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: Османова В.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 06.10.24

Москва, 2024

Постановка задачи

Вариант 18.

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.

Найти образец в строке наивным алгоритмом.

Общий метод и алгоритм решения

Использованные системные вызовы:

- int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
  void *(*start)(void *), void *arg) – создание потока
- int pthread_join (pthread_t THREAD_ID, void ** DATA) – ожидание завершения потока
- int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr) – инициализация мьютекса
- int pthread_mutex_lock(pthread_mutex_t *mutex) – блокировка мьютекса
- int pthread_mutex_unlock(pthread_mutex_t *mutex) – разблокировка мьютекса
- int pthread_mutex_destroy(pthread_mutex_t *mutex) – удаление мьютекса
- int open(const char *pathname, int flags, mode_t mode) - открытие\создание файла
- int close(int fd) – закрыть файл
- int lseek (int fd, off_t offset, int whence) - закрыть файл

В качестве аргументов командной строки программе подаются имя файла, количество потоков и подстрока, вхождения которой необходимо найти в файле.

Создаются массив указателей на потоки и массив, каждый элемент которого представляет собой структуру, содержащую необходимые потоку аргументы. Каждому потоку передаются файловый дескриптор (fd), номер потока (mod), общее количество потоков (N), подстрока для поиска (pattern), длина подстроки (pattern_len), указатель на указатель на первый элемент массива с найденными включениями (res), указатель на размер этого массива (res_len), ссылка на общий счётчик найденных включений (counter) и указатели на два мьютекса, которые отвечают за чтение из файла (mutex_file) и запись в массив включений (mutex_res).

Поток № k отвечает за проверку совпадения данной подстроки с подстроками файла, начинающихся с символов k, k + N, k + 2N и т.д. Таким образом каждый поток проверяет свой класс вычета по модулю количества потоков. В начале итерации поток блокирует мьютекс файла, перематывает файл на нужный символ, номер которого хранится в symb, считывает в буфер pattern_len символов, разблокирует мьютекс, проверяет, что ещё конец файла ещё не достигнут, и начинает сравнивать буфер с данной подстрокой. Если совпадение есть, блокируется мьютекс массива включений, запоминается текущее значение counter, затем оно инкрементируется. Также выполняется проверка, что массив не переполнен, в противном случае он увеличивается, а новый адрес массива и его размер записываются по адресам, которые есть у каждого потока. Далее мьютекс разблокируется и в массив вносится номер символа, начиная с которого найдено вхождение подстроки, т.к. значение counter уже инкрементировано, никакой другой поток не перезапишет это значение. После этого значение symb увеличивается на N, если совпадения не было, поток сразу переходит к этому шагу, и начинается следующая итерация.

После завершения всех потоков, программа сортирует массив вхождений, потому что потоки могут записывать туда результаты в любом порядке, и выводит в стандартный поток вывода номера символов, с которых начинается вхождение подстроки.

Код программы

main.c

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

#define MAX_THREADS 10
#define BUF_SIZE 1024
#define RES_SIZE 256

typedef struct {
    int mod;
    int N;
    int fd;
    char* pattern;
    int pattern_len;
    unsigned int** res;
    int* res_size;
    int* counter;
    pthread_mutex_t* mutex_res;
    pthread_mutex_t* mutex_file;
} Data;

int uint_cmp(const void* a1, const void* a2) {
    return (*(unsigned int*)a1 > *(int*)a2) - (*(unsigned int*)a1 < *(int*)a2);
}

int print_unsigned_int(const unsigned int num) {
    char buf[16];
    char res[32];
    int n = 0;
    int x = num;
    if (x == 0) {
        write(STDOUT_FILENO, "0\n", 2);
        return 0;
    }
    while (x) {
        buf[n] = (x % 10) + '0';
        x = x / 10;
        n++;
    }
    for (int i = 0; i < n; i++) {
        res[i] = buf[n - i - 1];
    }
    res[n] = '\n';
    write(STDOUT_FILENO, res, (n + 1));
    return 0;
}

int str_to_int(const char* string) {
    int num = 0;
    int i = 0;
    char c = string[i];
    while ((c >= '0') && (c <= '9')) {
        if ((num * 10 + (c - '0')) > 100) {
            return -1;
        }
        num = num * 10 + (c - '0');
```

