Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу
«Операционные системы»

Управление потоками и синхронизация

Группа: М80-207Б-20

Студент: Чекменев В.А.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата:

Москва, 2021

Постановка задачи

Составить и отладить программу на языке C++, обрабатывающую данные в многопоточном режиме. Использовать стандартные средства создания потоков операционной системы Unix. Предусмотреть возможность ограничить максимальное количество потоков, используемых в программе.

Вариант задания: 9.

9. Рассчитать детерминант матрицы

Общие сведения о программе

Программа состоит из файла **fin_version.cpp**, в котором реализован многопоточный алгоритм наложения матриц наращивания и эрозии на матрицу вещественных чисел.

Используются следующие системные вызовы

1. **thread() –** создает новый поток выполнения в программе.
2. **thread.join()** – для синхронного окончания выаполнения функций в потоках

Общий метод и алгоритм решения

Воспользуюсь алгоритмом LU — разложения

U = A, где A матрица, детерминант которой надо найти

- сделать цикл k = [1, n-1], где n — размер матрицы
- внутри сделать два двойных цикла и для каждого из них сохдать отдельный поток
- один для вычисление нижней треугольной матрицы

```
L[j][i]=U[j][i]/U[i][i]; // j – строки, i - столбцы
```

- другой для вычисления верхней треугольной

```
U[i][j]=U[i][j]-L[i][k-1]*U[k-1][j];
```

В конце алгоритма матрица L будет приведена в нижнему треугольному виду, а мтарица U будет приведена к верх треуг виду

Код программы

main.cpp:

```
#include <iostream>
#include <thread>
#include <mutex>
#include <unistd.h>
#include <vector>

using namespace std;

#define MAX_NUMBER_OF_THREADS 100

mutex mtx;
int lower_counter = 0, upper_counter = 0;

void show(vector<vector<double>> A, int n);
void L_loop(int &k, int &n, vector<vector<double>> &L,
        vector<vector<double>> &U);
void U_loop(int &k, int &n, vector<vector<double>> &L,
        vector<vector<double>> &U);
void LU(vector<vector<double>> A, vector<vector<double>> &L,
        vector<vector<double>> &U, int n);
double determinant(vector<vector<double>> &L,
        vector<vector<double>> &U);
void mult(vector<vector<double>> A, vector<vector<double>> B,
        vector<vector<double>> &R, int n);


void L_loop(int &k, int &n, vector<vector<double>> &L,
                    vector<vector<double>> &U) {
   mtx.lock();
   lower_counter++;
        if (lower_counter >= MAX_NUMBER_OF_THREADS) {
                cout << "Превышено максимальное кол-во потоков на один процесс\n";
                return;
        }
   cout << "In " << lower_counter
       << " loop for lower matrix: thread id = "
       << this_thread::get_id() << endl;
        for(int i = k-1; i < n; i++)
                for(int j = i; j < n; j++)
                        L[j][i]=U[j][i]/U[i][i];
   mtx.unlock();
}
```

```
void U_loop(int &k, int &n, vector<vector<double>> &L,
                    vector<vector<double>> &U) {
   mtx.lock();
   upper_counter++;
        if (lower_counter >= MAX_NUMBER_OF_THREADS) {
                cout << "Превышено максимальное кол-во потоков на один процесс\n";
                return;
        }
   cout << "In " << upper_counter
       << " loop for upper matrix: thread id = "
       << this_thread::get_id() << endl;
        for(int i = k; i < n; i++)
                for(int j = k-1; j < n; j++)
                        U[i][j]=U[i][j]-L[i][k-1]*U[k-1][j];
   mtx.unlock();

}

void LU(vector<vector<double>> A, vector<vector<double>> &L,
                vector<vector<double>> &U, int n) {
        U=A;

        for(int k = 1; k < n; k++) {
                // j - строки
                // i - столбцы
                // делим каждое значение в столбцах матрицы L
                // на значение диагонального элемента матрицы U,
                // который находится в том же столбце
      thread L_loop_th(L_loop, ref(k), ref(n), ref(L), ref(U));
                // j - столбцы
                // i - строки
                // вычитаем из строк i >= 1 элемент матрицы L,
                // строящий на той же позиции, домноженный на
                // диагональный элемент матрицы U того же столбца
                thread U_loop_th(U_loop, ref(k), ref(n), ref(L), ref(U));
                L_loop_th.join();
                U_loop_th.join();
        }

}

void mult(vector<vector<double>> A, vector<vector<double>> B,
                        vector<vector<double>> &R, int n)
{
        for(int i = 0; i < n; i++)
                for(int j = 0; j < n; j++)
                        for(int k = 0; k < n; k++)
```

```
                                R[i][j] += A[i][k] * B[k][j];
}

void show(vector<vector<double>> A, int n)
{
        for(int i = 0; i < n; i++) {
                for(int j = 0; j < n; j++)
                        cout <<" "<< A[i][j] << " ";
                cout << endl;
        }
}

double determinant(vector<vector<double>> &L, vector<vector<double>> &U)
{
   double det = 1;
   for (int i = 0; i < L.size(); ++i)
      det *= L[i][i]*U[i][i];
   return det;
}

int main()
{
        vector<vector<double>> A, L, U, R;

   int n;
   cout << "Enter size of the matrix: ";
   cin >> n;

   cout << "enter values:\n";
   for(size_t i = 0; i < n; ++i) {
      vector<double> temp, temp_for_A;
      for(size_t j = 0; j < n; ++j) {
         double t;
         cin >> t;
         temp_for_A.push_back(t);
         temp.push_back(0);
      }
      A.push_back(temp_for_A);
      L.push_back(temp);
      U.push_back(temp);
      R.push_back(temp);
   }

        LU(A,L,U,n);
        cout << "Fisrt matrix" << endl;
        show(A,n);
        cout << "U matrix" << endl;
        show(U,n);
```

```
        cout << "L matrix" << endl;
        show(L,n);
        mult(L,U,R,n);
        cout << "L*U matrix" << endl;
        show(R,n);
    cout << "determinant = " << determinant(L, U) << endl;
        return 0;
}
```

