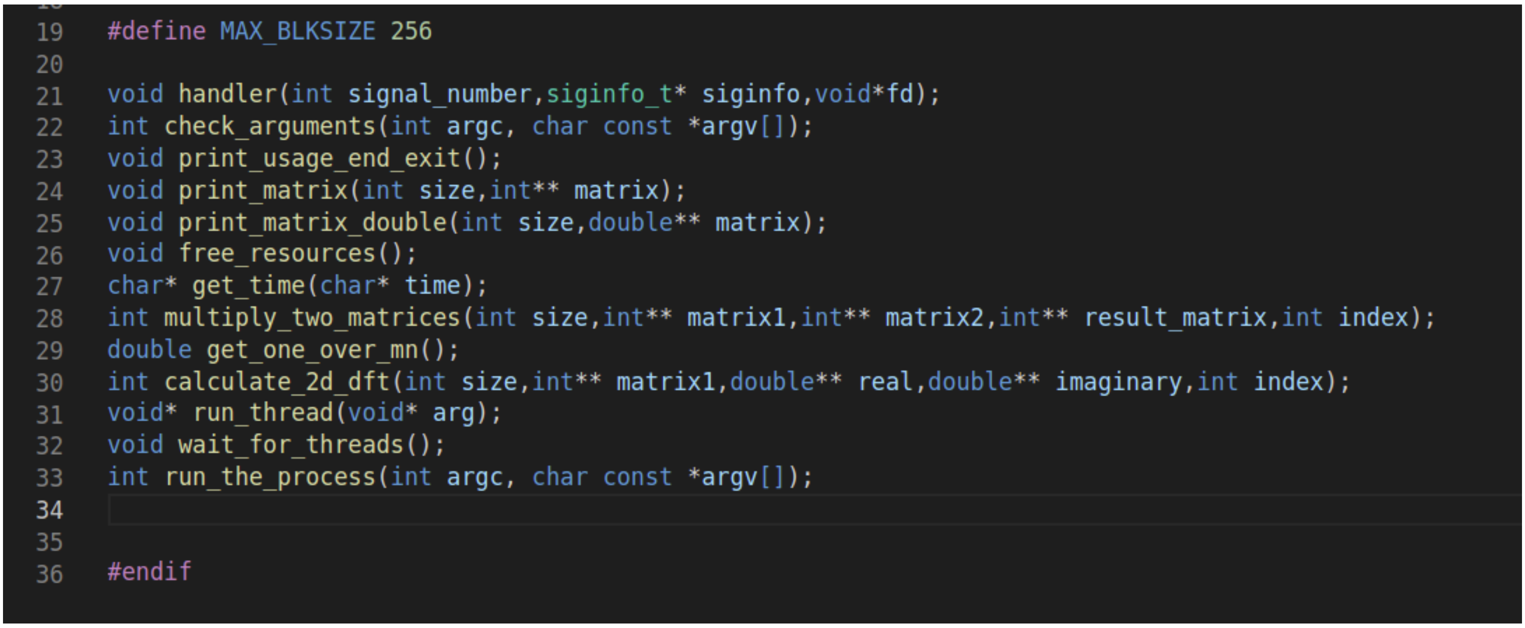
CSE 344 Systems Programming HW5 Report

Yakup Talha Yolcu
1801042609

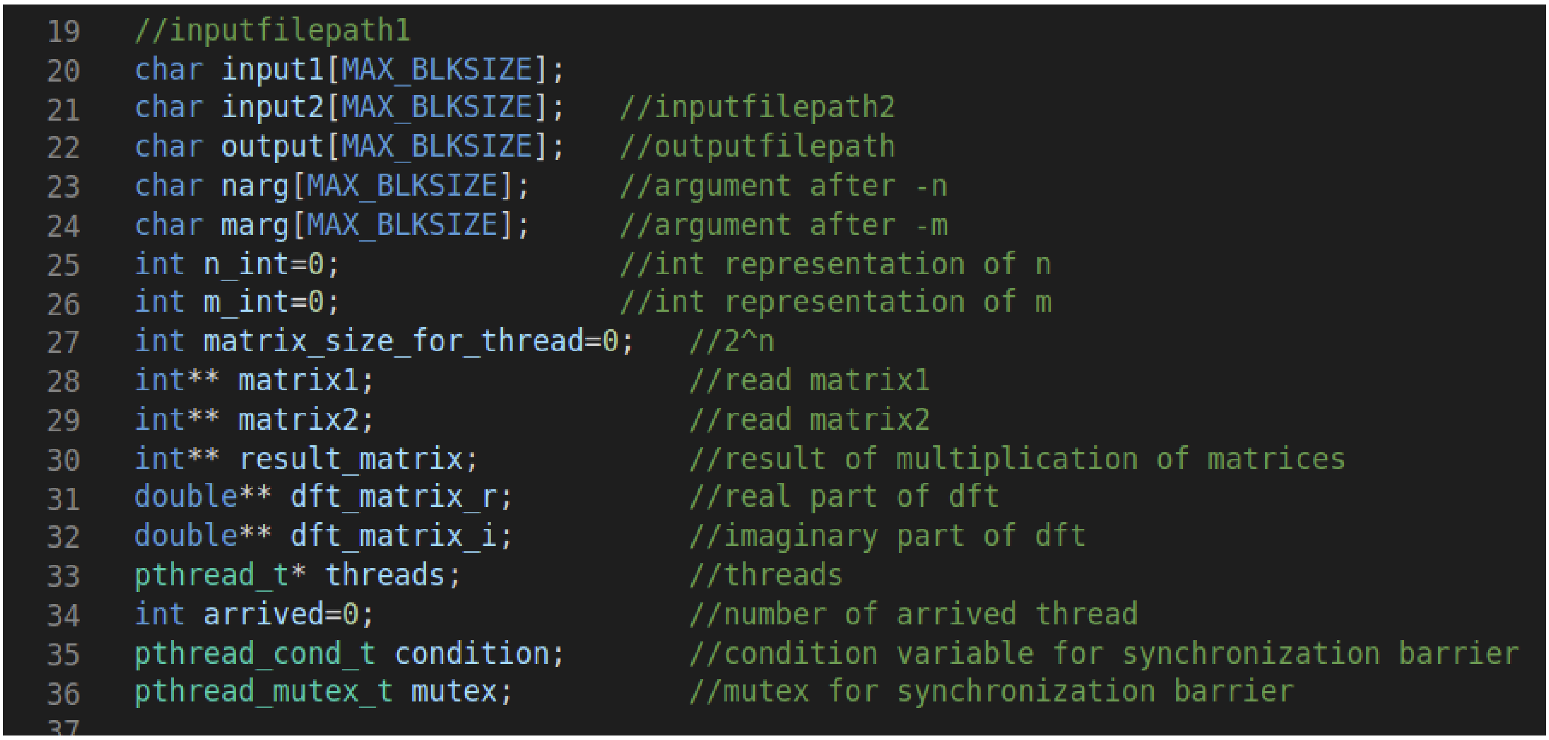
header file



```
#define MAX_BLKSIZE 256

void handler(int signal_number,siginfo_t* siginfo,void*fd);
int check_arguments(int argc, char const *argv[]);
void print_usage_end_exit();
void print_matrix(int size,int** matrix);
void print_matrix_double(int size,double** matrix);
void free_resources();
char* get_time(char* time);
int multiply_two_matrices(int size,int** matrix1,int** matrix2,int** result_matrix,int index);
double get_one_over_mn();
int calculate_2d_dft(int size,int** matrix1,double** real,double** imaginary,int index);
void* run_thread(void* arg);
void wait_for_threads();
int run_the_process(int argc, char const *argv[]);


#endif
```

