CSE344 – System Programming
REPORT - HW 3

Furkan ÖZEV
161044036

PROCESS:

The program to is developed involves 5 processes: a parent P1 and its 4 children P2, P3, P4 and P5.
Parent process will communicate with the child process and send and receive data.
To achieve this, I created bi-directional pipes between P1 and each of its children.
A bidirectional pipe between P1 and P2, a bi-directional pipe between P1 and P3, and so on.

MAIN IDEA:

The main idea is for Parent process to receive as input 2 square matrices, distribute the calculation of their product to its children process via pipes, collect from them the partial outputs, combine them, and then calculate the singular values of the product matrix.

P1 will create 4 children processes and each of them will be responsible for calculating one quarter of C.
P2 will calculate C11, P3 will calculate C12, P4 will calculate C21 and P5 will calculate C22.

In order to calculate Cij a process will need access to the i-th quarter row of A and j-th quarter column of B.

If P2 is to calculate C11 of C, it will take A11, A12, B11 and B21 quarters from P1.
If P3 is to calculate C12 of C, it will take A11, A12, B12 and B22 quarters from P1.
If P4 is to calculate C21 of C, it will take A21, A22, B11 and B21 quarters from P1.
If P5 is to calculate C22 of C, it will take A21, A22, B12 and B22 quarters from P1.

P1 will then send to each child process through the pipe the quarters of A and B that the child requires in order to calculate one quarter of C.

Once all children have completed their calculations, P1 will collect their outputs through the pipes and form the product matrix C.

P1 will finallly calculate all the singular values of C, print them to the terminal and exit.

PARENT PROCESS:

Parent Process (P1) receive as command-line arguments the paths of 2 regular files and a positive integer n.

./program -i inputPathA -j inputPathB -n NUMBER

Firstly I checked arguments amount, It must be 7.
I used the getopt() library method for parsing commandline arguments.
If the command line arguments are missing/invalid your program must print usage information and exit.
Then I checked NUMBER, it must be positive.

SIGHLD:

(NOTE: The section for handling the SIGCHLD signal is taken from page 557-558 of the textbook.)

I initialize numLiveChild with 4. It show number of live child process. It used for handling SIGCHLD signal.

For handle SIGCHLD, I changed SIGCHDL action because default action of SIGCHLD is immediately termination but we want to parent process wait all child terminate.
I blocked SIGCHLD to prevent its delivery if a child terminates before the parent commences the sigsuspend().

PIPE:

I used pipes to establish communication between parent and a child process.
I used 1 pipe for communication from a parent to a child, and another pipe for communication from a child to the parent. So I made a bidirectional pipe.
Since there will be 4 children in total, I made 4 pipes for both directions with pipe() function.

FORK:

I made the fork operation in a loop that turns 4 times. I kept the child process ids returned by the fork function in a global array. This array will be used for the SIGINT signal.

If return value of fork is -1, it will print error message and exit.
If return value of fork is 0, then we know it is child process. So, child process does its job.
Otherwise, we know it is parent process, and close unused ends for parent process.

AFTER FORK:

It will open input files, and will read (2 ^ n) x (2 ^ n) characters from each file.
These characters will be converted into its ASCII code integer equivalent in 2D arrays with using my convert() function.

To calculate the resulting quarters, I divided the matrices into 2 parts.
First part of Matrix A includes A11 and A12 quarters.
Second part of Matrix A includes A21 and A22 quarters.
First part of Matrix B includes B11 and B21 quarters.
Second part of Matrix B includes B12 and B22 quarters.

Then I send these parts to child processes with using my writer() function.
A11, A12, B11 and B21 quarters will be sent to Process 2 as input for calculate C11.
A11, A12, B12 and B22 quarters will be sent to Process 3 as input for calculate C12.
A21, A22, B11 and B21 quarters will be sent to Process 4 as input for calculate C21.
A21, A22, B12 and B22 quarters will be sent to Process 5 as input for calculate C22.

Then quarters of the resulting matrix calculated in Child Processes is read from Child Processes and written to the relevant quarters of matrixResult. For read operation I create reader() function.

