ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет по:

Лабораторной работе №1

Дисциплина:

Операционные системы

Выполнили:

Студенты гр. P33211

Бурдаев Игорь

Ломоносов Александр

Санкт-Петербург

2020 г.

Задание

Разработать программу на языке C, которая осуществляет следующие действия

- Создает область памяти размером A мегабайт, начинающихся с адреса B (если возможно) при помощи C = (malloc, mmap) заполненную случайными числами /dev/urandom в D потоков. Используя системные средства мониторинга, определите адрес начала в адресном пространстве процесса и характеристики выделенных участков памяти. Замеры виртуальной/физической памяти необходимо снять:

1. До аллокации
2. После аллокации
3. После заполнения участка данными
4. После деаллокации

- Записывает область памяти в файлы одинакового размера E мегабайт с использованием F = (блочного, некешируемого) обращения к диску. Размер блока ввода-вывода G байт. Преподаватель выдает в качестве задания последовательность записи/чтения блоков H = (последовательный, заданный или случайный)
- Генерацию данных и запись осуществлять в бесконечном цикле.
- В отдельных I потоках осуществлять чтение данных из файлов и подсчитывать агрегированные характеристики данных – J = (сумму, среднее значение, максимальное, минимальное значение).
- Чтение и запись данных в/из файла должна быть защищена примитивами синхронизации K = (futex, cv, sem, flock).
- По заданию преподавателя изменить приоритеты потоков и описать изменения в характеристиках программы.

Для запуска программы возможно использовать операционную систему Windows 10 или Debian/Ubuntu в виртуальном окружении.

Измерить значения затраченного процессорного времени на выполнение программы и на операции ввода-вывода используя системные утилиты.

Отследить трассу системных вызовов.

Используя stap построить графики системных характеристик.

Вариант

A = 214; B = 0x8B1598B0; C = mmap; D = 134; E = 188; F = block; G = 62; H = seq; I = 65; J = min; K = sem.

Код

```
#include <stdio.h>
#include <sys/mman.h>
#include <pthread.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include <limits.h>
#include <semaphore.h>
#include <stdbool.h>
#include <stdint.h>

#define A 214
#define B 0x8B1598B0
#define D 134
#define E 188
#define G 62
#define I 55
#define SUCCESS 0
#define ERROR_CREATE_THREAD -11

sem_t files_semaphore[2];
int file_count = 0;
int fds[2];

typedef struct {
    char* address;
    FILE* file;
    size_t size;
    int number;
```

```
} write_args;

typedef struct {
    int number;

} aggregate_args;

char* allocate_memory() {
    return mmap((void*) B, A * 1024 * 1024, PROT_READ | PROT_WRITE,
MAP_PRIVATE | MAP_ANONYMOUS, 0, 0);
}

void* generate_numbers(void* arg) {
    write_args* args = (write_args*) arg;
    int count = 0;
    count = fread(args->address, args->size, 1, args->file);
    if (count == 1) return SUCCESS;
    else return (void*) 1;
}

void fill_memory(char* addr) {
    int i;
    int status;
    pthread_t threads[D];
    write_args args[D];
    int extra_size = A * 1024 * 1024 % D;
    int size = (A * 1024 * 1024 - extra_size) / D;
    FILE* f = fopen("/dev/urandom", "rb");
    for (i = 0; i < D; i++) {
        args[i].address = addr + i * size;
        args[i].file = f;
```

```
        if (i == D - 1) args[i].size = (size_t) (size + extra_size);
        else args[i].size = (size_t) size;
        status = pthread_create(&threads[i], NULL, generate_numbers, (void*)
&args[i]);
        if (status != 0) {
            printf("error: can't create thread number:%d\n", i + 1);
            exit(ERROR_CREATE_THREAD);
        }
    }
    printf("filling the memory...\n");
    for (i = 0; i < D; i++) {
        pthread_join(threads[i], NULL);

    }
    printf("completed\n");
    fclose(f);

}

char* generate_filename(int number) {
    char* str_number = malloc(sizeof(number) + 1);
    char* filename = malloc(sizeof("File") + sizeof(number) + 1);

    strcpy(filename, "File");
    snprintf(str_number, 10, "%d", number + 1);
    strcat(filename, str_number);

    free(str_number);

    return filename;
}
```

```
void* write_thread(void* arg) {
    write_args* args = (write_args*) arg;
    int file_number = args->number % 2;

    sem_wait(&files_semaphore[file_number]);

    write(fds[file_number], args->address, args->size);

    sem_post(&files_semaphore[file_number]);
    return NULL;
}

void write_in_files(char* address) {
    int i;
    int last_size = A % E * 1024 * 1024;
    int count = (A * 1024 * 1024 - last_size) / E / 1024 / 1024;
    pthread_t threads[count + 1];
    write_args args[count + 1];
    printf("start writing...\n");
    for (i = 0; i < count; i++) {
        args[i].address = address;
        args[i].size = E * 1024 * 1024;
        args[i].number = i;
        pthread_create(&threads[i], NULL, write_thread, (void*) &args[i]);
        address += E * 1024 * 1024;
    }
    args[i].address = address;
    args[i].size = last_size;
    args[i].number = i;
    pthread_create(&threads[i], NULL, write_thread, (void*) &args[i]);
```

