Московский авиационный институт
(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Операционные системы»

Лабораторная работа № 3

Тема: Управление потоками в ОС

Студент: Волков М. А.
Группа: 80-207
Преподаватель: Миронов Е. С.
Дата:
Оценка:

Москва, 2020

Постановка задачи

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

Вариант 10: Наложить K раз медианный фильтр на матрицу, состоящую из целых чисел. Размер окна задается.

Алгоритм решения

Задача разбита на 4 файла: *vector.h, merge.h, matfunc.h, main.c*.

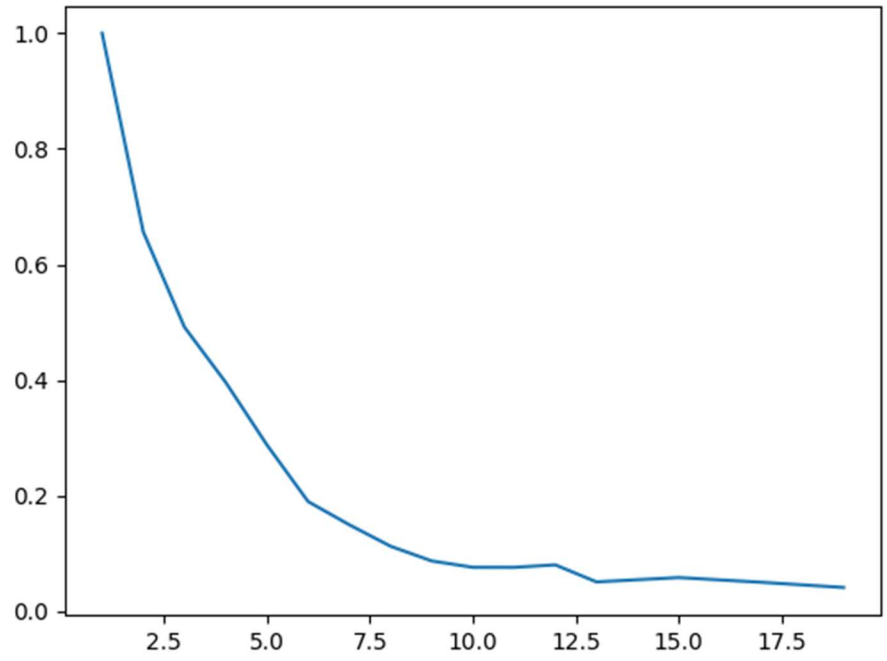
Самописный вектор в данном задании нужен для удобства использования функции сортировки merge. Сортировка используется при применении медианного фильтра. В заголовочном файле matfunc, опять же для удобства, определены операции с матрицей.

В *main.c* сделана функция *threadFunc,* которая нужна для выполнения задачи в разных потоках. В глобальных переменных *in, out* содержатся матрицы. Из их названия понятно, что in – является матрице, из которой берутся данные, а out – в которую заносятся новые данные.

В функции *main* происходит ввод размера матрицы, данных в матрицу, размер окна. Также в этой функции идет создание необходимого количества потоков, задаваемое пользователем в консоли перед стартом программы. Здесь же идет запись в файл времени выполнения программы, а затем небольшой скрипт на питоне делает график эффективности работы программы.

Эффективность вычисляется по формуле $\boldsymbol{S = \frac{T_1}{T_n * n}}$, где S – эффективность, $\boldsymbol{T_1}$ – время выполнения программы на одном потоке. $\boldsymbol{T_n}$ – время выполнения на n-ом потоке.

График:



Из этого графика видно, что эффективность с появлением каждого потока падает. Это связано с накладными расходами при создании потока и с использованием ОС нужных ей для работы потоков.

Листинг программы

vector.h

```
#ifndef VECTOR_H
#define VECTOR_H

#include "stdlib.h"
#include "stdio.h"

typedef struct vector {
    int *data;//тут надо подправить тип
    int size;
} vector;

void create(vector *v, int size) {
    v->data = (int *) malloc(sizeof(int) * size);//тут надо подправить тип
    v->size = size;
    int i;
    for (i = 0; i < size; ++i) { //если используете в виде данных структуру, желательно
убрать этот for или
        v->data[i] = '\0';           //сделать свою инициализацию
    }
}

void push_back(vector *v, const int x) {//тут надо подправить тип
    v->data = (int *) realloc(v->data, sizeof(int) * (v->size + 1));//тут надо подправить
тип
    v->data[v->size] = x;
    v->size++;
}

void pop(vector *v) {
    v->data = (int *) realloc(v->data, sizeof(int) * (v->size - 1));//тут надо подправить
тип
    v->size--;
}

void print_all(vector *v) {
    int i;
    for (i = 0; i < v->size; ++i) {
        printf("%d ", v->data[i]);
    }
    printf("\n");
}

int size(vector *v) {
    if (v == NULL) {
        return 0;
    }
    return v->size;
}

void destroy(vector *v) {
    v->size = 0;
    //free(v->data);
}

#endif
```