Демонстрация работы программы в терминале

Пример 1:

[suraba04@asusx512fl code_data]$ g++ -o fin_version -pthread fin_version.cpp

[suraba04@asusx512fl code_data]$ ./fin_version

Enter size of the matrix: 4

enter values:

1 2 3 4

3 4 2 4

5 8 0 6

-1 3 0 6

In 1 loop for lower matrix: thread id = 140351242622528

In 1 loop for upper matrix: thread id = 140351234229824

In 2 loop for lower matrix: thread id = 140351234229824

In 2 loop for upper matrix: thread id = 140351242622528

In 3 loop for lower matrix: thread id = 140351242622528

In 3 loop for upper matrix: thread id = 140351234229824

Fisrt matrix

1 2 3 4

3 4 2 4

5 8 0 6

-1 3 0 6

U matrix

1 2 3 4

0 -2 -7 -8

0 0 -8 -6

0 0 0 0.875

L matrix

1 0 0 0

3 1 0 0

5 1 1 0

-1 -2.5 1.8125 1

L*U matrix

1 2 3 4

3 4 2 4

5 8 0 6

-1 3 0 6

determinant = 14

[suraba04@asusx512fl code_data]$

Пример 2

[suraba04@asusx512fl code_data]$ ./fin_version

Enter size of the matrix: 3

enter values:

1 2 3

3 4 2

5 8 0

In 1 loop for lower matrix: thread id = 139705086703168

In 1 loop for upper matrix: thread id = 139705078310464

In 2 loop for lower matrix: thread id = 139705078310464

In 2 loop for upper matrix: thread id = 139705086703168

Fisrt matrix

1 2 3

3 4 2

5 8 0

U matrix

1 2 3

0 -2 -7

0 0 -8

L matrix

1 0 0

3 1 0

5 1 1

L*U matrix

1 2 3

3 4 2

5 8 0

determinant = 16

Использование утилиты strace

```
[suraba04@asusx512fl code_data]$ strace ./fin_version
execve("./fin_version", ["./fin_version"], 0x7fff10a9ff10 /* 72 vars */) = 0
brk(NULL)                               = 0x55cf22907000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fff136138d0) = -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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=212244, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 212244, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f759c747000
close(3)                                = 0
```

```
openat(AT_FDCWD, "/usr/lib/libstdc++.so.6", O_RDONLY|O_CLOEXEC)
= 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0@\220\t\
0\0\0\0\0"..., 832) = 832

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\212\260\345pT\
335\35\313\246\201\362\27\1j\374j"..., 36, 800) = 36

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=17969672, ...},
AT_EMPTY_PATH) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7f759c745000

mmap(NULL, 2185280, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0)
= 0x7f759c52f000

mmap(0x7f759c5c8000, 1048576, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x99000) = 0x7f759c5c8000

mmap(0x7f759c6c8000, 442368, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x199000) = 0x7f759c6c8000

mmap(0x7f759c734000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x204000) = 0x7f759c734000

mmap(0x7f759c742000, 10304, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f759c742000

close(3)                                = 0

openat(AT_FDCWD, "/usr/lib/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\
0\1\0\0\0\260\363\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1323472, ...},
AT_EMPTY_PATH) = 0

mmap(NULL, 1323032, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0)
= 0x7f759c3eb000

mprotect(0x7f759c3fa000, 1257472, PROT_NONE) = 0

mmap(0x7f759c3fa000, 630784, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0xf000) = 0x7f759c3fa000

mmap(0x7f759c494000, 622592, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xa9000) = 0x7f759c494000
```

```
mmap(0x7f759c52d000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x141000) = 0x7f759c52d000

close(3)                                = 0

openat(AT_FDCWD, "/usr/lib/libgcc_s.so.1", 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
0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=475944, ...},
AT_EMPTY_PATH) = 0

mmap(NULL, 107240, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f759c3d0000

mprotect(0x7f759c3d3000, 90112, PROT_NONE) = 0

mmap(0x7f759c3d3000, 73728, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x7f759c3d3000

mmap(0x7f759c3e5000, 12288, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x15000) = 0x7f759c3e5000

mmap(0x7f759c3e9000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f759c3e9000

close(3)                                = 0

openat(AT_FDCWD, "/usr/lib/libpthread.so.0", O_RDONLY|O_CLOEXEC)
= 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\
0\1\0\0\0\300\200\0\0\0\0\0\0"..., 832) = 832

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"..., 80, 792) = 80

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\7\310\371[O2Q\
320\205P!z\330\241\363\20"..., 68, 872) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=154040, ...},
AT_EMPTY_PATH) = 0

mmap(NULL, 131472, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f759c3af000

mprotect(0x7f759c3b6000, 81920, PROT_NONE) = 0
```

```
mmap(0x7f759c3b6000, 61440, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f759c3b6000

mmap(0x7f759c3c5000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x16000) = 0x7f759c3c5000

mmap(0x7f759c3ca000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000) = 0x7f759c3ca000

mmap(0x7f759c3cc000, 12688, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f759c3cc000

close(3)                                = 0

openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`|\
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"..., 80, 848) = 80

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0K@g7\5w\
10\300\344\306B4Zp<G"..., 68, 928) = 68

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2150424, ...},
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, 1880536, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0)
= 0x7f759c1e3000

mmap(0x7f759c209000, 1355776, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7f759c209000

mmap(0x7f759c354000, 311296, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x171000) = 0x7f759c354000

mmap(0x7f759c3a0000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x1bc000) = 0x7f759c3a0000

mmap(0x7f759c3a6000, 33240, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f759c3a6000

close(3)                                = 0
```

```