```
        i++;
        c = string[i];
    }
    if ((c == 0) && (i != 0)) {
        return num;
    } else {
        return -1;
    }
}

void* find_inclusions_by_mod(void* data) {
    Data* d = (Data*)data;
    int symb = d->mod;
    int i; //compare buf and pattern
    int bytes_read = 0;

    char* buf = malloc(BUF_SIZE * sizeof(char));
    if (!buf) {
        write(STDOUT_FILENO, "fail to alloc\n", 15);
        return NULL;
    }
    int buf_offset = 0; //current index in buf

    pthread_mutex_lock(d->mutex_file);
    lseek(d->fd, symb, SEEK_SET);
    bytes_read = read(d->fd, buf, BUF_SIZE);
    pthread_mutex_unlock(d->mutex_file);

    int my_res_size = RES_SIZE / d->N;
    int* my_res = malloc(sizeof(int) * my_res_size);
    int my_counter = 0;

    while (1) {
        if ((bytes_read - buf_offset) < d->pattern_len) {
            if (bytes_read < BUF_SIZE) break;

            pthread_mutex_lock(d->mutex_file);
            lseek(d->fd, symb, SEEK_SET);
            bytes_read = read(d->fd, buf, BUF_SIZE);
            buf_offset = 0;
            pthread_mutex_unlock(d->mutex_file);

            if (bytes_read < d->pattern_len) break;
        }

        i = 0;
        while ((d->pattern)[i] && (buf[buf_offset + i] == (d->pattern)[i])) {
            i++;
        }

        if (!(d->pattern)[i]) {
            my_res[my_counter] = symb;
            my_counter ++;
            if (my_counter >= my_res_size) {
                my_res_size *= 2;
                my_res = (unsigned int*)realloc(my_res, *(d->res_size) * sizeof(unsigned
int));
            }
        }
        symb += d->N;
        buf_offset += d->N;
    }

    pthread_mutex_lock(d->mutex_res);
```

```
    int index = *(d->counter);
    if (index + my_counter >= *(d->res_size)) {
        *(d->res_size) =  (*(d->res_size) + my_counter) * 2;
        *(d->res) = (unsigned int*)realloc(*(d->res), *(d->res_size) * sizeof(unsigned
int));
    }
    (*(d->counter)) = (*(d->counter)) + my_counter;
    pthread_mutex_unlock(d->mutex_res);

    for (int i = 0; i < my_counter; i++) {
        (*(d->res))[index + i] = my_res[i];
    }

    free(buf);
    return NULL;
}

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

    if (argc != 4) {
        char* msg = "enter: ./main.out <path to file> <number of threads> <pattern>\n";
        write(STDOUT_FILENO, msg, strlen(msg));
        return -1;
    }

    int N = str_to_int(argv[2]);
    if((N == -1) || (N > MAX_THREADS)) {
        char* msg = "invalid number of threads\n";
        write(STDOUT_FILENO, msg, strlen(msg));
        return -1;
    }

    pthread_t* threads = (pthread_t*)malloc(N * sizeof(pthread_t));
    Data* threads_data = (Data*)malloc(N * sizeof(Data));
    int fd = open(argv[1], O_RDONLY);
    if (fd == -1) {
        char* msg = "fail to open file\n";
        write(STDOUT_FILENO, msg, strlen(msg));
        return 1;
    }
    int pattern_len = strlen(argv[3]);
    char* pattern = (char*)malloc((pattern_len + 1) * sizeof(char));
    strcpy(pattern, argv[3]);
    int res_size = RES_SIZE;
    unsigned int* results = (unsigned int*)malloc(res_size * sizeof(unsigned int));
    int counter = 0;

