Факультет программной инженерии и компьютерной техники

Операционные системы

Лабораторная работа №1

Вариант № A=193;B=0xACDFC47B;C=malloc;D=119;E=150;F=nocache;G=19;H=seq;I=57;J=sum;K=cv

Преподаватель: Покид Александр Владимирович

Выполнил: Колоколов Артем Михайлович, P33112

Санкт-Петербург

2020

Задание

Разработать программу на языке C, которая осуществляет следующие действия

Создает область памяти размером 193 мегабайт,

начинающихся с адреса 0xACDFC47B (если возможно) при помощи C=malloc заполненную случайными числами /dev/urandom в 119 потоков.

Используя системные средства мониторинга определите адрес начала в адресном пространстве процесса и характеристики выделенных участков памяти.

Замеры виртуальной/физической памяти необходимо снять:

- До аллокации
- После аллокации
- После заполнения участка данными
- После деаллокации

Записывает область памяти в файлы одинакового размера 150 мегабайт с использованием F=некешируемого обращения к диску.

Размер блока ввода-вывода 19 байт.

Преподаватель выдает в качестве задания последовательность записи/чтения блоков H=последовательный

Генерацию данных и запись осуществлять в бесконечном цикле.

В отдельных 57 потоках осуществлять чтение данных из файлов и подсчитывать агрегированные характеристики данных - J=сумму.

Чтение и запись данных в/из файла должна быть защищена примитивами синхронизации K=cv.