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

}

void* aggregate_data(void* arg) {
    aggregate_args* args = (aggregate_args*) arg;
    int min = INT_MAX;
    char buffer[G];
    int i;
    int j;
    int file_number = args->number % 2;
    lseek(fds[file_number], 0, SEEK_SET);

    while (1) {
        sem_wait(&files_semaphore[file_number]);
        int count = read(fds[file_number], buffer, G);
        sem_post(&files_semaphore[file_number]);
        if (count == 0) break;
        for (i = 0; i < G / sizeof(int); i += sizeof(int)) {
            int num = 0;
            for (j = 0; j < sizeof(int); j++) {
                num = (num << 8) + buffer[i + j];
            }
            if (min > num) min = num;
        }
    }

    printf("thread%lu min value from file: File%d= %d\n", (uint64_t)
pthread_self(), file_number + 1, min);
    return NULL;
}
```

```
void start_aggregate_threads(const int files_count) {
    int i;
    aggregate_args args[I];
    pthread_t threads[I];
    printf("starting aggregating threads...\n");
    for (i = 0; i < I; i++) {// I
        args[i].number = i + 1;
        pthread_create(&threads[i], NULL, aggregate_data, (void*) &args[i]);
    }
}

int main(void) {
    for (int i = 0; i < 2; i++) {
        sem_init(&files_semaphore[i], 0, 1);
    }

    for (int i = 0; i < 2; i++) {
        char * filename = generate_filename(i);
        fds[i] = open(filename, O_RDWR | O_CREAT | O_TRUNC, (mode_t) 0600);
        free(filename);
    }
    printf("before allocation\n");
    getchar();
    bool is_aggregate_start = false;
    const int last_size = A % E * 1024 * 1024;
    file_count = (A * 1024 * 1024 - last_size) / E / 1024 / 1024 + 1;

    while (1) {
        char* address = allocate_memory();
        printf("after allocation\n");
```

```
        printf("starting allocating memory address: %p\n",(void*)address);
        getchar();
        fill_memory(address);
        printf("after filling memory\n");

        char ch;
        while ((ch = getchar()) != '\n' && ch != EOF);
        getchar();

