НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Операционные системы

Лабораторная работа № 1

Выполнил
студент

Гораш Вячеслав Игоревич

Группа № P33122

Преподаватель

Покид Александр Владимирович

г. Санкт-Петербург

2020

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.

A=118;B=0x3C28E4CC;C=malloc;D=116;E=14;F=block;G=23;H=random;I=22;J=min;K=sem

Исходный код

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <fcntl.h>
#include <limits.h>

#define MEM_SIZE 118*1024*1024
#define FIRST_ADDRESS 0x3C28E4CC
#define NUM_THREADS_MEMORY 116
#define NUM_THREADS_FILE 22
#define FILE_SIZE 14*1024*1024
#define BLOCK_SIZE 23
#define RANDOM_FILE_NAME "/dev/urandom"
#define FULL_FILES_COUNT ((MEM_SIZE) / (FILE_SIZE))

typedef struct{
    int thread_num;
    int * memory_region;
    int start_number;
    int end_number;
    FILE* file;
}memory_fill_params;

typedef struct{
    int * memory_region;
}file_write_params;

typedef struct{
    int file_number;
    int thread_number;
}file_read_params;
```

```
int cycle_stop = 0;
int min_value = INT_MAX;
//sem_t semaphore_urandom;
sem_t semaphore_file[FULL_FILES_COUNT+1];

int read_int_from_file(FILE *file) {
    int i = 0;
    fread(&i, 4, 1, file);
    return i;
}

void write_to_file(const int * memory_region, int start, int fd, int file_size){
    int num_blocks = file_size / BLOCK_SIZE;
    char * buffer = (char *) malloc(BLOCK_SIZE);
    char * v_memory = (char *) memory_region + start;
    for(int i = 0; i <= num_blocks * 2; i++){
        int block_number = rand() % (num_blocks + 1);
        int start_byte = block_number * BLOCK_SIZE;
        int block_size = block_number == num_blocks ? file_size - (BLOCK_SIZE*num_blocks) :
BLOCK_SIZE;
        for(int byte_num=0; byte_num < block_size; byte_num++){
            buffer[byte_num] = v_memory[start_byte+byte_num];
        }
        pwrite(fd, buffer, block_size, start_byte);
    }
    free(buffer);
}

void read_from_file(int * memory_region, int fd, int file_size){
    int num_blocks = file_size / BLOCK_SIZE;
    char * buffer = (char *) malloc(BLOCK_SIZE);
    char * v_memory = (char *) memory_region;
    for(int i = 0; i <= num_blocks * 2; i++){
        int block_number = rand() % (num_blocks + 1);
        int start_byte = block_number * BLOCK_SIZE;
```

```
        int block_size = block_number == num_blocks ? file_size - (BLOCK_SIZE*num_blocks) :
BLOCK_SIZE;
        pread(fd, buffer, block_size, start_byte);
        for(int byte_num=0; byte_num < block_size; byte_num++){
            v_memory[start_byte+byte_num] = buffer[byte_num];
        }
    }
    free(buffer);
}

void* fill_memory_thread(void * params_void){
    memory_fill_params * params = (memory_fill_params *) params_void;
    printf("[Generator %i] start\n", params->thread_num);
    do{
        for(int i=params->start_number; i<params->end_number; i++){
            //sem_wait(&semaphore_urandom);
            params->memory_region[i]=read_int_from_file(params->file);
            //sem_post(&semaphore_urandom);
        }
    } while (!cycle_stop);
    printf("[Generator %i] finish\n", params->thread_num);
    return NULL;
}

void* file_write_thread(void * params_void){
    file_write_params * params = (file_write_params *) params_void;
    printf("[Writer] start\n");
    do{
        //заполняем файлы
        for(int i=0; i<=FULL_FILES_COUNT; i++){
            char filename[11] = "file_0.bin\0";
            filename[5] = '0' + i;
            sem_wait(&semaphore_file[i]);
            int fd = open(filename, O_CREAT | O_WRONLY | O_TRUNC, 00666);
            if(fd == -1){
                printf("[Writer] can not open file %i\n", i);
```

