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: Савинова Екатерина Ильинична

Группа: М8О-207Б-21

Вариант: 15

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/savinova-kati/operating-systems

**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** - Системный вызов munmap удаляет все отражения из заданной области памяти, после чего все ссылки на данную область будут вызывать ошибку "неправильное обращение к памяти"

**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** - работает точно так же, как open за исключением того, что если путь, указанный в pathname , является относительным, то он интерпретируется относительно каталога, на который ссылается файловый дескриптор dirfd (а не относительно текущего рабочего каталога вызывающего процесса)

Флаги:

O\_RDONLY

Открыт только для чтения.

O\_WRONLY

Открыт только для записи.

O\_RDWR

Открыт для чтения и записи. Результат не определен, если этот флаг применяется к FIFO.

O\_APPEND

Если установлено, смещение файла устанавливается до конца файла перед каждой записью.

O\_CREATE

Создайте файл, если он не существует. Этот флаг требует указания аргумента режима .

**mprotect** - контролирует доступ к области памяти. Если программой производится запрещенный этой функцией доступ к памяти, то такая программа получает сигнал SIGSEGV.

prot состоит из следующих побитно и логически сложенных значений:

PROT\_NONE

Доступ к памяти запрещен.

PROT\_READ

Данные в памяти можно читать.

PROT\_WRITE

В память можно записать информацию.

PROT\_EXEC

Память может содержать исполняемый код.

При удачном завершении вызова возвращаемое значение равно нулю. При ошибке оно равно -1, а переменной errno присваивается номер ошибки.

**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

stat64:

integer stat64

integer\*8 statb(13)

character name\*(\*)

ierr = stat64 ( name, statb )

**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 file\_1;

string name;

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

cin >> name;

int truba[2];

int truba\_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 string\_r;

pid\_t id = fork();

if (id == -1){

perror("fork");

cout << "1";

exit(EXIT\_FAILURE);

}

else if (id == 0) {

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

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

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

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

execl("./child\_", to\_string(truba[0]).c\_str(), to\_string(truba[1]).c\_str(), to\_string(truba\_2[0]).c\_str(), to\_string(truba\_2[1]).c\_str(), name.c\_str(), NULL);

}

else {

cout<<"Enter amount of strings: ";

int amount;

cin >> amount;

cout << endl;

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

cin >> string\_r;

int s\_size = string\_r.size();

char str\_array[s\_size];

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

str\_array[k] = string\_r[k];

}

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

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

int flag\_;

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

//cout << truba\_2[0] << endl;

if (flag\_ == 1) {

cout << "The string does not fit the rule" << endl;

}

}

}

close(pipe\_1[0]);

close(pipe\_1[1]);

close(pipe\_2[0]);

close(pipe\_2[1]);

return 0;

}

**сhild\_.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(int argc, char \*argv[]){

string filename = argv[4];

int truba[2];

int truba\_2[2];

int flag1 = 1;

int flag2 = 22;

truba[0] = stoi(argv[0]);

truba[1] = stoi(argv[1]);

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

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

fstream file\_1;

file\_1.open(filename, fstream::in | fstream::out | fstream::app);

while(true) {

int stroka\_size;

read(truba[0], &stroka\_size, sizeof(int));

char stroka[stroka\_size];

read(truba[0], &stroka, sizeof(char)\*stroka\_size);

string result;

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

result.push\_back(stroka[i]);

}

if (stroka[0] >= 65 && stroka[0] <= 90) {

file\_1 << result << endl;

cout << "Added string " << result << " to file!" << endl;

write(truba\_2[1], &flag2, sizeof(int));

//cout << truba\_2[1] << endl;

} else {

write(truba\_2[1], &flag1, sizeof(int));

//cout << truba\_2[1] << endl;

}

}

return 0;

}

**Демонстрация работы программы**

litann@Annalit lab1 % sudo dtruss -f ./main

Password:

PID/THRD  SYSCALL(args) = return

Enter filename:

 1382/0xc73b:  fork() = 0 0

 1382/0xc73b:  munmap(0x102904000, 0x8C000) = 0 0

 1382/0xc73b:  munmap(0x102990000, 0x8000) = 0 0

 1382/0xc73b:  munmap(0x102998000, 0x4000) = 0 0

 1382/0xc73b:  munmap(0x10299C000, 0x4000) = 0 0

 1382/0xc73b:  munmap(0x1029A0000, 0x54000) = 0 0

 1382/0xc73b:  open(".\0", 0x100000, 0x0) = 3 0

 1382/0xc73b:  fcntl(0x3, 0x32, 0x16D893358) = 0 0

 1382/0xc73b:  close(0x3) = 0 0

 1382/0xc73b:  fsgetpath(0x16D893368, 0x400, 0x16D893348) = 32 0

 1382/0xc73b:  fsgetpath(0x16D893378, 0x400, 0x16D893358) = 14 0

 1382/0xc73b:  csrctl(0x0, 0x16D89377C, 0x4) = -1 Err#1

 1382/0xc73b:  \_\_mac\_syscall(0x1AF6E8143, 0x2, 0x16D8936D0) = 0 0

 1382/0xc73b:  csrctl(0x0, 0x16D89379C, 0x4) = -1 Err#1

 1382/0xc73b:  \_\_mac\_syscall(0x1AF6E5094, 0x5A, 0x16D893730) = 0 0

 1382/0xc73b:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D892CA0, 0x16D892C90, 0x1AF6E6CA1, 0xD) = 0 0

 1382/0xc73b:  sysctl([CTL\_KERN, 136, 0, 0, 0, 0] (2), 0x16D892D48, 0x16D892D40, 0x0, 0x0) = 0 0

 1382/0xc73b:  open("/\0", 0x20100000, 0x0) = 3 0

 1382/0xc73b:  openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0) = 4 0

 1382/0xc73b:  dup(0x4, 0x0, 0x0) = 5 0

 1382/0xc73b:  fstatat64(0x4, 0x16D892821, 0x16D892790) = 0 0

 1382/0xc73b:  openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0) = 6 0

 1382/0xc73b:  fcntl(0x6, 0x32, 0x16D892820) = 0 0

 1382/0xc73b:  dup(0x6, 0x0, 0x0) = 7 0

 1382/0xc73b:  dup(0x5, 0x0, 0x0) = 8 0

 1382/0xc73b:  close(0x3) = 0 0

 1382/0xc73b:  close(0x5) = 0 0

 1382/0xc73b:  close(0x4) = 0 0

 1382/0xc73b:  close(0x6) = 0 0

 1382/0xc73b:  shared\_region\_check\_np(0x16D892E50, 0x0, 0x0) = 0 0

 1382/0xc73b:  fsgetpath(0x16D893388, 0x400, 0x16D8932D8) = 82 0

 1382/0xc73b:  fcntl(0x8, 0x32, 0x16D893388) = 0 0

 1382/0xc73b:  close(0x8) = 0 0

 1382/0xc73b:  close(0x7) = 0 0

 1382/0xc73b:  getfsstat64(0x0, 0x0, 0x2) = 11 0

 1382/0xc73b:  getfsstat64(0x10257E090, 0x5D28, 0x2) = 11 0

 1382/0xc73b:  getattrlist("/\0", 0x16D8936C8, 0x16D893688) = 0 0

 1382/0xc73b:  fsgetpath(0x16D893308, 0x400, 0x16D8932E8) = 82 0

 1382/0xc73b:  stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld\_shared\_cache\_arm64e\0", 0x16D893770, 0x0) = 0 0

 1382/0xc73b:  stat64("/Users/litann/Desktop/lab1/main\0", 0x16D892AB0, 0x0) = 0 0

 1382/0xc73b:  open("/Users/litann/Desktop/lab1/main\0", 0x0, 0x0) = 3 0

 1382/0xc73b:  mmap(0x0, 0xC5C1, 0x1, 0x40002, 0x3, 0x0) = 0x1025FC000 0

 1382/0xc73b:  fcntl(0x3, 0x32, 0x16D892BC8) = 0 0

 1382/0xc73b:  close(0x3) = 0 0

 1382/0xc73b:  munmap(0x1025FC000, 0xC5C1) = 0 0

 1382/0xc73b:  stat64("/Users/litann/Desktop/lab1/main\0", 0x16D893020, 0x0) = 0 0