        write_in_files(address);
        if (!is_aggregate_start) start_aggregate_threads(file_count);
        is_aggregate_start = true;
        printf("%p", address);
        munmap(address, A * 1024 * 1024);
        printf("after deallocation\n");
        getchar();
    }
}
```

Замеры

До аллокации

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
231739	root	20	0	358M	30112	15056	S	0.0	0.8	0:01.31	/usr/lib/packag
231752	root	20	0	336M	19328	16684	S	0.0	0.5	0:00.00	/usr/sbin/Netwo
231753	root	20	0	336M	19328	16684	S	0.0	0.5	0:00.01	/usr/sbin/Netwo
231746	root	20	0	336M	19328	16684	S	0.0	0.5	0:00.13	/usr/sbin/Netwo
231902	lp	20	0	15324	6700	5900	S	0.0	0.2	0:00.00	/usr/lib/cups/n
232231	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.00	/usr/bin/gjs /u
232232	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.00	/usr/bin/gjs /u
232248	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.00	/usr/bin/gjs /u
232263	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.00	/usr/bin/gjs /u
232228	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.05	/usr/bin/gjs /u
232583	parallels	20	0	948M	51848	38972	S	0.0	1.5	0:00.00	/usr/libexec/gn
232584	parallels	20	0	948M	51848	38972	S	0.0	1.5	0:00.03	/usr/libexec/gn
232588	parallels	20	0	948M	51848	38972	S	0.0	1.5	0:00.00	/usr/libexec/gn
232593	parallels	20	0	948M	51848	38972	S	0.0	1.5	0:00.00	/usr/libexec/gn
232594	parallels	20	0	19248	4900	3512	S	0.0	0.1	0:00.02	bash
232930	parallels	20	0	2648	632	548	S	0.0	0.0	0:00.00	./main

После аллокации

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
231739	root	20	0	358M	30112	15056	S	0.0	0.8	0:01.31	/usr/lib/packa
231752	root	20	0	336M	19328	16684	S	0.0	0.5	0:00.00	/usr/sbin/Netw
231753	root	20	0	336M	19328	16684	S	0.0	0.5	0:00.01	/usr/sbin/Netw
231746	root	20	0	336M	19328	16684	S	0.0	0.5	0:00.13	/usr/sbin/Netw
231902	lp	20	0	15324	6700	5900	S	0.0	0.2	0:00.00	/usr/lib/cups/
232231	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.00	/usr/bin/gjs /
232232	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.00	/usr/bin/gjs /
232248	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.00	/usr/bin/gjs /
232263	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.00	/usr/bin/gjs /
232228	parallels	20	0	2546M	26604	21616	S	0.0	0.7	0:00.05	/usr/bin/gjs /
232583	parallels	20	0	948M	51940	38992	S	0.0	1.5	0:00.00	/usr/libexec/g
232584	parallels	20	0	948M	51940	38992	S	0.0	1.5	0:00.03	/usr/libexec/g
232588	parallels	20	0	948M	51940	38992	S	0.0	1.5	0:00.00	/usr/libexec/g
