Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Группа: М80-206Б-20

Студент: Кочев Д.О.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 01.12.2023

Москва, 2023

**Постановка задачи**

**Вариант 9.**

Рассчитать детерминант матрицы (используя определение детерминанта)

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* int pthread\_create(pthread\_t \*restrict thread, const pthread\_attr\_t \*restrict attr, void \*(\*start\_routine)(void \*), void \*restrict arg) - создает поток.
* pthread\_join(pthread\_t thread, void \*\*retval) - блокирует вызывающий поток, пока указанный поток не завершится.

Сначала программа считывает порядок матрицы. Так как детерминант мы находим по определению, каждый процесс берет на себя вычисление некоторого количества алгебраических дополнений элементов первой строки матрицы. Следовательно, максимальное количество используемых потоков в программе всегда равно порядку вводимой матрицы. Если количество потоков меньше, то программа старается распределить количество алгебраических дополнений поровну между всеми потоками. Далее в каждом процессе находятся алгебраические дополнения элементов матрицы с помощью рекурсивной функции. После завершения работы всех процессов их результаты складываются и выводится ответ.

**Код программы**

**lab2.c**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <sys/time.h>

typedef struct {

int parami;

int param\_numbers;

int number\_of\_thread;

} ThreadParams;

int\*\* arr;

int n;

int\* determinant;

void freeMemory() {

for (int i = 0; i < n; ++i) {

free(arr[i]);

}

free(arr);

}

void findMinor(int minor[n][n], int row, int col, int size) {

int minor\_plus[size][size];

for (int i = 0; i < size; i++)

for (int j = 0; j < size; j++)

minor\_plus[i][j] = minor[i][j];

int minorRow = 0, minorCol = 0;

for (int i = 0; i < size; ++i) {

if (i != row) {

for (int j = 0; j < size; ++j) {

if (j != col) {

minor[minorRow][minorCol] = minor\_plus[i][j];

minorCol++;

}

}

minorCol = 0;

minorRow++;

}

}

}

// Функция для нахождения определителя матрицы

int findDeterminant(int matrix[n][n], int size) {

if (size == 1) {

return matrix[0][0];

} else if (size == 2) {

return matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0];

} else {

int det = 0;

int sign = 1;

for (int i = 0; i < size; i++) {

int minor[n][n]; // ОБЯЗАТЕЛЬНО РАЗМЕРНОСТЬ ТАКАЯ ЖЕ, КАК В ОБЪЯВЛЕНИИ ФУНКЦИИ

for (int u = 0; u < size; u++){

for (int j = 0; j < size; j++)

minor[u][j] = matrix[u][j];

}

// for (int u = 0; u < size; u++){

// for (int j = 0; j < size; j++)

// printf("%d ", minor[u][j]);

// printf("\n");

// }

findMinor(minor, 0, i, size);

// for (int u = 0; u < size - 1; u ++){

// for (int j = 0; j < size - 1; j++)

// printf("%d ", minor[u][j]);

// printf("\n");

// }

// printf("\n");

det += sign \* matrix[0][i] \* findDeterminant(minor, size - 1);

sign = -sign;

}

return det;

}

}

int help\_algСomplement(int row, int column){

int arr\_copy[n][n];

for (int i = 0; i < n; i ++)

for (int j = 0; j < n; j++)

arr\_copy[i][j] = arr[i][j];

findMinor(arr\_copy, row, column, n);

return arr[row][column] \* findDeterminant(arr\_copy, n - 1);

}

void thread\_create(pthread\_t\* thread, const pthread\_attr\_t\* attr, void \*(\*start)(void \*), void\* arg) {

if (pthread\_create(thread, attr, start, arg) != 0) {

perror("create thread\n");

exit(-1);

}

}

void\* threadDistributor(void\* arg){

ThreadParams\* paramsss = (ThreadParams\*)arg;

int i = paramsss->parami;

int numbers = paramsss->param\_numbers;

// printf("i in Distributor: %d \n", paramsss->parami);

// printf("numbers: %d \n", paramsss->param\_numbers);

printf("номер потока: %d \n", paramsss->number\_of\_thread);

int s = 0;

for (int j = i; j < i + numbers; j++){

// printf("determinant before: %d \n", determinant[j]);

determinant[j] = help\_algСomplement(0, j);

// printf("determinant after: %d \n", determinant[j]);

// s += algСomplement(0, j);

}

pthread\_exit((void\*)&s);

// pthread\_exit(NULL);

}

int main(int argc, char\* argv[]){

if (argc != 2){

printf("Incorrect input of arguments\n");

return 1;