```
            sem_post(&semaphore_file[i]);
            return NULL;
        }
        //в последний файл сложим остатки
        int file_size = i == FULL_FILES_COUNT ? MEM_SIZE - (FULL_FILES_COUNT *
FILE_SIZE) : FILE_SIZE;
        write_to_file(params->memory_region, FILE_SIZE/4*i, fd, file_size);
        close(fd);
        sem_post(&semaphore_file[i]);
    }
  } while (!cycle_stop);
  printf("[Writer] finish\n");
  return NULL;
}

void* file_read_thread(void * params_void){
  file_read_params * params = (file_read_params *) params_void;
  printf("[Reader %i] start with file %i\n", params->thread_number, params->file_number);
  do{
    //заполняем файлы
    char filename[11] = "file_0.bin\0";
    filename[5] = '0' + params->file_number;
    sem_wait(&semaphore_file[params->file_number]);
    int fd = open(filename, O_RDONLY);
    if(fd == -1){
      sem_post(&semaphore_file[params->file_number]);
      printf("[Reader %i] can not open file %i\n", params->thread_number, params-
>file_number);
      sleep(1);
    }
    int file_size = params->file_number == FULL_FILES_COUNT ? MEM_SIZE -
(FULL_FILES_COUNT * FILE_SIZE) : FILE_SIZE;
    int * memory = (int *) malloc(file_size);
    read_from_file(memory, fd, file_size);
    close(fd);
    sem_post(&semaphore_file[params->file_number]);
    for(int i=0; i<file_size/4; i++){
```

```
            if(memory[i]<min_value){
                min_value = memory[i];
            }
        }
        free(memory);
    } while (!cycle_stop);
    printf("[Reader %i] finish\n", params->thread_number);
    return NULL;
}


void fill_memory(int * memory_region){
    cycle_stop=0;
    //memory filling params
    //sem_init(&semaphore_urandom, 0, 1);
    FILE *random_file = fopen(RANDOM_FILE_NAME, "r");
    pthread_t * memory_fillers = (pthread_t*) malloc(NUM_THREADS_MEMORY *
sizeof(pthread_t));
    memory_fill_params * memory_data = (memory_fill_params *)
malloc(NUM_THREADS_MEMORY * sizeof(memory_fill_params));
    int segment_size = (MEM_SIZE / 4 / NUM_THREADS_MEMORY) + 1;
    for(int i=0; i<NUM_THREADS_MEMORY; i++){
        memory_data[i].thread_num=i;
        memory_data[i].start_number= i * segment_size;
        memory_data[i].end_number = i != NUM_THREADS_MEMORY - 1 ? (i + 1) *
segment_size : MEM_SIZE / 4;
        memory_data[i].memory_region = memory_region;
        memory_data[i].file = random_file;
        pthread_create(&(memory_fillers[i]), NULL, fill_memory_thread, &memory_data[i]);
    }
    //file writing params
    for(int i=0; i<=FULL_FILES_COUNT; i++){
        sem_init(&semaphore_file[i], 0, 1);
    }
    pthread_t * file_writer = (pthread_t *) malloc(sizeof(pthread_t));
    file_write_params * file_write_data = (file_write_params *)
malloc(sizeof(file_write_params));
    file_write_data->memory_region = memory_region;
```

```
    pthread_create(file_writer, NULL, file_write_thread, (void *) file_write_data);


    //file reading params
    file_read_params * file_read_data = (file_read_params *)
malloc(NUM_THREADS_FILE*sizeof(file_read_params));
    pthread_t * file_readers = (pthread_t*) malloc(NUM_THREADS_FILE * sizeof(pthread_t));
    for(int i=0; i<NUM_THREADS_FILE; i++){
        file_read_data[i].thread_number=i;
        file_read_data[i].file_number = rand() % (FULL_FILES_COUNT+1);
        pthread_create(&(file_readers[i]), NULL, file_read_thread, &(file_read_data[i]));
    }


    sleep(1); //чтобы следующая надпись вывелась после старта всех потоков
    printf("Press [Enter] to interrupt infinite loop\n");
    getchar();
    printf("Waiting for all threads finish\n");
    cycle_stop=1;
    //wait for all threads
    for(int i=0; i<NUM_THREADS_MEMORY; i++){
        pthread_join(memory_fillers[i], NULL);
    }
    pthread_join(*file_writer, NULL);
    for(int i=0; i<NUM_THREADS_FILE; i++){
        pthread_join(file_readers[i], NULL);
    }
    //free memory
    free(memory_fillers);
    free(memory_data);
    free(file_writer);
    free(file_write_data);
    free(file_read_data);
    free(file_readers);
    //sem_destroy(&semaphore_urandom);
    for(int i=0; i<=FULL_FILES_COUNT; i++){
        sem_destroy(&semaphore_file[i]);
    }
```