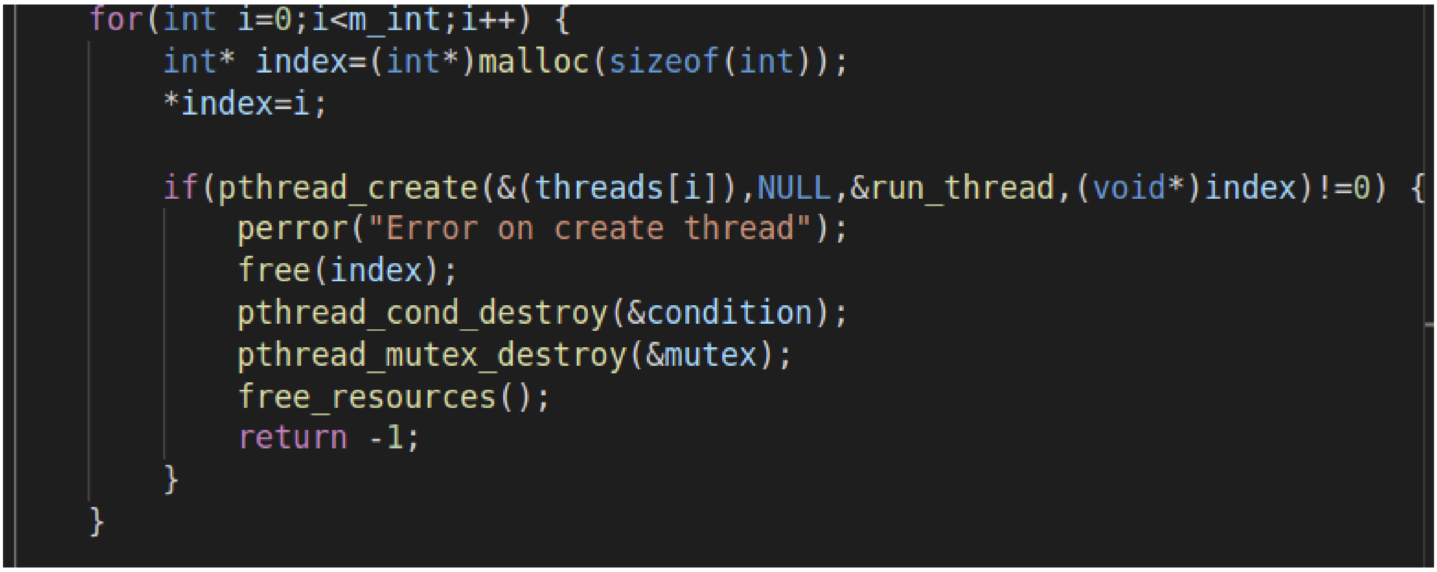
handler -> signal handler, checks sigint signal
check arguments -> takes directly argc and argv from main and firstly looks for argument count which should be 11, then looks for -i , -j, -o, -n and -m with order does not matter. If all arguments are OK, then it returns zero. Otherwise -1
print usage and exit -> if there is an inappropriate argument, then it is called and program exits.
free resources -> frees all the resources such as matrix1,matrix2,result matrix, dft matrix i, dft matrix r and threads array
get_ time -> returns the timestamp
multiply_two_matrices -> multiplies two matrices with size, result is written to the result_matrix. Index is taken as parameter because each thread makes calculation of some number of column. With using index, We determine the calculation point.
calculate_2d_dft -> takes size, multiplication matrix, it writes 2d dft result to the real and imaginary matrices. Again index is taken to determine calculation points
run_thread -> This function for threads. It is given as parameter to the pthread_create
wait_for_threads -> joining threads
run_the_process -> process function

variables:



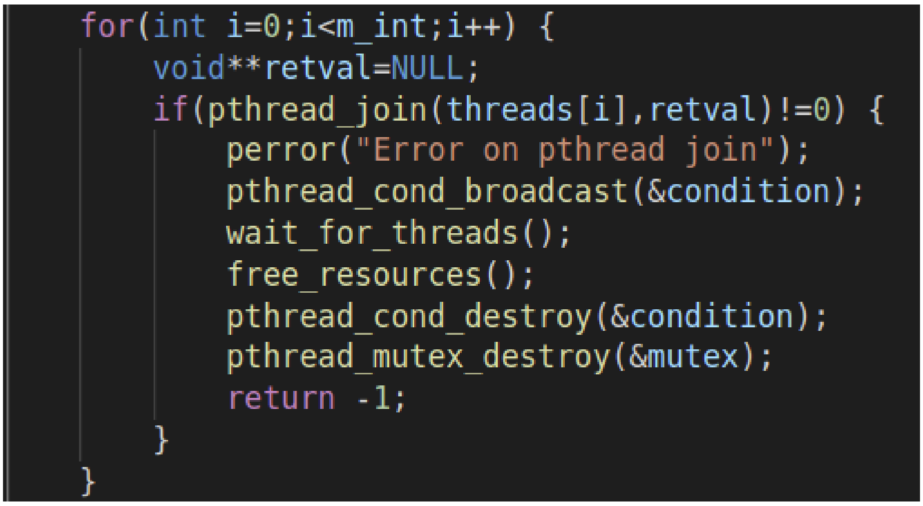
```
//inputfilepath1
char input1[MAX_BLKSIZE];
char input2[MAX_BLKSIZE];   //inputfilepath2
char output[MAX_BLKSIZE];   //outputfilepath
char narg[MAX_BLKSIZE];     //argument after -n
char marg[MAX_BLKSIZE];     //argument after -m
int n_int=0;                //int representation of n
int m_int=0;                //int representation of m
int matrix_size_for_thread=0;   //2^n
int** matrix1;                  //read matrix1
int** matrix2;                  //read matrix2
int** result_matrix;            //result of multiplication of matrices
double** dft_matrix_r;          //real part of dft
double** dft_matrix_i;          //imaginary part of dft
pthread_t* threads;             //threads
int arrived=0;                  //number of arrived thread
pthread_cond_t condition;       //condition variable for synchronization barrier
pthread_mutex_t mutex;          //mutex for synchronization barrier
```

thread creation:



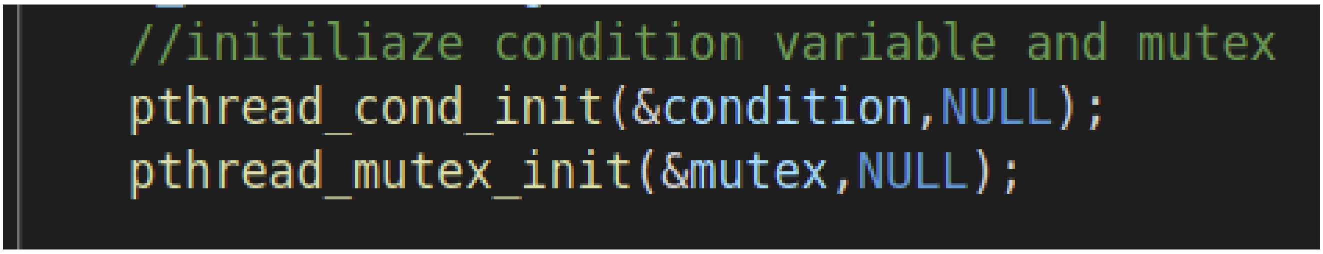
```
for(int i=0;i<m_int;i++) {
    int* index=(int*)malloc(sizeof(int));
    *index=i;

    if(pthread_create(&(threads[i]),NULL,&run_thread,(void*)index)!=0) {
        perror("Error on create thread");
        free(index);
        pthread_cond_destroy(&condition);
        pthread_mutex_destroy(&mutex);
        free_resources();
        return -1;
    }
}
```

waiting for all threads to finish:



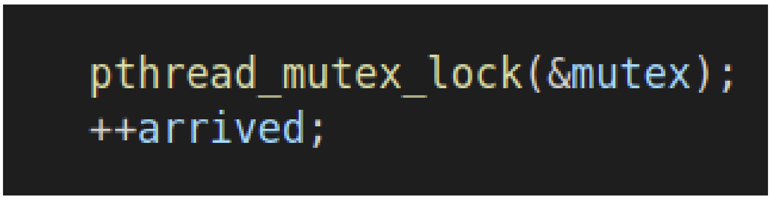
```
    for(int i=0;i<m_int;i++) {
        void**retval=NULL;
        if(pthread_join(threads[i],retval)!=0) {
            perror("Error on pthread join");
            pthread_cond_broadcast(&condition);
            wait_for_threads();
            free_resources();
            pthread_cond_destroy(&condition);
            pthread_mutex_destroy(&mutex);
            return -1;
        }
    }
```

initialization of condition variable and mutex

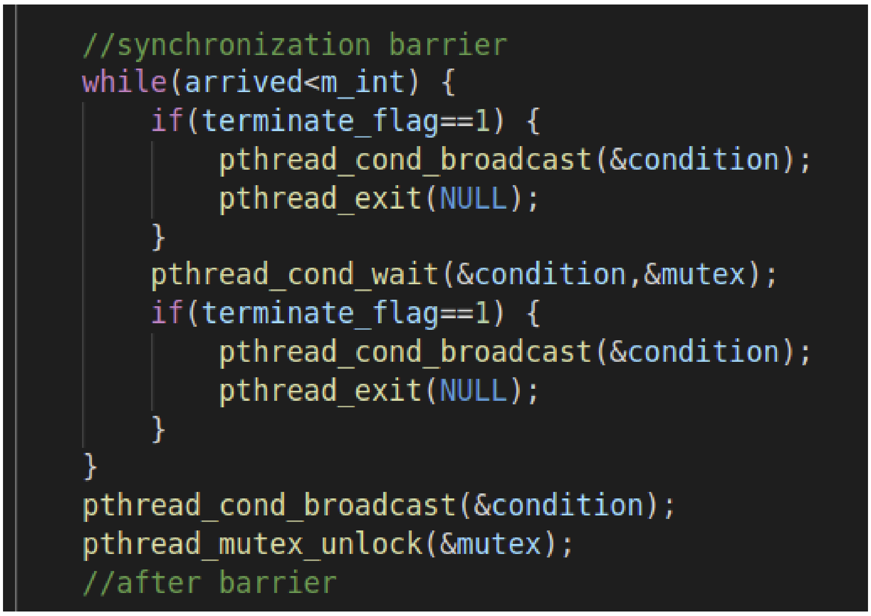


```
    //initiliaze condition variable and mutex
    pthread_cond_init(&condition,NULL);
    pthread_mutex_init(&mutex,NULL);
```

synchronization barrier:

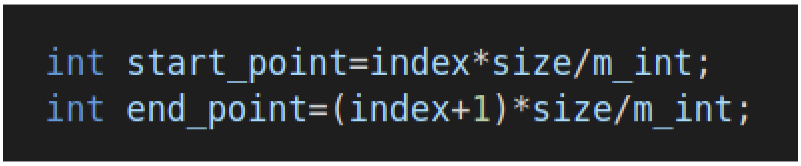


```
    pthread_mutex_lock(&mutex);
    ++arrived;
```



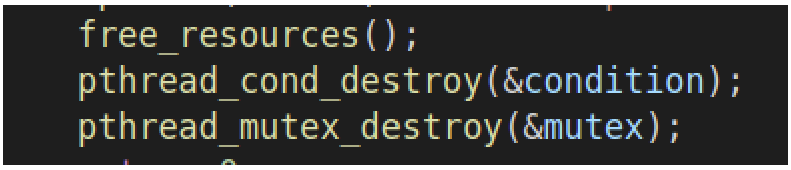
```
    //synchronization barrier
    while(arrived<m_int) {
        if(terminate_flag==1) {
            pthread_cond_broadcast(&condition);
            pthread_exit(NULL);
        }
        pthread_cond_wait(&condition,&mutex);
        if(terminate_flag==1) {
            pthread_cond_broadcast(&condition);
            pthread_exit(NULL);
        }
    }
    pthread_cond_broadcast(&condition);
    pthread_mutex_unlock(&mutex);
    //after barrier
```

determining calculation point:



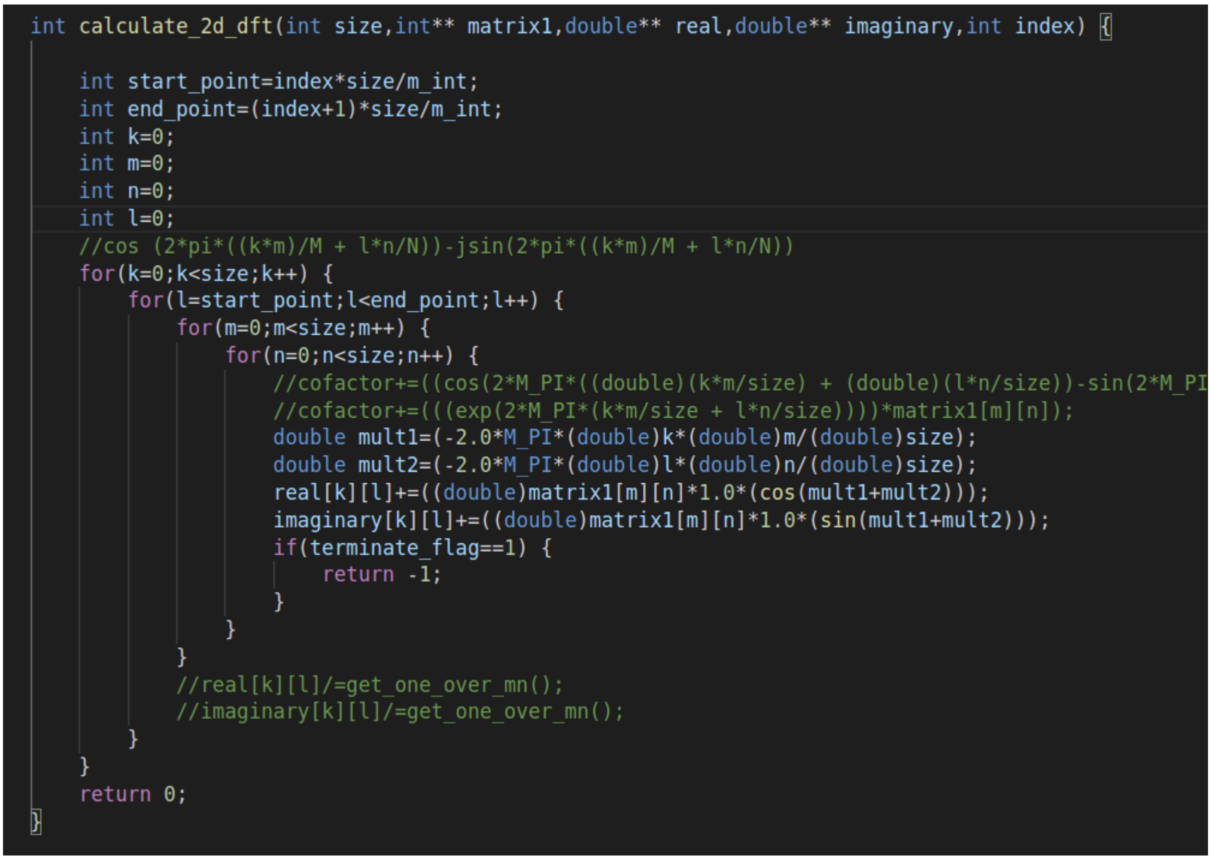
```
int start_point=index*size/m_int;
int end_point=(index+1)*size/m_int;
```