 1382/0xc73b:  stat64("/usr/lib/libc++.1.dylib\0", 0x16D891FF0, 0x0) = -1 Err#2

 1382/0xc73b:  stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libc++.1.dylib\0", 0x16D891FA0, 0x0) = -1 Err#2

 1382/0xc73b:  stat64("/usr/lib/system/libdispatch.dylib\0", 0x16D88FBD0, 0x0) = -1 Err#2

 1382/0xc73b:  stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/system/libdispatch.dylib\0", 0x16D88FB80, 0x0) = -1 Err#2

 1382/0xc73b:  stat64("/usr/lib/system/libdispatch.dylib\0", 0x16D88FBD0, 0x0) = -1 Err#2

 1382/0xc73b:  stat64("/usr/lib/libSystem.B.dylib\0", 0x16D891FF0, 0x0) = -1 Err#2

 1382/0xc73b:  stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x16D891FA0, 0x0) = -1 Err#2

 1382/0xc73b:  open("/dev/dtracehelper\0", 0x2, 0x0) = 3 0

 1382/0xc73b:  ioctl(0x3, 0x80086804, 0x16D891CE8) = 0 0

 1382/0xc73b:  close(0x3) = 0 0

 1382/0xc73b:  open("/Users/litann/Desktop/lab1/main\0", 0x0, 0x0) = 3 0

 1382/0xc73b:  \_\_mac\_syscall(0x1AF6E8143, 0x2, 0x16D8912E0) = 0 0

 1382/0xc73b:  map\_with\_linking\_np(0x16D890F00, 0x1, 0x16D890F30) = 0 0

 1382/0xc73b:  close(0x3) = 0 0

 1382/0xc73b:  mprotect(0x102570000, 0x4000, 0x1) = 0 0

 1382/0xc73b:  shared\_region\_check\_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

 1382/0xc73b:  access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2

 1382/0xc73b:  bsdthread\_register(0x1AF987E24, 0x1AF987E18, 0x4000) = 1073742303 0

 1382/0xc73b:  shm\_open(0x1AF84CF52, 0x0, 0x6D892B00) = 3 0

 1382/0xc73b:  fstat64(0x3, 0x16D891EB0, 0x0) = 0 0

 1382/0xc73b:  mmap(0x0, 0x4000, 0x1, 0x40001, 0x3, 0x0) = 0x102604000 0

 1382/0xc73b:  close(0x3) = 0 0

 1382/0xc73b:  ioctl(0x2, 0x4004667A, 0x16D891F5C) = 0 0

 1382/0xc73b:  mprotect(0x102610000, 0x4000, 0x0) = 0 0

 1382/0xc73b:  mprotect(0x10261C000, 0x4000, 0x0) = 0 0

 1382/0xc73b:  mprotect(0x102620000, 0x4000, 0x0) = 0 0

 1382/0xc73b:  mprotect(0x10262C000, 0x4000, 0x0) = 0 0

 1382/0xc73b:  mprotect(0x102630000, 0x4000, 0x0) = 0 0

 1382/0xc73b:  mprotect(0x10263C000, 0x4000, 0x0) = 0 0

 1382/0xc73b:  mprotect(0x102608000, 0x98, 0x1) = 0 0

 1382/0xc73b:  mprotect(0x102608000, 0x98, 0x3) = 0 0

 1382/0xc73b:  mprotect(0x102608000, 0x98, 0x1) = 0 0

 1382/0xc73b:  mprotect(0x102640000, 0x4000, 0x1) = 0 0

 1382/0xc73b:  mprotect(0x102644000, 0x98, 0x1) = 0 0

 1382/0xc73b:  mprotect(0x102644000, 0x98, 0x3) = 0 0

 1382/0xc73b:  mprotect(0x102644000, 0x98, 0x1) = 0 0

 1382/0xc73b:  mprotect(0x102608000, 0x98, 0x3) = 0 0

 1382/0xc73b:  mprotect(0x102608000, 0x98, 0x1) = 0 0

 1382/0xc73b:  mprotect(0x102640000, 0x4000, 0x3) = 0 0

 1382/0xc73b:  mprotect(0x102640000, 0x4000, 0x1) = 0 0

 1382/0xc73b:  objc\_bp\_assist\_cfg\_np(0x1AF621800, 0x80000018001C1048, 0x0) = -1 Err#5

