Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

Студент: Литовченко Анна Александровна

Группа: М8О-207Б-21

Вариант: 5

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/Annalitov/OS/lab1

**Постановка задачи**

Цель работы — приобретение практических навыков диагностики работы программного обеспечения.

**Strace**

Strace показывает все системные вызовы программы, которые она отправляет к системе во время выполнения, а также их параметры и результат выполнения. При необходимости можно подключиться к уже запущенному процессу.

**Общий метод и алгоритм решения**

Протестируем программу на примере лабораторной работы №2

В операционной системе macOs strace имеет аналог - dtruss

Описание работы dtruss

munmap - удаляет отображение

ftruncate - устанавливает файлу необходимый размер

mmap - создает новое отображение в памяти в адресном пространстве процесса

close - закрывает файловый дискриптор

open - получив в pathname имя файла, возвращает файловый дескриптор

openat - открывает файл в определенной директории

mprotect - изменяет защиту доступа на ту, которая указана prot для целых страниц, содержащих любую часть адресного пространства процесса, начиная с адреса addr и продолжая для байтов len

ioctl - изменяет базовые параметры устройства, представленного в виде специального файла.

sysctl - используется для изменения параметров ядра во время выполнения. Доступные параметры перечислены в разделе /proc/sys/

fsgetpath - получает путь, связанный с идентификатором узла файловой системы

stat64 - возвращают информацию о файле в буфер, на который указывает buf

**Исходный код**

**main.cpp**

#include <iostream>

#include <string>

#include <cstdlib>

#include <sys/types.h>

#include <unistd.h>

#include <fstream>

#include <errno.h>

#include <signal.h>

#include <sys/wait.h>

using namespace std;

int main(){

fstream f;

string filename;

cout<<"Enter a filename: "<<endl;

cin >> filename;

int fd\_1[2];

int fd\_2[2];

int pipe\_1[2];

int pipe\_2[2];

if (pipe(pipe\_1) == -1){

perror("pipe");

exit(EXIT\_FAILURE);

}

if (pipe(pipe\_2) == -1){

perror("pipe");

exit(EXIT\_FAILURE);

}

string num;

pid\_t id = fork();

if (id == -1){

perror("fork");

exit(EXIT\_FAILURE);

} else if (id == 0) {

fd\_1[0] = pipe\_1[0];

fd\_1[1] = pipe\_1[1];

fd\_2[0] = pipe\_2[0];

fd\_2[1] = pipe\_2[1];

execl("./child", to\_string(fd\_1[0]).c\_str(), to\_string(fd\_1[1]).c\_str(), to\_string(fd\_2[0]).c\_str(), to\_string(fd\_2[1]).c\_str(), filename.c\_str(), NULL);

} else {

cout<<"Enter number of operations: ";

int numberr;

cin>>numberr;

cout<<endl;

for (int i = 0; i < numberr; ++i) {

cout << "Enter a number: " << endl;

cin >> num;

int s\_size = num.size();

char str\_array[s\_size];

for (int k = 0; k < s\_size; ++k) {

str\_array[k] = num[k];

}

write(pipe\_1[1], &s\_size, sizeof(int));

write(pipe\_1[1], str\_array, sizeof(char)\*s\_size);

int flag\_0;

read(pipe\_2[0], &flag\_0, sizeof(int));

if (flag\_0 == 1) {

cout << "The number is prime or negative" << endl;

}

}

}

close(pipe\_1[0]);

close(pipe\_1[1]);

close(pipe\_2[0]);

close(pipe\_2[1]);

return 0;

}

**child.cpp**

#include <iostream>

#include <string>

#include <cstdlib>

#include <sys/types.h>

#include <unistd.h>

#include <fstream>

#include <errno.h>

#include <signal.h>

#include <sys/wait.h>

using namespace std;

int isNotPrime(int n)

{

if (n < 2) {

return 0;

} else {

for (int i = 2; i \* i < n + 1; i++) {

if (n % i == 0) {

return 1;

}

}

}

return 0;

}

int main(int argc, char \*argv[]){

string filename = argv[4];

int fd\_1[2];

int fd\_2[2];

int flag\_1 = 1;

int flag\_2 = 2;

fd\_1[0] = stoi(argv[0]);

fd\_1[1] = stoi(argv[1]);

fd\_2[0] = stoi(argv[2]);

fd\_2[1] = stoi(argv[3]);

fstream f;

f.open(filename, fstream::in | fstream::out | fstream::app);

while(true) {

int num\_size;

read(fd\_1[0], &num\_size, sizeof(int));

char num\_str[num\_size];

read(fd\_1[0], &num\_str, sizeof(char)\*num\_size);

string result;

for (int i = 0; i < num\_size; i++) {

result.push\_back(num\_str[i]);

