Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

Группа: М80-206Б-22

Студент: Жаднов М. Д.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 06.10.23

Москва, 2023

**Постановка задачи**

**Группа вариантов 2.**

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода. Стандартный поток вывода дочернего процесса перенаправляется в pipe1. Родительский процесс читает из pipe1 и прочитанное выводит в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами.

**Вариант 8.**

В файле записаны команды вида:«число число число<endline>». Дочерний процесс производит деление первого числа команда, на последующие числа в команде, а результат выводит в стандартный поток вывода. Если происходит деление на 0, то тогда дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип int. Количество чисел может быть произвольным.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс.
* int pipe(int \*fd); – создает неименованный канал, у которого первое поле отвечает за чтение, а второе - за запись.
* int execv(const char \*\_\_path, char \*const \*\_\_argv); - предоставляют новой программе список аргументов в виде массива указателей на строки, заканчивающиеся (char \*)0.
* int dup2(int, int); - создает копию файлового дескриптора oldfd *(1 поле)*, используя для нового дескриптора newfd *(2 поле)* файловый дескриптор (они становятся взаимозаменяемыми).
* \_exit(int status); – выходит из процесса с заданным статусом.
* pid\_t wait(int \*status); – приостаналивает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится.
* int read(int fd, void \*buffer, int nbyte); – читает nbyte байтов из файлового дескриптора fd в буффер buffer.

Программа parent.c принимает аргументом название файла, который нужно будет прочитать. Далее происходит проверка поданного файла на чтение, и если прочитался успешно, создаётся pipe и дочерний процесс (с дальнейшими проверками их создания, конечно же). Потом родительский процесс ждет дочерний, пока тот запустит программу child.c (с установкой на стандартный ввод из файлового дескриптора с открытым файлом и стандартный поток ошибок в pipe\_fd[1] (на запись)), в которой происходит работа над данными файла и вывод результата в стандартный вывод.

Если программа child.c завершилась неудачно (деление на нуль), то в pipe (стандартный поток ошибок для дочернего процесса) записывается ошибка.

Далее, подождав окончания дочернего процесса, родительский процесс проверяет, завершился ли дочерний процесс успехом или нет, и если нет, то выводит ошибку (деление на нуль) и завершается неуспехом. В случае успеха завершает свою работу успехом.

**Код программы**

**parent.c**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <string.h>

#include <stdbool.h>

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

if (argc != 2) {

perror("\nError: no filename\n");

exit(EXIT\_FAILURE);

}

int fd = open(argv[1], O\_RDONLY);

if(fd == -1){

perror("\nCan't open file\n");

\_exit(EXIT\_FAILURE);

}

int pipe\_fd[2];

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

perror("\npipe: Here is a problem\n");

\_exit(EXIT\_FAILURE);

}

pid\_t pid = fork();

if (pid == -1) {

perror("\nfork: Here is a problem\n");

\_exit(EXIT\_FAILURE);

}else{

if(pid == 0){ //child

close (pipe\_fd[0]);

dup2(fd, STDIN\_FILENO);

dup2(pipe\_fd[1], STDERR\_FILENO);

char\* args[] = {"./child", NULL};

if (execv(args[0], args) == -1) {

fprintf(stderr, "Unable to exec\n");

\_exit(EXIT\_FAILURE);

}

}else{ //parent

close(pipe\_fd[1]);

wait(0);

char ch\_status[17];

while(read(pipe\_fd[0], ch\_status, 1)){

if(strcmp(ch\_status, "") != 0){

printf("\nDivision by zero\n");

\_exit(EXIT\_FAILURE);

}

}

}

}

return 0;

}

**child.c**

#include "stdio.h"

#include "stdlib.h"

#include "unistd.h"

int main(){

int c = '\0';

int tmp = 0, res = 0;

int end\_of\_str = 0;

do{

c = getchar();

if(!end\_of\_str){

if(c>='0' && c<='9'){

tmp = tmp\*10 + c - '0';

}

if(c == ' ' || c == '\n' || c == EOF){

if(res == 0 && tmp != 0){

res = tmp;

