

Факультет ПИиКТ

Операционные системы

Лабораторная работа 1

Вариант:

A=160; B=0xC024CD24; C=mmap; D=94; E=68; F=block; G=143; H=seq; I=37; J=min; K=futex

Выполнил: Давтян Д. А.

Группа: P33022

Преподаватель: Покид А. В.

Санкт-Петербург

2020 г.

**Задание**

Разработать программу на языке С, которая осуществляет следующие действия

* Создает область памяти размером A *(160)* мегабайт, начинающихся с адреса B *(0xC024CD24)* (если возможно) при помощи C=mmap заполненную случайными числами /dev/urandom в D *(94)* 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 *(68)* мегабайт с использованием F=*блочного* обращения к диску. Размер блока ввода-вывода G *(143)* байт. Преподаватель выдает в качестве задания последовательность записи/чтения блоков H=*последовательный*
* Генерацию данных и запись осуществлять в бесконечном цикле.
* В отдельных I *(37)* потоках осуществлять чтение данных из файлов и подсчитывать агрегированные характеристики данных - J=*минимальное значение*.
* Чтение и запись данных в/из файла должна быть защищена примитивами синхронизации K=*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.

**Код (solution.c)**

#include <stdlib.h>

#include <sys/mman.h>

#include <errno.h>

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <string.h>

#include <signal.h>

#include <linux/futex.h>

#include <sys/syscall.h>

#include <sys/time.h>

#include <sys/ioctl.h>

#include <stdatomic.h>

#include "nyan\_cat.c"

// Wrapper for futex

int futex**(**int**\*** uaddr**,** int futex\_op**,** int val**,** const struct timespec**\*** timeout**,** int**\*** uaddr2**,** int val3**)** **{**

**return** syscall**(**SYS\_futex**,** uaddr**,** futex\_op**,** val**,** timeout**,** uaddr2**,** val3**);**

**}**

#define \_GNU\_SOURCE

#define MMAP\_ADDRESS (void\*) 0xC024CD24

#define MMAP\_LENGTH 160000000 // 160 MB

#define MMAP\_PROTECTION PROT\_READ | PROT\_WRITE

#define MMAP\_FLAGS MAP\_PRIVATE | MAP\_ANONYMOUS

#define MMAP\_FD -1 // This is ignored because of flags = MAP\_ANONYMOUS, but some systems require this to be -1

#define MMAP\_OFFSET 0

#define RANDOM\_SRC "/dev/urandom"

#define FILL\_THREADS 94

#define WRITE\_FILES\_SIZE 68000000 // 68 MB

#define IO\_BLOCK\_SIZE 143

#define FILE\_READ\_THREADS 37

#define FILE\_NAME\_PREFIX "file"

#define FILE\_MODE S\_IRWXU | S\_IRGRP | S\_IROTH

#define FILES\_NUM MMAP\_LENGTH/WRITE\_FILES\_SIZE + (MMAP\_LENGTH % WRITE\_FILES\_SIZE == 0 ? 0 : 1)

void startFileAnalyzingJob**(**int**);**

void**\*** fillThreadHandler**(**void**\*);**

void**\*** fileAnalyzeHandler**(**void**\*);**

void analyzeFileContent**(**char**\*,** int**,** int**);**

void generateData**();**

void writeData**();**

void writeRegionToFile**(**char**\*,** void**\*,** int**);**

void drawFileWritingProgress**(**char**\*,** int**,** int**);**

void clearFileWritingProgress**();**

void freeData**();**

void fileWriteLock**(**int**);**

void fileReadLock**(**int**);**

void fileWriteUnlock**(**int**);**

void fileReadUnlock**(**int**);**

void waitOnFutexValue**(**int**\*** addr**,** int value**);**

void wakeFutexBlocking**(**int**\*** addr**);**

void signalHandler**(**int**);**

void printAhh**();**

void printVan**();**

void**\*** targetRegionPtr**;**

int randomFd**;**

struct RWLock **{**

int fileCreated**;**

int syncLock**;**

int writeLock**;**

int readers**;** // To change this value, the syncLock must be acquired

**}** rwLocks**[**FILES\_NUM**]** **=** **{**0**};**

int main**()** **{**

signal**(**SIGINT**,** signalHandler**);**

signal**(**SIGSEGV**,** signalHandler**);**

printVan**();**

startFileAnalyzingJob**(**FILE\_READ\_THREADS**);**

**while** **(**1**)** **{**

generateData**();**

writeData**();**

freeData**();**

// printAhh();