}

int number;

int number\_1;

number = stoi(result);

number\_1 = abs(number);

if ( number > 0 && isNotPrime(number\_1) > 0 ) {

f << result << endl;

cout << "A number " << result << " is added to file!" << endl;

write(fd\_2[1], &flag\_2, sizeof(int));

} else {

write(fd\_2[1], &flag\_1, sizeof(int));

}

}

return 0

}

**Демонстрация работы программы**

**litann@Annalit lab2 % sudo dtruss -f ./main**

**Password:**

**PID/THRD SYSCALL(args) = return**

**Enter a filename:**

**969/0x241b: fork() = 0 0**

**969/0x241b: munmap(0x102DA4000, 0x8C000) = 0 0**

**969/0x241b: munmap(0x102E30000, 0x8000) = 0 0**

**969/0x241b: munmap(0x102E38000, 0x4000) = 0 0**

**969/0x241b: munmap(0x102E3C000, 0x4000) = 0 0**

**969/0x241b: munmap(0x102E40000, 0x54000) = 0 0**

**969/0x241b: open(".\0", 0x100000, 0x0) = 3 0**

**969/0x241b: fcntl(0x3, 0x32, 0x16D213358) = 0 0**

**969/0x241b: close(0x3) = 0 0**

**969/0x241b: fsgetpath(0x16D213368, 0x400, 0x16D213348) = 32 0**

**969/0x241b: fsgetpath(0x16D213378, 0x400, 0x16D213358) = 14 0**

**969/0x241b: csrctl(0x0, 0x16D21377C, 0x4) = -1 Err#1**

**969/0x241b: \_\_mac\_syscall(0x1AF6E8143, 0x2, 0x16D2136D0) = 0 0**

**969/0x241b: csrctl(0x0, 0x16D21379C, 0x4) = -1 Err#1**

**969/0x241b: \_\_mac\_syscall(0x1AF6E5094, 0x5A, 0x16D213730) = 0 0**

**969/0x241b: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D212CA0, 0x16D212C90, 0x1AF6E6CA1, 0xD) = 0 0**

**969/0x241b: sysctl([CTL\_KERN, 136, 0, 0, 0, 0] (2), 0x16D212D48, 0x16D212D40, 0x0, 0x0) = 0 0**

**969/0x241b: open("/\0", 0x20100000, 0x0) = 3 0**

**969/0x241b: openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0) = 4 0**

**969/0x241b: dup(0x4, 0x0, 0x0) = 5 0**

**969/0x241b: fstatat64(0x4, 0x16D212821, 0x16D212790) = 0 0**

**969/0x241b: openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0) = 6 0**

**969/0x241b: fcntl(0x6, 0x32, 0x16D212820) = 0 0**

**969/0x241b: dup(0x6, 0x0, 0x0) = 7 0**

**969/0x241b: dup(0x5, 0x0, 0x0) = 8 0**

**969/0x241b: close(0x3) = 0 0**

**969/0x241b: close(0x5) = 0 0**

**969/0x241b: close(0x4) = 0 0**

**969/0x241b: close(0x6) = 0 0**

**969/0x241b: shared\_region\_check\_np(0x16D212E50, 0x0, 0x0) = 0 0**

**969/0x241b: fsgetpath(0x16D213388, 0x400, 0x16D2132D8) = 82 0**

**969/0x241b: fcntl(0x8, 0x32, 0x16D213388) = 0 0**

**969/0x241b: close(0x8) = 0 0**

**969/0x241b: close(0x7) = 0 0**

**969/0x241b: getfsstat64(0x0, 0x0, 0x2) = 11 0**

**969/0x241b: getfsstat64(0x102BFE090, 0x5D28, 0x2) = 11 0**

**969/0x241b: getattrlist("/\0", 0x16D2136C8, 0x16D213688) = 0 0**

**969/0x241b: fsgetpath(0x16D213308, 0x400, 0x16D2132E8) = 82 0**

**969/0x241b: stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld\_shared\_cache\_arm64e\0", 0x16D213770, 0x0) = 0 0**