}

else if(res != 0 && tmp != 0){

res /= tmp;

}

else if(res == 0 && tmp == 0){

end\_of\_str = 1;

}

else if(res != 0 && tmp == 0){

fprintf(stderr, "Division by zero");

\_exit(EXIT\_FAILURE);

}

tmp = 0;

}

}

if(c == '\n' || c == EOF){

printf("%d\n", res);

end\_of\_str = 0;

res = 0;

}

}while(c != EOF);

return 0;

}

**Протокол работы программы**

**Тестирование:**

mishazhadnov@McB-airmi scr % ./parent test.txt

1

3

0

Division by zero

**Dtrace (аналог strace):**

mishazhadnov@McB-airmi scr % sudo dtruss -f ./parent test.txt

dtrace: system integrity protection is on, some features will not be available

PID/THRD SYSCALL(args) = return

1

3

0

Division by zero

**46558/0x4b5196:** fork() = 0 0

46558/0x4b5196: munmap(0x1141D5000, 0x9C000) = 0 0

46558/0x4b5196: munmap(0x114271000, 0x8000) = 0 0

46558/0x4b5196: munmap(0x114279000, 0x4000) = 0 0

46558/0x4b5196: munmap(0x11427D000, 0x4000) = 0 0

46558/0x4b5196: munmap(0x114281000, 0x54000) = 0 0

46558/0x4b5196: open(".\0", 0x100000, 0x0) = 3 0

46558/0x4b5196: fcntl(0x3, 0x32, 0x7FF7B81632A0) = 0 0

46558/0x4b5196: close(0x3) = 0 0

46558/0x4b5196: fsgetpath(0x7FF7B81632B0, 0x400, 0x7FF7B8163298) = 58 0

46558/0x4b5196: fsgetpath(0x7FF7B81632B0, 0x400, 0x7FF7B8163298) = 14 0

46558/0x4b5196: csrctl(0x0, 0x7FF7B81636BC, 0x4) = -1 1

46558/0x4b5196: \_\_mac\_syscall(0x7FF810C2E11B, 0x2, 0x7FF7B8163530) = 0 0

46558/0x4b5196: csrctl(0x0, 0x7FF7B81636CC, 0x4) = -1 1

46558/0x4b5196: \_\_mac\_syscall(0x7FF810C2B0A8, 0x5A, 0x7FF7B8163660) = 0 0

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

46558/0x4b5196: open("/\0", 0x20100000, 0x0) = 3 0

46558/0x4b5196: openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0) = 4 0

46558/0x4b5196: dup(0x4, 0x0, 0x0) = 5 0

46558/0x4b5196: fstatat64(0x4, 0x7FF7B8162401, 0x7FF7B8162800) = 0 0

46558/0x4b5196: openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0) = 6 0

46558/0x4b5196: fcntl(0x6, 0x32, 0x7FF7B8162490) = 0 0

46558/0x4b5196: dup(0x6, 0x0, 0x0) = 7 0

46558/0x4b5196: dup(0x5, 0x0, 0x0) = 8 0

46558/0x4b5196: close(0x3) = 0 0

46558/0x4b5196: close(0x5) = 0 0

46558/0x4b5196: close(0x4) = 0 0

46558/0x4b5196: close(0x6) = 0 0

46558/0x4b5196: shared\_region\_check\_np(0x7FF7B8162D88, 0x0, 0x0) = 0 0

46558/0x4b5196: fsgetpath(0x7FF7B81632E0, 0x400, 0x7FF7B8163218) = 83 0

46558/0x4b5196: fcntl(0x8, 0x32, 0x7FF7B81632E0) = 0 0

46558/0x4b5196: close(0x8) = 0 0

46558/0x4b5196: close(0x7) = 0 0

46558/0x4b5196: getfsstat64(0x0, 0x0, 0x2) = 8 0

46558/0x4b5196: getfsstat64(0x107DA2A10, 0x43C0, 0x2) = 8 0

46558/0x4b5196: getattrlist("/\0", 0x7FF7B8163170, 0x7FF7B81630E0) = 0 0

46558/0x4b5196: fsgetpath(0x7FF7B8162F60, 0x400, 0x7FF7B8162F48) = 83 0

46558/0x4b5196: stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld\_shared\_cache\_x86\_64h\0", 0x7FF7B81633C8, 0x0) = 0 0