**}**

**}**

void generateData**()** **{**

printf**(**"\n---------- GENERATING \033[1;35m♂️DATA♂️\033[0m ----------\n\n"**);**

printf**(**"Trying to pass \033[1;35m♂️fingers♂️\033[0m into computer memory's \033[1;35m♂️ass♂️\033[0m\n"**);**

targetRegionPtr **=** mmap**(**MMAP\_ADDRESS**,** MMAP\_LENGTH**,** MMAP\_PROTECTION**,** MMAP\_FLAGS**,** MMAP\_FD**,** MMAP\_OFFSET**);**

**if** **(**targetRegionPtr **==** MAP\_FAILED**)** **{**

printf**(**"\033[1;35mOh shit, I'm sorry\033[0m\n"**);**

printf**(**"Could not do mmap, errno = %d\n"**,** errno**);**

exit**(-**1**);**

**}** **else** **{**

printf**(**"mmap is done successfully, region created at %p\n"**,** targetRegionPtr**);**

**}**

printf**(**"Filling up memory's \033[1;35m♂️ass♂️\033[0m with \033[1;35m♂️data♂️\033[0m using %d threads\n"**,** FILL\_THREADS**);**

randomFd **=** open**(**RANDOM\_SRC**,** O\_RDONLY**);**

**if** **(**randomFd **<** 0**)** **{**

printf**(**"\033[1;35m- Oh shit, I'm sorry\033[0m\n"**);**

printf**(**"\033[1;35m- Sorry for what?\033[0m\n"**);**

printf**(**"- Could not open file %s, errno = %d\n"**,** RANDOM\_SRC**,** errno**);**

exit**(-**1**);**

**}**

pthread\_t threads**[**FILL\_THREADS**];**

int threadArg**[**FILL\_THREADS**];**

**for** **(**int i **=** 0**;** i **<** FILL\_THREADS**;** i**++)** **{**

threadArg**[**i**]** **=** i**;**

pthread\_create**(&**threads**[**i**],** **NULL,** fillThreadHandler**,** **&**threadArg**[**i**]);**

**}**

printf**(**"Waiting for threads to \033[1;35m♂️join♂️\033[0m our \033[1;35m♂️leather club♂️\033[0m...\n"**);**

**for** **(**int i **=** 0**;** i **<** FILL\_THREADS**;** i**++)**

pthread\_join**(**threads**[**i**],** **NULL);**

close**(**randomFd**);**

printf**(**"Memory \033[1;35m♂️fisting♂️\033[0m is done successfully with %d \033[1;35m♂️fingers♂️\033[0m\n"**,** FILL\_THREADS**);**

**}**

void writeData**()** **{**

printf**(**"\n---------- WRITING \033[1;35m♂️DATA♂️\033[0m ----------\n\n"**);**

**for** **(**int i **=** 0**;** i **<** FILES\_NUM**;** i**++)** **{**

char iString**[]** **=** **{**'0' **+** i**,** '\0'**};**

char filename**[**strlen**(**FILE\_NAME\_PREFIX**)** **+** strlen**(**iString**)];**

strcpy**(**filename**,** FILE\_NAME\_PREFIX**);**

strcat**(**filename**,** iString**);**

int actualSize **=** WRITE\_FILES\_SIZE**;**

**if** **(**i **==** FILES\_NUM **-** 1**)** // Is this the last file?

actualSize **=** MMAP\_LENGTH **-** WRITE\_FILES\_SIZE**\*(**FILES\_NUM **-** 1**);**

fileWriteLock**(**i**);**

writeRegionToFile**(**filename**,** targetRegionPtr **+** WRITE\_FILES\_SIZE **\*** i**,** actualSize**);**

rwLocks**[**i**].**fileCreated **=** 1**;**

wakeFutexBlocking**(&**rwLocks**[**i**].**fileCreated**);** // Wake if someone waits for file creation

fileWriteUnlock**(**i**);**

**}**

**}**

void writeRegionToFile**(**char**\*** name**,** void**\*** regionStart**,** int size**)** **{**

int blocksNum **=** size**/**IO\_BLOCK\_SIZE**;**

**if** **(**IO\_BLOCK\_SIZE**\***blocksNum **<** size**)**

blocksNum**++;**

printf**(**"Writing file '%s' from region at %p, size = %d, will write %d blocks\n"**,** name**,** regionStart**,** size**,** blocksNum**);**