}

int max\_threads = atoi(argv[1]);

if (max\_threads <= 0){

printf("Number of threads must be more then 0\n");

return 1;

}

printf("Please enter the matrix dimension: ");

scanf("%d", &n);

arr = (int\*\*)malloc(n \* sizeof(int\*));

determinant = (int\*)malloc(n \* sizeof(int));

for (int i = 0; i < n; i++){

arr[i] = (int\*)malloc(n \* sizeof(int));

for (int j = 0; j < n; j++)

scanf("%d", &arr[i][j]);

}

// for (int i = 0; i < n; i++){

// for (int j = 0; j < n; j++)

// printf("%d ", arr[i][j]);

// printf("\n");

// }

double numbers\_in\_threads\_doub = (double)n / max\_threads;

int numbers\_in\_threads = n / max\_threads;

if (numbers\_in\_threads\_doub != (double)numbers\_in\_threads)

numbers\_in\_threads += 1;

// printf("numbers\_in\_threads: %d \n", numbers\_in\_threads);

pthread\_t\* threads = (pthread\_t\*)malloc(max\_threads \* sizeof(pthread\_t));

// int\* thread\_args = (int\*)malloc(max\_threads \* sizeof(int));

ThreadParams params[n];

int k = 0;

struct timeval start\_time, end\_time;

gettimeofday(&start\_time, NULL); // начальное время

for (int i = 0; i < n; i += numbers\_in\_threads){

params[k].param\_numbers = numbers\_in\_threads;

params[k].parami = i;

params[k].number\_of\_thread = k + 1;

// printf("i in main: %d \n", i);

thread\_create(&threads[k], NULL, threadDistributor, (void\*)&params[k]);

k++;

}

for (int i = 0; i < k; i += 1){

pthread\_join(threads[i], NULL);

}

int det\_final = 0;

int check = 1;

for (int i = 0; i< n; i++){

det\_final += determinant[i] \* check;

// printf("%d ", determinant[i]);

check \*= -1;

}

gettimeofday(&end\_time, NULL); // конечное время

long long start\_ms = start\_time.tv\_sec \* 1000LL + start\_time.tv\_usec / 1000; // преобразование из микросекунд в миллисекунды

long long end\_ms = end\_time.tv\_sec \* 1000LL + end\_time.tv\_usec / 1000; // преобразование из микросекунд в миллисекунды

double search\_time = (end\_ms - start\_ms) / 1000.0; // искомо

printf("\nОтвет: %d \n", det\_final);

printf("\nвремя выполения: %.4f", search\_time);

// printf("\nstart time: %f", start\_ms / 1000.0);

// printf("\nend\_time: %f\n ", end\_ms / 1000.0);

freeMemory();

free(determinant);

free(threads);