merge.h

```
#ifndef OC_MERGE_H
#define OC_MERGE_H
#include "vector.h"
void merge(vector* data, int l, int r){
    if(l == r){
        return;
    }

    int m = (l + r) / 2;

    merge(data,l,m);
    merge(data,m+1, r);

    int middle[data->size];
    //create(&middle,0);
    int lhs=l,rhs = m+1;
    int count = 0;
    while(lhs != m+1 && rhs != r+1){
        if(data->data[lhs] < data->data[rhs]){
            //push_back(&middle,data->data[lhs]);
            middle[count] = data->data[lhs];
            ++count;
            lhs++;
        }
        else{
            //push_back(&middle,data->data[rhs]);
            middle[count] = data->data[rhs];
            ++count;
            rhs++;
        }
    }

    if(lhs != m + 1){
        for (int i = lhs; i <= m; ++i) {
//            push_back(&middle,data->data[i]);
            middle[count] = data->data[i];
            ++count;
        }
    }
    else if(rhs != r + 1){
        for (int i = rhs; i <= r; ++i) {
            //push_back(&middle,data->data[i]);
            middle[count] = data->data[i];
            ++count;
        }
    }

//    if(middle.size == 2) {
//        print_all(&middle);
//    }

    for (int i = l; i <= r; ++i) {
        data->data[i] = middle[i-l];
    }
    //destroy(&middle);
}
#endif //OC_MERGE_H
```

matfunc.h

```
#ifndef MATFUNC_H
#define MATFUNC_H

struct Matrix {
   int dimn;
   int dimm;
   int **matr;
};

void scanfMatrix(struct Matrix *a, int n, int m) { // Ввод матрицы
   a->dimn = n;
   a->dimm = m;
   a->matr = (int**)malloc(sizeof(int*)*n);
   for (int i=0; i<n; i++) {
      a->matr[i] = (int*) malloc(sizeof(int)*m);
   }

   for (int i=0; i<n; i++) {
      for (int j=0; j<m; j++) {
         scanf("%d", &a->matr[i][j]);
      }
   }
}

/*int random(int seed) {
    seed * 123456;
}*/

void scanf_With_Fill_Rand_Matrix(struct Matrix *a, int n, int m) { // Ввод с заполнением
матрицы
   a->dimn = n;
   a->dimm = m;
   a->matr = (int**)malloc(sizeof(int*)*n);
   for (int i=0; i<n; i++) {
      a->matr[i] = (int*) malloc(sizeof(int)*m);
   }
    srand(time(NULL));
   for (int i=0; i<n; i++) {
      for (int j=0; j<m; j++) {

            int num = rand() % 100;
        a->matr[i][j] = num;
      }
   }
}

void scanf_Matrix_with_Fill_0(struct Matrix* a, int n, int m){
    a->dimm = m;
    a->dimn = n;
    a->matr = (int**)malloc(sizeof(int*)*n);
    for (int i = 0; i < n; ++i) {
        a->matr[i] = (int*)malloc(sizeof(int)*m);
    }

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            a->matr[i][j] = 0;
        }
    }
}

void printMatrix(struct Matrix *a) { // Печать матрицы
   printf("\n");
   for (int i=0; i<a->dimn; i++) {
      for (int j=0; j<a->dimm; j++) {
         printf("%d ", a->matr[i][j]);
      }
      printf("\n");
   }
```

```
   printf("\n");
}

void fillMatrix(struct Matrix *a, int num) { // Заполнение матрицы
   for (int i=0; i<a->dimn; i++) {
      for (int j=0; j<a->dimm; j++) {
         a->matr[i][j] = num;
      }
   }
}

void swap (struct Matrix *a, struct Matrix *b) { // Обмен ссылками на матрицы одинаковой
размерности
    struct Matrix p; int **pp;
    p = *a; pp = a->matr;
    *a = *b; a->matr = b->matr;
    *b = p; b->matr = pp;
}

#endif
```