232593	parallels	20	0	948M	51940	38992	S	0.0	1.5	0:00.00	/usr/libexec/g
232594	parallels	20	0	19248	4900	3512	S	0.0	0.1	0:00.02	bash
232930	parallels	20	0	216M	632	548	S	0.0	0.0	0:00.00	./main

После заполнения участка данными

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
31813	parallels	20	0	602M	20512	10136	S	0.0	0.6	0:33.49	/usr/bin/prlcc
31853	parallels	20	0	164M	4740	4068	S	0.0	0.1	0:25.76	/usr/bin/prldnd
31832	parallels	20	0	164M	4740	4068	S	0.0	0.1	0:25.78	/usr/bin/prldnd
641	root	20	0	243M	9408	5700	S	0.0	0.3	0:24.18	/usr/lib/policy
232582	parallels	20	0	948M	52000	39012	S	0.0	1.5	0:01.79	/usr/libexec/gn
31559	parallels	20	0	312M	6244	4468	S	0.0	0.2	0:04.98	/usr/bin/ibus-d
31556	parallels	20	0	312M	6244	4468	S	0.0	0.2	0:07.90	/usr/bin/ibus-d
31426	parallels	20	0	147M	39624	10528	S	0.0	1.1	0:09.93	/usr/lib/xorg/X
754	root	20	0	306M	1696	1596	S	0.0	0.0	0:16.25	/usr/bin/prltoo
231840	parallels	20	0	3565M	219M	71628	S	0.0	6.3	0:00.33	/usr/bin/gnome-
199355	root	20	0	38264	10236	7428	S	0.0	0.3	0:02.29	/usr/sbin/cupsd
31357	parallels	20	0	13952	10600	3624	S	0.0	0.3	0:07.15	/usr/bin/dbus-d
1118	root	20	0	150M	3304	3020	S	0.0	0.1	0:29.71	prlshprint
31669	parallels	20	0	167M	5048	4736	S	0.0	0.1	0:01.56	/usr/libexec/ib
31981	parallels	20	0	404M	14256	6612	S	0.0	0.4	0:14.42	/usr/bin/prlsga
232930	parallels	20	0	312M	215M	1560	S	0.0	6.2	0:06.97	./main

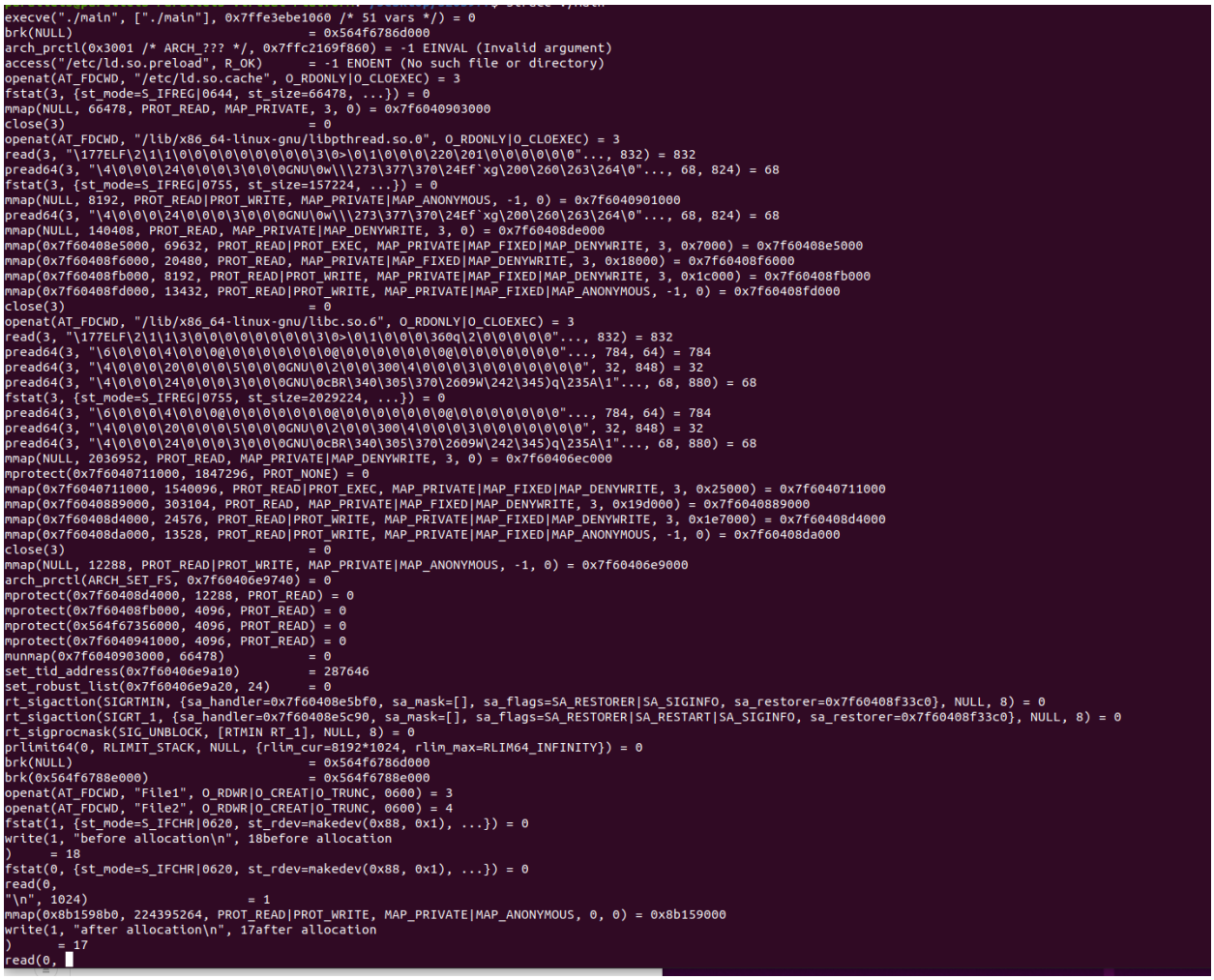
После деаллокации

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
31629	parallels	20	0	492M	20672	11408	S	0.0	0.6	0:00.88	/usr/libexec/x
231850	parallels	20	0	3565M	220M	71628	S	0.0	6.4	0:00.07	/usr/bin/gnome
31590	parallels	20	0	159M	4820	4604	S	0.0	0.1	0:01.37	/usr/libexec/a
31468	parallels	20	0	315M	5900	5628	S	0.0	0.2	0:00.91	/usr/libexec/g
31853	parallels	20	0	164M	4740	4068	S	0.0	0.1	0:25.95	/usr/bin/prldn
199358	root	20	0	176M	12468	10752	S	0.0	0.4	0:00.53	/usr/sbin/cups
599	root	20	0	131M	7304	6576	S	0.0	0.2	0:13.58	/usr/sbin/ther
31850	parallels	20	0	404M	14256	6612	S	0.0	0.4	0:25.53	/usr/bin/prlsg
31603	parallels	20	0	604M	7476	6504	S	0.0	0.2	0:01.46	/usr/libexec/x
648	root	20	0	131M	7304	6576	S	0.0	0.2	0:04.55	/usr/sbin/ther
31981	parallels	20	0	404M	14256	6612	S	0.0	0.4	0:14.54	/usr/bin/prlsg
75895	parallels	20	0	316M	6852	5900	S	0.0	0.2	0:08.83	/usr/libexec/g
31945	parallels	20	0	1854M	423M	18516	S	0.0	12.3	0:19.34	/snap/snap-sto
31835	parallels	20	0	6220	3244	2984	S	0.0	0.1	0:11.70	/usr/bin/prlsh
232930	parallels	20	0	522M	2176	1560	S	0.0	0.1	0:06.98	./main

Время выполнения