46558/0x4b5196: stat64("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/parent\0", 0x7FF7B81629F0, 0x0) = 0 0

46558/0x4b5196: open("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/parent\0", 0x0, 0x0) = 3 0

46558/0x4b5196: mmap(0x0, 0x364C, 0x1, 0x40002, 0x3, 0x0) = 0x107DE1000 0

46558/0x4b5196: fcntl(0x3, 0x32, 0x7FF7B8162B00) = 0 0

46558/0x4b5196: close(0x3) = 0 0

46558/0x4b5196: munmap(0x107DE1000, 0x364C) = 0 0

46558/0x4b5196: stat64("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/parent\0", 0x7FF7B8162F50, 0x0) = 0 0

46558/0x4b5196: stat64("/usr/lib/libSystem.B.dylib\0", 0x7FF7B8161FA0, 0x0) = -1 2

46558/0x4b5196: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x7FF7B8161F50, 0x0) = -1 2

46558/0x4b5196: stat64("/usr/lib/system/libdispatch.dylib\0", 0x7FF7B815FBA0, 0x0) = -1 2

46558/0x4b5196: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/system/libdispatch.dylib\0", 0x7FF7B815FB50, 0x0) = -1 2

46558/0x4b5196: stat64("/usr/lib/system/libdispatch.dylib\0", 0x7FF7B815FBA0, 0x0) = -1 2

46558/0x4b5196: open("/dev/dtracehelper\0", 0x2, 0x0) = 3 0

46558/0x4b5196: ioctl(0x3, 0x80086804, 0x7FF7B8161BA8) = 0 0

46558/0x4b5196: close(0x3) = 0 0

46558/0x4b5196: mprotect(0x107D9D000, 0x1000, 0x1) = 0 0

46558/0x4b5196: shared\_region\_check\_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

46558/0x4b5196: mprotect(0x107DA0000, 0x40000, 0x1) = 0 0

46558/0x4b5196: access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 2

46558/0x4b5196: bsdthread\_register(0x7FF810EF5BC4, 0x7FF810EF5BB0, 0x2000) = 1073742303 0