freeing resources:



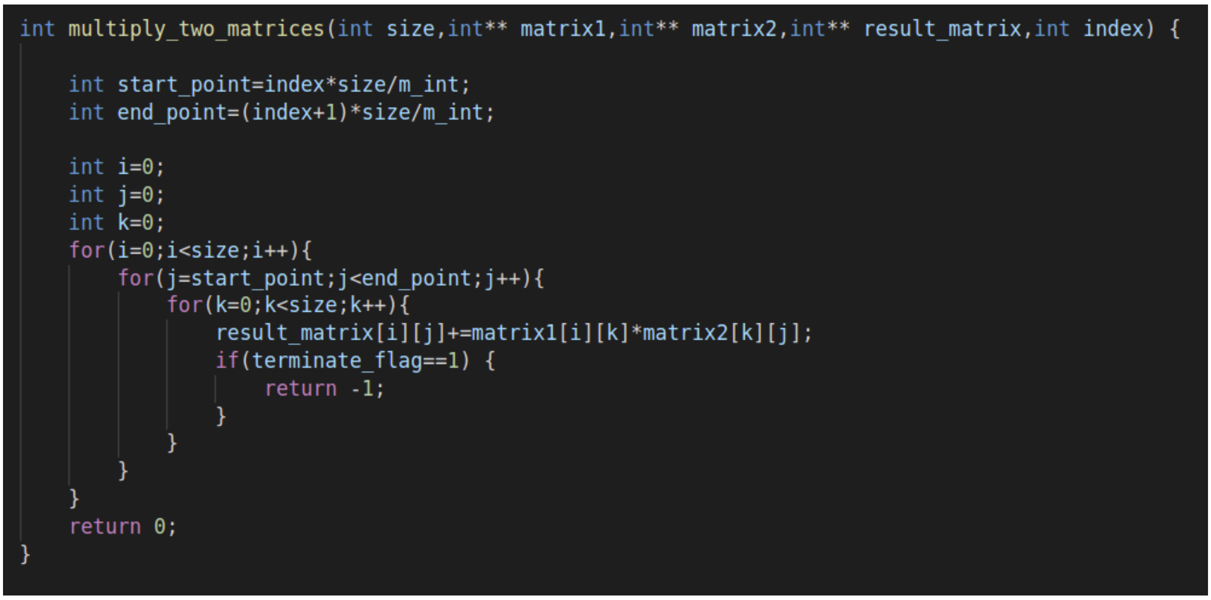
```
free_resources();
pthread_cond_destroy(&condition);
pthread_mutex_destroy(&mutex);
```

dft calculation:



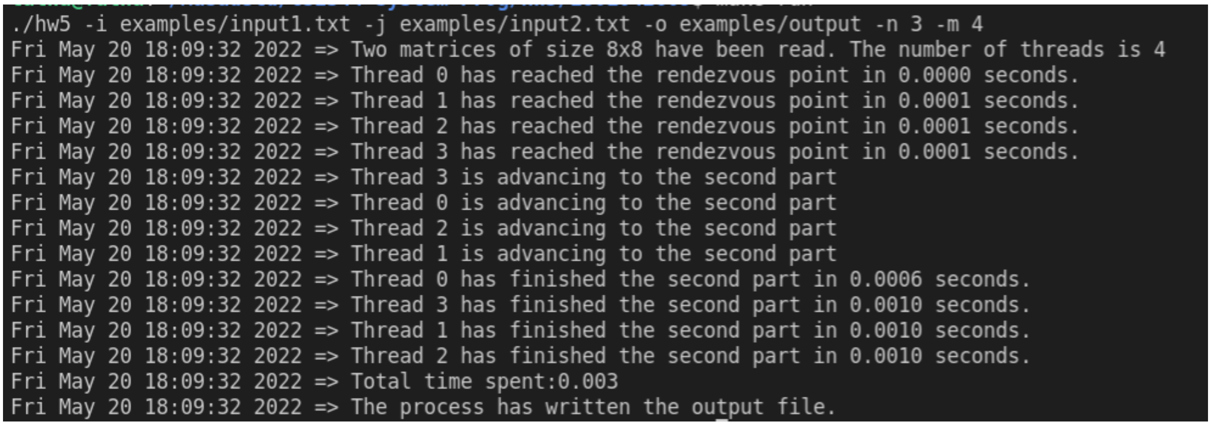
```
int calculate_2d_dft(int size,int** matrix1,double** real,double** imaginary,int index) {

    int start_point=index*size/m_int;
    int end_point=(index+1)*size/m_int;
    int k=0;
    int m=0;
    int n=0;
    int l=0;
    //cos (2*pi*((k*m)/M + l*n/N))-jsin(2*pi*((k*m)/M + l*n/N))
    for(k=0;k<size;k++) {
        for(l=start_point;l<end_point;l++) {
            for(m=0;m<size;m++) {
                for(n=0;n<size;n++) {
                    //cofactor+=((cos(2*M_PI*((double)(k*m/size) + (double)(l*n/size))-sin(2*M_PI
                    //cofactor+=(((exp(2*M_PI*(k*m/size + l*n/size))))*matrix1[m][n]);
                    double mult1=(-2.0*M_PI*(double)k*(double)m/(double)size);
                    double mult2=(-2.0*M_PI*(double)l*(double)n/(double)size);
                    real[k][l]+=((double)matrix1[m][n]*1.0*(cos(mult1+mult2)));
                    imaginary[k][l]+=((double)matrix1[m][n]*1.0*(sin(mult1+mult2)));
                    if(terminate_flag==1) {
                        return -1;
                    }
                }
            }
            //real[k][l]/=get_one_over_mn();
            //imaginary[k][l]/=get_one_over_mn();
        }
    }
    return 0;
}
```

matrix multipliaction:



```c
int multiply_two_matrices(int size,int** matrix1,int** matrix2,int** result_matrix,int index) {

    int start_point=index*size/m_int;
    int end_point=(index+1)*size/m_int;

    int i=0;
    int j=0;
    int k=0;
    for(i=0;i<size;i++){
        for(j=start_point;j<end_point;j++){
            for(k=0;k<size;k++){
                result_matrix[i][j]+=matrix1[i][k]*matrix2[k][j];
                if(terminate_flag==1) {
                    return -1;
                }
            }
        }
    }
    return 0;
}
```

