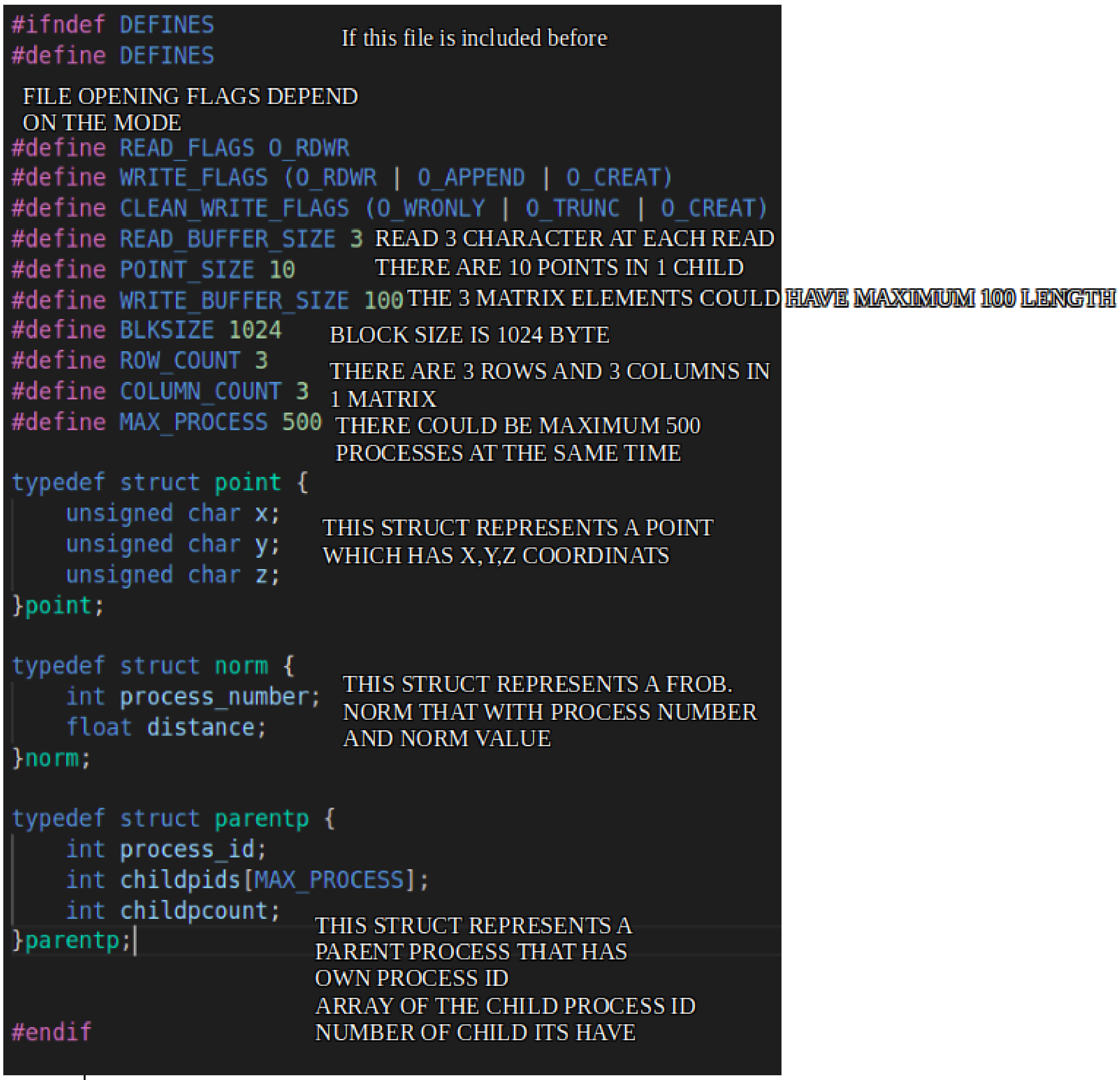
GEBZE TECHNICAL UNIVERSITY

SYSTEMS PROGRAMMING COURSE HW1 REPORT

Yakup Talha Yolcu
1801042609

1. DEFINE's

Firstly I have defines.h file to hold #defines and structs



```c
#ifndef DEFINES
#define DEFINES

#define READ_FLAGS O_RDWR
#define WRITE_FLAGS (O_RDWR | O_APPEND | O_CREAT)
#define CLEAN_WRITE_FLAGS (O_WRONLY | O_TRUNC | O_CREAT)
#define READ_BUFFER_SIZE 3
#define POINT_SIZE 10
#define WRITE_BUFFER_SIZE 100
#define BLKSIZE 1024
#define ROW_COUNT 3
#define COLUMN_COUNT 3
#define MAX_PROCESS 500

typedef struct point {
    unsigned char x;
    unsigned char y;
    unsigned char z;
}point;

typedef struct norm {
    int process_number;
    float distance;
}norm;

typedef struct parentp {
    int process_id;
    int childpids[MAX_PROCESS];
    int childpcount;
}parentp;


#endif
```