main.c

```
#include "time.h"
#include "stdio.h"
#include "vector.h"
#include "matfunc.h"
#include "stdlib.h"
#include <pthread.h>
#include "merge.h"

struct Matrix in;
struct Matrix out;

typedef struct tr_data {
    int x, y;
    int winX, winY;

} tr_data;

int k, n, m;

int max(int lhs, int rhs) {
    if (lhs > rhs) {
        return lhs;
    }
    return rhs;
}

int min(int lhs, int rhs) {
    if (lhs < rhs) {
        return lhs;
    }
    return rhs;
}

void *threadFunc(void *Tdata) {
    fflush(stdout);
    struct tr_data *id = (struct tr_data *) Tdata;
    vector pixels;
    create(&pixels, 0);

    for (int i = max(0, id->x - ((id->winX - 1) / 2)); i <= min(n - 1, id->x + ((id->winX
- 1) / 2)); ++i) {
        for (int j = max(0, id->y - ((id->winY - 1) / 2)); j <= min(m - 1, id->y + ((id-
>winY - 1) / 2)); ++j) {

            push_back(&pixels, in.matr[i][j]);
        }
    }
```

```
    merge(&pixels, 0, pixels.size - 1);

    if (m > 1 && n > 1) {
        out.matr[id->x][id->y] = pixels.data[pixels.size / 2];
        destroy(&pixels);
        return NULL;
    }
    else {
        out.matr[id->x][id->y] = in.matr[id->x][id->y];
        destroy(&pixels);
        return NULL;
    }

}


int main(int argc, char const *argv[]) {
    unsigned int max_threads = 1;
    if (argc > 1 && atoi(argv[1]) > 0) {
        max_threads = atoi(argv[1]);
    }

    int winX, winY;

    printf("Threads %d\nEnter size of matrix. Enter N and M\n> ", max_threads);
    fflush(stdout);
    scanf("%d %d", &n, &m);

    if (n == 0 || m == 0) {
        printf("I cannot create matrix with this n and m\n");
        return 0;
    }

    scanf_With_Fill_Rand_Matrix(&in, n, m);
    printf("Enter your window\n> ");
    fflush(stdout);
    scanf("%d %d", &winX, &winY);

    //условия задания окна
    if (winX != winY ||
        winX == 0 || winY == 0 ||
        winX > n || winY > m ||
        winX % 2 == 0) {
        printf("I cannot work with this window\n");
        return 0;
    }

    printf("Enter your K\n> ");
    fflush(stdout);
    scanf("%d", &k);

    pthread_t *threads = (pthread_t *) malloc(sizeof(pthread_t) * max_threads);

    double start, end;

    struct tr_data *args = (struct tr_data *) malloc(sizeof(struct tr_data) * n * m);
    start = clock();

    for (int h = 0; h < k; ++h) {
        scanf_Matrix_with_Fill_0(&out, n, m);
        int count = 0;
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < m; ++j) {
                args[count].x = i;
                args[count].y = j;
                args[count].winX = winX;
                args[count].winY = winY;
                ++count;
            }
        }
```

```
        for (unsigned int i = 0; i < n * m; i += max_threads) {
            for (unsigned int j = 0; j < max_threads; ++j) {
                //чтобы не запускался под конец новый процесс
                if (i + j >= n * m) {
                    break;
                }
                if (pthread_create(&threads[j], NULL, threadFunc, (void *) &args[i + j])
!= 0) {
                    perror("Cannot create thread\n");
                    return 1;
                }
            }

            //ждем завершения потоков
            for (int j = 0; j < max_threads; ++j) {
                if (i + j >= n * m) {
                    break;
                }
                if (pthread_join(threads[j], NULL)) {
                    perror("Thread didn't finished\n");
                    return 1;
                }
            }

        }
        swap(&in, &out);
    }
    end = clock();

    free(threads);
    free(args);

    printf("Result matrix:\n");

    FILE *file = fopen("log.txt", "a");
    FILE *thread = fopen("/mnt/d/Documents/Projects/c++/OC/venv/treads.txt", "a");
    FILE *time = fopen("/mnt/d/Documents/Projects/c++/OC/venv/time.txt", "a");

    double threTime = (end - start) / 1000;

    fprintf(file, "threads = %d\ntime = %fms\n\n", max_threads, (end - start) / 1000);
    fprintf(thread, "%d\n", max_threads);
    fprintf(time, "%f\n", threTime);

    fclose(file);
    fclose(thread);
    fclose(time);
    return 0;
}
```