int fd **=** open**(**name**,** O\_CREAT **|** O\_WRONLY**,** FILE\_MODE**);**

**if** **(**fd **<** 0**)** **{**

printf**(**"Could not open file '%s', errno = %d\n"**,** name**,** errno**);**

exit**(-**1**);**

**}**

**for** **(**int i **=** 0**;** i **<** blocksNum**;** i**++)** **{**

int actualBlockSize **=** IO\_BLOCK\_SIZE**;**

**if** **(**i **==** blocksNum **-** 1**)** // Is this the last block?

actualBlockSize **=** size **-** IO\_BLOCK\_SIZE**\*(**blocksNum **-** 1**);**

void**\*** writeStartPtr **=** regionStart **+** i**\***IO\_BLOCK\_SIZE**;**

ssize\_t wrote **=** write**(**fd**,** writeStartPtr**,** actualBlockSize**);**

drawFileWritingProgress**(**name**,** i**,** blocksNum**);**

fflush**(**stdout**);**

**if** **(**wrote **==** **-**1**)** **{**

printf**(**"\n"**);**

printf**(**"Could not write to file '%s' from memory at %p, size = %d, errno = %d\n"**,** name**,** writeStartPtr**,** actualBlockSize**,** errno**);**

exit**(-**1**);**

**}**

**}**

// printf("\r\n");

clearFileWritingProgress**();**

close**(**fd**);**

// printf("%c[2K", 27);

printf**(**"File \033[1;35m♂️filling♂️\033[0m took %d \033[1;35m♂️blocks♂️\033[0m\n\n"**,** blocksNum**);**

**}**

void freeData**()** **{**

munmap**(**targetRegionPtr**,** MMAP\_LENGTH**);**

**}**

void startFileAnalyzingJob**(**int threadNum**)** **{**

printf**(**"Started to analyze output \033[1;35m♂️files♂️\033[0m with %d \033[1;35m♂️dicks♂️\033[0m in background\n"**,** FILE\_READ\_THREADS**);**

pthread\_t**\*** threads **=** **(**pthread\_t**\*)**malloc**(**threadNum**\*sizeof(**pthread\_t**));**

int**\*** threadIndices **=** **(**int**\*)**malloc**(**threadNum**\*sizeof(**int**));**

**for** **(**int i **=** 0**;** i **<** threadNum**;** i**++)** **{**

threadIndices**[**i**]** **=** i**;**

pthread\_create**(&**threads**[**i**],** **NULL,** fileAnalyzeHandler**,** **&**threadIndices**[**i**]);**

**}**

**}**

void**\*** fileAnalyzeHandler**(**void**\*** vargPtr**)** **{**

int index **=** **\*(**int**\*)**vargPtr**;**

**while(**1**)** **{**

**for** **(**int i **=** 0**;** i **<** FILES\_NUM**;** i**++)** **{**

char iString**[]** **=** **{**'0' **+** i**,** '\0'**};**

char filename**[**strlen**(**FILE\_NAME\_PREFIX**)** **+** strlen**(**iString**)];**

strcpy**(**filename**,** FILE\_NAME\_PREFIX**);**

strcat**(**filename**,** iString**);**

analyzeFileContent**(**filename**,** i**,** index**);**

**}**

**}**

**}**

void analyzeFileContent**(**char**\*** fileName**,** int fileRwLockStructIdx**,** int analyzerIdx**)** **{**

fileReadLock**(**fileRwLockStructIdx**);**

int fd **=** open**(**fileName**,** O\_RDONLY**,** FILE\_MODE**);**

**if** **(**fd **<** 0 **&&** errno **==** ENOENT**)** **{** // File does not exist

printf**(**"\033[2K\033[2;49;37m[Analyzer %d] File '%s' doesn't exist yet, read skipped\033[0m\n"**,** analyzerIdx**,** fileName**);**

fileReadUnlock**(**fileRwLockStructIdx**);**

waitOnFutexValue**(&**rwLocks**[**fileRwLockStructIdx**].**fileCreated**,** 0**);** // Wait for file creation

**return;**

**}** **else** **if** **(**fd **<** 0**)** **{**

printf**(**"Error reading file '%s', errno = %d\n"**,** fileName**,** errno**);**

exit**(**1**);**

**}**

off\_t size **=** lseek**(**fd**,** 0L**,** SEEK\_END**);**

lseek**(**fd**,** 0**,** SEEK\_SET**);**

\_\_uint32\_t**\*** content **=** **(**\_\_uint32\_t**\*)** malloc**(**size**);**

ssize\_t readBytes **=** read**(**fd**,** content**,** size**);**

close**(**fd**);**

fileReadUnlock**(**fileRwLockStructIdx**);**

\_\_uint32\_t min **=** content**[**0**];**

**for** **(**size\_t i **=** 0**;** i **<** readBytes**/sizeof(**\_\_uint32\_t**);** i **+=** 1**)** **{**

**if** **(**content**[**i**]** **<** min**)**

min **=** content**[**i**];**

**}**

// http://www.climagic.org/mirrors/VT100\_Escape\_Codes.html

printf**(**"\033[2K\033[2;49;37m[Analyzer %d] Minimal 4-byte uint of file '%s' is %d\033[0m\n"**,** analyzerIdx**,** fileName**,** min**);**

