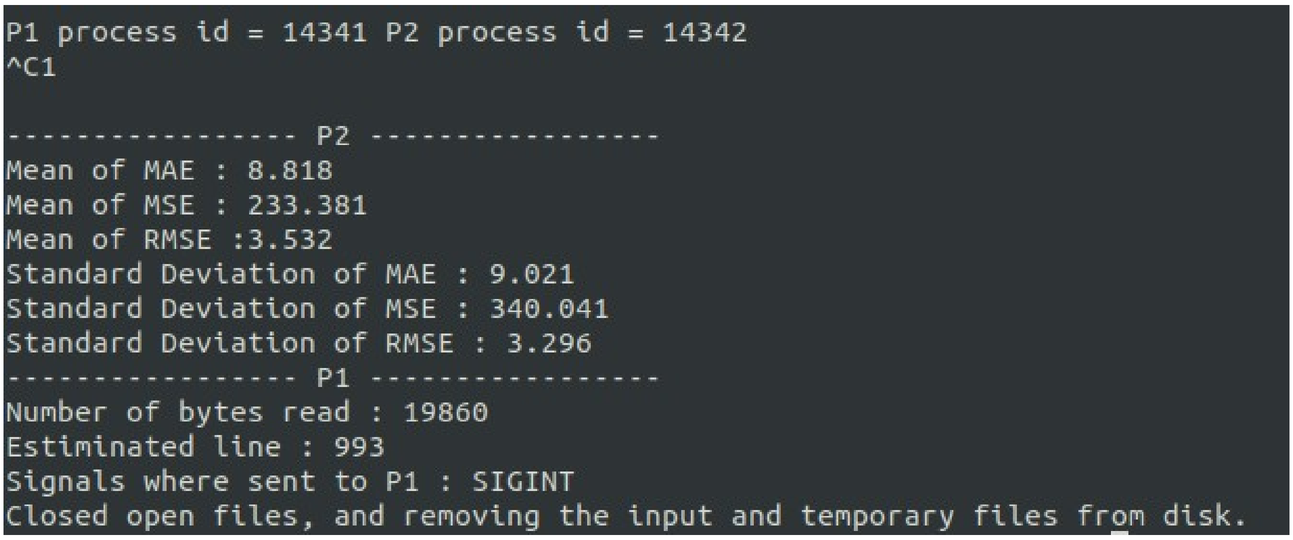


```
P1 process id = 10181 P2 process id = 10182
^Z^C1
                                              ctrl+z , ctrl +c

----------------- P2 -----------------
Mean of MAE : 8.818
Mean of MSE : 233.381
Mean of RMSE :3.532
^ZStandard Deviation of MAE : 9.021
Standard Deviation of MSE : 340.041
Standard Deviation of RMSE : 3.296
----------------- P1 -----------------
Number of bytes read : 19860
Estiminated line : 993
Signals where sent to P1 : SIGINT SIGSTOP

Closed open files, and removing the input and temporary files from disk.
```

Figure 1 : pressed CTRL+Z and CTRL+C when program runs



```
P1 process id = 14341 P2 process id = 14342
^C1

----------------- P2 -----------------
Mean of MAE : 8.818
Mean of MSE : 233.381
Mean of RMSE :3.532
Standard Deviation of MAE : 9.021
Standard Deviation of MSE : 340.041
Standard Deviation of RMSE : 3.296
----------------- P1 -----------------
Number of bytes read : 19860
Estiminated line : 993
Signals where sent to P1 : SIGINT
Closed open files, and removing the input and temporary files from disk.
```

Figure 2 : pressed just CTRL+C when program runs

P2 forked early by P1, and read the contents of the temporary file created by P1, line by line, and for every line it reads, it calculates the **mean absolute error (MAE), mean squared error (MSE) and root mean squared error (RMSE)** between the coordinates and the estimated line. This calculation be considered a critical section, and is not to be interrupted by **SIGINT/SIGSTOP** like critical section of P1. It removes the line it read from the file and write its own output to the file denoted by **outputPath** as

10 coordinates, the line equation (ax+b) and the three error estimates in a comma separated form.