```
real    0m17.934s
user    0m0.028s
sys     0m6.957s
```

Трасса системных вызовов



```
execve("./main", ["./main"], 0x7ffe3ebe1060 /* 51 vars */) = 0
brk(NULL)                               = 0x564f6786d000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc2169f860) = -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=66478, ...}) = 0
mmap(NULL, 66478, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f6040903000
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\0w\\\273\377\370\24Ef`xg\200\260\263\264\0"..., 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) = 0x7f6040901000
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0w\\\273\377\370\24Ef`xg\200\260\263\264\0"..., 68, 824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f60408de000
mmap(0x7f60408e5000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f60408e5000
mmap(0x7f60408f6000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f60408f6000
mmap(0x7f60408fb000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7f60408fb000
mmap(0x7f60408fd000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f60408fd000
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\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 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\0cBR\340\305\370\2609W\242\345)q\235A\1"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f60406ec000
mprotect(0x7f6040711000, 1847296, PROT_NONE) = 0
mmap(0x7f6040711000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f6040711000
mmap(0x7f6040889000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7f6040889000
mmap(0x7f60408d4000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f60408d4000
mmap(0x7f60408da000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f60408da000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f60406e9000
arch_prctl(ARCH_SET_FS, 0x7f60406e9740) = 0
mprotect(0x7f60408d4000, 12288, PROT_READ) = 0
mprotect(0x7f60408fb000, 4096, PROT_READ) = 0
mprotect(0x564f67356000, 4096, PROT_READ) = 0
mprotect(0x7f6040941000, 4096, PROT_READ) = 0
munmap(0x7f6040903000, 66478)           = 0
set_tid_address(0x7f60406e9a10)         = 287646
set_robust_list(0x7f60406e9a20, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7f60408e5bf0, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7f60408f33c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7f60408e5c90, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f60408f33c0}, 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)                               = 0x564f6786d000