SINGULAR VALUES:

P1 distributed the calculation of their product to its children, collected from them the partial outputs, combined them.
Now it need to calculate the singular values of the product matrix .
The matrix has singular values as much as the number of rows or columns.
So I created this amount of dynamic float arrays.
Then, call dsvd function with result matrix, amount of row, and this float array.

For singular value calculation, I copied from these external sources for internet.
It is stated in the homework pdf that this is allowed.

Then, print Matrix A, Matrix B, Result Matrix and all singular values of result matrix.

Then, all of the memory allocated for dynamic arrays are released with using free() function, and close the input files.

If parent comes end before child, It will wait for SIGCHLD until all children are dead.

After parent process will exit.

CHILD PROCESSES:

All child process will do same operations, just their input context different.
Parent sends this input content via pipe.

Firstly, It closed unused ends for child processes.
When child received a message (quarters) from parent, It reads bytes and convert into 2 integer matrix.
Then, multiply these 2 matrixs.
Then, It will convert the integer matrix resulting from the product into a string of characters to write a pipe.

Then, child sends a response to parent with using my writer2() function.

Then It will free memory, and exit successfull.

SIGNAL HANDLER OR CATCHER:

Firstly, It keep the errno

In case of CTRL-C, all 5 processes must exit gracefully.
When it caught the SIGINT signal, it will send kill signal to child processes, then terminate.

In case of SIGCHLD,
When it caught the SIGCHLD signal, it will perform a synchronous wait for each of its children.
It wait by using waitpid, then decrease number of live child process variable.
Then, it checks for error.

It will restore errno

HELPER FUNCTIONS:

READER FUNCTION:

It reads an quarter from child process and write this quarter relevant part.

WRITER FUNCTION:

It writes character array of 2 matrixs in child process's input pipe.

WRITER2 FUNCTION:

It writes character array of input matrix in relevant pipe.

CONVERT FUNCTION:

It converts characters to its ASCII code integer equivalent.

MULTIPLY FUNCTION:

It multiplys 2 matrixs.

PRINT FUNCTION:

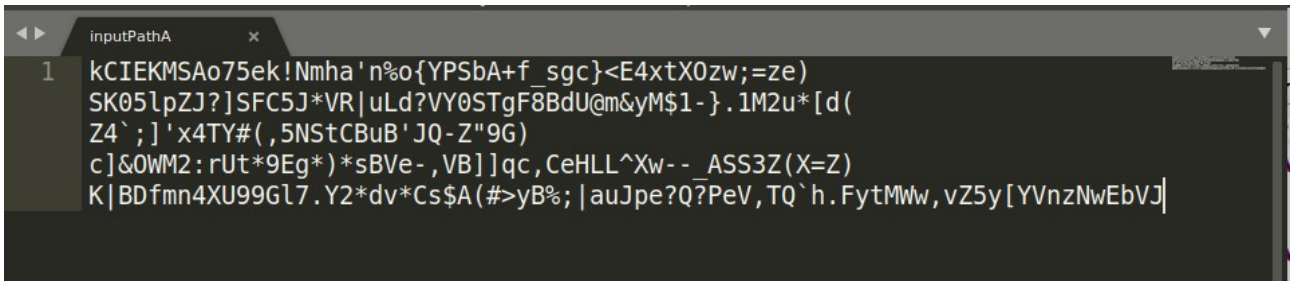
It prints matrixs.

DSVD FUNCTION:

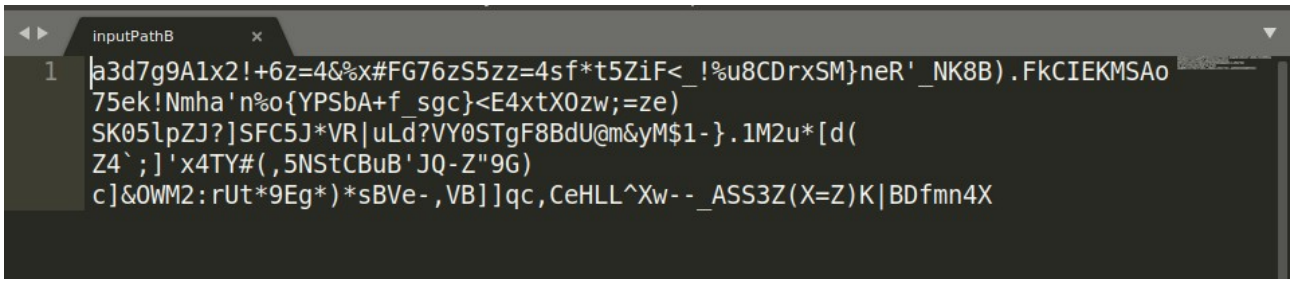
It calculate singular values of result matrix.
For singular value calculation, I copied from these external sources for internet.
It is stated in the homework pdf that this is allowed.

RUN EXAMPLE :

INPUT FILE – A: (256 characters)



INPUT FILE – B: (256 characters)



COMPILE & RUN:

```
furkan@furkan:~/Desktop/hw3$ make
gcc -o program program.c svd.c -lm
furkan@furkan:~/Desktop/hw3$ ./program -i inputPathA -j inputPathB -n 4

-------------------- Matrix - A --------------------

107     67      73      69      75      77      83      65      111     55      53      101     107     33      78      109
104     97      39      110     37      111     123     89      80      83      98      65      43      102     95      115
103     99      125     60      69      52      120     116     88      79      122     119     59      61      122     101
41      83      75      48      53      108     112     90      74      63      93      83      70      67      53      74
42      86      82      124     117     76      100     63      86      89      48      83      84      103     70      56
66      100     85      64      109     38      121     77      36      49      45      125     46      49      77      50
117     42      91      100     40      90      52      96      59      93      39      120     52      84      89      35
40      44      53      78      83      116     67      66      117     66      39      74      81      45      90      34
57      71      41      99      93      38      79      87      77      50      58      114     85      116     42      57
69      103     42      41      42      115     66      86      101     45      44      86      66      93      93      113
99      44      67      101     72      76      76      94      88      119     45      45      95      65      83      83
51      90      40      88      61      90      41      75      124     66      68      102     109     110     52      88
85      57      57      71      108     55      46      89      50      42      100     118     42      67      115     36
65      40      35      62      121     66      37      59      124     97      117     74      112     101     63      81
63      80      101     86      44      84      81      96      104     46      70      121     116     77      87      119
44      118     90      53      121     91      89      86      110     122     78      119     69      98      86      74

-------------------- Matrix - B --------------------

97      51      100     55      103     57      65      49      120     50      33      43      54      122     61      52
38      37      120     35      70      71      55      54      122     83      53      122     122     61      52      115
102     42      116     53      90      105     70      60      95      33      37      117     56      67      68      114
120     83      77      125     110     101     82      39      95      78      75      56      66      41      46      70
107     67      73      69      75      77      83      65      111     55      53      101     107     33      78      109
104     97      39      110     37      111     123     89      80      83      98      65      43      102     95      115
103     99      125     60      69      52      120     116     88      79      122     119     59      61      122     101
41      83      75      48      53      108     112     90      74      63      93      83      70      67      53      74
42      86      82      124     117     76      100     63      86      89      48      83      84      103     70      56
66      100     85      64      109     38      121     77      36      49      45      125     46      49      77      50
117     42      91      100     40      90      52      96      59      93      39      120     52      84      89      35
40      44      53      78      83      116     67      66      117     66      39      74      81      45      90      34
57      71      41      99      93      38      79      87      77      50      58      114     85      116     42      57
69      103     42      41      42      115     66      86      101     45      44      86      66      93      93      113
99      44      67      101     72      76      76      94      88      119     45      45      95      65      83      83
51      90      40      88      61      90      41      75      124     66      68      102     109     110     52      88
```

```
-------------------- Multiplication --------------------

96088   88789   95561   102347  100604  101759  102650  93774   119525  87350   75039   112618  96732   100274  90402   96793
110436  102716  107104  109603  103058  115726  114855  106723  128354  99916   87397   123269  101959  108500  103974  110768
116355  101079  120600  113506  113574  124031  120101  114321  138473  104853  87594   137138  112068  112208  111141  115860
91771   85966   91767   92236   86320   99635   100053  93315   107597  83061   75230   111517  87077   90877   90149   95599
103995  96057   100394  102794  102358  108346  109959  97400   120749  89046   79590   120183  98862   93901   96643   107290
88974   76894   92667   82882   87157   94353   92867   85478   109367  77964   69099   104622  88226   78396   85933   92513
94494   84871   90801   92805   94654   102446  100939  87924   110052  80601   69481   101257  84292   89926   87976   91676
86029   80254   80618   92324   85750   90373   95990   82997   98201   78316   67575   95770   81083   82455   81314   86577
87690   84467   86605   89550   89383   98565   94519   86756   109762  78260   69422   105346  89008   85715   85094   90830
87538   87483   87572   94583   88377   102279  98283   92108   114400  86042   73643   106337  93411   97054   88325   98228
98484   93296   94816   99856   100261  99430   106859  93785   112172  84733   75769   111708  91316   96744   89237   97072
91231   91901   89494   101897  96299   105852  101921  93257   116366  86715   73601   113348  96155   98961   89538   97207
91938   75318   86578   88880   85129   97838   90833   85391   106439  80918   64062   98127   86018   81976   84935   86862
96856   91431   89868   103017  96700   101594  101970  95028   112358  85876   70141   115537  94496   98265   91386   93751
102537  97149   101851  110290  105294  116427  110523  103810  128929  95148   82514   124248  104784  107905  98593   107445
108937  103281  111500  110349  110267  119931  120824  109754  132500  99609   84867   135909  110557  105298  108621  116528

---------- All Singular Values ----------

Singular Value - 0 :    1570820.000
Singular Value - 1 :    24811.854
Singular Value - 2 :    18006.904
Singular Value - 3 :    15354.021
Singular Value - 4 :    14055.641
Singular Value - 5 :    11779.852
Singular Value - 6 :    10052.306
Singular Value - 7 :    7475.138
Singular Value - 8 :    5828.765
Singular Value - 9 :    4350.116
Singular Value - 10 :   2995.213
Singular Value - 11 :   2468.767
Singular Value - 12 :   1876.744
Singular Value - 13 :   709.666
Singular Value - 14 :   5.142
Singular Value - 15 :   250.688

Child Process (pid = 12958) terminated
Child Process (pid = 12959) terminated
Child Process (pid = 12960) terminated
Child Process (pid = 12961) terminated
Parent Process (pid = 12957) terminated

furkan@furkan:~/Desktop/hw3$
```

