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 <sys/stat.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 {
    char* filename;
    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);

    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;
    write_args* args;
    threads = (pthread_t*) malloc((count + 1) * sizeof(pthread_t));
    args = (write_args*) malloc((count + 1) * sizeof(write_args));
    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
        int file_number = i % files_count;
        char* filename = generate_filename(file_number);
        args[i].number = i + 1;
        args[i].filename = filename;
        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++) {
        fds[i] = open(generate_filename(i), O_RDWR | O_CREAT | O_TRUNC,
(mode_t) 0600);
    }
    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");
        getchar();
        fill_memory(address);
        printf("after filling memory\n");
        getchar();

        write_in_files(address);
```

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

Замеры

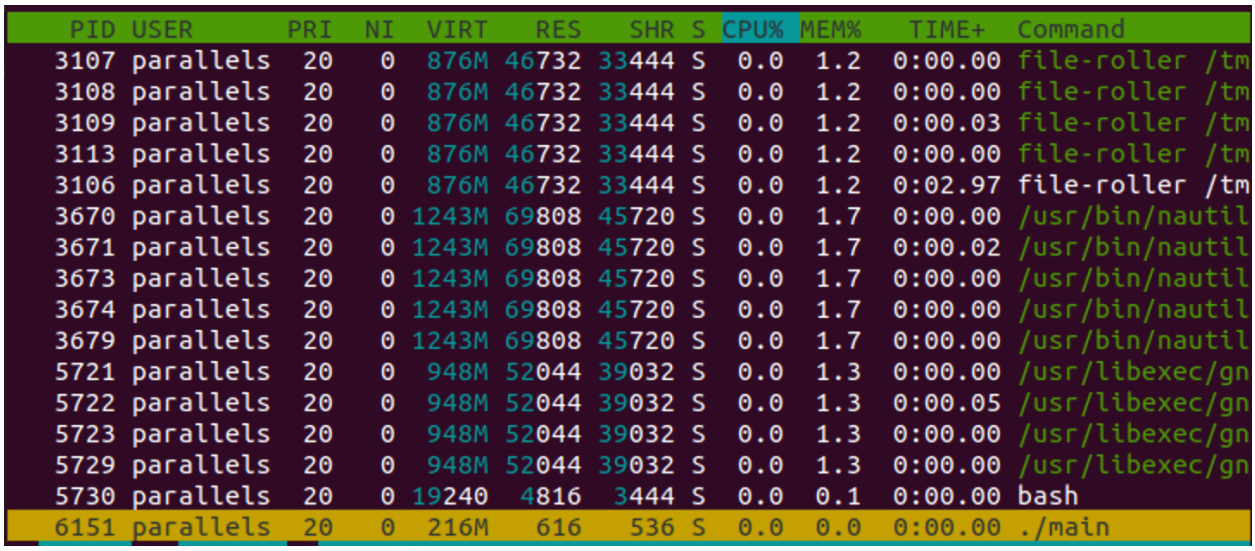
До аллокации

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
3109	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:00.03	file-roller /tm
3113	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:00.00	file-roller /tm
3106	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:02.97	file-roller /tm
3670	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3671	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.02	/usr/bin/nautil
3673	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3674	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3679	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3669	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:08.82	/usr/bin/nautil
5721	parallels	20	0	948M	52056	39044	S	0.0	1.3	0:00.00	/usr/libexec/gn
5722	parallels	20	0	948M	52056	39044	S	0.0	1.3	0:00.05	/usr/libexec/gn
5723	parallels	20	0	948M	52056	39044	S	0.0	1.3	0:00.00	/usr/libexec/gn
5729	parallels	20	0	948M	52056	39044	S	0.0	1.3	0:00.00	/usr/libexec/gn
5730	parallels	20	0	19240	4816	3444	S	0.0	0.1	0:00.00	bash
6151	parallels	20	0	2640	616	536	S	0.0	0.0	0:00.00	./main
6174	parallels	20	0	19240	4752	3364	S	0.0	0.1	0:00.01	bash

После аллокации

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
3109	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:00.03	file-roller /tm
3113	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:00.00	file-roller /tm
3106	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:02.97	file-roller /tm
3670	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3671	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.02	/usr/bin/nautil
3673	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3674	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3679	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3669	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:08.82	/usr/bin/nautil
5721	parallels	20	0	948M	52056	39044	S	0.0	1.3	0:00.00	/usr/libexec/gn
5722	parallels	20	0	948M	52056	39044	S	0.0	1.3	0:00.05	/usr/libexec/gn
5723	parallels	20	0	948M	52056	39044	S	0.0	1.3	0:00.00	/usr/libexec/gn
5729	parallels	20	0	948M	52056	39044	S	0.0	1.3	0:00.00	/usr/libexec/gn
5730	parallels	20	0	19240	4816	3444	S	0.0	0.1	0:00.00	bash
6151	parallels	20	0	2640	616	536	S	0.0	0.0	0:00.00	./main
6174	parallels	20	0	19240	4752	3364	S	0.0	0.1	0:00.01	bash

После заполнения участка данными

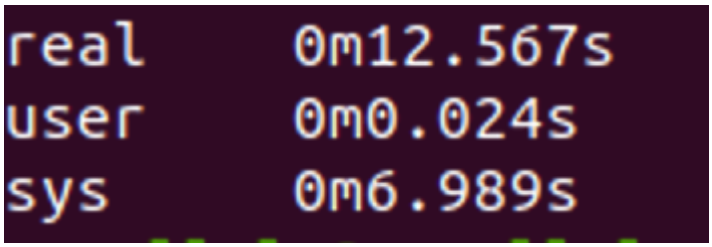


PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