free**(**content**);**

**}**

void fileReadLock**(**int fileIdx**)** **{**

// printf("read lock %d\n", fileIdx);

struct RWLock**\*** lock **=** **&**rwLocks**[**fileIdx**];**

int zero**;**

**do** **{**

zero **=** 0**;**

waitOnFutexValue**(&**lock**->**syncLock**,** 1**);**

**}** **while** **(!**atomic\_compare\_exchange\_strong**(&**lock**->**syncLock**,** **&**zero**,** 1**));**

// function sync area start

lock**->**readers**++;**

**if** **(**lock**->**readers **==** 1**)** **{** // We must acquire the write lock here

**do** **{**

zero **=** 0**;**

waitOnFutexValue**(&**lock**->**writeLock**,** 1**);**

**}** **while** **(!**atomic\_compare\_exchange\_strong**(&**lock**->**writeLock**,** **&**zero**,** 1**));**

**}**

// function sync area end

lock**->**syncLock **=** 0**;**

wakeFutexBlocking**(&**lock**->**syncLock**);**

// printf("read locked %d\n", fileIdx);

**}**

void fileReadUnlock**(**int fileIdx**)** **{**

// printf("read unlock %d\n", fileIdx);

struct RWLock**\*** lock **=** **&**rwLocks**[**fileIdx**];**

int zero**;**

**do** **{**

zero **=** 0**;**

waitOnFutexValue**(&**lock**->**syncLock**,** 1**);**

**}** **while** **(!**atomic\_compare\_exchange\_strong**(&**lock**->**syncLock**,** **&**zero**,** 1**));**

// function sync area start

lock**->**readers**--;**

**if** **(**lock**->**readers **==** 0**)** **{** // We must acquire the write lock here

lock**->**writeLock **=** 0**;**

wakeFutexBlocking**(&**lock**->**writeLock**);**

**}**

// function sync area end

lock**->**syncLock **=** 0**;**

wakeFutexBlocking**(&**lock**->**syncLock**);**

// printf("read unlocked %d\n", fileIdx);

**}**

void fileWriteLock**(**int fileIdx**)** **{**

struct RWLock**\*** lock **=** **&**rwLocks**[**fileIdx**];**

// printf("write lock %d\n", fileIdx);

int zero**;**

**do** **{**

zero **=** 0**;**

waitOnFutexValue**(&**lock**->**writeLock**,** 1**);**

**}** **while** **(!**atomic\_compare\_exchange\_strong**(&**lock**->**writeLock**,** **&**zero**,** 1**));**

// printf("write locked %d\n", fileIdx);

**}**

void fileWriteUnlock**(**int fileIdx**)** **{**

// printf("write unlock %d\n", fileIdx);

struct RWLock**\*** lock **=** **&**rwLocks**[**fileIdx**];**

lock**->**writeLock **=** 0**;**

wakeFutexBlocking**(&**lock**->**writeLock**);**

// printf("write unlocked %d\n", fileIdx);

**}**

void waitOnFutexValue**(**int**\*** addr**,** int value**)** **{**

**while** **(**1**)** **{**

int futexCode **=** futex**(**addr**,** FUTEX\_PRIVATE\_FLAG **|** FUTEX\_WAIT**,** value**,** **NULL,** **NULL,** 0**);**

**if** **(**futexCode **==** **-**1**)** **{**

**if** **(**errno **!=** EAGAIN**)** **{**

printf**(**"\033[1;35mOh shit, I'm sorry\033[0m\n"**);**

printf**(**"Error during futex waiting"**);**

exit**(**1**);**

**}** **else** **if** **(\***addr **!=** value**)** **{**

// Value doesn't match

**return;**

**}**

**}** **else** **if** **(**futexCode **==** 0**)** **{**

**if** **(\***addr **!=** value**)** **{**

// Wake up!

**return;**

**}**

**}** **else** **{**

abort**();**

**}**

**}**

**}**

void wakeFutexBlocking**(**int**\*** addr**)** **{**

**while** **(**1**)** **{**

int futexCode **=** futex**(**addr**,** FUTEX\_WAKE**,** 1**,** **NULL,** **NULL,** 0**);**

**if** **(**futexCode **==** **-**1**)** **{**

printf**(**"\033[1;35mOh shit, I'm sorry\033[0m\n"**);**

printf**(**"Error during futex blocking"**);**

exit**(**1**);**

**}** **else** **if** **(**futexCode **>=** 0**)** **{** // 0 or more waiters woke up

**return;**

**}**

**}**

**}**

void**\*** fillThreadHandler**(**void**\*** vargPtr**)** **{**

int chunkSize **=** MMAP\_LENGTH**/**FILL\_THREADS**;**

int threadIndex **=** **\*(**int**\*)**vargPtr**;**

void**\*** ptrStart **=** **(**void**\*)(**targetRegionPtr **+** threadIndex **\*** chunkSize**);**