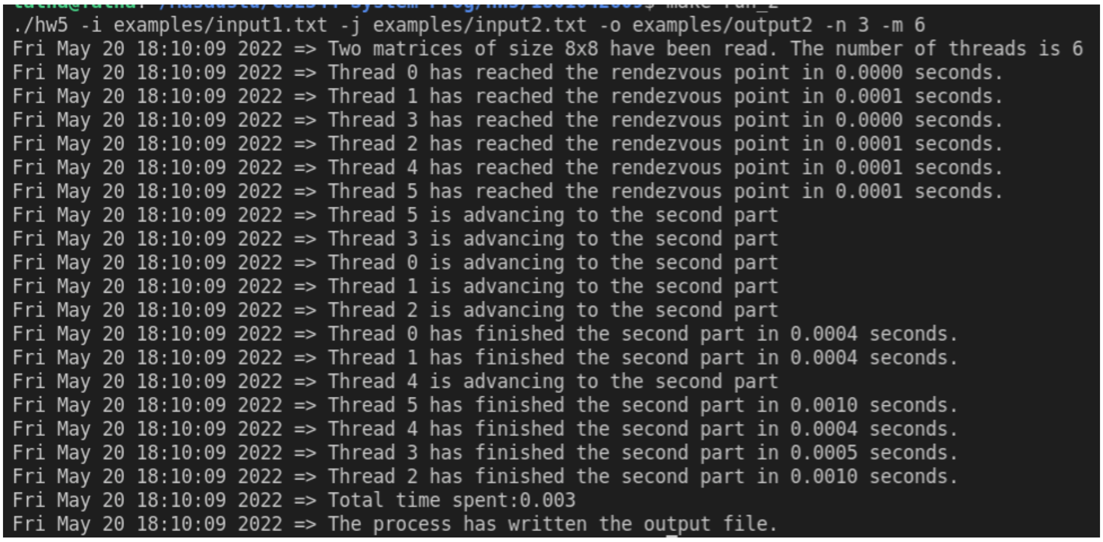
test 1:



```
./hw5 -i examples/input1.txt -j examples/input2.txt -o examples/output -n 3 -m 4
Fri May 20 18:09:32 2022 => Two matrices of size 8x8 have been read. The number of threads is 4
Fri May 20 18:09:32 2022 => Thread 0 has reached the rendezvous point in 0.0000 seconds.
Fri May 20 18:09:32 2022 => Thread 1 has reached the rendezvous point in 0.0001 seconds.
Fri May 20 18:09:32 2022 => Thread 2 has reached the rendezvous point in 0.0001 seconds.
Fri May 20 18:09:32 2022 => Thread 3 has reached the rendezvous point in 0.0001 seconds.
Fri May 20 18:09:32 2022 => Thread 3 is advancing to the second part
Fri May 20 18:09:32 2022 => Thread 0 is advancing to the second part
Fri May 20 18:09:32 2022 => Thread 2 is advancing to the second part
Fri May 20 18:09:32 2022 => Thread 1 is advancing to the second part
Fri May 20 18:09:32 2022 => Thread 0 has finished the second part in 0.0006 seconds.
Fri May 20 18:09:32 2022 => Thread 3 has finished the second part in 0.0010 seconds.
Fri May 20 18:09:32 2022 => Thread 1 has finished the second part in 0.0010 seconds.
Fri May 20 18:09:32 2022 => Thread 2 has finished the second part in 0.0010 seconds.
Fri May 20 18:09:32 2022 => Total time spent:0.003
Fri May 20 18:09:32 2022 => The process has written the output file.
```

test2:

We expect accelerate 1.5 times but it didn’t I have 8 cores in my computer



```
./hw5 -i examples/input1.txt -j examples/input2.txt -o examples/output2 -n 3 -m 6
Fri May 20 18:10:09 2022 => Two matrices of size 8x8 have been read. The number of threads is 6
Fri May 20 18:10:09 2022 => Thread 0 has reached the rendezvous point in 0.0000 seconds.
Fri May 20 18:10:09 2022 => Thread 1 has reached the rendezvous point in 0.0001 seconds.
Fri May 20 18:10:09 2022 => Thread 3 has reached the rendezvous point in 0.0000 seconds.
Fri May 20 18:10:09 2022 => Thread 2 has reached the rendezvous point in 0.0001 seconds.
Fri May 20 18:10:09 2022 => Thread 4 has reached the rendezvous point in 0.0001 seconds.
Fri May 20 18:10:09 2022 => Thread 5 has reached the rendezvous point in 0.0001 seconds.
Fri May 20 18:10:09 2022 => Thread 5 is advancing to the second part
Fri May 20 18:10:09 2022 => Thread 3 is advancing to the second part
Fri May 20 18:10:09 2022 => Thread 0 is advancing to the second part
Fri May 20 18:10:09 2022 => Thread 1 is advancing to the second part
Fri May 20 18:10:09 2022 => Thread 2 is advancing to the second part
Fri May 20 18:10:09 2022 => Thread 0 has finished the second part in 0.0004 seconds.
Fri May 20 18:10:09 2022 => Thread 1 has finished the second part in 0.0004 seconds.
Fri May 20 18:10:09 2022 => Thread 4 is advancing to the second part
Fri May 20 18:10:09 2022 => Thread 5 has finished the second part in 0.0010 seconds.
Fri May 20 18:10:09 2022 => Thread 4 has finished the second part in 0.0004 seconds.
Fri May 20 18:10:09 2022 => Thread 3 has finished the second part in 0.0005 seconds.
Fri May 20 18:10:09 2022 => Thread 2 has finished the second part in 0.0010 seconds.
Fri May 20 18:10:09 2022 => Total time spent:0.003
Fri May 20 18:10:09 2022 => The process has written the output file.
```