MAE FORMULA : $(\sum_{i=1}^{n} |(yi-y)|) \div n$ **MSE FORMULA** : $(\sum_{i=1}^{n} (yi-y)^2) \div n$

RMSE FORMULA : $\sqrt{((\sum_{i=1}^{n} (yi-y)^2) \div n)}$

If P2 gets more chance to execute than P1, it might not find anything to read in the temporary file. I'm using **sigsuspend** to make sure P2 waits until there is some input available in the file, if P1 is not yet done with it.
If p1 write a line in tempfile. It sends **SIGUSR2** signal to P2. Thus, P2 is never busy waiting. If a signal(**SIGUSR2**) comes from P1, it continues.

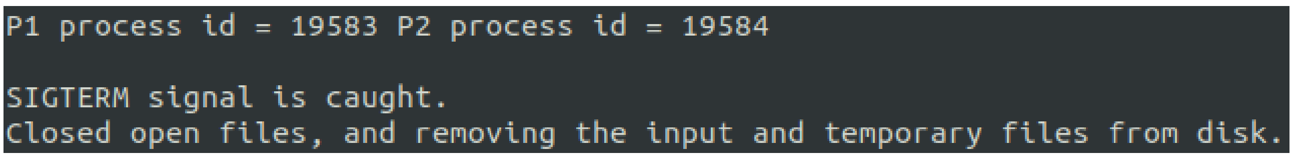
HOW P2 DETERMINES THAT P1 IS OVER

P1 sends **SIGUSR1** to P2 with this purpose. When P1 finishes reading the inputFile, it sends the **SIGUSR1** signal to P2. P2 changes the value of the variable of type **sig_atomic_t** to 0 by the signal handler. In this way, the P2 process runs without stopping until the tempfile ends and exits the loop.

SIGTERM SIGNAL STATUS

There is a global variable in process P1 and P2. When the SIGTERM signal is sent to P1 or P2, the signal sent process sends the SIGTERM signal to the other process. In this way, it exits the loop in two processes. Then, closing open files, and removing the input and temporary files from disk.

Example SIGTERM Result and Free Allocation :

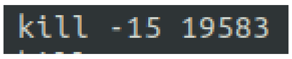


```
P1 process id = 19583 P2 process id = 19584

SIGTERM signal is caught.
Closed open files, and removing the input and temporary files from disk.
```

Figure 3 : Output of SIGTERM Result

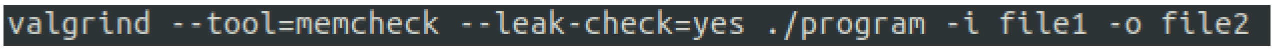
Send SIGTERM signal any two processes like following example :



```
kill -15 19583
```

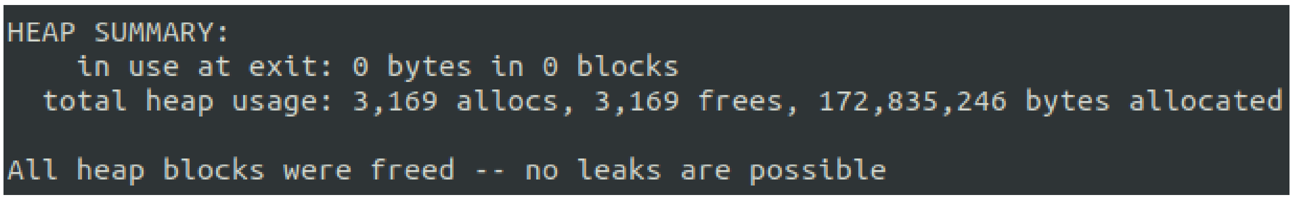
or 19584

When the results of Valgrind are examined, it is seen that all file and all received memories are free.



```
valgrind --tool=memcheck --leak-check=yes ./program -i file1 -o file2
```