**if** **(\*(**int**\*)**vargPtr **==** FILL\_THREADS **-** 1**)** // Am I filling the last chunk?

chunkSize **+=** MMAP\_LENGTH **-** MMAP\_LENGTH**/**FILL\_THREADS**\***FILL\_THREADS**;**

ssize\_t result **=** read**(**randomFd**,** ptrStart**,** chunkSize**);**

**if** **(**result **==** **-**1**)** **{**

printf**(**"Could not fill computer memory's \033[1;35m♂️ass♂️\033[0m at pointer %p, size = %d, errno = %d\n"**,** ptrStart**,** chunkSize**,** errno**);**

exit**(-**1**);**

**}**

**}**

void printAhh**()** **{**

// Secret code

**}**

void printVan**()** **{**

// Secret code

**}**

void drawFileWritingProgress**(**char**\*** fileName**,** int currentProgress**,** int maxProgress**)** **{**

static int previousFrame **=** **-**1**;**

int frame **=** **(**currentProgress**/**750**)** **%** NYAN\_CAT\_FRAMES**;**

char progressBar**[**41**];**

**for** **(**int j **=** 0**;** j **<** 40**;** j**++)**

progressBar**[**j**]** **=** j**/**40.0f **<** 1.0f**\***currentProgress**/**maxProgress **?** '>' **:** ' '**;**

progressBar**[**40**]** **=** '\0'**;**

printf**(**"\033[2K"**);**

printf**(**"Writing '%s': [%s] %d/%d blocks"**,** fileName**,** progressBar**,** currentProgress**,** maxProgress**);**

**if** **(**frame **!=** previousFrame**)** **{**

printf**(**"\n"**);**

**for** **(**int row **=** 0**;** row **<** NYAN\_CAT\_HEIGHT**;** row**++)** **{**

int startingColumn **=** NYAN\_CAT\_WIDTH **-** NYAN\_CAT\_WIDTH**\***currentProgress**/**maxProgress**;**

**for** **(**int column **=** startingColumn**;** column **<** NYAN\_CAT\_WIDTH**;** column**++)** **{**

char symbol **=** nyanCatColorData**[**frame**][**row**][**column**];**

printf**(**"\033[%sm%s\033[0m"**,** nyanCatColorMap**[**symbol**],** NYAN\_CAT\_SYMBOL**);**

**}**

printf**(**"\r\n"**);**

fflush**(**stdout**);**

**}**

**for** **(**int i **=** 0**;** i **<** NYAN\_CAT\_HEIGHT**+**1**;** i**++)**

printf**(**"\033[A"**);**

**}** **else** **{**

printf**(**"\r"**);**

**}**

fflush**(**stdout**);**

previousFrame **=** frame**;**

**}**

void clearFileWritingProgress**()** **{**

// Used buffering to prevent concurrent prints

char buffer**[**5**\*(**NYAN\_CAT\_HEIGHT**+**1**)** **+** 3**\*(**NYAN\_CAT\_HEIGHT**+**1**)** **+** 1**]** **=** **{**0**};**

size\_t ptr **=** 0**;**

**for** **(**int i **=** 0**;** i **<** NYAN\_CAT\_HEIGHT**+**1**;** i**++)** **{**

buffer**[**ptr**++]** **=** '\033'**;**

buffer**[**ptr**++]** **=** '['**;**

buffer**[**ptr**++]** **=** '2'**;**

buffer**[**ptr**++]** **=** 'K'**;**

buffer**[**ptr**++]** **=** '\n'**;**

**}**

**for** **(**int i **=** 0**;** i **<** NYAN\_CAT\_HEIGHT**+**1**;** i**++)** **{**

buffer**[**ptr**++]** **=** '\033'**;**

buffer**[**ptr**++]** **=** '['**;**

buffer**[**ptr**++]** **=** 'A'**;**

**}**

puts**(**buffer**);**

**}**

void signalHandler**(**int signum**)** **{**

**if** **(**signum **==** SIGINT**)** **{**

printf**(**"\n"**);**

exit**(**signum**);**

**}** **else** **if** **(**signum **==** SIGSEGV**)** **{**

printf**(**"♂️Fisting♂️ fault (♂️ass♂️ dumped)\n"**);**

exit**(**signum**);**

**}**

**}**

**Код (nyan\_cat.c)**

#define NYAN\_CAT\_HEIGHT 22

#define NYAN\_CAT\_WIDTH 48

#define NYAN\_CAT\_FRAMES 5

#define NYAN\_CAT\_SYMBOL "██"