    pthread_mutex_t mutex_res;
    pthread_mutex_init(&mutex_res, NULL);
    pthread_mutex_t mutex_file;
    pthread_mutex_init(&mutex_file, NULL);

    for (int i = 0; i < N; i++) {
        threads_data[i].fd = fd;
        threads_data[i].mod = i;
        threads_data[i].N = N;
        threads_data[i].pattern = pattern;
        threads_data[i].pattern_len = pattern_len;
        threads_data[i].res = &results;
        threads_data[i].res_size = &res_size;
        threads_data[i].counter = &counter;
        threads_data[i].mutex_res = &mutex_res;
        threads_data[i].mutex_file = &mutex_file;
    }
```

```
    for (int i = 0; i < N; i++) {
        pthread_create(threads + i, NULL, &find_inclusions_by_mod, threads_data + i);
    }

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

    qsort(results, counter, sizeof(int), &uint_cmp);

    for (int i = 0; i < counter; i++) {
        print_unsigned_int(results[i]);
    }

    free(threads);
    free(threads_data);
    free(pattern);
    free(results);

    close(fd);

    pthread_mutex_destroy(&mutex_res);
    pthread_mutex_destroy(&mutex_file);
    return 0;
}
```

Протокол работы программы

Тестирование:

```
$ ./main.out b.txt 5 program
30
75
182
243
284
327
406
457
513
573
648
754
```

b.txt:

In the world of technology, a program is essential.
Every time you run a program, you are engaging
with a piece of software designed to perform
a specific task. A well-written program can make
life easier by automating processes. As a programmer,
you spend hours creating a program that meets
user needs. Testing a program is just as important
as writing it, ensuring that each part of
the program functions correctly. When you finish
the program, you feel a sense of accomplishment.
A simple program can lead to more complex
systems, showing how one program can impact
many areas of life. So, whether you
are learning to program or are already experienced,
remember that every great achievement in technology
starts with a program!

Производительность на этом тесте:

Количество потоков	Среднее время выполнения, мс
1	4,3
5	3,1
10	1,8

Strace:

```
execve("./main.out", ["./main.out", "b.txt", "5", "program"], 0x7fffdc5e8b60 /* 57 vars */) = 0
brk(NULL)                               = 0x562379a32000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe2aa71eb0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f83b9cd5000
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=61763, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 61763, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f83b9cc5000
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<\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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2072888, ...}, 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, 2117488, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f83b9a00000
mmap(0x7f83b9a22000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x22000) =
0x7f83b9a22000
mmap(0x7f83b9b9a000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19a000) =
0x7f83b9b9a000
mmap(0x7f83b9bf2000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1f1000) =
0x7f83b9bf2000
mmap(0x7f83b9bf8000, 53104, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7f83b9bf8000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f83b9cc2000
arch_prctl(ARCH_SET_FS, 0x7f83b9cc2740) = 0
set_tid_address(0x7f83b9cc2a10)         = 13124
set_robust_list(0x7f83b9cc2a20, 24)     = 0
rseq(0x7f83b9cc3060, 0x20, 0, 0x53053053) = 0
mprotect(0x7f83b9bf2000, 16384, PROT_READ) = 0
mprotect(0x56237818f000, 4096, PROT_READ) = 0
mprotect(0x7f83b9d0a000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f83b9cc5000, 61763)           = 0
getrandom("\x18\x43\x59\xa9\xc6\x95\xb6\xda", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x562379a32000
brk(0x562379a53000)                     = 0x562379a53000
openat(AT_FDCWD, "b.txt", O_RDONLY)     = 3
rt_sigaction(SIGRT_1, {sa_handler=0x7f83b9a8c450, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f83b9a3c460}, 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) = 0x7f83b91ff000
mprotect(0x7f83b9200000, 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_P
ARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f83b99ff990, parent_tid=0x7f83b99ff990, exit_signal=0,
stack=0x7f83b91ff000, stack_size=0x7fff80, tls=0x7f83b99ff6c0}strace: Process 13125 attached
 => {parent_tid=[13125]}, 88) = 13125
[pid 13125] rseq(0x7f83b99fffe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 13124] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
[pid 13125] <... rseq resumed>)         = 0
[pid 13124] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
[pid 13125] set_robust_list(0x7f83b99ff9a0, 24 <unfinished ...>
[pid 13124] <... mmap resumed>)         = 0x7f83b89fe000
[pid 13124] mprotect(0x7f83b89ff000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid 13125] <... set_robust_list resumed>) = 0
[pid 13124] <... mprotect resumed>)     = 0
[pid 13125] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 13124] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0
[pid 13125] <... rt_sigprocmask resumed>NULL, 8) = 0
```