We see that is no leak and all free memory



```
HEAP SUMMARY:
    in use at exit: 0 bytes in 0 blocks
  total heap usage: 3,169 allocs, 3,169 frees, 172,835,246 bytes allocated

All heap blocks were freed -- no leaks are possible
```

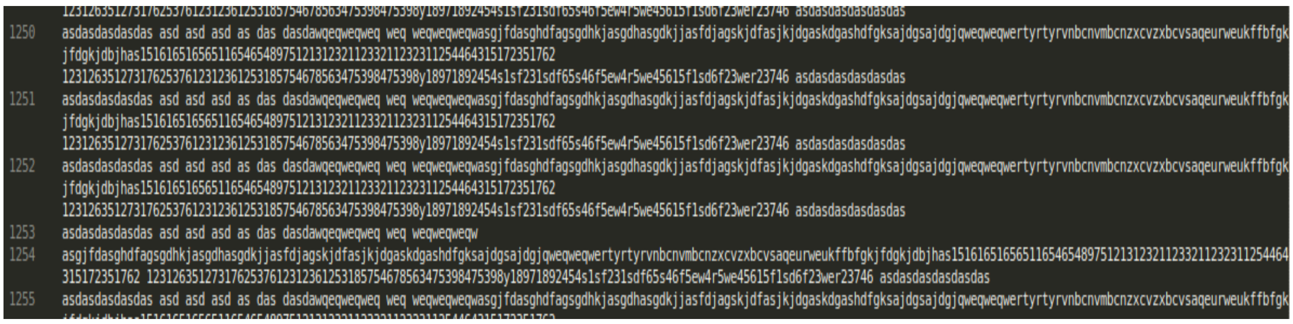
➝ **Test Program Examples:** The following commands write how you can run process. **Pay attention , inputFile is removed from P2! after working operation.**

You can execute with Absolute path or relative path for files.

./program -i inputFile -o outputFile
./program -i (AbsolutePath)/inputFile -o (AbsolutePath)/ outputFile

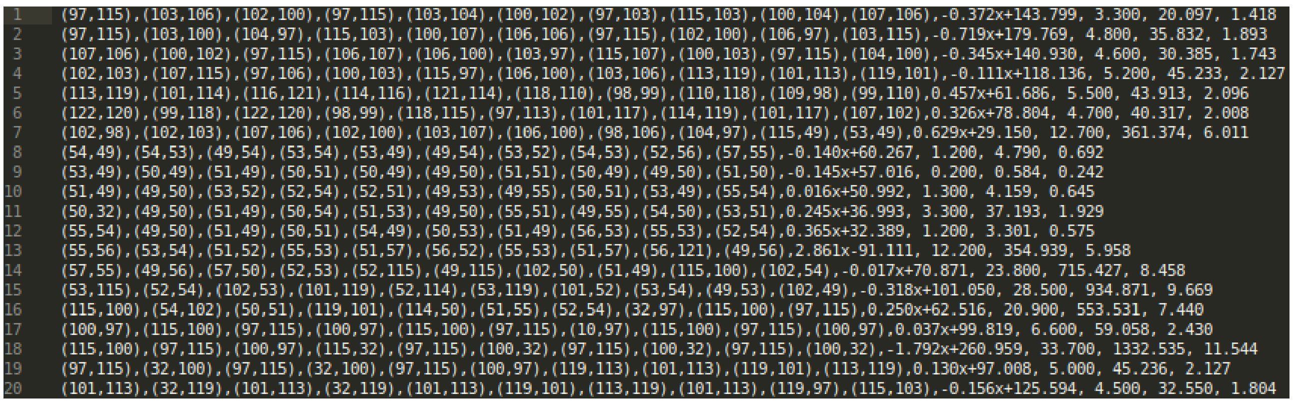
./program -o outputFile -i inputFile
./program -o (AbsolutePath)/inputFile -i (AbsolutePath)/ outputFile