46558/0x4b5196: shm\_open(0x7FF810DA0F5A, 0x0, 0x10D9F465) = 3 0

46558/0x4b5196: fstat64(0x3, 0x7FF7B8161DF0, 0x0) = 0 0

46558/0x4b5196: mmap(0x0, 0x3000, 0x1, 0x40001, 0x3, 0x0) = 0x107DE3000 0

46558/0x4b5196: close(0x3) = 0 0

46558/0x4b5196: ioctl(0x2, 0x4004667A, 0x7FF7B8161EA4) = 0 0

46558/0x4b5196: mprotect(0x107DEB000, 0x1000, 0x0) = 0 0

46558/0x4b5196: mprotect(0x107DF2000, 0x1000, 0x0) = 0 0

46558/0x4b5196: mprotect(0x107DF3000, 0x1000, 0x0) = 0 0

46558/0x4b5196: mprotect(0x107DFA000, 0x1000, 0x0) = 0 0

46558/0x4b5196: mprotect(0x107DE6000, 0x98, 0x1) = 0 0

46558/0x4b5196: mprotect(0x107DE6000, 0x98, 0x3) = 0 0

46558/0x4b5196: mprotect(0x107DE6000, 0x98, 0x1) = 0 0

46558/0x4b5196: mprotect(0x107DFB000, 0x1000, 0x1) = 0 0

46558/0x4b5196: mprotect(0x107DFC000, 0x98, 0x1) = 0 0

46558/0x4b5196: mprotect(0x107DFC000, 0x98, 0x3) = 0 0

46558/0x4b5196: mprotect(0x107DFC000, 0x98, 0x1) = 0 0

46558/0x4b5196: mprotect(0x107DE6000, 0x98, 0x3) = 0 0

46558/0x4b5196: mprotect(0x107DE6000, 0x98, 0x1) = 0 0

46558/0x4b5196: mprotect(0x107DFB000, 0x1000, 0x3) = 0 0

46558/0x4b5196: mprotect(0x107DFB000, 0x1000, 0x1) = 0 0

46558/0x4b5196: mprotect(0x107DA0000, 0x40000, 0x3) = 0 0

46558/0x4b5196: mprotect(0x107DA0000, 0x40000, 0x1) = 0 0

46558/0x4b5196: issetugid(0x0, 0x0, 0x0) = 0 0

46558/0x4b5196: mprotect(0x107DA0000, 0x40000, 0x3) = 0 0

46558/0x4b5196: getentropy(0x7FF7B8161950, 0x20, 0x0) = 0 0

46558/0x4b5196: mprotect(0x107DA0000, 0x40000, 0x1) = 0 0

46558/0x4b5196: getpid(0x0, 0x0, 0x0) = 46558 0

46558/0x4b5196: mprotect(0x107DA0000, 0x40000, 0x3) = 0 0

46558/0x4b5196: mprotect(0x107DA0000, 0x40000, 0x1) = 0 0

46558/0x4b5196: getattrlist("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/parent\0", 0x7FF7B8161DD0, 0x7FF7B8161DE8) = 0 0

46558/0x4b5196: access("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr\0", 0x4, 0x0) = 0 0

46558/0x4b5196: open("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr\0", 0x0, 0x0) = 3 0

46558/0x4b5196: fstat64(0x3, 0x7FE9D77044B0, 0x0) = 0 0

46558/0x4b5196: csrctl(0x0, 0x7FF7B816205C, 0x4) = -1 1

46558/0x4b5196: fgetattrlist(0x3, 0x7FF7B8162070, 0x7FF7B8162090) = 0 0

46558/0x4b5196: \_\_mac\_syscall(0x7FF81B4B2719, 0x2, 0x7FF7B8162090) = 0 0

46558/0x4b5196: fcntl(0x3, 0x32, 0x7FF7B8161D00) = 0 0

46558/0x4b5196: close(0x3) = 0 0

46558/0x4b5196: open("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/Info.plist\0", 0x0, 0x0) = -1 2

46558/0x4b5196: proc\_info(0x2, 0xB5DE, 0xD) = 64 0

46558/0x4b5196: csops\_audittoken(0xB5DE, 0x10, 0x7FF7B8161FE0) = -1 22

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

46558/0x4b5196: csops(0xB5DE, 0x0, 0x7FF7B8162444) = 0 0

46558/0x4b5196: sysctlbyname(kern.system\_version\_compat, 0x1A, 0x0, 0x0, 0x7FF7B8162474) = 0 0

46558/0x4b5196: mprotect(0x107DA0000, 0x40000, 0x3) = 0 0

46558/0x4b5196: open("test.txt\0", 0x0, 0x0) = 3 0

**46558/0x4b5196:** pipe(0x0, 0x0, 0x0) = 4 0

**46558/0x4b5196:** fork() = 46559 0

46558/0x4b5196: close(0x5) = 0 0

**46559/0x4b519c:** fork() = 0 0

46559/0x4b519c: thread\_selfid(0x0, 0x0, 0x0) = 4936092 0

46559/0x4b519c: bsdthread\_register(0x7FF810EF5BC4, 0x7FF810EF5BB0, 0x2000) = -1 22

46559/0x4b519c: mprotect(0x107DFC000, 0x98, 0x3) = 0 0

46559/0x4b519c: mprotect(0x107DFC000, 0x98, 0x1) = 0 0

46559/0x4b519c: close(0x4) = 0 0

46559/0x4b519c: dup2(0x3, 0x0, 0x0) = 0 0

46559/0x4b519c: dup2(0x5, 0x2, 0x0) = 2 0

dtrace: error on enabled probe ID 1688 (ID 285: syscall::execve:return): invalid address (0x107d9cf85) in action #12 at DIF offset 12