Тесты и протокол исполнения

Тест:

```
Threads 2
Enter size of matrix. Enter N and M
> 10 10

15 49 44 71 83 22 83 62 25 54
93 45 87 46 63 80 82 13 10 17
25 18 49 9 99 37 56 13 81 37
51 68 72 26 69 25 26 41 65 27
99 33 50 91 87 65 87 18 70 97
42 3 48 97 88 63 50 3 13 60
48 89 0 51 19 39 22 29 10 46
43 84 89 41 65 85 57 87 68 17
98 86 94 8 70 98 43 65 81 24
23 40 33 16 55 64 64 61 78 77

Enter your window
> 5 5
Enter your K
> 1
Result matrix:

45 46 49 49 63 62 62 37 54 25
49 49 51 49 63 56 62 37 41 37
49 49 51 50 65 62 63 41 54 37
49 49 51 63 63 56 63 41 41 27
48 49 50 50 50 41 41 39 41 37
50 51 51 65 57 51 57 46 46 41
50 51 65 65 63 63 65 57 50 46
48 48 48 63 55 57 63 60 57 60
84 48 51 64 55 57 64 64 61 65
84 43 55 65 64 64 65 65 65 68

Process finished with exit code 0
```

strace:

```
execve("./lab3", ["./lab3", "3"], 0x7ffd1b05d1d8 /* 19 vars */) = 0
brk(NULL)                               = 0x55807b7a2000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe800ac310) = -1 EINVAL (Invalid argument)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=36270, ...}) = 0
mmap(NULL, 36270, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f037fbe1000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\201\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\261\23\327o"...,
68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f037fbdf000
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\261\23\327o"...,
68, 824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f037fbbc000
mmap(0x7f037fbc3000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x7000) = 0x7f037fbc3000
mmap(0x7f037fbd4000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) =
0x7f037fbd4000
mmap(0x7f037fbd9000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1c000) = 0x7f037fbd9000
```

```
mmap(0x7f037fbdb000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1,
0) = 0x7f037fbdb000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) =
784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68,
880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) =
784
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68,
880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f037f9ca000
mprotect(0x7f037f9ef000, 1847296, PROT_NONE) = 0
mmap(0x7f037f9ef000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x25000) = 0x7f037f9ef000
mmap(0x7f037fb67000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) =
0x7f037fb67000
mmap(0x7f037fbb2000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1e7000) = 0x7f037fbb2000
mmap(0x7f037fbb8000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1,
0) = 0x7f037fbb8000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f037f9c7000
arch_prctl(ARCH_SET_FS, 0x7f037f9c7740) = 0
mprotect(0x7f037fbb2000, 12288, PROT_READ) = 0
mprotect(0x7f037fbd9000, 4096, PROT_READ) = 0
mprotect(0x558079ed6000, 4096, PROT_READ) = 0
mprotect(0x7f037fc17000, 4096, PROT_READ) = 0
munmap(0x7f037fbe1000, 36270)           = 0
set_tid_address(0x7f037f9c7a10)         = 3086
set_robust_list(0x7f037f9c7a20, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f037fbc3bf0, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f037fbd13c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f037fbc3c90, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f037fbd13c0}, NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...}) = 0
brk(NULL)                               = 0x55807b7a2000
brk(0x55807b7c3000)                     = 0x55807b7c3000
write(1, "Threads 3\nEnter size of matrix. "..., 46Threads 3
Enter size of matrix. Enter N and M
) = 46
write(1, "> ", 2> )                     = 2
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...}) = 0
read(0, 5 5
"5 5\n", 1024)                  = 4
write(1, "Enter your window\n", 18Enter your window
)     = 18
write(1, "> ", 2> )                     = 2
read(0, 3 3
"3 3\n", 1024)                  = 4
write(1, "Enter your K\n", 13Enter your K
)          = 13
write(1, "> ", 2> )                     = 2
```

```
read(0, 1
"1\n", 1024)                    = 2
clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=5564204}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f037f1c6000
mprotect(0x7f037f1c7000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f037f9c5fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[3087], tls=0x7f037f9c6700,
child_tidptr=0x7f037f9c69
d0) = 3087
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f037e9c5000
mprotect(0x7f037e9c6000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f037f1c4fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f1c5700,
child_tidptr=0x7f037f1c59d0)
 = 3088
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f037e1c4000
mprotect(0x7f037e1c5000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f037e9c3fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037e9c4700,