3107	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:00.00	file-roller /tm
3108	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:00.00	file-roller /tm
3109	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:00.03	file-roller /tm
3113	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:00.00	file-roller /tm
3106	parallels	20	0	876M	46732	33444	S	0.0	1.2	0:02.97	file-roller /tm
3670	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3671	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.02	/usr/bin/nautil
3673	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3674	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
3679	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:00.00	/usr/bin/nautil
5721	parallels	20	0	948M	52044	39032	S	0.0	1.3	0:00.00	/usr/libexec/gn
5722	parallels	20	0	948M	52044	39032	S	0.0	1.3	0:00.05	/usr/libexec/gn
5723	parallels	20	0	948M	52044	39032	S	0.0	1.3	0:00.00	/usr/libexec/gn
5729	parallels	20	0	948M	52044	39032	S	0.0	1.3	0:00.00	/usr/libexec/gn
5730	parallels	20	0	19240	4816	3444	S	0.0	0.1	0:00.00	bash
6151	parallels	20	0	216M	616	536	S	0.0	0.0	0:00.00	./main

После деаллокации

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1642	parallels	20	0	4071M	341M	123M	S	0.0	8.7	0:23.77	/usr/bin/gnome-
1641	parallels	20	0	4071M	341M	123M	S	0.7	8.7	0:23.16	/usr/bin/gnome-
2039	parallels	20	0	3590M	324M	147M	S	0.0	8.3	0:16.73	/usr/lib/firefo
2192	parallels	20	0	2732M	341M	152M	S	0.7	8.7	0:40.79	/usr/lib/firefo
2357	parallels	20	0	2732M	341M	152M	S	0.7	8.7	0:24.97	/usr/lib/firefo
2026	parallels	20	0	3590M	324M	147M	S	0.7	8.3	0:32.79	/usr/lib/firefo
2359	parallels	20	0	2732M	341M	152M	S	0.7	8.7	0:19.16	/usr/lib/firefo
1493	parallels	20	0	384M	7704	5928	S	0.0	0.2	0:01.48	/usr/bin/ibus-d
1542	parallels	20	0	667M	57828	36732	S	0.0	1.4	0:00.17	/usr/libexec/ib
6750	parallels	20	0	432M	26660	14948	S	0.0	0.7	0:00.07	/usr/libexec/tr
3669	parallels	20	0	1243M	69808	45720	S	0.0	1.7	0:08.87	/usr/bin/nautil
1295	parallels	20	0	20516	11844	8224	S	0.0	0.3	0:00.75	/lib/systemd/sy
306	root	19	-1	70084	18144	16480	S	0.0	0.5	0:00.51	/lib/systemd/sy
1307	parallels	20	0	242M	7792	6776	S	0.0	0.2	0:00.04	/usr/bin/gnome-
6754	parallels	20	0	432M	26660	14948	S	0.0	0.7	0:00.01	/usr/libexec/tr
6151	parallels	20	0	522M	2096	1488	S	0.0	0.1	0:06.77	./main

Время выполнения



```
real    0m12.567s
user    0m0.024s
sys     0m6.989s
```

Трасса системных вызовов

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/OS$ strace ./main
execve("./main", ["./main"], 0x7ffe241366d0 /* 51 vars */) = 0
brk(NULL)                               = 0x55e660561000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fff5d7bd080) = -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) = 0x7f9b9dc2f000
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) = 0x7f9b9dc2d000
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) = 0x7f9b9dc0a000
mmap(0x7f9b9dc11000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f9
b9dc11000
mmap(0x7f9b9dc22000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f9b9dc22000
mmap(0x7f9b9dc27000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7f
9b9dc27000
mmap(0x7f9b9dc29000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9b9d
c29000
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", 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", 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) = 0x7f9b9da18000
mprotect(0x7f9b9da3d000, 1847296, PROT_NONE) = 0
mmap(0x7f9b9da3d000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x
```

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.