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7f759c1e1000

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7f759c1de000

arch_prctl(ARCH_SET_FS, 0x7f759c1de740) = 0

mprotect(0x7f759c3a0000, 12288, PROT_READ) = 0

mprotect(0x7f759c3ca000, 4096, PROT_READ) = 0

mprotect(0x7f759c3e9000, 4096, PROT_READ) = 0

mprotect(0x7f759c52d000, 4096, PROT_READ) = 0

mprotect(0x7f759c734000, 53248, PROT_READ) = 0

mprotect(0x55cf21546000, 4096, PROT_READ) = 0

mprotect(0x7f759c7a9000, 8192, PROT_READ) = 0

munmap(0x7f759c747000, 212244)          = 0

set_tid_address(0x7f759c1dea10)         = 36989

set_robust_list(0x7f759c1dea20, 24)     = 0

rt_sigaction(SIGRTMIN, {sa_handler=0x7f759c3b6b70, sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f759c3c2870},
NULL, 8) = 0

rt_sigaction(SIGRT_1, {sa_handler=0x7f759c3b6c10, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO,
sa_restorer=0x7f759c3c2870}, 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

brk(NULL)                               = 0x55cf22907000

brk(0x55cf22928000)                     = 0x55cf22928000

futex(0x7f759c7426bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0

futex(0x7f759c7426c8, FUTEX_WAKE_PRIVATE, 2147483647) = 0

newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88,
0x4), ...}, AT_EMPTY_PATH) = 0
```

```
write(1, "Enter size of the matrix: ", 26Enter size of the
matrix: ) = 26

newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88,
0x4), ...}, AT_EMPTY_PATH) = 0

read(0, 4

"4\n", 1024)                    = 2

write(1, "enter values:\n", 14enter values:

)         = 14

read(0,  1  2  3  4

 3  4  2  4

 5  8  0  6

 -1  3  0  " 1  2  3  4 \n", 1024)         = 13

read(0, " 3  4  2  4 \n", 1024)         = 13

read(0, " 5  8  0  6 \n", 1024)         = 13

read(0, 6

" -1  3  0  6\n", 1024)         = 13

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|
MAP_STACK, -1, 0) = 0x7f759b9dd000

mprotect(0x7f759b9de000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone(child_stack=0x7f759c1dcef0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDIn 1 loop
for lower matrix: thread id = 140143107102272

, parent_tid=[37096], tls=0x7f759c1dd640,
child_tidptr=0x7f759c1dd910) = 37096

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|
MAP_STACK, -1, 0) = 0x7f759b1dc000

mprotect(0x7f759b1dd000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
```

```
clone(child_stack=0x7f759b9dbef0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDIn 1 loop
for upper matrix: thread id = 140143098709568

, parent_tid=[0], tls=0x7f759b9dc640,
child_tidptr=0x7f759b9dc910) = 37097

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone(child_stack=0x7f759b9dbef0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDIn 2 loop
for lower matrix: thread id = 140143098709568

, parent_tid=[0], tls=0x7f759b9dc640,
child_tidptr=0x7f759b9dc910) = 37098

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone(child_stack=0x7f759c1dcef0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
parent_tid=[37099], tls=0x7f759c1dd640,
child_tidptr=0x7f759c1dd910) = 37099

In 2 loop for upper matrix: thread id = 140143107102272

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone(child_stack=0x7f759c1dcef0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDIn 3 loop
for lower matrix: thread id = 140143107102272

, parent_tid=[37100], tls=0x7f759c1dd640,
child_tidptr=0x7f759c1dd910) = 37100

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone(child_stack=0x7f759b9dbef0, flags=CLONE_VM|CLONE_FS|
CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
```

```
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTIDIn 3 loop
for upper matrix: thread id = 140143098709568

, parent_tid=[37101], tls=0x7f759b9dc640,
child_tidptr=0x7f759b9dc910) = 37101

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

write(1, "Fisrt matrix\n", 13Fisrt matrix

)          = 13

write(1, " 1  2  3  4 \n", 13 1  2  3  4

)          = 13

write(1, " 3  4  2  4 \n", 13 3  4  2  4

)          = 13

write(1, " 5  8  0  6 \n", 13 5  8  0  6

)          = 13

write(1, " -1  3  0  6 \n", 14 -1  3  0  6

)         = 14

write(1, "U matrix\n", 9U matrix

)               = 9

write(1, " 1  2  3  4 \n", 13 1  2  3  4

)          = 13

write(1, " 0  -2  -7  -8 \n", 16 0  -2  -7  -8

)       = 16

write(1, " 0  0  -8  -6 \n", 15 0  0  -8  -6

)        = 15

write(1, " 0  0  0  0.875 \n", 17 0  0  0  0.875

)      = 17

write(1, "L matrix\n", 9L matrix

)               = 9

write(1, " 1  0  0  0 \n", 13 1  0  0  0

)          = 13
```

```
write(1, " 3  1  0  0 \n", 13 3  1  0  0
)          = 13
write(1, " 5  1  1  0 \n", 13 5  1  1  0
)          = 13
write(1, " -1  -2.5  1.8125  1 \n", 22 -1  -2.5  1.8125  1
) = 22
write(1, "L*U matrix\n", 11L*U matrix
)            = 11
write(1, " 1  2  3  4 \n", 13 1  2  3  4
)          = 13
write(1, " 3  4  2  4 \n", 13 3  4  2  4
)          = 13
write(1, " 5  8  0  6 \n", 13 5  8  0  6
)          = 13
write(1, " -1  3  0  6 \n", 14 -1  3  0  6
)         = 14
write(1, "determinant = 14\n", 17determinant = 14
)      = 17
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal
seek)
exit_group(0)                           = ?
+++ exited with 0 +++
[suraba04@asusx512fl code_data]$
```