 1382/0xc73b:  issetugid(0x0, 0x0, 0x0) = 0 0

 1382/0xc73b:  getentropy(0x16D8919A8, 0x20, 0x0) = 0 0

 1382/0xc73b:  getpid(0x0, 0x0, 0x0) = 1382 0

 1382/0xc73b:  csops(0x566, 0x10, 0x16D891FC0) = 0 0

 1382/0xc73b:  csops\_audittoken(0x566, 0x10, 0x16D892020) = 0 0

 1382/0xc73b:  proc\_info(0x2, 0x566, 0xD) = 64 0

 1382/0xc73b:  csops\_audittoken(0x566, 0x10, 0x16D8920B0) = 0 0

 1382/0xc73b:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D8923F0, 0x16D8923E0, 0x1B27F0D3D, 0x15) = 0 0

 1382/0xc73b:  sysctl([CTL\_KERN, 134, 0, 0, 0, 0] (2), 0x16D892498, 0x16D892480, 0x0, 0x0) = 0 0

 1382/0xc73b:  csops(0x566, 0x0, 0x16D89254C) = 0 0

 1382/0xc73b:  mprotect(0x10257C000, 0x40000, 0x1) = 0 0

 1382/0xc73b:  getrlimit(0x1008, 0x16D893088, 0x0) = 0 0

 1382/0xc73b:  fstat64(0x1, 0x16D893080, 0x0) = 0 0

 1382/0xc73b:  ioctl(0x1, 0x4004667A, 0x16D8930CC) = 0 0

 1382/0xc73b:  write\_nocancel(0x1, "Enter filename: \n\0", 0x11) = 17 0

 1382/0xc73b:  fstat64(0x0, 0x16D893110, 0x0) = 0 0

 1382/0xc73b:  ioctl(0x0, 0x4004667A, 0x16D89315C) = 0 0

123.txt

Enter amount of strings:  1382/0xc73b:  read\_nocancel(0x0, "123.txt\n\0", 0x1000) = 8 0

 1382/0xc73b:  pipe(0x0, 0x0, 0x0) = 3 0

 1382/0xc73b:  pipe(0x0, 0x0, 0x0) = 5 0

 1382/0xc73b:  fork() = 1383 0

 1383/0xc75c:  fork() = 0 0

 1383/0xc75c:  thread\_selfid(0x0, 0x0, 0x0) = 51036 0

 1383/0xc75c:  bsdthread\_register(0x1AF987E24, 0x1AF987E18, 0x4000) = -1 Err#22

 1382/0xc73b:  write\_nocancel(0x1, "Enter amount of strings: \0", 0x19) = 25 0

 1383/0xc75c:  mprotect(0x102644000, 0x98, 0x3) = 0 0

 1383/0xc75c:  mprotect(0x102644000, 0x98, 0x1) = 0 0

dtrace: error on enabled probe ID 1688 (ID 285: syscall::execve:return): invalid address (0x10256fdb5) in action #12 at DIF offset 12