**46559/0x4b519d:** fork() = 0 0

46559/0x4b519d: mprotect(0x113488000, 0x8000, 0x1) = 0 0

46559/0x4b519d: thread\_selfid(0x0, 0x0, 0x0) = 4936093 0

46559/0x4b519d: shared\_region\_check\_np(0x7FF7B0EBD948, 0x0, 0x0) = 0 0

46559/0x4b519d: thread\_selfid(0x0, 0x0, 0x0) = 4936093 0

46559/0x4b519d: getpid(0x0, 0x0, 0x0) = 46559 0

46559/0x4b519d: proc\_info(0xF, 0xB5DF, 0x0) = 0 0

46559/0x4b519d: munmap(0x1133EC000, 0x9C000) = 0 0

46559/0x4b519d: munmap(0x113488000, 0x8000) = 0 0

46559/0x4b519d: munmap(0x113490000, 0x4000) = 0 0

46559/0x4b519d: munmap(0x113494000, 0x4000) = 0 0

46559/0x4b519d: munmap(0x113498000, 0x54000) = 0 0

46559/0x4b519d: open(".\0", 0x100000, 0x0) = 4 0

46559/0x4b519d: fcntl(0x4, 0x32, 0x7FF7B0EBD2C0) = 0 0

46559/0x4b519d: close(0x4) = 0 0

46559/0x4b519d: fsgetpath(0x7FF7B0EBD2D0, 0x400, 0x7FF7B0EBD2B8) = 57 0

46559/0x4b519d: fsgetpath(0x7FF7B0EBD2D0, 0x400, 0x7FF7B0EBD2B8) = 14 0

46559/0x4b519d: csrctl(0x0, 0x7FF7B0EBD6DC, 0x4) = -1 1

46559/0x4b519d: \_\_mac\_syscall(0x7FF810C2E11B, 0x2, 0x7FF7B0EBD550) = 0 0

46559/0x4b519d: csrctl(0x0, 0x7FF7B0EBD6EC, 0x4) = -1 1

46559/0x4b519d: \_\_mac\_syscall(0x7FF810C2B0A8, 0x5A, 0x7FF7B0EBD680) = 0 0

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

46559/0x4b519d: open("/\0", 0x20100000, 0x0) = 4 0

46559/0x4b519d: openat(0x4, "System/Cryptexes/OS\0", 0x100000, 0x0) = 6 0

46559/0x4b519d: dup(0x6, 0x0, 0x0) = 7 0

46559/0x4b519d: fstatat64(0x6, 0x7FF7B0EBC421, 0x7FF7B0EBC820) = 0 0

46559/0x4b519d: openat(0x6, "System/Library/dyld/\0", 0x100000, 0x0) = 8 0

46559/0x4b519d: fcntl(0x8, 0x32, 0x7FF7B0EBC4B0) = 0 0

46559/0x4b519d: dup(0x8, 0x0, 0x0) = 9 0

46559/0x4b519d: dup(0x7, 0x0, 0x0) = 10 0

46559/0x4b519d: close(0x4) = 0 0

46559/0x4b519d: close(0x7) = 0 0

46559/0x4b519d: close(0x6) = 0 0

46559/0x4b519d: close(0x8) = 0 0

46559/0x4b519d: shared\_region\_check\_np(0x7FF7B0EBCDA8, 0x0, 0x0) = 0 0

46559/0x4b519d: fsgetpath(0x7FF7B0EBD300, 0x400, 0x7FF7B0EBD238) = 83 0

46559/0x4b519d: fcntl(0xA, 0x32, 0x7FF7B0EBD300) = 0 0

46559/0x4b519d: close(0xA) = 0 0

46559/0x4b519d: close(0x9) = 0 0

46559/0x4b519d: getfsstat64(0x0, 0x0, 0x2) = 8 0

46559/0x4b519d: getfsstat64(0x10F048A10, 0x43C0, 0x2) = 8 0

46559/0x4b519d: getattrlist("/\0", 0x7FF7B0EBD190, 0x7FF7B0EBD100) = 0 0

46559/0x4b519d: fsgetpath(0x7FF7B0EBCF80, 0x400, 0x7FF7B0EBCF68) = 83 0

46559/0x4b519d: stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld\_shared\_cache\_x86\_64h\0", 0x7FF7B0EBD3E8, 0x0) = 0 0