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

Код

```
#define _GNU_SOURCE
#include <sys/mman.h>
#include <stdio.h>
#include <limits.h>
#include <pthread.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdint.h>
#include <math.h>
```

```
#include <fcntl.h>
#include <string.h>

#define A 193
#define D 119
#define E 150
#define G 19
#define I 57

int r_cond = 1;
int w_cond = 1;
int number_of_files;
unsigned char *ptr;

typedef struct{
    char * start;
    size_t count;
    FILE * urandom;
} thread_args;

typedef struct {
    pthread_cond_t *cond_vars;
    pthread_mutex_t *mutexes;
} lock_tool;

void* write_random_numbers(void * in_data){
    thread_args *data = (thread_args*) in_data;
    fread((void *) data->start, 1, data->count, data->urandom);
    return NULL;
}

void fill_memory(char * memory_region){
    char * thread_data_start = memory_region;
    FILE * urandom = fopen("/dev/urandom", "r");
    size_t part =  A * 1024 * 1024 / D;
    pthread_t threads[D];
    thread_args threads_data[D];

    for (int i = 0; i < D; i++) {
        if(i==D-1){
            threads_data[i].count = part + (A * 1024 * 1024 % D);
        }else{
            threads_data[i].count = part;
        }
        threads_data[i].urandom = urandom;
        threads_data[i].start = thread_data_start;
        pthread_create(&(threads[i]), NULL, write_random_numbers, &(threads_
data));
        thread_data_start+=part;
    }
```

```
    for (int i = 0; i < D; i++) pthread_join(threads[i], NULL);

    fclose(urandom);
}

void* thread_func_write_files(void* args ){
    lock_tool *lt = (lock_tool*)args;
    int file_handle;
    int block_size;
    int number_of_blocks;
    unsigned char *address;
    char filename[8];

    struct stat buff;
    do{
        for(int i = 0; i < number_of_files; i++){
            pthread_mutex_lock(&lt->mutexes[i]);
            sprintf(filename, "file_%d", i);
            file_handle = open(filename, O_CREAT | O_WRONLY | O_DIRECT | O_T
RUNC, __S_IREAD | __S_IWRITE);
            if(file_handle == -1){
                printf("Smth went wrong with file reading.\n");
                break;
            }

            stat(filename, &buff);
            block_size = (int) buff.st_blksize;
            number_of_blocks = E * 1024 * 256 * sizeof(int) / block_size;
            char *block = (char *) malloc(2 * block_size - 1);
            char *wblock = (char *) (((uintptr_t) block + block_size - 1) &
~((uintptr_t) block_size - 1));

            for(int j = 0; j < number_of_blocks; j++){
                address = ptr + i * (E * 1024 * 1024) + block_size * j;

                memcpy(wblock, address, block_size);
                if(pwrite(file_handle, wblock, block_size, j*block_size) ==
-1){

                    close(file_handle);
                    free(block);
                }
            }
            close(file_handle);
            free(block);
            pthread_cond_broadcast(&lt->cond_vars[i]);
            pthread_mutex_unlock(&lt->mutexes[i]);
            printf("End write\n");
        }
    }while(w_cond);
    return NULL;
```

```
}

void* thread_func_read_files(void* args){
    lock_tool *lt =(lock_tool*) args;
    char filename[8];
    long long number_of_blocks = E * 1024 * 1024 / G;
    do{
        for(int i = 0; i < number_of_files; i++){

            pthread_mutex_lock(&lt->mutexes[i]);
            printf("Waiting cond var for file #%d\n", i);
            pthread_cond_wait(&lt->cond_vars[i], &lt->mutexes[i]);

            long long sum = 0;
            sprintf(filename, "file_%d", i);
            FILE *file = fopen(filename, "r");
            char buffer[G];

            for(long long j = 0; j < number_of_blocks; j++){

                if (fread(&buffer, 1, G, file) != G){
                    continue;
                }
                for(int k = 0; k < G; ++k){
                    int num = *((int *) &buffer[k]);
                    sum += num;
                }
            }
            printf("File #%d has sum=%lld\n", i, sum);
            fclose(file);

            pthread_mutex_unlock(&lt->mutexes[i]);
        }
    }while(r_cond);
    return NULL;
}

void fill_files(){
    number_of_files = A / E;

    lock_tool writer_tools;
    lock_tool reader_tools;

    pthread_cond_t *cond_vars = malloc(sizeof(pthread_cond_t) * number_of_fi
les);
    pthread_mutex_t *mutexes = malloc(sizeof(pthread_mutex_t) * number_of_fi
les);
    for(int i = 0; i < number_of_files; i++){
        pthread_mutex_init(&mutexes[i], NULL);
        pthread_cond_init(&cond_vars[i], NULL);
```

```
        writer_tools.cond_vars = cond_vars;
        writer_tools.mutexes = mutexes;

        reader_tools.cond_vars = cond_vars;
        reader_tools.mutexes = mutexes;
    }

    pthread_t w_thread;
    pthread_create(&w_thread, NULL, thread_func_write_files, &writer_tools);


    pthread_t r_threads[I];
    for(int i = 0; i < I; i++){
        pthread_create(&r_threads[i], NULL, thread_func_read_files, &reader_
tools);
    }
    printf("Press key to stop\n");
    getchar();
    r_cond = 0;
    for(int i = 0; i < I; i++) pthread_join(r_threads[i], NULL);
    w_cond = 0;
    pthread_join(w_thread, NULL);
    free(cond_vars);
    free(mutexes);
}
int main (void)
{
    printf("Start create and fill memory area\n");
    long long int memory_size = A * 1024 * 1024;
    printf("Снять до аллокации\n");
    getchar();
    char *memory_region = malloc(memory_size);
    ptr = (unsigned char *)memory_region;
    printf("Снять после аллокации\n");
    getchar();
    fill_memory(memory_region);
    printf("Start create and fill files\n");
    fill_files();
    printf("Снять после заполнения участка данными\n");
    getchar();
    free(memory_region);
    printf("Снять после деаллокации\n");
    getchar();
}
```