/\*\*

\* 0: background color

\* 1: red

\* 2: orange

\* 3: yellow

\* 4: green

\* 5: cyan

\* 6: magenta

\* 7: white

\* 8: dark grey

\* 9: grey

\* a: cookie color

\* b: light rose

\* c: contrast rose

\*/

char**\*** nyanCatColorMap**[**'c'**+**1**]** **=** **{**

**[**'0'**]** **=** "38;5;17"**,**

**[**'1'**]** **=** "38;5;196"**,**

**[**'2'**]** **=** "38;5;208"**,**

**[**'3'**]** **=** "38;5;226"**,**

**[**'4'**]** **=** "38;5;46"**,**

**[**'5'**]** **=** "36;1"**,**

**[**'6'**]** **=** "35;1"**,**

**[**'7'**]** **=** "37;1"**,**

**[**'8'**]** **=** "38;5;234"**,**

**[**'9'**]** **=** "38;5;246"**,**

**[**'a'**]** **=** "38;5;223"**,**

**[**'b'**]** **=** "38;5;207"**,**

**[**'c'**]** **=** "38;5;126"

**};**

char**\*** nyanCatColorData**[**NYAN\_CAT\_FRAMES**][**NYAN\_CAT\_HEIGHT**]** **=** **{**

**{**

"000000000000000000000000000000000000000000000000"**,**

"000000000000000000000008888888888888888800000000"**,**

"00011111100000011111118aaaaaaaaaaaaaaaaa80000000"**,**

"1111111111111111111118aaabbbbbbbbbbbbbaaa8000000"**,**

"1111111111111111111118aabbbbbbcbbcbbbbbaa8000000"**,**

"1112222221111112222228abbcbbbbbbbbbbbbbba8000000"**,**

"2222222222222222222228abbbbbbbbbb88bbcbba8088000"**,**

"2222222222222222222228abbbbbbbbb8998bbbba8899800"**,**

"2223333332222228888338abbbbbcbbb89998bbba8999800"**,**

"3333333333333338998838abbbbbbbbb8999988889999800"**,**

"3333333333333338899888abbbcbbbbb8999999999999800"**,**

"3334444443333334889988abbbbbbbc89999999999999980"**,**

"4444444444444444488998abcbbbbbb89997899999789980"**,**

"4444444444444444448888abbbbbbbb89998899989889980"**,**

"4445555554444445555588abbbbbcbb89cc999999999cc80"**,**

"5555555555555555555558aabcbbbbb89cc989989989cc80"**,**

"5555555555555555555558aaabbbbbbb8999888888899800"**,**

"55566666655555566666888aaaaaaaaaa899999999998000"**,**

"666666666666666666689998888888888888888888880000"**,**

"666666666666666666689988089980000089980899800000"**,**

"666000000666666000088880088800000008880088000000"**,**

"000000000000000000000000000000000000000000000000"

**},** **{**

"000000000000000000000000000000000000000000000000"**,**

"000000000000000000000008888888888888888800000000"**,**

"00011111100000011111118aaaaaaaaaaaaaaaaa80000000"**,**

"1111111111111111111118aaabbbbbbbbbbbbbaaa8000000"**,**

"1111111111111111111118aabbbbbbcbbcbbbbbaa8000000"**,**

"1112222221111112222228abbcbbbbbbbbbbbbbba8000000"**,**

"2222222222222222222228abbbbbbbbbbb88bcbba8008800"**,**

"2222222222222222222228abbbbbbbbbb8998bbba8089980"**,**

"2223333332222223333338abbbbbcbbbb89998bba8899980"**,**

"3333333333333333883338abbbbbbbbbb899998888999980"**,**

"3333333333333338998338abbbcbbbbbb899999999999980"**,**

"3334444443333338998888abbbbbbbcb8999999999999998"**,**

"4444444444444444899998abcbbbbbbb8999789999978998"**,**

"4444444444444444488998abbbbbbbbb8999889998988998"**,**

"4445555554444445555888abbbbbcbbb89cc999999999cc8"**,**

"5555555555555555555558aabcbbbbbb89cc989989989cc8"**,**

"5555555555555555555558aaabbbbbbbb899988888889980"**,**

"55566666655555566666688aaaaaaaaaaa89999999999800"**,**

"666666666666666666668998888888888888888888888000"**,**

"666666666666666666668998089980000008998089980000"**,**

"666000000666666000008880008880000000888008880000"**,**

"000000000000000000000000000000000000000000000000"