```
[pid 13124]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_P
ARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f83b91fe990, parent_tid=0x7f83b91fe990, exit_signal=0,
stack=0x7f83b89fe000, stack_size=0x7fff80, tls=0x7f83b91fe6c0} <unfinished ...>
[pid 13125] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 13124] <... clone3 resumed> => {parent_tid=[13126]}, 88) = 13126
strace: Process 13126 attached
[pid 13124] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 13125] <... mmap resumed>)         = 0x7f83b09fe000
[pid 13124] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13124] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
[pid 13126] rseq(0x7f83b91fefe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 13124] <... mmap resumed>)         = 0x7f83b01fd000
[pid 13126] <... rseq resumed>)         = 0
[pid 13125] munmap(0x7f83b09fe000, 56631296 <unfinished ...>
[pid 13124] mprotect(0x7f83b01fe000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid 13126] set_robust_list(0x7f83b91fe9a0, 24 <unfinished ...>
[pid 13124] <... mprotect resumed>)     = 0
[pid 13126] <... set_robust_list resumed>) = 0
[pid 13125] <... munmap resumed>)       = 0
[pid 13124] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid 13125] munmap(0x7f83b8000000, 10477568 <unfinished ...>
[pid 13124] <... rt_sigprocmask resumed>[], 8) = 0
[pid 13125] <... munmap resumed>)       = 0
[pid 13124]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_P
ARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f83b09fd990, parent_tid=0x7f83b09fd990, exit_signal=0,
stack=0x7f83b01fd000, stack_size=0x7fff80, tls=0x7f83b09fd6c0} <unfinished ...>
[pid 13126] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 13125] mprotect(0x7f83b4000000, 135168, PROT_READ|PROT_WRITE) = 0
strace: Process 13127 attached
[pid 13124] <... clone3 resumed> => {parent_tid=[13127]}, 88) = 13127
[pid 13125] lseek(3, 0, SEEK_SET <unfinished ...>
[pid 13127] rseq(0x7f83b09fdfe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 13126] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13124] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 13125] <... lseek resumed>)        = 0
[pid 13127] <... rseq resumed>)         = 0
[pid 13126] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 13124] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13125] read(3,  <unfinished ...>
[pid 13124] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
[pid 13127] set_robust_list(0x7f83b09fd9a0, 24 <unfinished ...>
[pid 13126] <... mmap resumed>)         = 0x7f83a81fd000
[pid 13124] <... mmap resumed>)         = 0x7f83b81fd000
[pid 13126] munmap(0x7f83a81fd000, 65024000 <unfinished ...>
[pid 13124] mprotect(0x7f83b81fe000, 8388608, PROT_READ|PROT_WRITE) = 0
[pid 13127] <... set_robust_list resumed>) = 0
[pid 13126] <... munmap resumed>)       = 0
[pid 13124] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid 13126] munmap(0x7f83b0000000, 2084864 <unfinished ...>
[pid 13127] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 13124] <... rt_sigprocmask resumed>[], 8) = 0
[pid 13127] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13126] <... munmap resumed>)       = 0
[pid 13124]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_P
ARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f83b89fd990, parent_tid=0x7f83b89fd990, exit_signal=0,
stack=0x7f83b81fd000, stack_size=0x7fff80, tls=0x7f83b89fd6c0} <unfinished ...>
[pid 13127] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 13126] mprotect(0x7f83ac000000, 135168, PROT_READ|PROT_WRITE) = 0
[pid 13127] <... mmap resumed>)         = 0x7f83a4000000
[pid 13124] <... clone3 resumed> => {parent_tid=[13128]}, 88) = 13128
strace: Process 13128 attached
[pid 13124] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 13127] munmap(0x7f83a8000000, 67108864 <unfinished ...>
[pid 13126] futex(0x7ffe2aa71ec0, FUTEX_WAIT_PRIVATE, 2, NULL <unfinished ...>
[pid 13125] <... read resumed>"In the world of technology, a pr"..., 1024) = 764
[pid 13124] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13127] <... munmap resumed>)       = 0
[pid 13124] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
[pid 13125] futex(0x7ffe2aa71ec0, FUTEX_WAKE_PRIVATE, 1 <unfinished ...>
[pid 13128] rseq(0x7f83b89fdfe0, 0x20, 0, 0x53053053 <unfinished ...>
```