```
    fclose(random_file);
}



int main() {
    printf("Program is starting...\n");
    printf("Pause before memory allocation. Press [Enter] to continue\n");
    getchar();
    printf("Memory allocation...\n");
    int *memory_region = malloc(MEM_SIZE);
    //printf("%i", memory_region[10050000]);
    printf("Pause after memory allocation. Press [Enter] to continue\n");
    getchar();
    printf("Memory filling...\n");
    fill_memory(memory_region);
    //printf("%i", memory_region[10050000]);
    printf("Pause after memory filling. Press [Enter] to continue\n");
    getchar();
    printf("Minimal value is %i\n", min_value);
    printf("Memory releasing...\n");
    free(memory_region);
    return 0;
}
```

Измерение характеристик

Память. Общий замер (free)

До аллокации

```
               всего        занято        свободно      общая  буф./врем.   доступно
Память:      7930152      2963016      1846196       126152      3120940      4541728
Подкачка:      2097148            0      2097148
```

После аллокации

```
               всего        занято        свободно      общая  буф./врем.   доступно
Память:      7930152      2966588      1842132       126276      3121432      4538044
Подкачка:      2097148            0      2097148
```

После заполнения

```
               всего        занято        свободно      общая  буф./врем.    доступно
Память:      7930152      3195456      1612100       126316      3122596      4309128
Подкачка:      2097148            0      2097148
```

После деаллокации

```
               всего        занято        свободно      общая  буф./врем.    доступно
Память:      7930152      2992404      1814412       126448      3123336      4512052
Подкачка:      2097148            0      2097148
```

Память. Замер процесса (pmap -x [pid])

До аллокации

```
8500:   ./lab1
Адрес           Кб     RSS   Dirty Mode  Mapping
00005568e4bc7000       4       4       0 r---- lab1
00005568e4bc8000       4       4       0 r-x-- lab1
00005568e4bc9000       4       4       0 r---- lab1
00005568e4bca000       4       4       4 r---- lab1
00005568e4bcb000       4       4       4 rw--- lab1
00005568e6683000     132       4       4 rw---   [ anon ]
00007f1a591f0000      12       8       8 rw---   [ anon ]
00007f1a591f3000     148     144       0 r---- libc-2.31.so
00007f1a59218000    1504     720       0 r-x-- libc-2.31.so
00007f1a59390000     296     124       0 r---- libc-2.31.so
00007f1a593da000       4       0       0 ----- libc-2.31.so
00007f1a593db000      12      12      12 r---- libc-2.31.so
00007f1a593de000      12      12      12 rw--- libc-2.31.so
00007f1a593e1000      16      16      16 rw---   [ anon ]
00007f1a593e5000      28      28       0 r---- libpthread-2.31.so
00007f1a593ec000      68      68       0 r-x-- libpthread-2.31.so
00007f1a593fd000      20       0       0 r---- libpthread-2.31.so
00007f1a59402000       4       4       4 r---- libpthread-2.31.so
00007f1a59403000       4       4       4 rw--- libpthread-2.31.so
00007f1a59404000      24      12      12 rw---   [ anon ]
00007f1a59424000       4       4       0 r---- ld-2.31.so
00007f1a59425000     140     140       0 r-x-- ld-2.31.so
00007f1a59448000      32      32       0 r---- ld-2.31.so
00007f1a59451000       4       4       4 r---- ld-2.31.so
00007f1a59452000       4       4       4 rw--- ld-2.31.so
00007f1a59453000       4       4       4 rw---   [ anon ]
00007fff378f5000     132      12      12 rw---   [ stack ]
00007fff379f1000      12       0       0 r----   [ anon ]
00007fff379f4000       4       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 --x--   [ anon ]
---------------- ------- ------- -------
всего Кб     2644    1380     104
```

После аллокации (выделен адрес начала сегмента)