}

**Протокол работы программы**

$ ./a.out 3

Please enter the matrix dimension: 4

1 2 3 1

4 5 6 2

7 8 10 6

8 3 9 7

номер потока: 1

номер потока: 2

Ответ: 76

$ ./a.out 2

Please enter the matrix dimension: 4

1 2 3 1

4 5 6 2

7 8 10 6

8 3 9 7

номер потока: 1

номер потока: 2

Ответ: 76

$ ./a.out 4

Please enter the matrix dimension: 4

19 2 3 1

4 5 6 2

7 8 10 6

8 3 9 18

номер потока: 1

номер потока: 2

номер потока: 3

номер потока: 4

Ответ: -713

$ ./a.out 2

Please enter the matrix dimension: 5

19 2 3 1 1

4 5 6 2 2

7 8 10 6 3

8 3 9 18 4

9 8 7 6 5

номер потока: 1

номер потока: 2

Ответ: -4370

$ ./a.out 7

Please enter the matrix dimension: 5

19 2 3 1 1

4 5 6 2 2

7 8 10 6 3

8 3 9 18 4

9 8 7 6 5

номер потока: 1

номер потока: 3

номер потока: 2

номер потока: 4

номер потока: 5

Ответ: -4370

$ ./a.out 7

Please enter the matrix dimension: 7

19 2 3 1 1 2 9

4 5 6 2 2 10 8

7 8 10 6 3 3 1

8 3 9 18 4 2 2

9 8 7 6 5 -1 3

0 1 11 23 1 2 4

1 24 2 7 9 7 64

номер потока: 1

номер потока: 2

номер потока: 7

номер потока: 3

номер потока: 4

номер потока: 5

номер потока: 6

Ответ: 5179890

$ ./a.out 8

Please enter the matrix dimension: 7

19 2 3 1 1 2 9

4 5 6 2 2 10 8

7 8 10 6 3 3 1

8 3 9 18 4 2 2

9 8 7 6 5 -1 3

0 1 11 23 1 2 4

1 24 2 7 9 7 64

номер потока: 1

номер потока: 2

номер потока: 3

номер потока: 5

номер потока: 4

номер потока: 7

номер потока: 6

Ответ: 5179890

$ ./a.out 4

Please enter the matrix dimension: 7

19 2 3 1 1 2 9

4 5 6 2 2 10 8

7 8 10 6 3 3 1

8 3 9 18 4 2 2

9 8 7 6 5 -1 3

0 1 11 23 1 2 4

1 24 2 7 9 7 64

номер потока: 1

номер потока: 3

номер потока: 2

номер потока: 4

Ответ: 5179890

**Таблица зависимости времени выполнения от исходных данных и количества потоков:**

| **Порядок матрицы** | **Число потоков** | **Время исполнения (с)** | **Ускорение** | **Эффективность** |
| --- | --- | --- | --- | --- |
| 11 | 1 | 5.0220 | 1 | 1 |
| 11 | 2 | 3.1640 | 1.59 | 0.795 |
| 11 | 4 | 1.6830 | 2.98 | 0.745 |
| 11 | 6 | 1.2810 | 3.92 | 0.653 |
| 11 | 11 | 1.1290 | 4.45 | 0.404 |
| 9 | 1 | 0.0620 | 1 | 1 |
| 9 | 9 | 0.0180 | 3.44 | 0.382 |
| 7 | 1 | 0.0060 | 1 | 1 |
| 7 | 7 | 0.0010 | 6 | 0.85 |

**Strace:**

$ strace -f ./a.out 2

execve("./a.out", ["./a.out", "2"], 0x7ffd268697c0 /\* 36 vars \*/) = 0

brk(NULL) = 0x559786f32000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffd8c3b0c00) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fb6e42c4000

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=18151, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 18151, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fb6e42bf000

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\0P\237\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 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 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

mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fb6e4097000

mmap(0x7fb6e40bf000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fb6e40bf000

mmap(0x7fb6e4254000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fb6e4254000

mmap(0x7fb6e42ac000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7fb6e42ac000

mmap(0x7fb6e42b2000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fb6e42b2000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fb6e4094000

arch\_prctl(ARCH\_SET\_FS, 0x7fb6e4094740) = 0

set\_tid\_address(0x7fb6e4094a10) = 3863

set\_robust\_list(0x7fb6e4094a20, 24) = 0

rseq(0x7fb6e40950e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7fb6e42ac000, 16384, PROT\_READ) = 0

mprotect(0x559785386000, 4096, PROT\_READ) = 0

mprotect(0x7fb6e42fe000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7fb6e42bf000, 18151) = 0

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x4), ...}, AT\_EMPTY\_PATH) = 0

getrandom("\xeb\xfe\x18\x76\xab\x28\xa4\x68", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x559786f32000

brk(0x559786f53000) = 0x559786f53000

newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x4), ...}, AT\_EMPTY\_PATH) = 0

write(1, "Please enter the matrix dimensio"..., 35Please enter the matrix dimension: ) = 35

read(0, 4

"4\n", 1024) = 2

read(0, 1 2 3 4

"1 2 3 4\n", 1024) = 8

read(0, 67 34 0 9

"67 34 0 9\n", 1024) = 10

read(0, 12 3 4 1

"12 3 4 1\n", 1024) = 9

read(0, 2 45 60

"2 45 60\n", 1024) = 8

read(0, 89

"89\n", 1024) = 3

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7fb6e4128870, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7fb6e40d9520}, NULL, 8) = 0

rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7fb6e3893000

mprotect(0x7fb6e3894000, 8388608, PROT\_READ|PROT\_WRITE) = 0

rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fb6e4093910, parent\_tid=0x7fb6e4093910, exit\_signal=0, stack=0x7fb6e3893000, stack\_size=0x7fff00, tls=0x7fb6e4093640}strace: Process 3992 attached

=> {parent\_tid=[3992]}, 88) = 3992

[pid 3863] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 3992] rseq(0x7fb6e4093fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 3863] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 3992] <... rseq resumed>) = 0

[pid 3863] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 3992] set\_robust\_list(0x7fb6e4093920, 24 <unfinished ...>

[pid 3863] <... mmap resumed>) = 0x7fb6e3092000

[pid 3992] <... set\_robust\_list resumed>) = 0

[pid 3863] mprotect(0x7fb6e3093000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 3992] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 3863] <... mprotect resumed>) = 0

[pid 3992] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 3863] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 3992] write(1, "\320\275\320\276\320\274\320\265\321\200 \320\277\320\276\321\202\320\276\320\272\320\260: 1 \n", 28номер потока: 1