➝ **inputFile Examples:** all character must be ASCII character like following example.



```
1231263512731762537612312361253185754678563475398475398y18971892454s1sf231sdf65s46f5ew4r5we45615f1sd6f23wer23746 asdasdasdasdasdas
asdasdasdasdas asd asd asd as das dasdawqeqweqweq weq weqweqweqwasgjfdasghdfagsgdhkjasgdhasgdkjjasfdjagskjdfasjkjdgaskdgashdfgksajdgsajdgjqweqweqwertyrtyrvnbcnvmbcnzxcvzxbcvsaqeurweukffbfgk
jfdgkjdbjhas1516165165651165465489751213123211233211232311254464315172351762
1231263512731762537612312361253185754678563475398475398y18971892454s1sf231sdf65s46f5ew4r5we45615f1sd6f23wer23746 asdasdasdasdasdas
asdasdasdasdas asd asd asd as das dasdawqeqweqweq weq weqweqweqwasgjfdasghdfagsgdhkjasgdhasgdkjjasfdjagskjdfasjkjdgaskdgashdfgksajdgsajdgjqweqweqwertyrtyrvnbcnvmbcnzxcvzxbcvsaqeurweukffbfgk
jfdgkjdbjhas1516165165651165465489751213123211233211232311254464315172351762
1231263512731762537612312361253185754678563475398475398y18971892454s1sf231sdf65s46f5ew4r5we45615f1sd6f23wer23746 asdasdasdasdasdas
asdasdasdasdas asd asd asd as das dasdawqeqweqweq weq weqweqweqwasgjfdasghdfagsgdhkjasgdhasgdkjjasfdjagskjdfasjkjdgaskdgashdfgksajdgsajdgjqweqweqwertyrtyrvnbcnvmbcnzxcvzxbcvsaqeurweukffbfgk
jfdgkjdbjhas1516165165651165465489751213123211233211232311254464315172351762
1231263512731762537612312361253185754678563475398475398y18971892454s1sf231sdf65s46f5ew4r5we45615f1sd6f23wer23746 asdasdasdasdasdas
asdasdasdasdas asd asd asd as das dasdawqeqweqweq weq weqweqweqw
asgjfdasghdfagsgdhkjasgdhasgdkjjasfdjagskjdfasjkjdgaskdgashdfgksajdgsajdgjqweqweqwertyrtyrvnbcnvmbcnzxcvzxbcvsaqeurweukffbfgkjfdgkjdbjhas1516165165651165465489751213123211233211232311254464
315172351762 1231263512731762537612312361253185754678563475398475398y18971892454s1sf231sdf65s46f5ew4r5we45615f1sd6f23wer23746 asdasdasdasdasdas
asdasdasdasdas asd asd asd as das dasdawqeqweqweq weq weqweqweqwasgjfdasghdfagsgdhkjasgdhasgdkjjasfdjagskjdfasjkjdgaskdgashdfgksajdgsajdgjqweqweqwertyrtyrvnbcnvmbcnzxcvzxbcvsaqeurweukffbfgk
```

→ outputFile Example :