brk(0x564f6788e000)                     = 0x564f6788e000
openat(AT_FDCWD, "File1", O_RDWR|O_CREAT|O_TRUNC, 0600) = 3
openat(AT_FDCWD, "File2", O_RDWR|O_CREAT|O_TRUNC, 0600) = 4
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
write(1, "before allocation\n", 18before allocation
)      = 18
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
read(0,
"\n", 1024)                     = 1
mmap(0x8b1598b0, 224395264, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, 0, 0) = 0x8b159000
write(1, "after allocation\n", 17after allocation
)       = 17
read(0,
```

Статистика системных вызовов

% time	seconds	usecs/call	calls	errors	syscall
97.47	900.121380	1094	822400	11103	futex
2.22	20.533791	50	405773	1	read
0.25	2.352859	42779	55		lseek
0.02	0.185351	2808	66		write
0.02	0.166210	870	191		madvise
0.01	0.065446	342	191		clone
0.00	0.021616	161	134		munmap
0.00	0.019640	102	192		set_robust_list
0.00	0.016929	87	193		mprotect
0.00	0.012356	61	202		mmap
0.00	0.000009	2	4		close
0.00	0.000000	0	6		fstat
0.00	0.000000	0	3		brk
0.00	0.000000	0	2		rt_sigaction
0.00	0.000000	0	1		rt_sigprocmask
0.00	0.000000	0	1	1	ioctl
0.00	0.000000	0	8		pread64
0.00	0.000000	0	1	1	access
0.00	0.000000	0	1		execve
0.00	0.000000	0	2	1	arch_prctl
0.00	0.000000	0	1		set_tid_address
0.00	0.000000	0	6		openat
0.00	0.000000	0	1		prlimit64
100.00	923.495587		1229434	11107	total

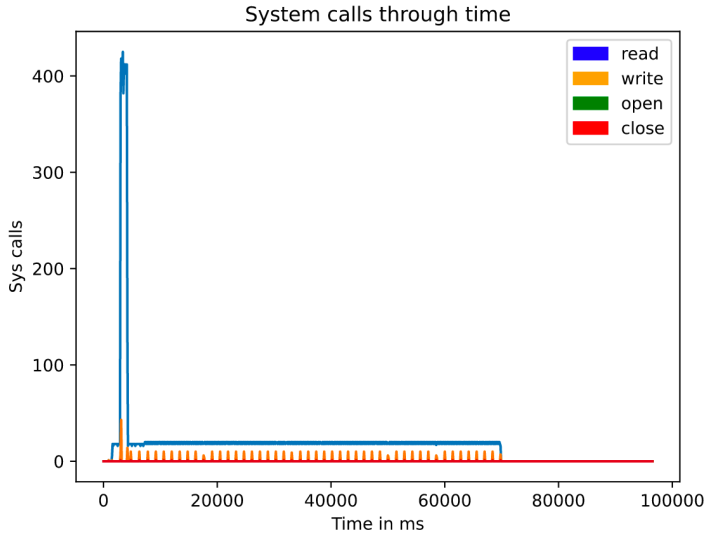
Proc/maps

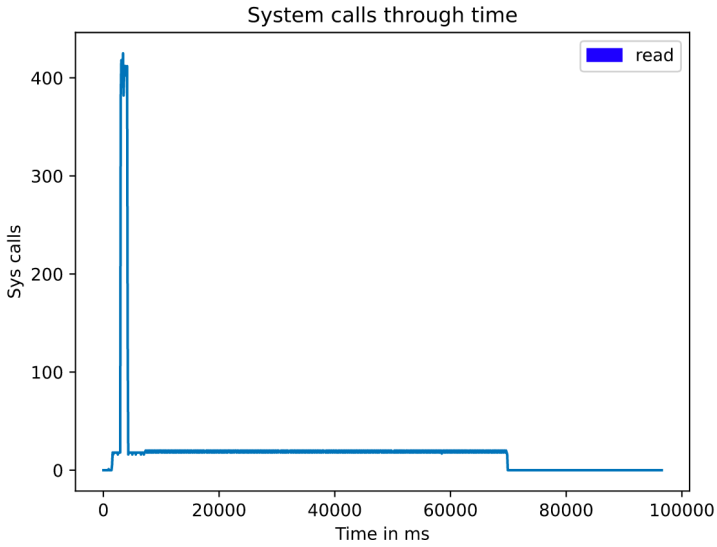
```
55d013b34000-55d013b35000 r--p 00000000 08:05 2233852                    /home/parallels/Desktop/s263977/main
55d013b35000-55d013b36000 r-xp 00001000 08:05 2233852                    /home/parallels/Desktop/s263977/main
55d013b36000-55d013b37000 r--p 00002000 08:05 2233852                    /home/parallels/Desktop/s263977/main
55d013b37000-55d013b38000 r--p 00002000 08:05 2233852                    /home/parallels/Desktop/s263977/main
55d013b38000-55d013b39000 rw-p 00003000 08:05 2233852                    /home/parallels/Desktop/s263977/main
55d014f0e000-55d014f2f000 rw-p 00000000 00:00 0                          [heap]
7f5fd77dc000-7f5fd77df000 rw-p 00000000 00:00 0
