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: М8О-213Б-23

Студент: Барменков А.С.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 01.11.24

Москва, 2024

Постановка задачи

Вариант 10.

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, 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 входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

Решить систему линейных уравнений методом Гаусса

Общий метод и алгоритм решения

Использованные системные вызовы:

- read — это системный вызов, который используется для считывания данных из файлового дескриптора.
- write — это системный вызов, который используется для записи данных в файловый дескриптор.
- pthread_create — создает новый поток выполнения.
- pthread_join — заставляет вызывающий поток ожидать завершения потока, идентификатор которого передан в аргументе thread
- pthread_mutex_init() — инициализируем мьютекс
- pthread_mutex_lock() – блокируем мьютекс
- pthread_mutex_unlock() – разблокируем мьютекс
- pthread_cond_wait освобождает мьютекс и переводит поток в состояние ожидания на условной переменной cond
- pthread_cond_destroy освобождает ресурсы, связанные с условной переменной, когда она больше не нужна.
- pthread_mutex_destroy освобождает ресурсы, связанные с мьютексом, после того как он больше не нужен.

Эта программа выполняет параллельные вычисления для решения системы линейных уравнений методом Гаусса с использованием многопоточности и примитивов синхронизации.
Программа определяет максимальное количество переменных системы уравнений (MAX_N) и количество потоков (numThreads), которые пользователь может настроить.
Используются примитивы синхронизации: мьютекс (pthread_mutex_t mutex) и условная переменная (pthread_cond_t cond). Эти примитивы нужны для координации работы потоков.
Пользователь вводит количество переменных n и количество потоков numThreads.
Затем вводятся коэффициенты расширенной матрицы системы уравнений. Ввод осуществляется построчно, в виде чисел с плавающей точкой.
Программа создает numThreads потоков с помощью pthread_create. Каждый поток выполняет функцию GaussianElimination, которая выполняет прямой ход метода Гаусса.
Потоки распределяют работу по вычислению системы уравнений, используя их идентификаторы threadId, чтобы определять, какую часть матрицы обрабатывать.
В функции GaussianElimination каждый поток:
Нормализует строку матрицы, чтобы главный элемент стал равным 1. Только один поток обрабатывает строку за раз, используя мьютекс.
Использует синхронизацию (WaitForSync), чтобы все потоки завершили обработку строки, прежде чем продолжить.
Убирает элементы ниже главного элемента в остальных строках, распределяя эту работу между потоками.
После завершения прямого хода метод Гаусса, программа выполняет обратную подстановку в функции BackSubstitution, чтобы найти решения уравнений.
Программа выводит результаты решения системы уравнений (значения переменных) с использованием функции WriteDouble.
Мьютекс используется для защиты критических секций, чтобы не допустить одновременного изменения данных несколькими потоками.
Условная переменная (pthread_cond_t cond) и счетчик counter используются для синхронизации работы потоков, чтобы все они синхронно переходили к следующему этапу вычислений.