**},** **{**

"000000000000000000000000000000000000000000000000"**,**

"000000000000000000000000000000000000000000000000"**,**

"110000001111111100000008888888888888888800000000"**,**

"11111111111111111111118aaaaaaaaaaaaaaaaa80000000"**,**

"1111111111111111111118aaabbbbbbbbbbbbbaaa8000000"**,**

"2211111122222222111118aabbbbbbcbbcbbbbbaa8000000"**,**

"2222222222222222222228abbcbbbbbbbbbbbbbba8000000"**,**

"2222222222222222222228abbbbbbbbbbb88bcbba8008800"**,**

"3322222233333333222228abbbbbbbbbb8998bbba8089980"**,**

"3333333333333333333338abbbbbcbbbb89998bba8899980"**,**

"3333333333333333333338abbbbbbbbbb899998888999980"**,**

"4433333344444443333338abbbcbbbbbb899999999999980"**,**

"4444444444444444444488abbbbbbbcb8999999999999998"**,**

"4444444444444444888888abcbbbbbbb8999789999978998"**,**

"5544444455555558899998abbbbbbbbb8999889998988998"**,**

"5555555555555558999888abbbbbcbbb89cc999999999cc8"**,**

"5555555555555555888858aabcbbbbbb89cc989989989cc8"**,**

"6655555566666666555558aaabbbbbbbb899988888889980"**,**

"66666666666666666666688aaaaaaaaaaa89999999999800"**,**

"666666666666666666666898888888888888888888888000"**,**

"006666660000000066666899808998000000899808998000"**,**

"000000000000000000000888000888000000088800888000"

**},** **{**

"000000000000000000000000000000000000000000000000"**,**

"000000000000000000000000000000000000000000000000"**,**

"110000001111111100000008888888888888888800000000"**,**

"11111111111111111111118aaaaaaaaaaaaaaaaa80000000"**,**

"1111111111111111111118aaabbbbbbbbbbbbbaaa8000000"**,**

"2211111122222222111118aabbbbbbcbbcbbbbbaa8000000"**,**

"2222222222222222222228abbcbbbbbbbbbbbbbba8000000"**,**

"2222222222222222222228abbbbbbbbbbb88bcbba8008800"**,**

"3322222233333333222228abbbbbbbbbb8998bbba8089980"**,**

"3333333333333333333338abbbbbcbbbb89998bba8899980"**,**

"3333333333333333333338abbbbbbbbbb899998888999980"**,**

"4433333344444443333338abbbcbbbbbb899999999999980"**,**

"4444444444444444444888abbbbbbbcb8999999999999998"**,**

"4444444444444444488998abcbbbbbbb8999789999978998"**,**

"5544444455555555899998abbbbbbbbb8999889998988998"**,**

"5555555555555558998888abbbbbcbbb89cc999999999cc8"**,**

"5555555555555558998558aabcbbbbbb89cc989989989cc8"**,**

"6655555566666666885558aaabbbbbbbb899988888889980"**,**

"66666666666666666666688aaaaaaaaaaa89999999999800"**,**

"666666666666666666668998888888888888888888888000"**,**

"006666660000000066668998008998000008998089980000"**,**

"000000000000000000008880000888000000888008880000"

**},** **{**

"000000000000000000000000000000000000000000000000"**,**

"000000000000000000000000000000000000000000000000"**,**

"000111111000000111111118888888888888888800000000"**,**

"11111111111111111111118aaaaaaaaaaaaaaaaa80000000"**,**

"1111111111111111111118aaabbbbbbbbbbbbbaaa8000000"**,**

"1112222221111112222228aabbbbbbcbbcbbbbbaa8000000"**,**

"2222222222222222222228abbcbbbbbbbbbbbbbba8000000"**,**

"2222222222222222222228abbbbbbbbbb88bbcbba8088000"**,**

"2223333332222223333338abbbbbbbbb8998bbbba8899800"**,**

"3333333333333333333338abbbbbcbbb89998bbba8999800"**,**

"3333333333333338888338abbbbbbbbb8999988889999800"**,**

"3334444443333389998888abbbcbbbbb8999999999999800"**,**

"4444444444444488999988abbbbbbbc89999999999999980"**,**

"4444444444444444888898abcbbbbbb89997899999789980"**,**

"4445555554444445555888abbbbbbbb89998899989889980"**,**

"5555555555555555555558abbbbbcbb89cc999999999cc80"**,**

"5555555555555555555558aabcbbbbb89cc989989989cc80"**,**

"5556666665555556666688aaabbbbbbb8999888888899800"**,**

"66666666666666666668888aaaaaaaaaa899999999998000"**,**

"666666666666666666899988888888888888888888880000"**,**

"666000000666666000899808998000000899808998000000"**,**

"000000000000000000888000888000000088800888000000"

**},**

**};**

**Выполнение**

С помощью GDB я остановил программу в нужных местах и измерил используемую память с помощью top.