Ход выполнения

```
top - 15:56:34 up 21:26,  0 users,  load average: 0.52, 0.58, 0.59
Tasks:  19 total,   1 running,  18 sleeping,   0 stopped,   0 zombie
%Cpu(s): 15.7 us, 10.1 sy,  0.0 ni, 73.8 id,  0.0 wa,  0.4 hi,  0.0 si,  0.0 st
MiB Mem :   8101.2 total,    882.0 free,   6995.2 used,    224.0 buff/cache
MiB Swap:  24576.0 total,  23110.9 free,   1465.1 used.    975.4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
27751 ifelsee+  20   0 1209868 119140 110812 S   1.7   1.4   2:44.23 node
13071 ifelsee+  20   0   18916   2144   1456 R   0.3   0.0   0:00.79 top
    1 root      20   0    8936    472    428 S   0.0   0.0   0:00.21 init
27656 root      20   0    8948    492    436 S   0.0   0.0   0:00.01 init
27657 ifelsee+  20   0   10656   1832   1800 S   0.0   0.0   0:00.04 sh
27658 ifelsee+  20   0   10656   1868   1836 S   0.0   0.0   0:00.06 sh
27663 ifelsee+  20   0   10656   1844   1812 S   0.0   0.0   0:00.01 sh
27665 ifelsee+  20   0 1003728  47240  39424 S   0.0   0.6   0:15.93 node
27687 ifelsee+  20   0  894364  35116  10448 S   0.0   0.4  27:03.61 node
27772 ifelsee+  20   0  608964  35484  14796 S   0.0   0.4   0:09.50 node
27783 ifelsee+  20   0 1802740  35052  28904 S   0.0   0.4   0:16.00 cpptools
27811 ifelsee+  20   0  562700  26144  16396 S   0.0   0.3   0:00.29 node
11418 ifelsee+  20   0 4889508  25112   6268 S   0.0   0.3   0:00.44 cpptools-srv
11462 root      20   0    8948    496    440 S   0.0   0.0   0:00.01 init
11463 ifelsee+  20   0   18036   4560   4288 S   0.0   0.1   0:00.13 bash
11711 root      20   0    8948    484    428 S   0.0   0.0   0:00.00 init
11712 ifelsee+  20   0   18036   4484   4388 S   0.0   0.1   0:00.16 bash
12929 ifelsee+  20   0   18172   4676   4580 S   0.0   0.1   0:00.04 bash
13437 ifelsee+  20   0  830932 154512    620 S   0.0   1.9   0:53.94 lab1
```

	VIRT	RES
До аллокации	10688	624
После аллокации	208324	628
После заполнения участка данными	1158612	154560
После деаллокации	960976	956

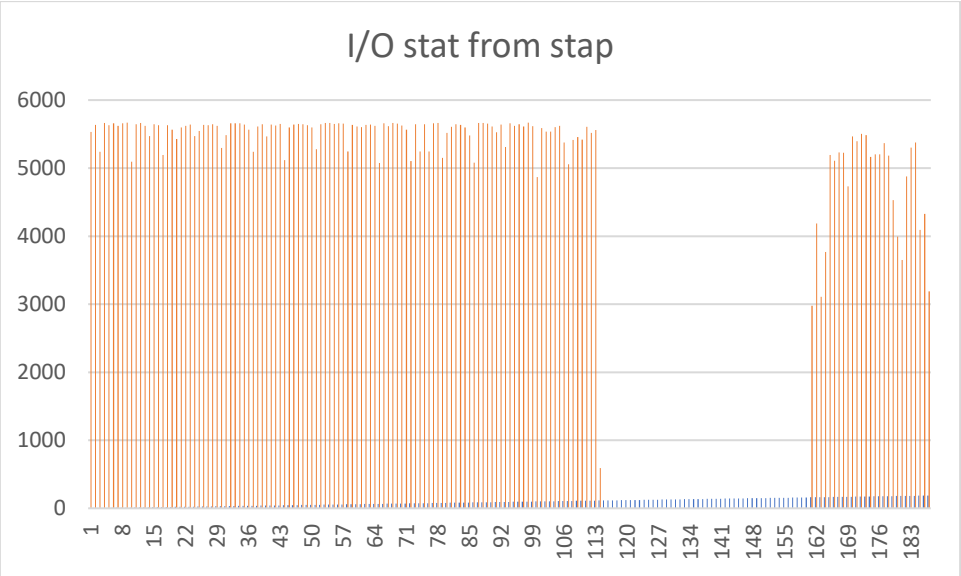
Для измерения значения затраченного процессорного времени на выполнение программы и на операции ввода-вывода использовал iostat и получил следующие данные

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          18.17    0.73    9.02    6.12    0.00   65.95

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
loop0             0.04         0.05         0.00         0.00       1804          0          0
loop1             0.00         0.03         0.00         0.00       1064          0          0
loop2             0.00         0.03         0.00         0.00       1049          0          0
loop3             0.40         0.41         0.00         0.00      14516          0          0
loop4             0.27         0.27         0.00         0.00       9680          0          0
loop5             0.00         0.01         0.00         0.00        348          0          0
loop6             0.00         0.03         0.00         0.00       1067          0          0
sda              42.17        28.47       547.79       458.00    1005239   19343457   16172609
```

Отследить трассу системных вызовов.