If this file is included before

FILE OPENING FLAGS DEPEND ON THE MODE

READ 3 CHARACTER AT EACH READ

THERE ARE 10 POINTS IN 1 CHILD

THE 3 MATRIX ELEMENTS COULD HAVE MAXIMUM 100 LENGTH

BLOCK SIZE IS 1024 BYTE

THERE ARE 3 ROWS AND 3 COLUMNS IN 1 MATRIX

THERE COULD BE MAXIMUM 500 PROCESSES AT THE SAME TIME

THIS STRUCT REPRESENTS A POINT WHICH HAS X,Y,Z COORDINATS

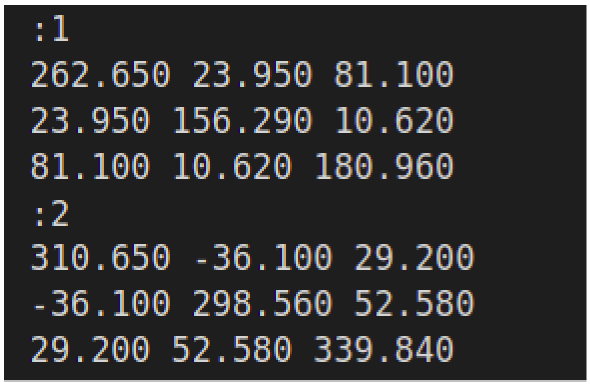
THIS STRUCT REPRESENTS A FROB. NORM THAT WITH PROCESS NUMBER AND NORM VALUE

THIS STRUCT REPRESENTS A PARENT PROCESS THAT HAS
OWN PROCESS ID
ARRAY OF THE CHILD PROCESS ID
NUMBER OF CHILD ITS HAVE

2. FUNCTIONS

These functions are belong to the parent process

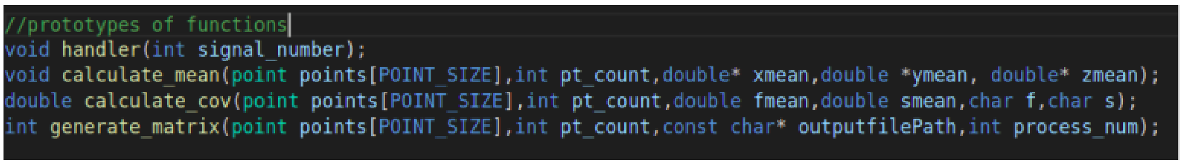
1. **handler** -> signal handler for SIGINT and other signals
2. **print_usage_and_exit** -> If input is invalid, then I call this function
3. **start_reading-file** -> This function reads the input file. While reading, new processes (childs) is created if they need to be. Fork and execve is in this function. . After reading is done, child processes should be finish their job, so parent waits for childrens. Return value is int and in case of error, it returns -1, otherwise it returns number of created processes.
4. **Print points** -> this function takes point array and number of point it has. Then prints x y and z coordinats of that points.
5. **Start reading output file** -> This function is executing after child processes are finished their job. Parent process takes output file and depend on the process count it reads the output file. I read the file byte by byte. My sample output file is like that:



```
:1
262.650 23.950 81.100
23.950 156.290 10.620
81.100 10.620 180.960
:2
310.650 -36.100 29.200
-36.100 298.560 52.580
29.200 52.580 339.840
```

   1 and 2 represents the process order. And matrix is written then. I read the file byte by byte because I don't know exactly how many characters in any line. Firstly, I read the first line until \n character. Then I seperated the buffer into : and %d Then I read until seeing a 3rd \n character. So I got 9 float numbers with spaces and 2 \n in them. So I seperated with them using sscanf function. This function returns 0 if it is success, otherwise return -1