```
8500:   ./lab1
Адрес           Кб     RSS   Dirty Mode  Mapping
00005568e4bc7000       4       4       0 r---- lab1
00005568e4bc8000       4       4       0 r-x-- lab1
00005568e4bc9000       4       4       0 r---- lab1
00005568e4bca000       4       4       4 r---- lab1
00005568e4bcb000       4       4       4 rw--- lab1
00005568e6683000     132       4       4 rw---   [ anon ]
00007f1a51bef000  120848      12      12 rw---   [ anon ]
00007f1a591f3000     148     144       0 r---- libc-2.31.so
00007f1a59218000    1504     720       0 r-x-- libc-2.31.so
00007f1a59390000     296     124       0 r---- libc-2.31.so
00007f1a593da000       4       0       0 ----- libc-2.31.so
00007f1a593db000      12      12      12 r---- libc-2.31.so
00007f1a593de000      12      12      12 rw--- libc-2.31.so
00007f1a593e1000      16      16      16 rw---   [ anon ]
00007f1a593e5000      28      28       0 r---- libpthread-2.31.so
00007f1a593ec000      68      68       0 r-x-- libpthread-2.31.so
00007f1a593fd000      20       0       0 r---- libpthread-2.31.so
00007f1a59402000       4       4       4 r---- libpthread-2.31.so
00007f1a59403000       4       4       4 rw--- libpthread-2.31.so
00007f1a59404000      24      12      12 rw---   [ anon ]
00007f1a59424000       4       4       0 r---- ld-2.31.so
00007f1a59425000     140     140       0 r-x-- ld-2.31.so
00007f1a59448000      32      32       0 r---- ld-2.31.so
00007f1a59451000       4       4       4 r---- ld-2.31.so
00007f1a59452000       4       4       4 rw--- ld-2.31.so
00007f1a59453000       4       4       4 rw---   [ anon ]
00007fff378f5000     132      12      12 rw---   [ stack ]
00007fff379f1000      12       0       0 r----   [ anon ]
00007fff379f4000       4       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 --x--   [ anon ]
---------------- ------- ------- -------
всего Кб   123480    1384     108
```

После заполнения

```
00007f1a09783000       4       0       0 -----   [ anon ]
00007f1a09784000    8192       8       8 rw---   [ anon ]
00007f1a48000000   14344   14344   14344 rw---   [ anon ]
00007f1a48e02000   51192       0       0 -----   [ anon ]
00007f1a51bef000  120848  120844  120844 rw---   [ anon ]
00007f1a591f3000     148     144       0 r---- libc-2.31.so
00007f1a59218000    1504    1040       0 r-x-- libc-2.31.so
00007f1a59390000     296     184       0 r---- libc-2.31.so
00007f1a593da000       4       0       0 ----- libc-2.31.so
00007f1a593db000      12      12      12 r---- libc-2.31.so
00007f1a593de000      12      12      12 rw--- libc-2.31.so
00007f1a593e1000      16      16      16 rw---   [ anon ]
00007f1a593e5000      28      28       0 r---- libpthread-2.31.so
00007f1a593ec000      68      68       0 r-x-- libpthread-2.31.so
00007f1a593fd000      20       0       0 r---- libpthread-2.31.so
00007f1a59402000       4       4       4 r---- libpthread-2.31.so
00007f1a59403000       4       4       4 rw--- libpthread-2.31.so
00007f1a59404000      24      12      12 rw---   [ anon ]
00007f1a59424000       4       4       0 r---- ld-2.31.so
00007f1a59425000     140     140       0 r-x-- ld-2.31.so
00007f1a59448000      32      32       0 r---- ld-2.31.so
00007f1a59451000       4       4       4 r---- ld-2.31.so
00007f1a59452000       4       4       4 rw--- ld-2.31.so
00007f1a59453000       4       4       4 rw---   [ anon ]
00007fff378f5000     132      12      12 rw---   [ stack ]
00007fff379f1000      12       0       0 r----   [ anon ]
00007fff379f4000       4       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 --x--   [ anon ]
---------------- ------- ------- -------
всего Кб   746088  223064  221408
```

Замер времени (time ./lab1)

```
real    0m40,892s
user    0m43,476s
sys     0m54,687s
```

Замер времени для ввода-вывода (sudo strace -c -fp [pid])

% time	seconds	usecs/call	calls	errors	syscall
84,16	4249,211985	1103	3851233	1813957	futex
13,12	662,457625	23	28083594		pread64
2,48	125,413936	11	10759299		pwrite64
0,22	11,289260	361	31255		read
0,01	0,291123	291123	1		clock_nanosleep
0,00	0,048580	349	139		clone
0,00	0,044015	153	286		write
0,00	0,031166	202	154		munmap
0,00	0,028506	863	33		openat
0,00	0,028282	181	156		mmap
0,00	0,025567	183	139		set_robust_list
0,00	0,023609	151	156		mprotect
0,00	0,022525	162	139		madvise
0,00	0,013881	420	33		close
0,00	0,000000	0	1		fstat
0,00	0,000000	0	1	1	ioctl
100.00	5048,930060		42726619	1813958	total

Трасса системных вызовов (sudo strace -fp [pid])

