vФедеральное государственное автономное образовательное учреждение высшего образования

«**Санкт-Петербургский национальный исследовательский**

университет информационных технологий, механики и оптики»

Факультет ПИиКТ

Дисциплина: Операционные системы

Лабораторная работа № 1

Выполнил: Камышанская Ксения Васильевна

Преподаватель: Покид Александр Владимирович

Группа: P33122

Вариант: A=49; B=0x82DC563B; C=malloc; D=127; E=152; F=block; G=145; H=random; I=55; J=avg; K=futex

Санкт-Петербург 2020г.

Задание

Разработать программу на языке С, которая осуществляет следующие действия

- Создает область памяти размером **49** мегабайт, начинающихся с адреса **0x82DC563B** при помощи **malloc** заполненную случайными числами /dev/urandom в **127** 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 **152** мегабайт с использованием **блочного** обращения к диску. Размер блока ввода-вывода **145** байт. Последовательность записи/чтения блоков - **случайная**
- Генерацию данных и запись осуществлять в бесконечном цикле.
- В отдельных **55** потоках осуществлять чтение данных из файлов и подсчитывать агрегированные характеристики данных - **среднее значение**.
- Чтение и запись данных в/из файла должна быть защищена примитивами синхронизации **futex**.
- 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.

Выполнение

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <linux/futex.h>
#include <syscall.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
```

```
const int SIZE_MEMORY = 49*1024*1024;
const int SIZE_FILE = 152*1024*1024;
const int SIZE_BLOCK = 145;
const int NUM_THREADS_READ = 127;
const int NUM_THREADS_WR = 5;
const int NUM_THREADS_AVG = 55;

char* memory;

typedef struct FutexRead
{
	int fileFutexRead;
	char nameFile[2];
}FutexRead;

typedef struct DataForRead
{
	int fileDescriptor;
	int numberOfBytes;
	char* adrMemory;
} DataForRead;

int futex_wait(int *addr, int val) { return syscall(SYS_futex, addr, FUTEX_WAIT, val,
NULL, NULL, 0); }
int futex_wake(int *addr, int val) { return syscall(SYS_futex, addr, FUTEX_WAKE, val,
NULL, NULL, 0); }

void* readFile(void* args){
	DataForRead *data = (DataForRead*) args;
	read (data->fileDescriptor, data->adrMemory, data->numberOfBytes);
	pthread_exit(0);
}

void* fillFile(void* args){

	FutexRead *fut = (FutexRead*) args;
	int flags = O_TRUNC | O_CREAT | O_WRONLY ;
	mode_t mode = S_IRUSR | S_IWUSR;
	int file_wr = open (fut->nameFile, flags, mode);

	for (int i = 0; i < SIZE_FILE; i += SIZE_BLOCK)
	{
		const char * buffer = memory + rand() % (SIZE_MEMORY - SIZE_BLOCK + 1);
		write(file_wr, buffer, SIZE_BLOCK);
	}
	close (file_wr);
	futex_wake(&(fut->fileFutexRead), NUM_THREADS_AVG/NUM_THREADS_WR);
	pthread_exit(0);
}

void* avg(void* args){

	FutexRead *fut = (FutexRead*) args;
	futex_wait(&(fut->fileFutexRead), 0);
	int buffer[SIZE_BLOCK];
	int file_avg;
	int avg = 0;
```

```
	int offset;

	file_avg = open (fut->nameFile, O_RDONLY);

	for (int i = 0; i < 2*(SIZE_FILE/SIZE_BLOCK); ++i)
	{
		offset = rand() % SIZE_FILE - SIZE_BLOCK + 1;
		lseek (file_avg, offset, SEEK_SET);
		read (file_avg, buffer, SIZE_BLOCK);
		for (int i = 0; i < SIZE_BLOCK; ++i) avg += buffer[i];
	}
	avg = avg/(2*(SIZE_FILE/SIZE_BLOCK));
	close (file_avg);
	pthread_exit(0);
}

int main()
{
	//before allocation
	memory = (char*)malloc(SIZE_MEMORY);
	//after allocation
	int numberOfBytes = SIZE_MEMORY/NUM_THREADS_READ;
	int fileRand = open ("/dev/urandom", O_RDONLY);
	pthread_t thread_read[NUM_THREADS_READ+1];