child_tidptr=0x7f037e9c49d0)
 = 3089
clone(child_stack=0x7f037e9c3fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037e9c4700,
child_tidptr=0x7f037e9c49d0)
 = 3090
clone(child_stack=0x7f037f1c4fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f1c5700,
child_tidptr=0x7f037f1c59d0)
 = 3091
clone(child_stack=0x7f037f9c5fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f9c6700,
child_tidptr=0x7f037f9c69d0)
 = 3092
clone(child_stack=0x7f037f9c5fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f9c6700,
child_tidptr=0x7f037f9c69d0)
 = 3093
clone(child_stack=0x7f037f1c4fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f1c5700,
child_tidptr=0x7f037f1c59d0)
 = 3094
clone(child_stack=0x7f037e9c3fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037e9c4700,
child_tidptr=0x7f037e9c49d0)
 = 3095
clone(child_stack=0x7f037e9c3fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037e9c4700,
child_tidptr=0x7f037e9c49d0)
 = 3096
clone(child_stack=0x7f037f1c4fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
```

```
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f1c5700,
child_tidptr=0x7f037f1c59d0)
 = 3097
clone(child_stack=0x7f037f9c5fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f9c6700,
child_tidptr=0x7f037f9c69d0)
 = 3098
clone(child_stack=0x7f037f9c5fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f9c6700,
child_tidptr=0x7f037f9c69d0)
 = 3099
clone(child_stack=0x7f037f1c4fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f1c5700,
child_tidptr=0x7f037f1c59d0)
 = 3100
clone(child_stack=0x7f037e9c3fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037e9c4700,
child_tidptr=0x7f037e9c49d0)
 = 3101
clone(child_stack=0x7f037e9c3fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037e9c4700,
child_tidptr=0x7f037e9c49d0)
 = 3102
clone(child_stack=0x7f037f1c4fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f1c5700,
child_tidptr=0x7f037f1c59d0)
 = 3103
clone(child_stack=0x7f037f9c5fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[3104], tls=0x7f037f9c6700,
child_tidptr=0x7f037f9c69
d0) = 3104
clone(child_stack=0x7f037f9c5fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f9c6700,
child_tidptr=0x7f037f9c69d0)
 = 3105
clone(child_stack=0x7f037f1c4fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f1c5700,
child_tidptr=0x7f037f1c59d0)
 = 3106
clone(child_stack=0x7f037e9c3fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037e9c4700,
child_tidptr=0x7f037e9c49d0)
 = 3107
clone(child_stack=0x7f037e9c3fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037e9c4700,
child_tidptr=0x7f037e9c49d0)
 = 3108
clone(child_stack=0x7f037f1c4fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f1c5700,
child_tidptr=0x7f037f1c59d0)
```

```
 = 3109
clone(child_stack=0x7f037f9c5fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f9c6700,
child_tidptr=0x7f037f9c69d0)
 = 3110
clone(child_stack=0x7f037f9c5fb0,
flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CL
ONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[0], tls=0x7f037f9c6700,
child_tidptr=0x7f037f9c69d0)
 = 3111
clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=8676425}) = 0
write(1, "Result matrix:\n", 15Result matrix:
)        = 15
openat(AT_FDCWD, "log.txt", O_WRONLY|O_CREAT|O_APPEND, 0666) = 3
lseek(3, 0, SEEK_END)                   = 590
openat(AT_FDCWD, "/mnt/d/Documents/Projects/c++/OC/venv/treads.txt",
O_WRONLY|O_CREAT|O_APPEND, 0666) = 4
lseek(4, 0, SEEK_END)                   = 41
openat(AT_FDCWD, "/mnt/d/Documents/Projects/c++/OC/venv/time.txt",
O_WRONLY|O_CREAT|O_APPEND, 0666) = 5
lseek(5, 0, SEEK_END)                   = 209
fstat(3, {st_mode=S_IFREG|0777, st_size=590, ...}) = 0
fstat(4, {st_mode=S_IFREG|0777, st_size=41, ...}) = 0
fstat(5, {st_mode=S_IFREG|0777, st_size=209, ...}) = 0
write(3, "threads = 3\ntime = 3.112000ms\n\n", 31) = 31
close(3)                                = 0
write(4, "3\n", 2)                      = 2
close(4)                                = 0
write(5, "3.112000\n", 9)               = 9
close(5)                                = 0
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод:

Благодаря проделанной работе я научился создавать программу, процесс которой можно распараллелить на другие потоки, чтобы выполнялся процесс быстрее. Узнал, что можно параллелить только независимые друг от друга процессы, иначе программа, даже с использованием многопоточности, будет выполняться последовательно. Также на практике осознал, что многопоточные программы отнюдь не такие простые в применении, как на первый взгляд кажется, и эффективность работы программы не всегда растет пропорционально росту потоков.

Список литературы

1. Pthreads: Потоки в русле POSIX - Habr
2. Матричные фильтры обработки изображений - Habr