**969/0x241b: stat64("/Users/litann/Desktop/lab2/main\0", 0x16D212AB0, 0x0) = 0 0**

**969/0x241b: open("/Users/litann/Desktop/lab2/main\0", 0x0, 0x0) = 3 0**

**969/0x241b: mmap(0x0, 0xC5C1, 0x1, 0x40002, 0x3, 0x0) = 0x102C7C000 0**

**969/0x241b: fcntl(0x3, 0x32, 0x16D212BC8) = 0 0**

**969/0x241b: close(0x3) = 0 0**

**969/0x241b: munmap(0x102C7C000, 0xC5C1) = 0 0**

**969/0x241b: stat64("/Users/litann/Desktop/lab2/main\0", 0x16D213020, 0x0) = 0 0**

**969/0x241b: stat64("/usr/lib/libc++.1.dylib\0", 0x16D211FF0, 0x0) = -1 Err#2**

**969/0x241b: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libc++.1.dylib\0", 0x16D211FA0, 0x0) = -1 Err#2**

**969/0x241b: stat64("/usr/lib/system/libdispatch.dylib\0", 0x16D20FBD0, 0x0) = -1 Err#2**

**969/0x241b: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/system/libdispatch.dylib\0", 0x16D20FB80, 0x0) = -1 Err#2**