	DataForRead mas[NUM_THREADS_READ+1];
	for(int i=0; i<NUM_THREADS_READ;++i){
		mas[i].fileDescriptor = fileRand;
		mas[i].adrMemory = memory + i*numberOfBytes;
		mas[i].numberOfBytes = numberOfBytes;
	}

	for (int i = 0; i < NUM_THREADS_READ; ++i)
		pthread_create(&thread_read[i], NULL, readFile, &mas[i]);

	for (int i = 0; i < NUM_THREADS_READ; ++i)
		pthread_join(thread_read[i], NULL);

	if(SIZE_MEMORY % NUM_THREADS_READ != 0){
		mas[NUM_THREADS_READ+1].fileDescriptor = fileRand;
		mas[NUM_THREADS_READ+1].numberOfBytes = SIZE_MEMORY % NUM_THREADS_READ;
		mas[NUM_THREADS_READ+1].adrMemory = mas[NUM_THREADS_READ].adrMemory +
mas[NUM_THREADS_READ+1].numberOfBytes;
		pthread_create(&thread_read[NUM_THREADS_READ+1], NULL, readFile,
&mas[NUM_THREADS_READ+1]);
	}

	//after data filling
	close(fileRand);

	FutexRead fut[NUM_THREADS_WR];
	for (int i = 0; i < NUM_THREADS_WR; ++i)
	{
		fut[i].fileFutexRead = 0;
	}
	while(1){
		pthread_t thread_wr[NUM_THREADS_WR];
```

```
            pthread_t thread_avg[NUM_THREADS_AVG];
            for (int i = 0; i < NUM_THREADS_WR; ++i)
            {
                  for (int j = i*(NUM_THREADS_AVG/NUM_THREADS_WR);
                   j < (NUM_THREADS_AVG/NUM_THREADS_WR) +
i*(NUM_THREADS_AVG/NUM_THREADS_WR); ++j)
                  {
                        pthread_create(&thread_avg[j], NULL, avg, &fut[i]);
                  }
                  pthread_create(&thread_wr[i], NULL, fillFile, &fut[i]);
            }

            for (int i = 0; i < NUM_THREADS_WR; ++i){
                  pthread_join(thread_wr[i], NULL);
            }
            for (int j = 0; j < NUM_THREADS_AVG; ++j){
                  pthread_join(thread_avg[j], NULL);
                  printf("%d\n", j);
            }
      }
      char x;
      printf("%s ","Введите любую букву");
      scanf("%s",&x);

      free(memory);
      //after free
      return 0;
}
```

Замеры виртуальной/физической памяти (ps -u)

	виртуальной	**физической**
До аллокации	10688	652
После аллокации	60868	676
После заполнения участка данными	159296	51052
После деаллокации	109116	876

Значения затраченного процессорного времени на выполнение программы и на операции ввода-вывода

time ./main

```
real    5m10.540s
user    1m29.531s
sys     37m17.391s
```

sudo strace -c -f ./main

% time	seconds	usecs/call	calls	errors	syscall
45.01	12855.525953	106	120911692	1	read
42.94	12265.157437	101	120911561	115	lseek
11.61	3317.150993	6783539	489	15	futex
0.42	120.666109	21	5496041		write
0.00	1.101921	5769	191		mprotect
0.00	0.900398	4413	204		mmap
0.00	0.844590	1121	753		rt_sigprocmask
0.00	0.679052	3611	188		madvise
0.00	0.651365	9869	66		openat
0.00	0.371247	1964	189		set_robust_list
0.00	0.314715	1710	184		munmap
0.00	0.032446	172	188		clone
0.00	0.022741	3248	7		fstat
0.00	0.004681	70	66		close
0.00	0.000000	0	4		brk
0.00	0.000000	0	2		rt_sigaction
0.00	0.000000	0	6		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	1		prlimit64
100.00	28563.423648	115	247321837	133	total

Отслеживание трассы системных вызовов (sudo strace -f ./main)