VALGRIND RESULT :

Valgrind is a programming tool for memory debugging, memory leak detection, and profiling.

```
==13606==
==13606== HEAP SUMMARY:
==13606==     in use at exit: 0 bytes in 0 blocks
==13606==   total heap usage: 77 allocs, 77 frees, 10,496 bytes allocated
==13606==
==13606== All heap blocks were freed -- no leaks are possible
==13606==
==13606== For counts of detected and suppressed errors, rerun with: -v
==13606== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

NOTICED:

1- I used the getopt() library method for parsing command line arguments.

2- If the command line arguments are missing/invalid my program will print usage information and exit

3- I used system calls for all file I/O purposes, I didn’t use standard C library functions for file I/O.

4- In case of an error, I exit program by printing to stderr a nicely formatted informative errno based message.

5- I free all resources explicitly and I closed all files.

6- I prevent the zombie process with the waitpid and suspend mechanisms.

7- Compilation is warning-free

8- For singular value calculation, I copied from these external sources for internet. It is stated in the homework pdf that this is allowed.

9- I tested the signal scenarios in the homework myself.

10- The section for handling the SIGCHLD signal is taken from page 557-558 of the textbook.

11- I used bi-directional pipes between P1 and each of its children.

13- If you want to test the signals, you can test it by putting sleep () where necessary.