**969/0x241b: stat64("/usr/lib/system/libdispatch.dylib\0", 0x16D20FBD0, 0x0) = -1 Err#2**

**969/0x241b: stat64("/usr/lib/libSystem.B.dylib\0", 0x16D211FF0, 0x0) = -1 Err#2**

**969/0x241b: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x16D211FA0, 0x0) = -1 Err#2**

**969/0x241b: open("/dev/dtracehelper\0", 0x2, 0x0) = 3 0**

**969/0x241b: ioctl(0x3, 0x80086804, 0x16D211CE8) = 0 0**

**969/0x241b: close(0x3) = 0 0**

**969/0x241b: open("/Users/litann/Desktop/lab2/main\0", 0x0, 0x0) = 3 0**

**969/0x241b: \_\_mac\_syscall(0x1AF6E8143, 0x2, 0x16D2112E0) = 0 0**

**969/0x241b: map\_with\_linking\_np(0x16D210F00, 0x1, 0x16D210F30) = 0 0**

**969/0x241b: close(0x3) = 0 0**

**969/0x241b: mprotect(0x102BF0000, 0x4000, 0x1) = 0 0**

**969/0x241b: shared\_region\_check\_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0**

**969/0x241b: access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2**

**969/0x241b: bsdthread\_register(0x1AF987E24, 0x1AF987E18, 0x4000) = 1073742303 0**

**969/0x241b: shm\_open(0x1AF84CF52, 0x0, 0x6D212B00) = 3 0**

**969/0x241b: fstat64(0x3, 0x16D211EB0, 0x0) = 0 0**

**969/0x241b: mmap(0x0, 0x4000, 0x1, 0x40001, 0x3, 0x0) = 0x102C84000 0**

**969/0x241b: close(0x3) = 0 0**

**969/0x241b: ioctl(0x2, 0x4004667A, 0x16D211F5C) = 0 0**

**969/0x241b: mprotect(0x102C90000, 0x4000, 0x0) = 0 0**

**969/0x241b: mprotect(0x102C9C000, 0x4000, 0x0) = 0 0**

**969/0x241b: mprotect(0x102CA0000, 0x4000, 0x0) = 0 0**

**969/0x241b: mprotect(0x102CAC000, 0x4000, 0x0) = 0 0**

**969/0x241b: mprotect(0x102CB0000, 0x4000, 0x0) = 0 0**

**969/0x241b: mprotect(0x102CBC000, 0x4000, 0x0) = 0 0**

**969/0x241b: mprotect(0x102C88000, 0x98, 0x1) = 0 0**

**969/0x241b: mprotect(0x102C88000, 0x98, 0x3) = 0 0**

**969/0x241b: mprotect(0x102C88000, 0x98, 0x1) = 0 0**

**969/0x241b: mprotect(0x102CC0000, 0x4000, 0x1) = 0 0**

**969/0x241b: mprotect(0x102CC4000, 0x98, 0x1) = 0 0**

**969/0x241b: mprotect(0x102CC4000, 0x98, 0x3) = 0 0**

**969/0x241b: mprotect(0x102CC4000, 0x98, 0x1) = 0 0**

**969/0x241b: mprotect(0x102C88000, 0x98, 0x3) = 0 0**

**969/0x241b: mprotect(0x102C88000, 0x98, 0x1) = 0 0**

**969/0x241b: mprotect(0x102CC0000, 0x4000, 0x3) = 0 0**

**969/0x241b: mprotect(0x102CC0000, 0x4000, 0x1) = 0 0**

**969/0x241b: objc\_bp\_assist\_cfg\_np(0x1AF621800, 0x80000018001C1048, 0x0) = -1 Err#5**

**969/0x241b: issetugid(0x0, 0x0, 0x0) = 0 0**

**969/0x241b: getentropy(0x16D2119A8, 0x20, 0x0) = 0 0**

**969/0x241b: getpid(0x0, 0x0, 0x0) = 969 0**

**969/0x241b: csops(0x3C9, 0x10, 0x16D211FC0) = 0 0**

**969/0x241b: csops\_audittoken(0x3C9, 0x10, 0x16D212020) = 0 0**

**969/0x241b: proc\_info(0x2, 0x3C9, 0xD) = 64 0**

**969/0x241b: csops\_audittoken(0x3C9, 0x10, 0x16D2120B0) = 0 0**

**969/0x241b: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D2123F0, 0x16D2123E0, 0x1B27F0D3D, 0x15) = 0 0**

**969/0x241b: sysctl([CTL\_KERN, 134, 0, 0, 0, 0] (2), 0x16D212498, 0x16D212480, 0x0, 0x0) = 0 0**

**969/0x241b: csops(0x3C9, 0x0, 0x16D21254C) = 0 0**

**969/0x241b: mprotect(0x102BFC000, 0x40000, 0x1) = 0 0**

**969/0x241b: getrlimit(0x1008, 0x16D213088, 0x0) = 0 0**

**969/0x241b: fstat64(0x1, 0x16D213080, 0x0) = 0 0**

**969/0x241b: ioctl(0x1, 0x4004667A, 0x16D2130CC) = 0 0**

**969/0x241b: write\_nocancel(0x1, "Enter a filename: \n\0", 0x13) = 19 0**

**969/0x241b: fstat64(0x0, 0x16D213110, 0x0) = 0 0**

**969/0x241b: ioctl(0x0, 0x4004667A, 0x16D21315C) = 0 0**

**123.txt**

**Enter number of operations: 969/0x241b: read\_nocancel(0x0, "123.txt\n\0", 0x1000) = 8 0**

**969/0x241b: pipe(0x0, 0x0, 0x0) = 3 0**

**969/0x241b: pipe(0x0, 0x0, 0x0) = 5 0**

**969/0x241b: fork() = 971 0**

**971/0x24e9: fork() = 0 0**

**971/0x24e9: thread\_selfid(0x0, 0x0, 0x0) = 9449 0**

**971/0x24e9: bsdthread\_register(0x1AF987E24, 0x1AF987E18, 0x4000) = -1 Err#22**

**971/0x24e9: mprotect(0x102CC4000, 0x98, 0x3) = 0 0**

**969/0x241b: write\_nocancel(0x1, "Enter number of operations: \0", 0x1C) = 28 0**

**971/0x24e9: mprotect(0x102CC4000, 0x98, 0x1) = 0 0**

**dtrace: error on enabled probe ID 1688 (ID 285: syscall::execve:return): invalid address (0x102befda1) in action #12 at DIF offset 12**