```
execve("./lab1", ["./lab1"], 0x7ffddeac2a68 /* 19 vars */) = 0
brk(NULL)                               = 0x1add000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffecaca1360) = -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=81874, ...}) = 0
mmap(NULL, 81874, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fd972434000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/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@\201\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\1\0\0\300\4\0\0\0\30\0\0\0\0\0\0"..., 48, 792) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\217\350\220\r\206\4\302=\236\260H\221\0\323\0\225"..., 68, 840) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=304656, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd972432000
mmap(NULL, 135600, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd972410000
mmap(0x7fd972417000, 65536, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7fd972417000
mmap(0x7fd972427000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x17000) = 0x7fd972427000
mmap(0x7fd97242c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b000) = 0x7fd97242c000
mmap(0x7fd97242e000, 12720, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd97242e000
close(3)                                = 0
openat(AT_FDCWD, "/lib64/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\340\202\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\0"..., 784, 64) = 784
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\1\0\0\300\4\0\0\0\330\1\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0&\2\315-\346\334g*^\3047\212\316\vJh"..., 68, 896) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=3226024, ...}) = 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, 1880736, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fd972244000
mprotect(0x7fd97226a000, 1687552, PROT_NONE) = 0
mmap(0x7fd97226a000, 1372160, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7fd97226a000
mmap(0x7fd9723b9000, 311296, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x175000) = 0x7fd9723b9000
mmap(0x7fd972406000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c1000) = 0x7fd972406000
mmap(0x7fd97240c000, 12960, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fd97240c000
close(3)                                = 0
```

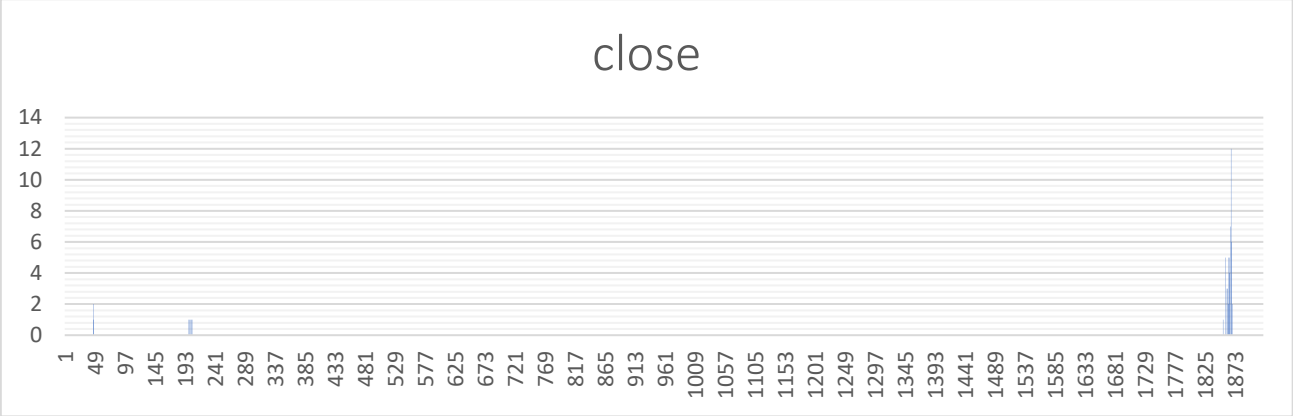
```
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fd972241000
arch_prctl(ARCH_SET_FS, 0x7fd972241740) = 0
mprotect(0x7fd972406000, 12288, PROT_READ) = 0
mprotect(0x7fd97242c000, 4096, PROT_READ) = 0
mprotect(0x403000, 4096, PROT_READ)     = 0
mprotect(0x7fd972473000, 4096, PROT_READ) = 0
munmap(0x7fd972434000, 81874)           = 0
set_tid_address(0x7fd972241a10)         = 11347
set_robust_list(0x7fd972241a20, 24)     = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7fd972417ba0, sa_mask=[], sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7fd9724241e0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7fd972417c40, sa_mask=[], sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7fd9724241e0}, 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
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
brk(NULL)                               = 0x1add000
brk(0x1afe000)                          = 0x1afe000
brk(NULL)                               = 0x1afe000
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
```

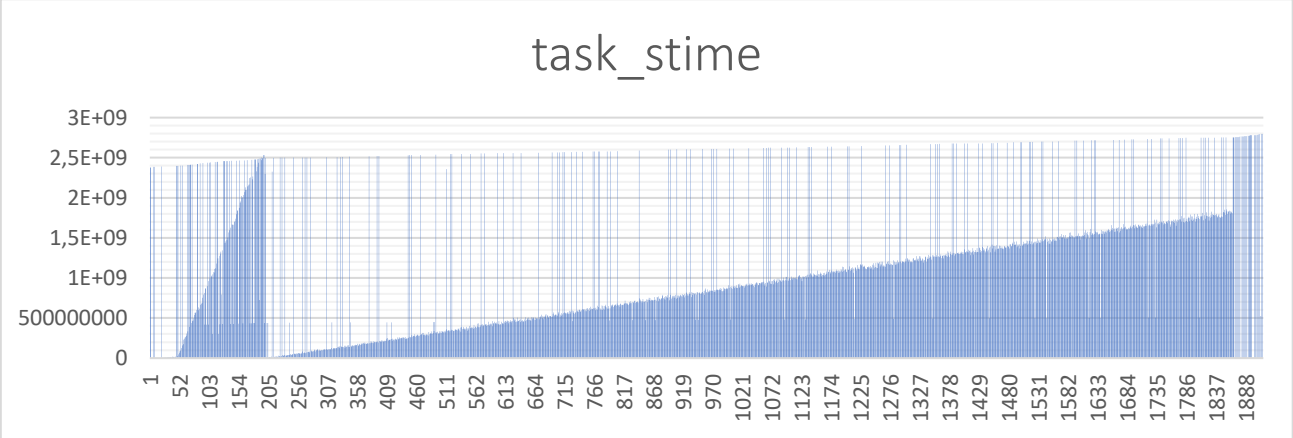
```
openat(AT_FDCWD, "/dev/urandom", O_RDONLY) = 3
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f950dd0d000
mprotect(0x7f950dd0e000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=0x7f950e50cef0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
1307
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f950d50c000
mprotect(0x7f950d50d000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=0x7f950dd0bef0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
1308
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f950cd0b000
mprotect(0x7f950cd0c000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
```