```
[pid 11856] pread64(9,  <unfinished ...>
[pid 11855] <... pread64 resumed>"V,\1P\322\366\21Y\204\377\36\v\3072\357\26\210\271\216\316\2
22!\364", 23, 6957431) = 23
[pid 11852] <... pread64 resumed>"\260\210]u\322-\26o\310\3\334K\374K\33\363\30d\3040\356\21}"
, 23, 11228669) = 23
[pid 11851] pread64(5,  <unfinished ...>
[pid 11850] pwrite64(4, "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 23, 4735838 <unfinis
hed ...>
[pid 11744] <... read resumed>"N\213s-\206\333\201f\37\ftsZ;\212<I\233d\271\361\17\316\6\333\2
06\313\2377pFl"..., 4096) = 4096
[pid 11863] <... pread64 resumed>"-\270jdD\360\n%\30\10\6O\336,\301\356\315\210\220v)~\32", 23
, 6629244) = 23
[pid 11861] pread64(10,  <unfinished ...>
[pid 11857] pread64(8,  <unfinished ...>
[pid 11856] <... pread64 resumed>"L\201u\f\21m\3052KS\373*:\267\363~\223\227K\363g>\1", 23, 27
88612) = 23
[pid 11855] pread64(7,  <unfinished ...>
[pid 11852] pread64(6,  <unfinished ...>
[pid 11851] <... pread64 resumed>"\343\200\313jz\3\360\206dG\5\226sQQC\207O\2527\t\377\216", 2
3, 2647231) = 23
[pid 11850] <... pwrite64 resumed>)     = 23
[pid 11744] futex(0x557407ff6ba0, FUTEX_WAKE_PRIVATE, 1 <unfinished ...>
[pid 11863] pread64(11,  <unfinished ...>
[pid 11861] <... pread64 resumed>"\350\263\361\23J,7\243/Qs\32\274\213)\252\177\322\5\n\3303u"
, 23, 10239278) = 23
[pid 11857] <... pread64 resumed>"\217\t\320\200\n\334\313B\366\317\360\372\224Y\276\326C\330\
226\220\226\373\332", 23, 8513105) = 23
[pid 11856] pread64(9,  <unfinished ...>
[pid 11855] <... pread64 resumed>"\r\25;W^Z\343\274\2\353H\247\325\v\277l\325\0n\332)&\230", 2
3, 14397517) = 23
[pid 11852] <... pread64 resumed>"\374\247\301\332A48\240\v9\323R\266\262qAB\267\367\353\345\3
15\202", 23, 13848093) = 23
[pid 11851] pread64(5,  <unfinished ...>
[pid 11850] pwrite64(4, "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 23, 9529268 <unfinis
hed ...>
[pid 11744] <... futex resumed>)        = 1
[pid 11739] <... futex resumed>)        = 0
[pid 11863] <... pread64 resumed>"\26\177'XU\216\256\247\343H\314\261sX!\311\346\360B\362jP\36
3", 23, 2966931) = 23
[pid 11861] pread64(10,  <unfinished ...>
[pid 11857] pread64(8,  <unfinished ...>
```

Stap

```
Pass 1: parsed user script and 477 library scripts using 105340virt/90896res/7372shr/83496data
 kb, in 180usr/40sys/221real ms.
Pass 2: analyzed script: 11 probes, 7 functions, 99 embeds, 8 globals using 173456virt/160076r
es/8436shr/151612data kb, in 1070usr/90sys/1171real ms.
Pass 3: translated to C into "/tmp/stapE39sdK/stap_c1e922999065c02617dde7578b868903_74844_src.
c" using 173456virt/160076res/8436shr/151612data kb, in 10usr/0sys/3real ms.
Pass 4: compiled C into "stap_c1e922999065c02617dde7578b868903_74844.ko" in 2980usr/410sys/335
2real ms.
Pass 5: starting run.
starting probe
^C            name     open     read   MB tot    B avg    write   MB tot    B avg
           lab1       34 27171789      595       22 10759299      235       22
Pass 5: run completed in 30usr/200sys/197308real ms.
```

Вывод

В ходе работы я ознакомился со взаимодействием программы на Си и операционной систем (семафоры, работа с файлами) и средствами мониторинга операционной системы.