7f5fd77df000-7f5fd7804000 r--p 00000000 08:05 2627295                    /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f5fd7804000-7f5fd797c000 r-xp 00025000 08:05 2627295                    /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f5fd797c000-7f5fd79c6000 r--p 0019d000 08:05 2627295                    /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f5fd79c6000-7f5fd79c7000 ---p 001e7000 08:05 2627295                    /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f5fd79c7000-7f5fd79ca000 r--p 001e7000 08:05 2627295                    /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f5fd79ca000-7f5fd79cd000 rw-p 001ea000 08:05 2627295                    /usr/lib/x86_64-linux-gnu/libc-2.31.so
7f5fd79cd000-7f5fd79d1000 rw-p 00000000 00:00 0
7f5fd79d1000-7f5fd79d8000 r--p 00000000 08:05 2628198                    /usr/lib/x86_64-linux-gnu/libpthread-2.31.so
7f5fd79d8000-7f5fd79e9000 r-xp 00007000 08:05 2628198                    /usr/lib/x86_64-linux-gnu/libpthread-2.31.so
7f5fd79e9000-7f5fd79ee000 r--p 00018000 08:05 2628198                    /usr/lib/x86_64-linux-gnu/libpthread-2.31.so
7f5fd79ee000-7f5fd79ef000 r--p 0001c000 08:05 2628198                    /usr/lib/x86_64-linux-gnu/libpthread-2.31.so
7f5fd79ef000-7f5fd79f0000 rw-p 0001d000 08:05 2628198                    /usr/lib/x86_64-linux-gnu/libpthread-2.31.so
7f5fd79f0000-7f5fd79f6000 rw-p 00000000 00:00 0
7f5fd7a07000-7f5fd7a08000 r--p 00000000 08:05 2627082                    /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f5fd7a08000-7f5fd7a2b000 r-xp 00001000 08:05 2627082                    /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f5fd7a2b000-7f5fd7a33000 r--p 00024000 08:05 2627082                    /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f5fd7a34000-7f5fd7a35000 r--p 0002c000 08:05 2627082                    /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f5fd7a35000-7f5fd7a36000 rw-p 0002d000 08:05 2627082                    /usr/lib/x86_64-linux-gnu/ld-2.31.so
7f5fd7a36000-7f5fd7a37000 rw-p 00000000 00:00 0
7ffc3ff3c000-7ffc3ff5d000 rw-p 00000000 00:00 0                          [stack]
7ffc3fff1000-7ffc3fff5000 r--p 00000000 00:00 0                          [vvar]
7ffc3fff5000-7ffc3fff7000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
```

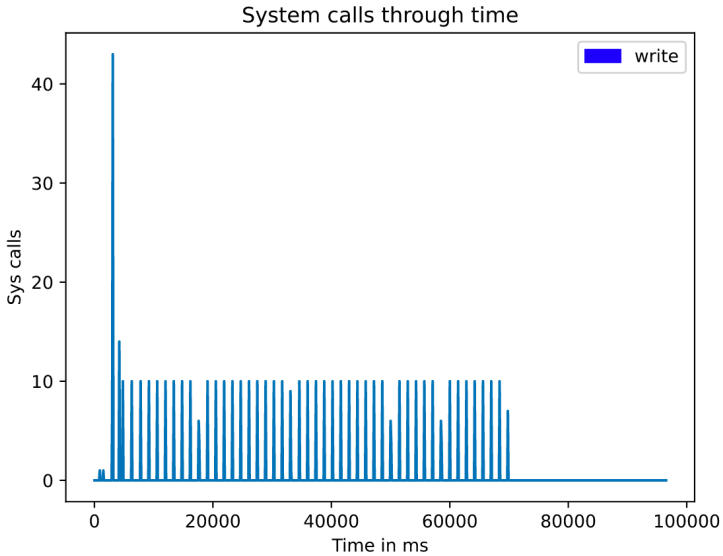
stap скрипт

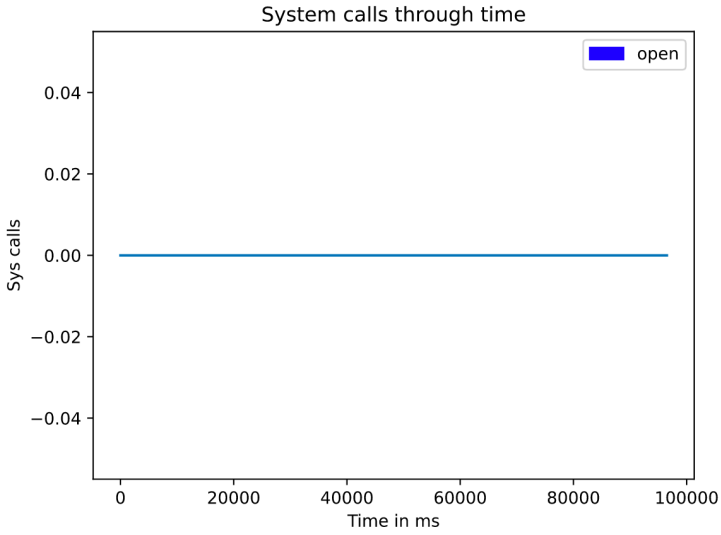
```
global read, write, start, open, close