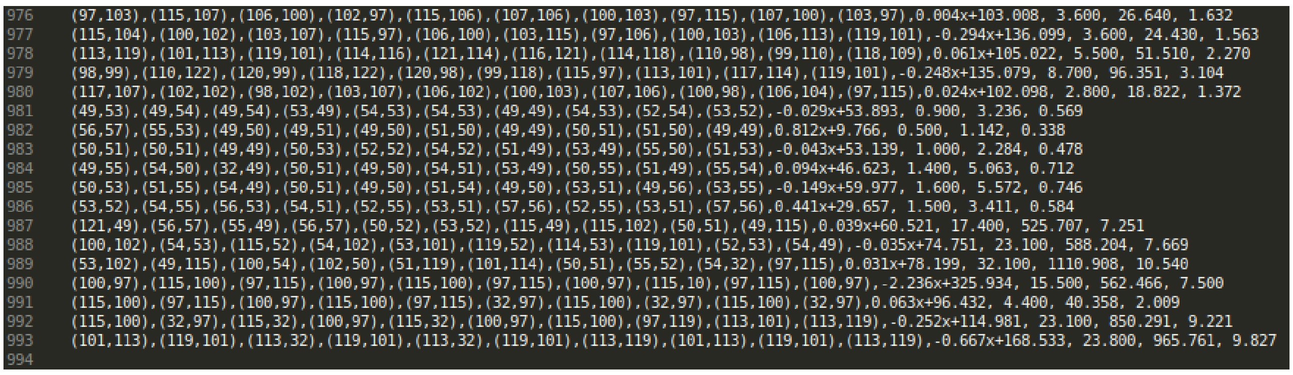


```
(97,115),(103,106),(102,100),(97,115),(103,104),(100,102),(97,103),(115,103),(100,104),(107,106),-0.372x+143.799, 3.300, 20.097, 1.418
(97,115),(103,100),(104,97),(115,103),(100,107),(106,106),(97,115),(102,100),(106,97),(103,115),-0.719x+179.769, 4.800, 35.832, 1.893
(107,106),(100,102),(97,115),(106,107),(106,100),(103,97),(115,107),(100,103),(97,115),(104,100),-0.345x+140.930, 4.600, 30.385, 1.743
(102,103),(107,115),(97,106),(100,103),(115,97),(106,100),(103,106),(113,119),(101,113),(119,101),-0.111x+118.136, 5.200, 45.233, 2.127
(113,119),(101,114),(116,121),(114,116),(121,114),(118,110),(98,99),(110,118),(109,98),(99,110),0.457x+61.686, 5.500, 43.913, 2.096
(122,120),(99,118),(122,120),(98,99),(118,115),(97,113),(101,117),(114,119),(101,117),(107,102),0.326x+78.804, 4.700, 40.317, 2.008
(102,98),(102,103),(107,106),(102,100),(103,107),(106,100),(98,106),(104,97),(115,49),(53,49),0.629x+29.150, 12.700, 361.374, 6.011
(54,49),(54,53),(49,54),(53,54),(53,49),(49,54),(53,52),(54,53),(52,56),(57,55),-0.140x+60.267, 1.200, 4.790, 0.692
(53,49),(50,49),(51,49),(50,51),(50,49),(49,50),(51,51),(50,49),(49,50),(51,50),-0.145x+57.016, 0.200, 0.584, 0.242
(51,49),(49,50),(53,52),(52,54),(52,51),(49,53),(49,55),(50,51),(53,49),(55,54),0.016x+50.992, 1.300, 4.159, 0.645
(50,32),(49,50),(51,49),(50,54),(51,53),(49,50),(55,51),(49,55),(54,50),(53,51),0.245x+36.993, 3.300, 37.193, 1.929
(55,54),(49,50),(51,49),(50,51),(54,49),(50,53),(51,49),(56,53),(55,53),(52,54),0.365x+32.389, 1.200, 3.301, 0.575
(55,56),(53,54),(51,52),(55,53),(51,57),(56,52),(55,53),(51,57),(56,121),(49,56),2.861x-91.111, 12.200, 354.939, 5.958
(57,55),(49,56),(57,50),(52,53),(52,115),(49,115),(102,50),(51,49),(115,100),(102,54),-0.017x+70.871, 23.800, 715.427, 8.458
(53,115),(52,54),(102,53),(101,119),(52,114),(53,119),(101,52),(53,54),(49,53),(102,49),-0.318x+101.050, 28.500, 934.871, 9.669
(115,100),(54,102),(50,51),(119,101),(114,50),(51,55),(52,54),(32,97),(115,100),(97,115),0.250x+62.516, 20.900, 553.531, 7.440
(100,97),(115,100),(97,115),(100,97),(115,100),(97,115),(10,97),(115,100),(97,115),(100,97),0.037x+99.819, 6.600, 59.058, 2.430
(115,100),(97,115),(100,97),(115,32),(97,115),(100,32),(97,115),(100,32),(97,115),(100,32),-1.792x+260.959, 33.700, 1332.535, 11.544
(97,115),(32,100),(97,115),(32,100),(97,115),(100,97),(119,113),(101,113),(119,101),(113,119),0.130x+97.008, 5.000, 45.236, 2.127
(101,113),(32,119),(101,113),(32,119),(101,113),(119,101),(113,119),(101,113),(119,97),(115,103),-0.156x+125.594, 4.500, 32.550, 1.804
```