 1383/0xc75d:  fork() = 0 0

 1383/0xc75d:  mprotect(0x101298000, 0x8000, 0x1) = 0 0

 1383/0xc75d:  thread\_selfid(0x0, 0x0, 0x0) = 51037 0

 1383/0xc75d:  shared\_region\_check\_np(0x16EF17870, 0x0, 0x0) = 0 0

 1383/0xc75d:  thread\_selfid(0x0, 0x0, 0x0) = 51037 0

 1383/0xc75d:  getpid(0x0, 0x0, 0x0) = 1383 0

 1383/0xc75d:  proc\_info(0xF, 0x567, 0x0) = 0 0

 1383/0xc75d:  munmap(0x10120C000, 0x8C000) = 0 0

 1383/0xc75d:  munmap(0x101298000, 0x8000) = 0 0

 1383/0xc75d:  munmap(0x1012A0000, 0x4000) = 0 0

 1383/0xc75d:  munmap(0x1012A4000, 0x4000) = 0 0

 1383/0xc75d:  munmap(0x1012A8000, 0x54000) = 0 0

 1383/0xc75d:  open(".\0", 0x100000, 0x0) = 7 0

 1383/0xc75d:  fcntl(0x7, 0x32, 0x16EF17328) = 0 0

 1383/0xc75d:  close(0x7) = 0 0

 1383/0xc75d:  fsgetpath(0x16EF17338, 0x400, 0x16EF17318) = 34 0

 1383/0xc75d:  fsgetpath(0x16EF17348, 0x400, 0x16EF17328) = 14 0

 1383/0xc75d:  csrctl(0x0, 0x16EF1774C, 0x4) = -1 Err#1

 1383/0xc75d:  \_\_mac\_syscall(0x1AF6E8143, 0x2, 0x16EF176A0) = 0 0

 1383/0xc75d:  csrctl(0x0, 0x16EF1776C, 0x4) = -1 Err#1

 1383/0xc75d:  \_\_mac\_syscall(0x1AF6E5094, 0x5A, 0x16EF17700) = 0 0

 1383/0xc75d:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16EF16C70, 0x16EF16C60, 0x1AF6E6CA1, 0xD) = 0 0

 1383/0xc75d:  sysctl([CTL\_KERN, 136, 0, 0, 0, 0] (2), 0x16EF16D18, 0x16EF16D10, 0x0, 0x0) = 0 0

 1383/0xc75d:  open("/\0", 0x20100000, 0x0) = 7 0

 1383/0xc75d:  openat(0x7, "System/Cryptexes/OS\0", 0x100000, 0x0) = 8 0

 1383/0xc75d:  dup(0x8, 0x0, 0x0) = 9 0

 1383/0xc75d:  fstatat64(0x8, 0x16EF167F1, 0x16EF16760) = 0 0

 1383/0xc75d:  openat(0x8, "System/Library/dyld/\0", 0x100000, 0x0) = 10 0

 1383/0xc75d:  fcntl(0xA, 0x32, 0x16EF167F0) = 0 0

 1383/0xc75d:  dup(0xA, 0x0, 0x0) = 11 0

 1383/0xc75d:  dup(0x9, 0x0, 0x0) = 12 0

 1383/0xc75d:  close(0x7) = 0 0

 1383/0xc75d:  close(0x9) = 0 0

 1383/0xc75d:  close(0x8) = 0 0

 1383/0xc75d:  close(0xA) = 0 0

 1383/0xc75d:  shared\_region\_check\_np(0x16EF16E20, 0x0, 0x0) = 0 0

 1383/0xc75d:  fsgetpath(0x16EF17358, 0x400, 0x16EF172A8) = 82 0

 1383/0xc75d:  fcntl(0xC, 0x32, 0x16EF17358) = 0 0

 1383/0xc75d:  close(0xC) = 0 0

 1383/0xc75d:  close(0xB) = 0 0

 1383/0xc75d:  getfsstat64(0x0, 0x0, 0x2) = 11 0

 1383/0xc75d:  getfsstat64(0x100EF60B0, 0x5D28, 0x2) = 11 0

 1383/0xc75d:  getattrlist("/\0", 0x16EF17698, 0x16EF17658) = 0 0

 1383/0xc75d:  fsgetpath(0x16EF172D8, 0x400, 0x16EF172B8) = 82 0

 1383/0xc75d:  stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld\_shared\_cache\_arm64e\0", 0x16EF17740, 0x0) = 0 0

 1383/0xc75d:  stat64("/Users/litann/Desktop/lab1/child\_\0", 0x16EF16A80, 0x0) = 0 0

 1383/0xc75d:  open("/Users/litann/Desktop/lab1/child\_\0", 0x0, 0x0) = 7 0

 1383/0xc75d:  mmap(0x0, 0xBD43, 0x1, 0x40002, 0x7, 0x0) = 0x100F74000 0

 1383/0xc75d:  fcntl(0x7, 0x32, 0x16EF16B98) = 0 0

 1383/0xc75d:  close(0x7) = 0 0

 1383/0xc75d:  munmap(0x100F74000, 0xBD43) = 0 0

 1383/0xc75d:  stat64("/Users/litann/Desktop/lab1/child\_\0", 0x16EF16FF0, 0x0) = 0 0

 1383/0xc75d:  stat64("/usr/lib/libc++.1.dylib\0", 0x16EF15FC0, 0x0) = -1 Err#2

 1383/0xc75d:  stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libc++.1.dylib\0", 0x16EF15F70, 0x0) = -1 Err#2