probe begin {
   start = gettimeofday_s()
}
probe syscall.write {
   if (pid() == target())
    write += 1
}

probe syscall.read {
   if (pid() == target())
    read += 1
}

probe syscall.open {
   if (pid() == target())
    open += 1
}

probe syscall.close {
   if (pid() == target())
    close += 1
}

probe timer.ms(100) {
   printf("%d\t%d\t%d\t%d\t%16d\n", read, write, open, close, task_stime())
   read=0
   write=0
   open = 0
   close = 0
}
```

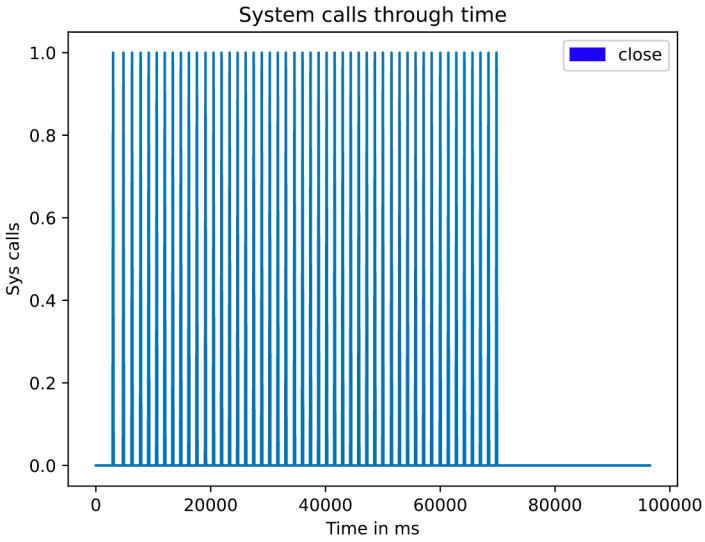
stap графики








System calls through time
open
0.04
0.02
0.00
−0.02
−0.04
Sys calls
0
20000
40000
60000
80000
100000
Time in ms


System calls through time
close
1.0
0.8
0.6
0.4
0.2
0.0
Sys calls
0
20000
40000
60000
80000
100000
Time in ms

Вывод

В ходе лабораторной работы мы получили опыт написания программ на языке C с использованием системных вызовов. А также познакомились с командами операционной системы Linux.