6. **Calculate frobenius norm** -> This function takes a matrix and calculates its frobenius norm by given formula
7. **Calculate difference** -> This function takes matrices and matrix count. In 2 for loop, calculates the minimum difference between 2 matrix with respect to their frobenius norm.It returns the mininmum distance.
8. **Read process number** -> This function takes file descriptor pointer to don't lose file descriptor, and It reads the process number as I mentioned above. It reads : and integer value. It returns the integer value. In case of error it returns -1



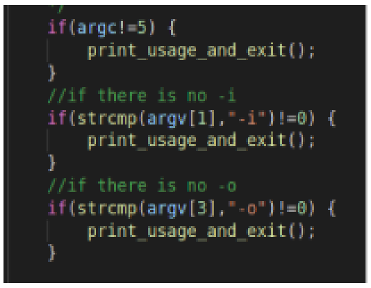
```
//prototypes of functions
void handler(int signal_number);
void calculate_mean(point points[POINT_SIZE],int pt_count,double* xmean,double *ymean, double* zmean);
double calculate_cov(point points[POINT_SIZE],int pt_count,double fmean,double smean,char f,char s);
int generate_matrix(point points[POINT_SIZE],int pt_count,const char* outputfilePath,int process_num);
```

These functions are belong to the child process

1. **handler** -> Works same with parent
2. **calculate mean** -> This function calculate point array and how many points it has. Because of we can't return the 3 three mean value, I just want to take these parameters by call by reference. Then I calculate x,y and z and assigned their values.
3. **Calculate_cov** -> This function finds one covarience with respect to given parameters. Fmeans represents the mean of the first cov element. Smean is respectively. F and S represents the char as 'x','y' or 'z' if we need to calculate covxx then we sent the fmean=xmean, smean=xmean, f='x' and s='x'. It returns the calculated covarience value
4. **generate matrix** -> It takes points array and size, it also takes the outputfile path to write to the file and it also takes current process Id. This function writes results to the output file. IN case of error, it return -1. Otherwise it return 0

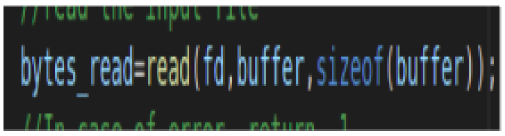
3. DESIGN DECISIONS

3. DESIGN DECISIONS



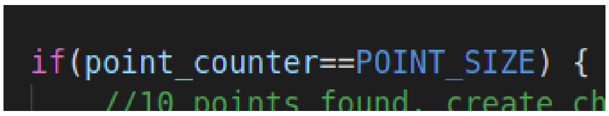
```
if(argc!=5) {
    print_usage_and_exit();
}
//if there is no -i
if(strcmp(argv[1],"-i")!=0) {
    print_usage_and_exit();
}
//if there is no -o
if(strcmp(argv[3],"-o")!=0) {
    print_usage_and_exit();
}
```

My input validity check controls the argument count. Then is there any -i in the command , then is there any -o in the command.



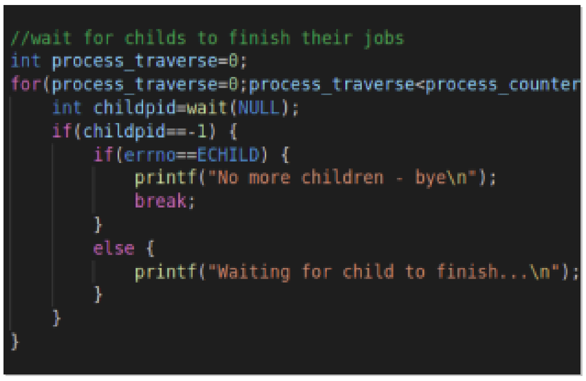
```
//read the input file
bytes_read=read(fd,buffer,sizeof(buffer));
//In case of error, return -1
```

I read three bytes to get x y and z coordinats. Size of buffer is 3



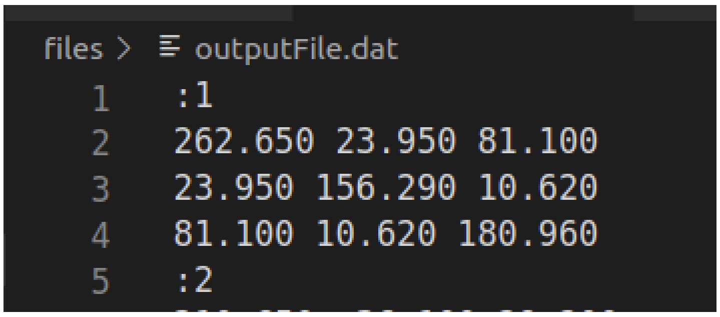
```
if(point_counter==POINT_SIZE) {
    //10 points found, create ch
```