 1383/0xc75d:  stat64("/usr/lib/system/libdispatch.dylib\0", 0x16EF13BA0, 0x0) = -1 Err#2

 1383/0xc75d:  stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/system/libdispatch.dylib\0", 0x16EF13B50, 0x0) = -1 Err#2

 1383/0xc75d:  stat64("/usr/lib/system/libdispatch.dylib\0", 0x16EF13BA0, 0x0) = -1 Err#2

 1383/0xc75d:  stat64("/usr/lib/libSystem.B.dylib\0", 0x16EF15FC0, 0x0) = -1 Err#2

 1383/0xc75d:  stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x16EF15F70, 0x0) = -1 Err#2

 1383/0xc75d:  open("/dev/dtracehelper\0", 0x2, 0x0) = 7 0

 1383/0xc75d:  ioctl(0x7, 0x80086804, 0x16EF15CB8) = 0 0

 1383/0xc75d:  close(0x7) = 0 0

 1383/0xc75d:  open("/Users/litann/Desktop/lab1/child\_\0", 0x0, 0x0) = 7 0

 1383/0xc75d:  \_\_mac\_syscall(0x1AF6E8143, 0x2, 0x16EF152B0) = 0 0

 1383/0xc75d:  map\_with\_linking\_np(0x16EF14F30, 0x1, 0x16EF14F60) = 0 0

 1383/0xc75d:  close(0x7) = 0 0

 1383/0xc75d:  mprotect(0x100EEC000, 0x4000, 0x1) = 0 0

 1383/0xc75d:  shared\_region\_check\_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

 1383/0xc75d:  access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2

 1383/0xc75d:  bsdthread\_register(0x1AF987E24, 0x1AF987E18, 0x4000) = 1073742303 0

 1383/0xc75d:  shm\_open(0x1AF84CF52, 0x0, 0xFFFFFFFFAF8ED4C8) = 7 0

 1383/0xc75d:  fstat64(0x7, 0x16EF15E80, 0x0) = 0 0

 1383/0xc75d:  mmap(0x0, 0x4000, 0x1, 0x40001, 0x7, 0x0) = 0x100F7C000 0

 1383/0xc75d:  close(0x7) = 0 0

 1383/0xc75d:  ioctl(0x2, 0x4004667A, 0x16EF15F2C) = 0 0

 1383/0xc75d:  mprotect(0x100F88000, 0x4000, 0x0) = 0 0

 1383/0xc75d:  mprotect(0x100F94000, 0x4000, 0x0) = 0 0

 1383/0xc75d:  mprotect(0x100F98000, 0x4000, 0x0) = 0 0

 1383/0xc75d:  mprotect(0x100FA4000, 0x4000, 0x0) = 0 0

 1383/0xc75d:  mprotect(0x100FA8000, 0x4000, 0x0) = 0 0

 1383/0xc75d:  mprotect(0x100FB4000, 0x4000, 0x0) = 0 0

 1383/0xc75d:  mprotect(0x100F80000, 0x98, 0x1) = 0 0

 1383/0xc75d:  mprotect(0x100F80000, 0x98, 0x3) = 0 0

 1383/0xc75d:  mprotect(0x100F80000, 0x98, 0x1) = 0 0

 1383/0xc75d:  mprotect(0x100FB8000, 0x4000, 0x1) = 0 0

 1383/0xc75d:  mprotect(0x100FBC000, 0x98, 0x1) = 0 0

 1383/0xc75d:  mprotect(0x100FBC000, 0x98, 0x3) = 0 0

 1383/0xc75d:  mprotect(0x100FBC000, 0x98, 0x1) = 0 0

 1383/0xc75d:  mprotect(0x100F80000, 0x98, 0x3) = 0 0

 1383/0xc75d:  mprotect(0x100F80000, 0x98, 0x1) = 0 0

 1383/0xc75d:  mprotect(0x100FB8000, 0x4000, 0x3) = 0 0

 1383/0xc75d:  mprotect(0x100FB8000, 0x4000, 0x1) = 0 0

 1383/0xc75d:  objc\_bp\_assist\_cfg\_np(0x1AF621800, 0x80000018001C1048, 0x0) = -1 Err#5