**971/0x24ea: fork() = 0 0**

**971/0x24ea: mprotect(0x102CD0000, 0x8000, 0x1) = 0 0**

**971/0x24ea: thread\_selfid(0x0, 0x0, 0x0) = 9450 0**

**971/0x24ea: shared\_region\_check\_np(0x16D2CB870, 0x0, 0x0) = 0 0**

**971/0x24ea: thread\_selfid(0x0, 0x0, 0x0) = 9450 0**

**971/0x24ea: getpid(0x0, 0x0, 0x0) = 971 0**

**971/0x24ea: proc\_info(0xF, 0x3CB, 0x0) = 0 0**

**971/0x24ea: munmap(0x102C44000, 0x8C000) = 0 0**

**971/0x24ea: munmap(0x102CD0000, 0x8000) = 0 0**

**971/0x24ea: munmap(0x102CD8000, 0x4000) = 0 0**

**971/0x24ea: munmap(0x102CDC000, 0x4000) = 0 0**

**971/0x24ea: munmap(0x102CE0000, 0x54000) = 0 0**

**971/0x24ea: open(".\0", 0x100000, 0x0) = 7 0**

**971/0x24ea: fcntl(0x7, 0x32, 0x16D2CB328) = 0 0**

**971/0x24ea: close(0x7) = 0 0**

**971/0x24ea: fsgetpath(0x16D2CB338, 0x400, 0x16D2CB318) = 33 0**

**971/0x24ea: fsgetpath(0x16D2CB348, 0x400, 0x16D2CB328) = 14 0**

**971/0x24ea: csrctl(0x0, 0x16D2CB74C, 0x4) = -1 Err#1**

**971/0x24ea: \_\_mac\_syscall(0x1AF6E8143, 0x2, 0x16D2CB6A0) = 0 0**

**971/0x24ea: csrctl(0x0, 0x16D2CB76C, 0x4) = -1 Err#1**

**971/0x24ea: \_\_mac\_syscall(0x1AF6E5094, 0x5A, 0x16D2CB700) = 0 0**

**971/0x24ea: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D2CAC70, 0x16D2CAC60, 0x1AF6E6CA1, 0xD) = 0 0**

**971/0x24ea: sysctl([CTL\_KERN, 136, 0, 0, 0, 0] (2), 0x16D2CAD18, 0x16D2CAD10, 0x0, 0x0) = 0 0**

**971/0x24ea: open("/\0", 0x20100000, 0x0) = 7 0**

**971/0x24ea: openat(0x7, "System/Cryptexes/OS\0", 0x100000, 0x0) = 8 0**

**971/0x24ea: dup(0x8, 0x0, 0x0) = 9 0**

**971/0x24ea: fstatat64(0x8, 0x16D2CA7F1, 0x16D2CA760) = 0 0**

**971/0x24ea: openat(0x8, "System/Library/dyld/\0", 0x100000, 0x0) = 10 0**

**971/0x24ea: fcntl(0xA, 0x32, 0x16D2CA7F0) = 0 0**

**971/0x24ea: dup(0xA, 0x0, 0x0) = 11 0**

**971/0x24ea: dup(0x9, 0x0, 0x0) = 12 0**

**971/0x24ea: close(0x7) = 0 0**

**971/0x24ea: close(0x9) = 0 0**

**971/0x24ea: close(0x8) = 0 0**

**971/0x24ea: close(0xA) = 0 0**

**971/0x24ea: shared\_region\_check\_np(0x16D2CAE20, 0x0, 0x0) = 0 0**

**971/0x24ea: fsgetpath(0x16D2CB358, 0x400, 0x16D2CB2A8) = 82 0**

**971/0x24ea: fcntl(0xC, 0x32, 0x16D2CB358) = 0 0**

**971/0x24ea: close(0xC) = 0 0**

**971/0x24ea: close(0xB) = 0 0**

**971/0x24ea: getfsstat64(0x0, 0x0, 0x2) = 11 0**

**971/0x24ea: getfsstat64(0x102B420B0, 0x5D28, 0x2) = 11 0**

**971/0x24ea: getattrlist("/\0", 0x16D2CB698, 0x16D2CB658) = 0 0**

**971/0x24ea: fsgetpath(0x16D2CB2D8, 0x400, 0x16D2CB2B8) = 82 0**

**971/0x24ea: stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld\_shared\_cache\_arm64e\0", 0x16D2CB740, 0x0) = 0 0**

**971/0x24ea: stat64("/Users/litann/Desktop/lab2/child\0", 0x16D2CAA80, 0x0) = 0 0**

**971/0x24ea: open("/Users/litann/Desktop/lab2/child\0", 0x0, 0x0) = 7 0**

**971/0x24ea: mmap(0x0, 0xBDA2, 0x1, 0x40002, 0x7, 0x0) = 0x102BC0000 0**

**971/0x24ea: fcntl(0x7, 0x32, 0x16D2CAB98) = 0 0**

**971/0x24ea: close(0x7) = 0 0**

**971/0x24ea: munmap(0x102BC0000, 0xBDA2) = 0 0**

**971/0x24ea: stat64("/Users/litann/Desktop/lab2/child\0", 0x16D2CAFF0, 0x0) = 0 0**

**971/0x24ea: stat64("/usr/lib/libc++.1.dylib\0", 0x16D2C9FC0, 0x0) = -1 Err#2**

**971/0x24ea: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libc++.1.dylib\0", 0x16D2C9F70, 0x0) = -1 Err#2**

**971/0x24ea: stat64("/usr/lib/system/libdispatch.dylib\0", 0x16D2C7BA0, 0x0) = -1 Err#2**

**971/0x24ea: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/system/libdispatch.dylib\0", 0x16D2C7B50, 0x0) = -1 Err#2**