```
ifelseelif@DESKTOP-TGK9FA3:~/lab1$ strace ./lab1
execve("./lab1", ["./lab1"], 0x7fffdd03d240 /* 19 vars */) = 0
brk(NULL)                               = 0x7fffe8f8d000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffff1102240) = -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=29824, ...}) = 0
mmap(NULL, 29824, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f069d268000
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\0O\305\3743\364B\2216\244\224\306@\261\23\327o"..., 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) = 0x7f069d2a0000
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\261\23\327o"..., 68, 824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f069d245000
mmap(0x7f069d24c000, 69632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7f069d24c000
mmap(0x7f069d25d000, 20480, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18000) = 0x7f069d25d000
mmap(0x7f069d262000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7f069d262000
mmap(0x7f069d264000, 13432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f069d264000
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\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 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\0\363\377?\332\200\270\27\304d\245n\355Y\377\t\334"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f069d050000
mprotect(0x7f069d075000, 1847296, PROT_NONE) = 0
mmap(0x7f069d075000, 1540096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7f069d075000
mmap(0x7f069d1ed000, 303104, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0x7f069d1ed000
mmap(0x7f069d238000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f069d238000
mmap(0x7f069d23e000, 13528, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f069d23e000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f069d040000
arch_prctl(ARCH_SET_FS, 0x7f069d040740) = 0
mprotect(0x7f069d238000, 12288, PROT_READ) = 0
mprotect(0x7f069d262000, 4096, PROT_READ) = 0
mprotect(0x7f069d2a7000, 4096, PROT_READ) = 0
mprotect(0x7f069d29d000, 4096, PROT_READ) = 0
munmap(0x7f069d268000, 29824)           = 0
set_tid_address(0x7f069d040a10)         = 22854
```



часть вывода pmap

```
4885:   ./lab1
Address           Kbytes     RSS   Dirty Mode  Mapping
0000555f0cda1000       4       4       0 r---- lab1
0000555f0cda2000       4       4       0 r-x-- lab1
0000555f0cda3000       4       4       0 r---- lab1
0000555f0cda4000       4       4       4 r---- lab1
```

```
0000555f0cda5000       4       4       4 rw--- lab1
0000555f0d484000     132      40      40 rw---   [ anon ]
00007f32d7f90000       4       0       0 -----   [ anon ]
00007f32d7f91000    8192       8       8 rw---   [ anon ]
00007f32d8791000       4       0       0 -----   [ anon ]
00007f32d8792000    8192       8       8 rw---   [ anon ]
00007f32d8f92000       4       0       0 -----   [ anon ]
00007f32d8f93000    8192       8       8 rw---   [ anon ]
00007f32d9793000       4       0       0 -----   [ anon ]
00007f32d9794000    8192       8       8 rw---   [ anon ]
00007f32e4000000     132       8       8 rw---   [ anon ]
00007f32e4021000   65404       0       0 -----   [ anon ]
00007f32e8000000     132       8       8 rw---   [ anon ]
00007f32e8021000   65404       0       0 -----   [ anon ]
00007f32ec000000     132       8       8 rw---   [ anon ]
00007f32ec021000   65404       0       0 -----   [ anon ]
00007f32f0000000     132       8       8 rw---   [ anon ]
00007f32f0021000   65404       0       0 -----   [ anon ]
00007f32f4000000     132       8       8 rw---   [ anon ]
```

Вывод

Во время выполнения лабораторной работы я познакомился с языком C и использовал утилиты для мониторинга программы, написанной на этом языке программирования. Особенно было интересно использовать на практике механизмы синхронизации потоков.

В ходе работы возникли трудности с утилитой stap. Для этого пришлось установить разные версии ubuntu, но даже это не решило полностью проблему.