 1383/0xc75d:  issetugid(0x0, 0x0, 0x0) = 0 0

 1383/0xc75d:  getentropy(0x16EF15978, 0x20, 0x0) = 0 0

 1383/0xc75d:  getpid(0x0, 0x0, 0x0) = 1383 0

 1383/0xc75d:  csops(0x567, 0x10, 0x16EF15F90) = 0 0

 1383/0xc75d:  csops\_audittoken(0x567, 0x10, 0x16EF15FF0) = 0 0

 1383/0xc75d:  proc\_info(0x2, 0x567, 0xD) = 64 0

 1383/0xc75d:  csops\_audittoken(0x567, 0x10, 0x16EF16080) = 0 0

 1383/0xc75d:  sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16EF163C0, 0x16EF163B0, 0x1B27F0D3D, 0x15) = 0 0

 1383/0xc75d:  sysctl([CTL\_KERN, 134, 0, 0, 0, 0] (2), 0x16EF16468, 0x16EF16450, 0x0, 0x0) = 0 0

 1383/0xc75d:  csops(0x567, 0x0, 0x16EF1651C) = 0 0

 1383/0xc75d:  mprotect(0x100EF4000, 0x40000, 0x1) = 0 0

 1383/0xc75d:  getrlimit(0x1008, 0x16EF17208, 0x0) = 0 0

 1383/0xc75d:  open\_nocancel("123.txt\0", 0x20A, 0x1B6) = 7 0

 1383/0xc75d:  lseek(0x7, 0x0, 0x2) = 0 0

2

 1382/0xc73b:  read\_nocancel(0x0, "2\n\0", 0x1000) = 2 0

 1382/0xc73b:  write\_nocancel(0x1, "\n\0", 0x1) = 1 0

aaaaff

The string does not fit the rule

 1382/0xc73b:  read\_nocancel(0x0, "aaaaff\n\0", 0x1000) = 7 0

 1382/0xc73b:  write(0x4, "\006\0", 0x4) = 4 0

 1382/0xc73b:  write(0x4, "aaaaff\0", 0x6) = 6 0

 1383/0xc75d:  read(0x3, "\006\0", 0x4) = 4 0

 1383/0xc75d:  read(0x3, "aaaaff\0", 0x6) = 6 0

 1383/0xc75d:  write(0x6, "\001\0", 0x4) = 4 0

 1382/0xc73b:  read(0x5, "\001\0", 0x4) = 4 0

 1382/0xc73b:  write\_nocancel(0x1, "The string does not fit the rule\n\0", 0x21) = 33 0

Affff

Added string Affff to file!

 1382/0xc73b:  read\_nocancel(0x0, "Affff\n\0", 0x1000) = 6 0

 1382/0xc73b:  write(0x4, "\005\0", 0x4) = 4 0

 1383/0xc75d:  read(0x3, "\005\0", 0x4) = 4 0

 1382/0xc73b:  write(0x4, "Affff\0", 0x5) = 5 0

 1383/0xc75d:  read(0x3, "Affff\0", 0x5) = 5 0

 1383/0xc75d:  fstat64(0x7, 0x16EF16F60, 0x0) = 0 0

 1383/0xc75d:  write\_nocancel(0x7, "Affff\n\0", 0x6) = 6 0

 1383/0xc75d:  fstat64(0x1, 0x16EF17010, 0x0) = 0 0

 1383/0xc75d:  ioctl(0x1, 0x4004667A, 0x16EF1705C) = 0 0

 1383/0xc75d:  write\_nocancel(0x1, "Added string Affff to file!\n\0", 0x1C) = 28 0

 1383/0xc75d:  write(0x6, "\026\0", 0x4) = 4 0

 1382/0xc73b:  read(0x5, "\026\0", 0x4) = 4 0

 1382/0xc73b:  close(0x3) = 0 0

 1382/0xc73b:  close(0x4) = 0 0

 1382/0xc73b:  close(0x5) = 0 0

 1382/0xc73b:  close(0x6) = 0 0

 1382/0xc73b:  lseek(0x0, 0xFFFFFFFFFFFFFFFF, 0x1) = 31148 0

**Выводы**

Благодаря данной лабораторной работе, я узнала о существовании такой утилиты, как dtruss. Также я научилась получать информацию из диагностики своих программ