46559/0x4b519d: stat64("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/child\0", 0x7FF7B0EBCA10, 0x0) = 0 0

46559/0x4b519d: open("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/child\0", 0x0, 0x0) = 4 0

46559/0x4b519d: mmap(0x0, 0x32D0, 0x1, 0x40002, 0x4, 0x0) = 0x10F087000 0

46559/0x4b519d: fcntl(0x4, 0x32, 0x7FF7B0EBCB20) = 0 0

46559/0x4b519d: close(0x4) = 0 0

46559/0x4b519d: munmap(0x10F087000, 0x32D0) = 0 0

46559/0x4b519d: stat64("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/child\0", 0x7FF7B0EBCF70, 0x0) = 0 0

46559/0x4b519d: stat64("/usr/lib/libSystem.B.dylib\0", 0x7FF7B0EBBFC0, 0x0) = -1 2

46559/0x4b519d: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x7FF7B0EBBF70, 0x0) = -1 2

46559/0x4b519d: stat64("/usr/lib/system/libdispatch.dylib\0", 0x7FF7B0EB9BC0, 0x0) = -1 2

46559/0x4b519d: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/system/libdispatch.dylib\0", 0x7FF7B0EB9B70, 0x0) = -1 2

46559/0x4b519d: stat64("/usr/lib/system/libdispatch.dylib\0", 0x7FF7B0EB9BC0, 0x0) = -1 2

46559/0x4b519d: open("/dev/dtracehelper\0", 0x2, 0x0) = 4 0

46559/0x4b519d: ioctl(0x4, 0x80086804, 0x7FF7B0EBBBC8) = 0 0

46559/0x4b519d: close(0x4) = 0 0

46559/0x4b519d: mprotect(0x10F043000, 0x1000, 0x1) = 0 0

46559/0x4b519d: shared\_region\_check\_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

46559/0x4b519d: mprotect(0x10F046000, 0x40000, 0x1) = 0 0

46559/0x4b519d: access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 2

46559/0x4b519d: bsdthread\_register(0x7FF810EF5BC4, 0x7FF810EF5BB0, 0x2000) = 1073742303 0