If I reached 10 points, then I use fork and created a child process. Then I set the command line arguments and environment variables. Then I called execve function.



```c
//wait for childs to finish their jobs
int process_traverse=0;
for(process_traverse=0;process_traverse<process_counter
    int childpid=wait(NULL);
    if(childpid==-1) {
        if(errno==ECHILD) {
            printf("No more children - bye\n");
            break;
        }
        else {
            printf("Waiting for child to finish...\n");
        }
    }
}
```

In here, I wait all childs to finish their jobs.



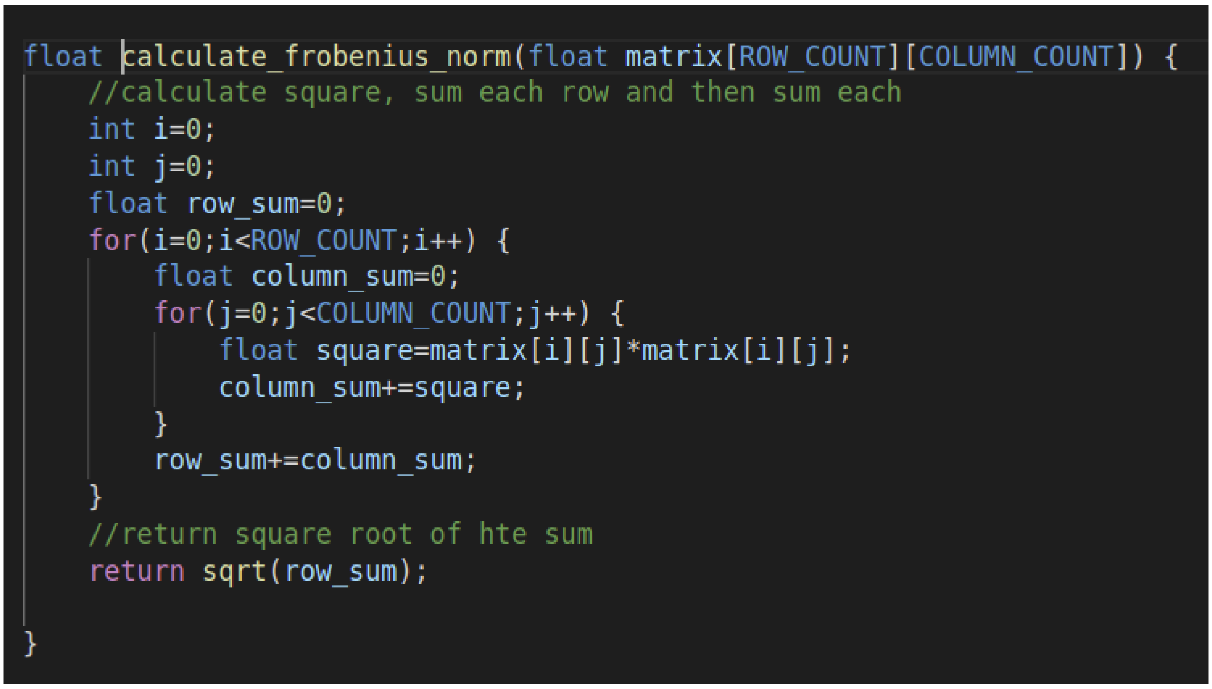
files > outputFile.dat

```
:1
262.650 23.950 81.100
23.950 156.290 10.620
81.100 10.620 180.960
:2
```

While reading output file, firstly I read the : and process number. After these, I read three lines byte by byte. I did it like these because We don’t know exactly how many character in the file. After seeing a \n, I take the buffer and I use sscanf to get the floats. After reading 3rd \n, this means that we are done with 1 process.

