Федеральное 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;
		sprintf(fut[i].nameFile,"%d",i);
	}
	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:

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

Вывод