....

....

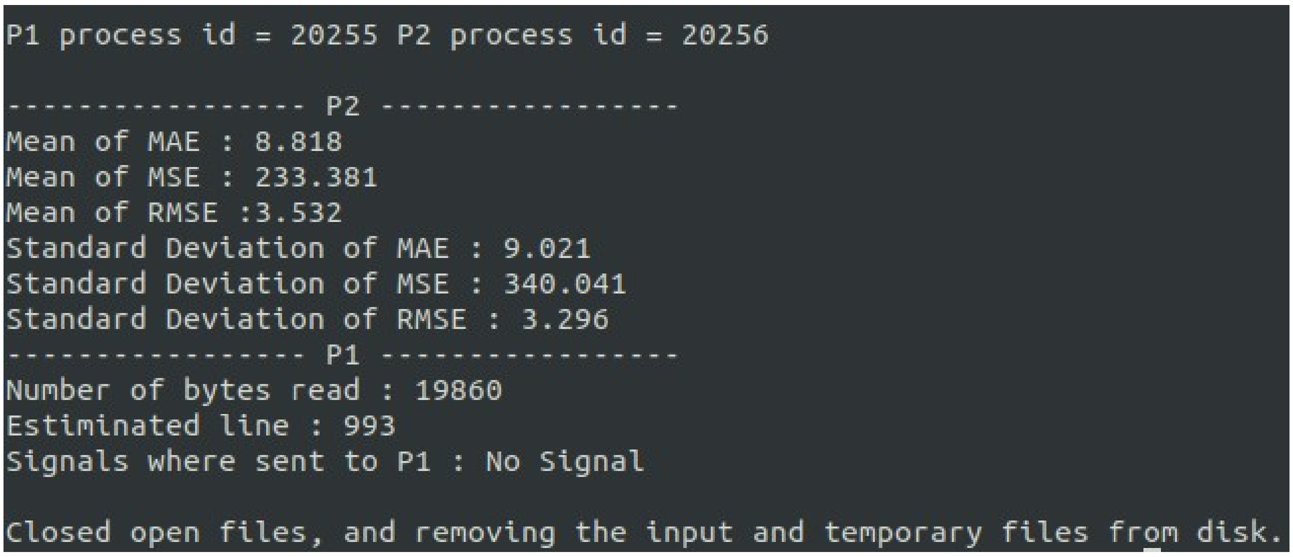
....



```
(97,103),(115,107),(106,100),(102,97),(115,106),(107,106),(100,103),(97,115),(107,100),(103,97),0.004x+103.008, 3.600, 26.640, 1.632
(115,104),(100,102),(103,107),(115,97),(106,100),(103,115),(97,106),(100,103),(106,113),(119,101),-0.294x+136.099, 3.600, 24.430, 1.563
(113,119),(101,113),(119,101),(114,116),(121,114),(116,121),(114,118),(110,98),(99,110),(118,109),0.061x+105.022, 5.500, 51.510, 2.270
(98,99),(110,122),(120,99),(118,122),(120,98),(99,118),(115,97),(113,101),(117,114),(119,101),-0.248x+135.079, 8.700, 96.351, 3.104
(117,107),(102,102),(98,102),(103,107),(106,102),(100,103),(107,106),(100,98),(106,104),(97,115),0.024x+102.098, 2.800, 18.822, 1.372
(49,53),(49,54),(49,54),(53,49),(54,53),(54,53),(49,49),(54,53),(52,54),(53,52),-0.029x+53.893, 0.900, 3.236, 0.569
(56,57),(55,53),(49,50),(49,51),(49,50),(51,50),(49,49),(50,51),(51,50),(49,49),0.812x+9.766, 0.500, 1.142, 0.338
(50,51),(50,51),(49,49),(50,53),(52,52),(54,52),(51,49),(53,49),(55,50),(51,53),-0.043x+53.139, 1.000, 2.284, 0.478
(49,55),(54,50),(32,49),(50,51),(49,50),(54,51),(53,49),(50,55),(51,49),(55,54),0.094x+46.623, 1.400, 5.063, 0.712
(50,53),(51,55),(54,49),(50,51),(49,50),(51,54),(49,50),(53,51),(49,56),(53,55),-0.149x+59.977, 1.600, 5.572, 0.746
(53,52),(54,55),(56,53),(54,51),(52,55),(53,51),(57,56),(52,55),(53,51),(57,56),0.441x+29.657, 1.500, 3.411, 0.584
(121,49),(56,57),(55,49),(56,57),(50,52),(53,52),(115,49),(115,102),(50,51),(49,115),0.039x+60.521, 17.400, 525.707, 7.251
(100,102),(54,53),(115,52),(54,102),(53,101),(119,52),(114,53),(119,101),(52,53),(54,49),-0.035x+74.751, 23.100, 588.204, 7.669