Код программы

main.c

```
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>

#define MAX_N 10
#define MAX_LINE_LENGTH 100

double matrix[MAX_N][MAX_N + 1];
int n, numThreads;
pthread_mutex_t mutex;
pthread_cond_t cond;
int counter = 0;

void IntToString(int value, char* buffer) {
    int i = 0;
    int isNegative = (value < 0);

    if (isNegative) {
        value = -value;
    }

    do {
        buffer[i++] = (value % 10) + '0';
        value /= 10;
    } while (value);

    if (isNegative) {
        buffer[i++] = '-';
    }
```

```
    buffer[i] = '\0';

    // Разворачиваем строку
    for (int j = 0; j < i / 2; j++) {
        char temp = buffer[j];
        buffer[j] = buffer[i - j - 1];
        buffer[i - j - 1] = temp;
    }
}

void DoubleToString(double value, char* buffer) {
    int intPart = (int)value;
    double fracPart = value - intPart;
    IntToString(intPart, buffer);

    int len = strlen(buffer);
    buffer[len++] = '.';

    // Обрабатываем дробную часть
    for (int i = 0; i < 6; i++) { // 6 знаков после запятой
        fracPart *= 10;
        int digit = (int)fracPart;
        buffer[len++] = digit + '0';
        fracPart -= digit;
    }
    buffer[len] = '\0';
}

void WriteString(const char* str) {
    write(1, str, strlen(str)); // 1 - стандартный вывод (stdout)
}

void WriteDouble(double value) {
    char buffer[50];
    DoubleToString(value, buffer);
    WriteString(buffer);
}

void WriteInt(int value) {
    char buffer[12];
    IntToString(value, buffer);
    WriteString(buffer);
}

void ReadLine(char* buffer, int size) {
    read(0, buffer, size);
    buffer[strcspn(buffer, "\n")] = 0; // Удаляем символ новой строки
}

void ReadDoubles(double* values, int count) {
    char line[MAX_LINE_LENGTH];
    ReadLine(line, sizeof(line));

    char* token = strtok(line, " ");
    for (int i = 0; i < count; i++) {
        if (token != NULL) {
            values[i] = atof(token);
            token = strtok(NULL, " ");
        }
    }
}

void WaitForSync() {
    pthread_mutex_lock(&mutex);
    counter++;
    if (counter < numThreads) {
        pthread_cond_wait(&cond, &mutex);
    } else {
        counter = 0;
```

```
            pthread_cond_broadcast(&cond);
        }
        pthread_mutex_unlock(&mutex);
    }

    void* GaussianElimination(void* arg) {
        int threadId = *(int*)arg;
        for (int k = 0; k < n; k++) {
            if (k % numThreads == threadId) {
                pthread_mutex_lock(&mutex);
                double pivot = matrix[k][k];
                for (int j = k; j <= n; j++) {
                    matrix[k][j] /= pivot;
                }
                pthread_mutex_unlock(&mutex);
            }

            WaitForSync();

            for (int i = k + 1; i < n; i++) {
                if (i % numThreads == threadId) {
                    pthread_mutex_lock(&mutex);
                    double factor = matrix[i][k];
                    for (int j = k; j <= n; j++) {
                        matrix[i][j] -= factor * matrix[k][j];
                    }
                    pthread_mutex_unlock(&mutex);
                }
            }

            WaitForSync();
        }
        return NULL;
    }

    void BackSubstitution(double* result) {
        for (int i = n - 1; i >= 0; i--) {
            result[i] = matrix[i][n];
            for (int j = i + 1; j < n; j++) {
                result[i] -= matrix[i][j] * result[j];
            }
        }
    }

    int main() {
        char inputLine[MAX_LINE_LENGTH];

        WriteString("Введите количество переменных (n <= 10): ");
        ReadLine(inputLine, sizeof(inputLine)); // Считываем строку
        n = atoi(inputLine); // Получаем количество переменных

        WriteString("Введите количество потоков: ");
        ReadLine(inputLine, sizeof(inputLine)); // Считываем строку
        numThreads = atoi(inputLine); // Получаем количество потоков

        if (numThreads <= 0 || numThreads > MAX_N) {
            WriteString("Неверное количество потоков. Установлено значение по умолчанию: 4\n");
            numThreads = 4;
        }

        WriteString("Введите элементы расширенной матрицы: \n");
        for (int i = 0; i < n; i++) {
            ReadDoubles(matrix[i], n + 1); // Считываем всю строку сразу
        }

        pthread_t threads[MAX_N];
        int threadIds[MAX_N];
        pthread_mutex_init(&mutex, NULL);
        pthread_cond_init(&cond, NULL);
```

```
    for (int i = 0; i < numThreads; i++) {
        threadIds[i] = i;
        pthread_create(&threads[i], NULL, GaussianElimination, &threadIds[i]);
    }

    for (int i = 0; i < numThreads; i++) {
        pthread_join(threads[i], NULL);
    }

    double result[MAX_N];
    BackSubstitution(result);

    WriteString("Решение системы:\n");
    for (int i = 0; i < n; i++) {
        WriteString("x");
        WriteInt(i + 1);
        WriteString(" = ");
        WriteDouble(result[i]);
        WriteString("\n");
    }

    pthread_mutex_destroy(&mutex);
    pthread_cond_destroy(&cond);

    return 0;
}
```