|  |  |  |
| --- | --- | --- |
| Использование памяти | | |
|  | VIRT (вирт.) | RES (физ.) |
| До аллокации | 3564368 | 1.8g |
| После аллокации | 4915964 | 3.1g |
| После заполнения данными | 4640148 | 2.9g |
| После деаллокации | 2401836 | 751344 |

**Общее потребление**

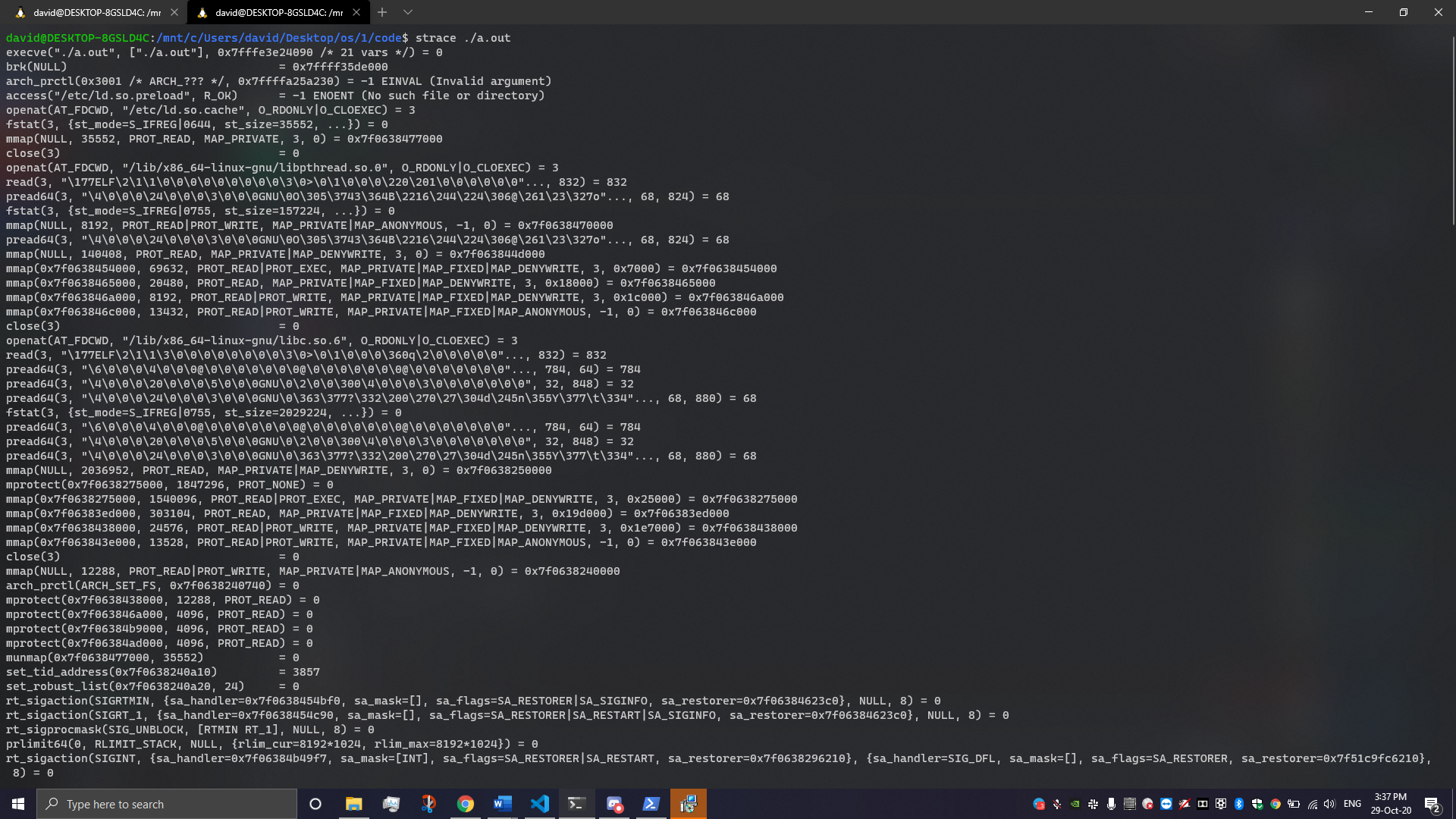
%CPU min = 36  
%CPU max = 45

**Чтение/запись (iostat)**



Для отслеживания системных вызовов воспользуюсь **strace**.

$ strace ./a.out



**Вывод**

Сделав эту лабораторную работу, я научился взаимодействовать с операционной системой с помощью системных вызовов, измерять потребление памяти и использование процессорного времени, научился ставить брейкпоинты в gdb и научился рисовать анимированного Nyancat’а в консоли. А еще я научился писать на С и почитал интересную книгу Таненбаума про операционные системы.