```
[pid 11306] <... rt_sigprocmask resumed>[], 8) = 0
[pid 11309] set_robust_list(0x7f950d50b920, 24 <unfinished ...>
[pid 11310] <... set_robust_list resumed>) = 0
strace: Process 11307 attached
[pid 11306] clone(child_stack=0x7f9508d01ef0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_S
[pid 11309] <... set_robust_list resumed>) = 0
[pid 11308] set_robust_list(0x7f950dd0c920, 24 <unfinished ...>
[pid 11310] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 11309] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 11308] <... set_robust_list resumed>) = 0
[pid 11307] set_robust_list(0x7f950e50d920, 24 <unfinished ...>
[pid 11306] <... clone resumed>, parent_tid=[11318], tls=0x7f9508d02640, child_tidptr=0x7
[pid 11310] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 11309] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 11308] rt_sigprocmask(SIG_SETMASK, [], strace: Process 11317 attached
strace: Process 11316 attached
strace: Process 11315 attached
strace: Process 11314 attached
```

```
[pid 11347] <... write resumed>)        = 2
[pid 11411] +++ exited with 0 +++
[pid 11410] +++ exited with 0 +++
munmap(0x7fd96f23b000, 8392704)         = 0
write(1, "1\n", 21
)                       = 2
munmap(0x7fd9677ff000, 8392704)         = 0
write(1, "2\n", 22
)                       = 2
munmap(0x7fd965ffc000, 8392704)         = 0
write(1, "3\n", 23
)                       = 2
munmap(0x7fd96ca36000, 8392704)         = 0
write(1, "4\n", 24
)                       = 2
munmap(0x7fd96d237000, 8392704)         = 0
write(1, "5\n", 25
)                       = 2
munmap(0x7fd971a40000, 8392704)         = 0
write(1, "6\n", 26
)                       = 2
munmap(0x7fd97123f000, 8392704)         = 0
write(1, "7\n", 27
)                       = 2
munmap(0x7fd97023d000, 8392704)         = 0
write(1, "8\n", 28
)                       = 2
munmap(0x7fd96fa3c000, 8392704)         = 0
write(1, "9\n", 29
)                       = 2
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
```

Графики системных характеристик.





Вывод

В этой лабораторной работе я научилась выделять память с помощью malloc и заполнять ее данными, работать с потоками, синхронизировать их с помощью futex и работать с файлами на языке C. Также я изучила утилиты для анализа работы процессов, такие как ps, strace, stap и time.