Протокол работы программы

Тестирование:

```
Введите количество переменных (n <= 10): 3
Введите количество потоков: 3
Введите элементы расширенной матрицы:
2 -3 1 1
4 3 -3 3
1 7 -4 3
[New Thread 0x7ffff7a00640 (LWP 7480)]
[New Thread 0x7ffff7000640 (LWP 7481)]
[New Thread 0x7ffff6600640 (LWP 7482)]
[Thread 0x7ffff7000640 (LWP 7481) exited]
[Thread 0x7ffff7a00640 (LWP 7480) exited]
[Thread 0x7ffff6600640 (LWP 7482) exited]
Решение системы:
x1 = 2.999999
x2 = 3.999999
x3 = 6.999999
[Inferior 1 (process 7477) exited normally]
```

```
Введите количество переменных (n <= 10): 3
Введите количество потоков: 3
Введите элементы расширенной матрицы:
1 1 1 9
2 -3 4 13
3 4 5 40
[New Thread 0x7ffff7a00640 (LWP 7466)]
[New Thread 0x7ffff7000640 (LWP 7467)]
[New Thread 0x7ffff6600640 (LWP 7468)]
Решение системы:
x1 = 1.000000
x2 = 3.000000
x3 = 5.000000
[Thread 0x7ffff6600640 (LWP 7468) exited]
[Thread 0x7ffff7000640 (LWP 7467) exited]
[Thread 0x7ffff7a00640 (LWP 7466) exited]
[Inferior 1 (process 7464) exited normally]
```