<unfinished ...>

[pid 3863] <... rt\_sigprocmask resumed>[], 8) = 0

[pid 3992] <... write resumed>) = 28

[pid 3863] clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fb6e3892910, parent\_tid=0x7fb6e3892910, exit\_signal=0, stack=0x7fb6e3092000, stack\_size=0x7fff00, tls=0x7fb6e3892640} <unfinished ...>

[pid 3992] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXECstrace: Process 3993 attached

) = 3

[pid 3863] <... clone3 resumed> => {parent\_tid=[3993]}, 88) = 3993

[pid 3993] rseq(0x7fb6e3892fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 3863] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 3992] newfstatat(3, "", <unfinished ...>

[pid 3863] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 3993] <... rseq resumed>) = 0

[pid 3863] futex(0x7fb6e4093910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 3992, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 3992] <... newfstatat resumed>{st\_mode=S\_IFREG|0644, st\_size=18151, ...}, AT\_EMPTY\_PATH) = 0

[pid 3993] set\_robust\_list(0x7fb6e3892920, 24 <unfinished ...>

[pid 3992] mmap(NULL, 18151, PROT\_READ, MAP\_PRIVATE, 3, 0 <unfinished ...>

[pid 3993] <... set\_robust\_list resumed>) = 0

[pid 3992] <... mmap resumed>) = 0x7fb6e42bf000

[pid 3993] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 3992] close(3 <unfinished ...>

[pid 3993] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 3992] <... close resumed>) = 0

[pid 3993] write(1, "\320\275\320\276\320\274\320\265\321\200 \320\277\320\276\321\202\320\276\320\272\320\260: 2 \n", 28 <unfinished ...>

[pid 3992] mmap(NULL, 134217728, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0номер потока: 2

<unfinished ...>

[pid 3993] <... write resumed>) = 28

[pid 3992] <... mmap resumed>) = 0x7fb6db092000

[pid 3993] futex(0x7fb6e4300a48, FUTEX\_WAIT\_PRIVATE, 2, NULL <unfinished ...>

[pid 3992] munmap(0x7fb6db092000, 16179200) = 0

[pid 3992] munmap(0x7fb6e0000000, 50929664) = 0

[pid 3992] mprotect(0x7fb6dc000000, 135168, PROT\_READ|PROT\_WRITE) = 0

[pid 3992] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

[pid 3992] read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

[pid 3992] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH) = 0

[pid 3992] mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fb6e3072000

[pid 3992] mmap(0x7fb6e3075000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7fb6e3075000

[pid 3992] mmap(0x7fb6e308c000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7fb6e308c000

[pid 3992] mmap(0x7fb6e3090000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7fb6e3090000

[pid 3992] close(3) = 0

[pid 3992] mprotect(0x7fb6e3090000, 4096, PROT\_READ) = 0

[pid 3992] munmap(0x7fb6e42bf000, 18151) = 0

[pid 3992] futex(0x7fb6e4300a48, FUTEX\_WAKE\_PRIVATE, 1 <unfinished ...>

[pid 3993] <... futex resumed>) = 0

[pid 3992] <... futex resumed>) = 1

[pid 3993] futex(0x7fb6e4300a48, FUTEX\_WAKE\_PRIVATE, 1 <unfinished ...>

[pid 3992] futex(0x7fb6e3091210, FUTEX\_WAKE\_PRIVATE, 2147483647 <unfinished ...>

[pid 3993] <... futex resumed>) = 0

[pid 3992] <... futex resumed>) = 0

[pid 3993] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 3992] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 3993] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 3992] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 3993] madvise(0x7fb6e3092000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 3992] madvise(0x7fb6e3893000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 3993] <... madvise resumed>) = 0

[pid 3992] <... madvise resumed>) = 0

[pid 3993] exit(0 <unfinished ...>

[pid 3992] exit(0 <unfinished ...>

[pid 3993] <... exit resumed>) = ?

[pid 3992] <... exit resumed>) = ?

[pid 3993] +++ exited with 0 +++

[pid 3863] <... futex resumed>) = 0

[pid 3992] +++ exited with 0 +++

write(1, "\320\236\321\202\320\262\320\265\321\202: 4724 \n", 18Ответ: 4724

) = 18

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

У меня получилось реализовать программу с использованием многопоточности и доказать, что таким я сокращаю время работы своего кода. Стало очевидно, что в задачах с большими данными многопоточность - незаменимый инструмент, который может сократить время выполнения в несколько раз. Было очень интересно продумывать работу каждого потока и организовывать логику их выполнения, следя за тем, чтобы не происходили “Data Race”. Уверен, эти знания пригодятся мне в будущем.