```
[pid 13124] <... mmap resumed>)         = 0x7f83b37ff000
[pid 13128] <... rseq resumed>)         = 0
[pid 13128] set_robust_list(0x7f83b89fd9a0, 24 <unfinished ...>
[pid 13124] mprotect(0x7f83b3800000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid 13125] <... futex resumed>)        = 1
[pid 13126] <... futex resumed>)        = 0
[pid 13124] <... mprotect resumed>)     = 0
[pid 13128] <... set_robust_list resumed>) = 0
[pid 13127] mprotect(0x7f83a4000000, 135168, PROT_READ|PROT_WRITE <unfinished ...>
[pid 13126] lseek(3, 1, SEEK_SET <unfinished ...>
[pid 13124] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid 13125] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid 13128] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 13127] <... mprotect resumed>)     = 0
[pid 13126] <... lseek resumed>)        = 1
[pid 13124] <... rt_sigprocmask resumed>[], 8) = 0
[pid 13125] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13128] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13126] read(3,  <unfinished ...>
[pid 13124]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_P
ARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f83b3fff990, parent_tid=0x7f83b3fff990, exit_signal=0,
stack=0x7f83b37ff000, stack_size=0x7fff80, tls=0x7f83b3fff6c0} <unfinished ...>
[pid 13126] <... read resumed>"n the world of technology, a pro"..., 1024) = 763
[pid 13125] madvise(0x7f83b91ff000, 8368128, MADV_DONTNEED <unfinished ...>
[pid 13128] futex(0x7ffe2aa71ec0, FUTEX_WAIT_PRIVATE, 2, NULL <unfinished ...>
[pid 13126] futex(0x7ffe2aa71ec0, FUTEX_WAKE_PRIVATE, 1 <unfinished ...>
[pid 13128] <... futex resumed>)        = -1 EAGAIN (Resource temporarily unavailable)
[pid 13125] <... madvise resumed>)      = 0
[pid 13127] lseek(3, 2, SEEK_SET <unfinished ...>
[pid 13126] <... futex resumed>)        = 0
strace: Process 13129 attached
[pid 13124] <... clone3 resumed> => {parent_tid=[13129]}, 88) = 13129
[pid 13125] exit(0 <unfinished ...>
[pid 13127] <... lseek resumed>)        = 2
[pid 13126] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid 13124] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 13129] rseq(0x7f83b3ffffe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 13125] <... exit resumed>)         = ?
[pid 13127] read(3,  <unfinished ...>
[pid 13126] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13124] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13129] <... rseq resumed>)         = 0
[pid 13124] futex(0x7f83b91fe990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 13126, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 13129] set_robust_list(0x7f83b3fff9a0, 24 <unfinished ...>
[pid 13128] futex(0x7ffe2aa71ec0, FUTEX_WAIT_PRIVATE, 2, NULL <unfinished ...>
[pid 13127] <... read resumed>" the world of technology, a prog"..., 1024) = 762
[pid 13126] madvise(0x7f83b89fe000, 8368128, MADV_DONTNEED <unfinished ...>
[pid 13125] +++ exited with 0 +++
[pid 13129] <... set_robust_list resumed>) = 0
[pid 13127] futex(0x7ffe2aa71ec0, FUTEX_WAKE_PRIVATE, 1 <unfinished ...>
[pid 13126] <... madvise resumed>)      = 0
[pid 13129] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 13127] <... futex resumed>)        = 1
[pid 13126] exit(0 <unfinished ...>
[pid 13128] <... futex resumed>)        = 0
[pid 13129] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13128] lseek(3, 3, SEEK_SET <unfinished ...>
[pid 13127] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid 13126] <... exit resumed>)         = ?
[pid 13129] futex(0x7ffe2aa71ec0, FUTEX_WAIT_PRIVATE, 2, NULL <unfinished ...>
[pid 13128] <... lseek resumed>)        = 3
[pid 13128] read(3,  <unfinished ...>
[pid 13124] <... futex resumed>)        = 0
[pid 13128] <... read resumed>"the world of technology, a progr"..., 1024) = 761
[pid 13127] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13126] +++ exited with 0 +++
[pid 13128] futex(0x7ffe2aa71ec0, FUTEX_WAKE_PRIVATE, 1 <unfinished ...>
[pid 13124] futex(0x7f83b09fd990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 13127, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 13127] madvise(0x7f83b01fd000, 8368128, MADV_DONTNEED <unfinished ...>
[pid 13128] <... futex resumed>)        = 1
```