Strace:

```
$ strace -f ./main
execve("./main", ["./main"], 0x7ffe0ef18f38 /* 46 vars */) = 0
brk(NULL)                               = 0x5eef76be0000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd59d9d890) = -1 EINVAL (Недопустимый аргумент)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75dccd9d3000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=58047, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 58047, PROT_READ, MAP_PRIVATE, 3, 0) = 0x75dccd9c4000
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"..., 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"..., 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"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, 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"..., 784, 64) = 784
mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75dccd600000
mprotect(0x75dccd628000, 2023424, PROT_NONE) = 0
mmap(0x75dccd628000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) =
0x75dccd628000
mmap(0x75dccd7bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x75dccd7bd000
mmap(0x75dccd816000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) =
0x75dccd816000
mmap(0x75dccd81c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x75dccd81c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x75dccd9c1000
arch_prctl(ARCH_SET_FS, 0x75dccd9c1740) = 0
set_tid_address(0x75dccd9c1a10)         = 7630
set_robust_list(0x75dccd9c1a20, 24)     = 0
rseq(0x75dccd9c20e0, 0x20, 0, 0x53053053) = 0
mprotect(0x75dccd816000, 16384, PROT_READ) = 0
mprotect(0x5eef76074000, 4096, PROT_READ) = 0
mprotect(0x75dccda0d000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x75dccd9c4000, 58047)           = 0
write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\273\320\270\321\207\320\265\321\201\321\202\320"..., 68Введите количество переменных (n <= 10): ) = 68
read(0,
"\n", 100)                      = 1
write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\273\320\270\321\207\320\265\321\201\321\202\320"..., 52Введите количество потоков: ) = 52
read(0,
"\n", 100)                      = 1
write(1, "\320\235\320\265\320\262\320\265\321\200\320\275\320\276\320\265
\320\272\320\276\320\273\320\270\321\207\320\265\321\201\321"..., 121Неверное количество потоков. Установлено значение по
умолчанию: 4
) = 121
write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\215\320\273\320\265\320\274\320\265\320\275\321\202\321\213 "..., 72Введите элементы расширенной матрицы:
) = 72
rt_sigaction(SIGRT_1, {sa_handler=0x75dccd691870, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x75dccd642520}, 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) = 0x75dcccc00000
mprotect(0x75dcccc01000, 8388608, PROT_READ|PROT_WRITE) = 0
getrandom("\x66\x71\xfc\x57\x10\xb9\xbb\x65", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5eef76be0000
brk(0x5eef76c01000)                     = 0x5eef76c01000
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=0x75dccd400910, parent_tid=0x75dccd400910, exit_signal=0,
stack=0x75dcccc00000, stack_size=0x7fff00, tls=0x75dccd400640}strace: Process 7631 attached
```

```
 => {parent_tid=[7631]}, 88) = 7631
[pid  7631] rseq(0x75dccd400fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  7630] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
[pid  7631] <... rseq resumed>)         = 0
[pid  7630] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
[pid  7631] set_robust_list(0x75dccd400920, 24 <unfinished ...>
[pid  7630] <... mmap resumed>)         = 0x75dccc200000
[pid  7631] <... set_robust_list resumed>) = 0
[pid  7631] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  7630] mprotect(0x75dccc201000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  7631] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  7630] <... mprotect resumed>)     = 0
[pid  7631] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  7630] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  7631] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  7630] <... rt_sigprocmask resumed>[], 8) = 0
[pid  7630]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75dccca00910, parent_tid=0x75dccca00910, exit_signal=0,
stack=0x75dccc200000, stack_size=0x7fff00, tls=0x75dccca00640} <unfinished ...>
[pid  7631] madvise(0x75dcccc00000, 8368128, MADV_DONTNEED) = 0
[pid  7631] exit(0)                     = ?
strace: Process 7632 attached
[pid  7631] +++ exited with 0 +++
[pid  7632] rseq(0x75dccca00fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  7630] <... clone3 resumed> => {parent_tid=[7632]}, 88) = 7632
[pid  7632] <... rseq resumed>)         = 0
[pid  7630] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  7632] set_robust_list(0x75dccca00920, 24 <unfinished ...>
[pid  7630] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  7632] <... set_robust_list resumed>) = 0
[pid  7630] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
[pid  7632] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  7630] <... mmap resumed>)         = 0x75dccb800000
[pid  7632] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  7632] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  7630] mprotect(0x75dccb801000, 8388608, PROT_READ|PROT_WRITE) = 0
[pid  7632] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  7632] madvise(0x75dccc200000, 8368128, MADV_DONTNEED) = 0
[pid  7630] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  7632] exit(0 <unfinished ...>
[pid  7630] <... rt_sigprocmask resumed>[], 8) = 0
[pid  7632] <... exit resumed>)         = ?
[pid  7630]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75dccc000910, parent_tid=0x75dccc000910, exit_signal=0,
stack=0x75dccb800000, stack_size=0x7fff00, tls=0x75dccc000640} <unfinished ...>
[pid  7632] +++ exited with 0 +++
strace: Process 7633 attached
[pid  7630] <... clone3 resumed> => {parent_tid=[7633]}, 88) = 7633
[pid  7633] rseq(0x75dccc000fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  7630] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  7633] <... rseq resumed>)         = 0
[pid  7630] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  7633] set_robust_list(0x75dccc000920, 24 <unfinished ...>
[pid  7630] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x75dccae00000
[pid  7633] <... set_robust_list resumed>) = 0
[pid  7630] mprotect(0x75dccae01000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  7633] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  7630] <... mprotect resumed>)     = 0
[pid  7633] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  7630] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  7633] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  7630] <... rt_sigprocmask resumed>[], 8) = 0
[pid  7633] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  7630]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75dccb600910, parent_tid=0x75dccb600910, exit_signal=0,
stack=0x75dccae00000, stack_size=0x7fff00, tls=0x75dccb600640} <unfinished ...>
```

```
[pid  7633] madvise(0x75dccb800000, 8368128, MADV_DONTNEEDstrace: Process 7634 attached
 <unfinished ...>
[pid  7630] <... clone3 resumed> => {parent_tid=[7634]}, 88) = 7634
[pid  7633] <... madvise resumed>)      = 0
[pid  7630] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  7634] rseq(0x75dccb600fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  7633] exit(0 <unfinished ...>
[pid  7630] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  7630] futex(0x75dccc000910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 7633, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  7634] <... rseq resumed>)         = 0
[pid  7633] <... exit resumed>)         = ?
[pid  7634] set_robust_list(0x75dccb600920, 24 <unfinished ...>
[pid  7633] +++ exited with 0 +++
[pid  7630] <... futex resumed>)        = 0
[pid  7634] <... set_robust_list resumed>) = 0
[pid  7630] futex(0x75dccb600910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 7634, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  7634] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
[pid  7634] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
[pid  7634] madvise(0x75dccae00000, 8368128, MADV_DONTNEED) = 0
[pid  7634] exit(0)                     = ?
[pid  7630] <... futex resumed>)        = 0
[pid  7634] +++ exited with 0 +++
write(1, "\320\240\320\265\321\210\320\265\320\275\320\270\320\265 \321\201\320\270\321\201\321\202\320\265\320\274\321\213:\n",
31Решение системы:
) = 31
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Язык Си с библиотеками предоставляет возможность построения многопоточных приложений, дает инструменты для работы с потоками, их ограничения для безопасности. Все это делает разработку на Си многообразнее и интереснее.