(53,102),(49,115),(100,54),(102,50),(51,119),(101,114),(50,51),(55,52),(54,32),(97,115),0.031x+78.199, 32.100, 1110.908, 10.540
(100,97),(115,100),(97,115),(100,97),(115,100),(97,115),(100,97),(115,10),(97,115),(100,97),-2.236x+325.934, 15.500, 562.466, 7.500
(115,100),(97,115),(100,97),(115,100),(97,115),(32,97),(115,100),(32,97),(115,100),(32,97),0.063x+96.432, 4.400, 40.358, 2.009
(115,100),(32,97),(115,32),(100,97),(115,32),(100,97),(115,100),(97,119),(113,101),(113,119),-0.252x+114.981, 23.100, 850.291, 9.221
(101,113),(119,101),(113,32),(119,101),(113,32),(119,101),(113,119),(101,113),(119,101),(113,119),-0.667x+168.533, 23.800, 965.761, 9.827

```

There is 19860 character in inputFile



```
P1 process id = 20255 P2 process id = 20256

----------------- P2 -----------------
Mean of MAE : 8.818
Mean of MSE : 233.381
Mean of RMSE :3.532
Standard Deviation of MAE : 9.021
Standard Deviation of MSE : 340.041
Standard Deviation of RMSE : 3.296
----------------- P1 -----------------
Number of bytes read : 19860
Estiminated line : 993
Signals where sent to P1 : No Signal

Closed open files, and removing the input and temporary files from disk.
```

Finally, I left example of inputFile.

inputFile has 19860 character = 20 * 993 (Estiminated Line)

→ Compiler and Run :

Makefiles Commands:

```
make  #that command compiler
make clean #that command cleans all object files
```

To run :

```
./program -i inputPath -o outputPath
```

If you want to see outputs. Check above **outputs.**

→ Note : 1) Don't forget that process P2 is removing temp and inputfiles when run the code.
2) If you want to try ctrl + z and ctrl + c, I recommend you to keep plenty of characters inside the input file.

Thanks :)