After reading a process,I calculate the frobenius norm:

float frobenius_norm=calculate_frobenius_norm(matrix);

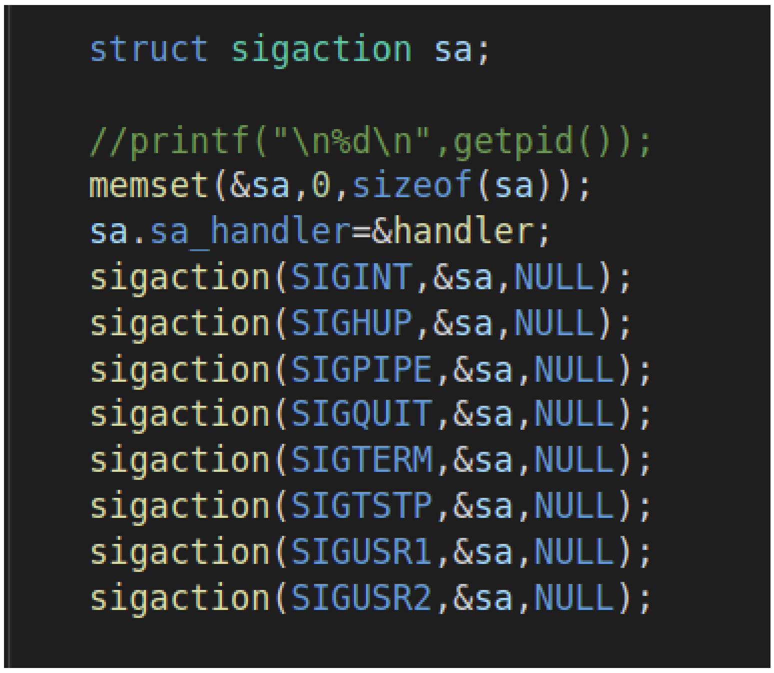


```c
float calculate_frobenius_norm(float matrix[ROW_COUNT][COLUMN_COUNT]) {
    //calculate square, sum each row and then sum each
    int i=0;
    int j=0;
    float row_sum=0;
    for(i=0;i<ROW_COUNT;i++) {
        float column_sum=0;
        for(j=0;j<COLUMN_COUNT;j++) {
            float square=matrix[i][j]*matrix[i][j];
            column_sum+=square;
        }
        row_sum+=column_sum;
    }
    //return square root of hte sum
    return sqrt(row_sum);

}
```

After finding frobenius norm I save this value into my norm struct array. After finding all distances, I compare them and find the result.

MY SIGNAL HANDLING MECHANISM:



```
struct sigaction sa;

//printf("\n%d\n",getpid());
memset(&sa,0,sizeof(sa));
sa.sa_handler=&handler;
sigaction(SIGINT,&sa,NULL);
sigaction(SIGHUP,&sa,NULL);
sigaction(SIGPIPE,&sa,NULL);
sigaction(SIGQUIT,&sa,NULL);
sigaction(SIGTERM,&sa,NULL);
sigaction(SIGTSTP,&sa,NULL);
sigaction(SIGUSR1,&sa,NULL);
sigaction(SIGUSR2,&sa,NULL);
```

I decided to take the same actions against these signals
SIGHUP,SIGPIPE,SIGQUIT,SIGTERM

SIGTSTP,SIGUSR1,SIGUSR2

In parent and child I have these code

```
sig_atomic_t flag=0;

void handler(int signal_number) {

   if(signal_number==SIGINT || signal_number==SIGHUP ||
signal_number==SIGPIPE || signal_number==SIGQUIT ||
signal_number==SIGTERM || signal_number==SIGTSTP ||
signal_number==SIGUSR1 || signal_number==SIGUSR2) {

        flag=1;

   }

}
```

4. ACHIEVED REQUIREMENTS

I can read the input file successfully, I got the x y and z coordinates from characters ASCII values. I can manipulate values between 0-126. After finding 10 points which is 30 characters, I create a child process and save its id into struct. Then in the child process, I create a 3x3 matrix with respect to given 10 coordinats. After finding matrix, I wrote it to the output file successfully. If there is an interrupt happens with SIGINT or whatever, I immediately close my open files, do cleaning and return from child process. In parent process, I wait all childrens to finish. If there is an interrupt happens, After all childrens finishes their jobs, parent process immediately do its cleaning and close open files and terminate. If there is no interrupt happens, and all childs finished their job,then I read the output file successfully and calculate frobenius norm for each matrix. After calculating it, I find the closest 2 pair and print it to the console.

5. FAILED REQUIREMENTS

I have a concern about program execution command. If it is given like

```
./processP -o outputFilePath  -i inputFilePath
```

Then my program will print usage and exit. Because input is not valid. According to the pdf, -i should come before the -o. I designed it that way.