**971/0x24ea: stat64("/usr/lib/system/libdispatch.dylib\0", 0x16D2C7BA0, 0x0) = -1 Err#2**

**971/0x24ea: stat64("/usr/lib/libSystem.B.dylib\0", 0x16D2C9FC0, 0x0) = -1 Err#2**

**971/0x24ea: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x16D2C9F70, 0x0) = -1 Err#2**

**971/0x24ea: open("/dev/dtracehelper\0", 0x2, 0x0) = 7 0**

**971/0x24ea: ioctl(0x7, 0x80086804, 0x16D2C9CB8) = 0 0**

**971/0x24ea: close(0x7) = 0 0**

**971/0x24ea: open("/Users/litann/Desktop/lab2/child\0", 0x0, 0x0) = 7 0**

**971/0x24ea: \_\_mac\_syscall(0x1AF6E8143, 0x2, 0x16D2C92B0) = 0 0**

**971/0x24ea: map\_with\_linking\_np(0x16D2C8F30, 0x1, 0x16D2C8F60) = 0 0**

**971/0x24ea: close(0x7) = 0 0**

**971/0x24ea: mprotect(0x102B38000, 0x4000, 0x1) = 0 0**

**971/0x24ea: shared\_region\_check\_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0**

**971/0x24ea: access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2**

**971/0x24ea: bsdthread\_register(0x1AF987E24, 0x1AF987E18, 0x4000) = 1073742303 0**

**971/0x24ea: shm\_open(0x1AF84CF52, 0x0, 0xFFFFFFFFAF8ED4C8) = 7 0**

**971/0x24ea: fstat64(0x7, 0x16D2C9E80, 0x0) = 0 0**

**971/0x24ea: mmap(0x0, 0x4000, 0x1, 0x40001, 0x7, 0x0) = 0x102BC8000 0**

**971/0x24ea: close(0x7) = 0 0**

**971/0x24ea: ioctl(0x2, 0x4004667A, 0x16D2C9F2C) = 0 0**

**971/0x24ea: mprotect(0x102BD4000, 0x4000, 0x0) = 0 0**

**971/0x24ea: mprotect(0x102BE0000, 0x4000, 0x0) = 0 0**

**971/0x24ea: mprotect(0x102BE4000, 0x4000, 0x0) = 0 0**

**971/0x24ea: mprotect(0x102BF0000, 0x4000, 0x0) = 0 0**

**971/0x24ea: mprotect(0x102BF4000, 0x4000, 0x0) = 0 0**

**971/0x24ea: mprotect(0x102C00000, 0x4000, 0x0) = 0 0**

**971/0x24ea: mprotect(0x102BCC000, 0x98, 0x1) = 0 0**

**971/0x24ea: mprotect(0x102BCC000, 0x98, 0x3) = 0 0**

**971/0x24ea: mprotect(0x102BCC000, 0x98, 0x1) = 0 0**

**971/0x24ea: mprotect(0x102C04000, 0x4000, 0x1) = 0 0**

**971/0x24ea: mprotect(0x102C08000, 0x98, 0x1) = 0 0**

**971/0x24ea: mprotect(0x102C08000, 0x98, 0x3) = 0 0**

**971/0x24ea: mprotect(0x102C08000, 0x98, 0x1) = 0 0**

**971/0x24ea: mprotect(0x102BCC000, 0x98, 0x3) = 0 0**

**971/0x24ea: mprotect(0x102BCC000, 0x98, 0x1) = 0 0**

**971/0x24ea: mprotect(0x102C04000, 0x4000, 0x3) = 0 0**

**971/0x24ea: mprotect(0x102C04000, 0x4000, 0x1) = 0 0**

**971/0x24ea: objc\_bp\_assist\_cfg\_np(0x1AF621800, 0x80000018001C1048, 0x0) = -1 Err#5**

**971/0x24ea: issetugid(0x0, 0x0, 0x0) = 0 0**

**971/0x24ea: getentropy(0x16D2C9978, 0x20, 0x0) = 0 0**

**971/0x24ea: getpid(0x0, 0x0, 0x0) = 971 0**

**971/0x24ea: csops(0x3CB, 0x10, 0x16D2C9F90) = 0 