```
[pid 13127] <... madvise resumed>)      = 0
[pid 13128] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid 13129] <... futex resumed>)        = 0
[pid 13127] exit(0 <unfinished ...>
[pid 13128] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 13129] lseek(3, 4, SEEK_SET <unfinished ...>
[pid 13128] madvise(0x7f83b81fd000, 8368128, MADV_DONTNEED <unfinished ...>
[pid 13127] <... exit resumed>)         = ?
[pid 13128] <... madvise resumed>)      = 0
[pid 13129] <... lseek resumed>)        = 4
[pid 13128] exit(0)                     = ?
[pid 13129] read(3,  <unfinished ...>
[pid 13128] +++ exited with 0 +++
[pid 13124] <... futex resumed>)        = 0
[pid 13124] futex(0x7f83b3fff990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 13129, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid 13129] <... read resumed>"he world of technology, a progra"..., 1024) = 760
[pid 13127] +++ exited with 0 +++
[pid 13129] futex(0x7ffe2aa71ec0, FUTEX_WAKE_PRIVATE, 1) = 0
[pid 13129] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
[pid 13129] madvise(0x7f83b37ff000, 8368128, MADV_DONTNEED) = 0
[pid 13129] exit(0)                     = ?
[pid 13124] <... futex resumed>)        = 0
[pid 13129] +++ exited with 0 +++
munmap(0x7f83b91ff000, 8392704)         = 0
write(1, "30\n", 330
)                       = 3
write(1, "75\n", 375
)                       = 3
write(1, "182\n", 4182
)                      = 4
write(1, "243\n", 4243
)                      = 4
write(1, "284\n", 4284
)                      = 4
write(1, "327\n", 4327
)                      = 4
write(1, "406\n", 4406
)                      = 4
write(1, "457\n", 4457
)                      = 4
write(1, "513\n", 4513
)                      = 4
write(1, "573\n", 4573
)                      = 4
write(1, "648\n", 4648
)                      = 4
write(1, "754\n", 4754
)                      = 4
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Работа над этой программой потребовала глубокого понимания многопоточного программирования в C. Были реализованы механизмы создания и синхронизации потоков с помощью библиотеки pthread, включая использование мьютексов для безопасного доступа к общим ресурсам. Особое внимание было уделено оптимизации алгоритма поиска подстрок, что повысило производительность программы при увеличении числа потоков. В результате была создана программа, демонстрирующая понимание концепций многопоточного программирования и обработки данных в C.