Сравнение эффективности работы:

Рассмотрим выполнение программы на больших матрицах 1000x1000

Напишем программу на пайтон, чтобы сгенерировать 1000000 чисел от 1 до 10

```
# generate random integer values
from random import seed
from random import randint
# seed random number generator
seed(1)
# generate some integers
for _ in range(1000000):
	value = randint(1, 10)
	print(value, end=' ')
```

не буду приводить код программы без использования многопоточности, так как это почти то же самое, что и наша программа

применим утилиту time для сравнения времени работы

[suraba04@asusx512fl code_data]$ time ./one_thread 100

real 0m0,015s

user 0m0,013s

sys 0m0,000s

[suraba04@asusx512fl code_data]$ time ./threaded 100

real 0m0,025s

user 0m0,021s

sys 0m0,000s

ускорение эффективности: 0,6 < 1 => эффективность многопоточной программмы ниже

[suraba04@asusx512fl code_data]$ time ./one_thread 10

real 0m0,002s

user 0m0,002s

sys 0m0,000s

[suraba04@asusx512fl code_data]$ time ./threaded 10

real 0m0,003s

user 0m0,003s

sys 0m0,000s

ускорение эффективности: 0,7 < 1 => эффективность многопоточной программмы ниже

[suraba04@asusx512fl code_data]$ time ./one_thread 1000

real 0m9,043s

user 0m8,985s

sys 0m0,014s

[suraba04@asusx512fl code_data]$ time ./threaded 1000

real 0m9,047s

user 0m9,011s

sys 0m0,085s

ускорение эффективности: 1 == 1 => эффективность многопоточной программмы равна эффективности однопоточной программы

[suraba04@asusx512fl code_data]$ time ./one_thread 10000

real 0m25,990s

user 0m15,084s

sys 0m7,601s

[suraba04@asusx512fl code_data]$ time ./threaded 10000

real 0m24,538s

user 0m14,075s

sys 0m3,440s

ускорение эффективности: 1,06 > 1 => эффективность многопоточной программмы выше

Вывод

В данной ЛР я познакомился с многопоточностью и синхронизацией потоков (с помощью двоичного семафора — мьютекса) в языке c++, в операционной системе линукс. Мне нужно было написать программу для вычисления детерминанта матрицы. Я погуглил и решил выбрать алгоритм ЛУ разложения, потому что его сложность кубическая в отличие от алгоритма миноров, сложность которого факториал. В данной лр, чтобы использовать все плюсы многопоточности, мне нужно было рассматривать большие матрицы, которые плохо считаются с помощью алгоритма миноров. Однако меня не удивил тот факт, что многопоточная программа работает дольше однопоточной, так как потоки нужны для распараллеливания сложных вычислений, а у меня нужно всего лишь n раз поделить одно число на другое. Поэтому программа и работала бы также не будь использованы мной сис вызовы для генерации и синхронизации потоков, однако системных вызовов получается много => увеличивается время выполнения программы. Я смотрел в интернете алогритмы для вычислени определителя матрицы в многопоточном режиме, однако код, написанный там, мало похож на мой.