0**

**971/0x24ea: csops\_audittoken(0x3CB, 0x10, 0x16D2C9FF0) = 0 0**

**971/0x24ea: proc\_info(0x2, 0x3CB, 0xD) = 64 0**

**971/0x24ea: csops\_audittoken(0x3CB, 0x10, 0x16D2CA080) = 0 0**

**971/0x24ea: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D2CA3C0, 0x16D2CA3B0, 0x1B27F0D3D, 0x15) = 0 0**

**971/0x24ea: sysctl([CTL\_KERN, 134, 0, 0, 0, 0] (2), 0x16D2CA468, 0x16D2CA450, 0x0, 0x0) = 0 0**

**971/0x24ea: csops(0x3CB, 0x0, 0x16D2CA51C) = 0 0**

**971/0x24ea: mprotect(0x102B40000, 0x40000, 0x1) = 0 0**

**971/0x24ea: getrlimit(0x1008, 0x16D2CB1E8, 0x0) = 0 0**

**971/0x24ea: open\_nocancel("123.txt\0", 0x20A, 0x1B6) = 7 0**

**971/0x24ea: lseek(0x7, 0x0, 0x2) = 0 0**

**2**

**Enter a number:**

**969/0x241b: read\_nocancel(0x0, "2\n\0", 0x1000) = 2 0**

**969/0x241b: write\_nocancel(0x1, "\n\0", 0x1) = 1 0**

**969/0x241b: write\_nocancel(0x1, "Enter a number: \n\0", 0x11) = 17 0**

**0**

**The number is prime or negative**

**Enter a number:**

**969/0x241b: read\_nocancel(0x0, "0\n\0", 0x1000) = 2 0**

**969/0x241b: write(0x4, "\001\0", 0x4) = 4 0**

**969/0x241b: write(0x4, "0\0", 0x1) = 1 0**

**971/0x24ea: read(0x3, "\001\0", 0x4) = 4 0**

**971/0x24ea: read(0x3, "0\0", 0x1) = 1 0**

**971/0x24ea: write(0x6, "\001\0", 0x4) = 4 0**

**969/0x241b: read(0x5, "\001\0", 0x4) = 4 0**

**969/0x241b: write\_nocancel(0x1, "The number is prime or negative\n\0", 0x20) = 32 0**

**969/0x241b: write\_nocancel(0x1, "Enter a number: \n\0", 0x11) = 17 0**

**12**

**A number 12 is added to file!**

**969/0x241b: read\_nocancel(0x0, "12\n\0", 0x1000) = 3 0**

**969/0x241b: write(0x4, "\002\0", 0x4) = 4 0**

**969/0x241b: write(0x4, "12\0", 0x2) = 2 0**

**971/0x24ea: read(0x3, "\002\0", 0x4) = 4 0**

**971/0x24ea: read(0x3, "12\0", 0x2) = 2 0**

**971/0x24ea: fstat64(0x7, 0x16D2CAF40, 0x0) = 0 0**

**971/0x24ea: write\_nocancel(0x7, "12\n\0", 0x3) = 3 0**

**971/0x24ea: fstat64(0x1, 0x16D2CAFF0, 0x0) = 0 0**

**971/0x24ea: ioctl(0x1, 0x4004667A, 0x16D2CB03C) = 0 0**

**971/0x24ea: write\_nocancel(0x1, "A number 12 is added to file!\n\0", 0x1E) = 30 0**

**971/0x24ea: write(0x6, "\002\0", 0x4) = 4 0**

**969/0x241b: read(0x5, "\002\0", 0x4) = 4 0**

**969/0x241b: close(0x3) = 0 0**

**969/0x241b: close(0x4) = 0 0**

**969/0x241b: close(0x5) = 0 0**

**969/0x241b: close(0x6) = 0 0**

**969/0x241b: lseek(0x0, 0xFFFFFFFFFFFFFFFF, 0x1) = 14927 0**

**Выводы**

В результате данной лабораторной работы я узнала о возможностях утилиты dtruss, а также о том, как много информации может дать диагностика программы для разработчика.