46559/0x4b519d: shm\_open(0x7FF810DA0F5A, 0x0, 0x10D9F465) = 4 0

46559/0x4b519d: fstat64(0x4, 0x7FF7B0EBBE10, 0x0) = 0 0

46559/0x4b519d: mmap(0x0, 0x3000, 0x1, 0x40001, 0x4, 0x0) = 0x10F089000 0

46559/0x4b519d: close(0x4) = 0 0

46559/0x4b519d: ioctl(0x2, 0x4004667A, 0x7FF7B0EBBEC4) = -1 25

46559/0x4b519d: ioctl(0x2, 0x40487413, 0x7FF7B0EBBEC8) = -1 25

46559/0x4b519d: mprotect(0x10F091000, 0x1000, 0x0) = 0 0

46559/0x4b519d: mprotect(0x10F098000, 0x1000, 0x0) = 0 0

46559/0x4b519d: mprotect(0x10F099000, 0x1000, 0x0) = 0 0

46559/0x4b519d: mprotect(0x10F0A0000, 0x1000, 0x0) = 0 0

46559/0x4b519d: mprotect(0x10F08C000, 0x98, 0x1) = 0 0

46559/0x4b519d: mprotect(0x10F08C000, 0x98, 0x3) = 0 0

46559/0x4b519d: mprotect(0x10F08C000, 0x98, 0x1) = 0 0

46559/0x4b519d: mprotect(0x10F0A1000, 0x1000, 0x1) = 0 0

46559/0x4b519d: mprotect(0x10F0A2000, 0x98, 0x1) = 0 0

46559/0x4b519d: mprotect(0x10F0A2000, 0x98, 0x3) = 0 0

46559/0x4b519d: mprotect(0x10F0A2000, 0x98, 0x1) = 0 0

46559/0x4b519d: mprotect(0x10F08C000, 0x98, 0x3) = 0 0

46559/0x4b519d: mprotect(0x10F08C000, 0x98, 0x1) = 0 0

46559/0x4b519d: mprotect(0x10F0A1000, 0x1000, 0x3) = 0 0

46559/0x4b519d: mprotect(0x10F0A1000, 0x1000, 0x1) = 0 0

46559/0x4b519d: mprotect(0x10F046000, 0x40000, 0x3) = 0 0

46559/0x4b519d: mprotect(0x10F046000, 0x40000, 0x1) = 0 0

46559/0x4b519d: issetugid(0x0, 0x0, 0x0) = 0 0

46559/0x4b519d: mprotect(0x10F046000, 0x40000, 0x3) = 0 0

46559/0x4b519d: getentropy(0x7FF7B0EBB970, 0x20, 0x0) = 0 0

46559/0x4b519d: mprotect(0x10F046000, 0x40000, 0x1) = 0 0

46559/0x4b519d: getpid(0x0, 0x0, 0x0) = 46559 0

46559/0x4b519d: mprotect(0x10F046000, 0x40000, 0x3) = 0 0

46559/0x4b519d: mprotect(0x10F046000, 0x40000, 0x1) = 0 0

46559/0x4b519d: getattrlist("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/child\0", 0x7FF7B0EBBDF0, 0x7FF7B0EBBE08) = 0 0

46559/0x4b519d: access("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr\0", 0x4, 0x0) = 0 0

46559/0x4b519d: open("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr\0", 0x0, 0x0) = 4 0

46559/0x4b519d: fstat64(0x4, 0x7FAF76F044B0, 0x0) = 0 0

46559/0x4b519d: csrctl(0x0, 0x7FF7B0EBC07C, 0x4) = -1 1

46559/0x4b519d: fgetattrlist(0x4, 0x7FF7B0EBC090, 0x7FF7B0EBC0B0) = 0 0

46559/0x4b519d: \_\_mac\_syscall(0x7FF81B4B2719, 0x2, 0x7FF7B0EBC0B0) = 0 0

46559/0x4b519d: fcntl(0x4, 0x32, 0x7FF7B0EBBD20) = 0 0

46559/0x4b519d: close(0x4) = 0 0

46559/0x4b519d: open("/Users/mishazhadnov/Desktop/wD/OS\_labs\_3t/lab1/scr/Info.plist\0", 0x0, 0x0) = -1 2

46559/0x4b519d: proc\_info(0x2, 0xB5DF, 0xD) = 64 0

46559/0x4b519d: csops\_audittoken(0xB5DF, 0x10, 0x7FF7B0EBC000) = -1 22

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid kernel access in action #11 at DIF offset 28

46559/0x4b519d: csops(0xB5DF, 0x0, 0x7FF7B0EBC464) = 0 0

46559/0x4b519d: sysctlbyname(kern.system\_version\_compat, 0x1A, 0x0, 0x0, 0x7FF7B0EBC494) = 0 0

46559/0x4b519d: mprotect(0x10F046000, 0x40000, 0x3) = 0 0

46559/0x4b519d: getrlimit(0x1008, 0x7FF7B0EBD5F0, 0x0) = 0 0

46559/0x4b519d: fstat64(0x0, 0x7FF7B0EBD5D8, 0x0) = 0 0

dtrace: error on enabled probe ID 1714 (ID 959: syscall::read\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

46559/0x4b519d: fstat64(0x1, 0x7FF7B0EBD438, 0x0) = 0 0

46559/0x4b519d: ioctl(0x1, 0x4004667A, 0x7FF7B0EBD484) = 0 0

dtrace: error on enabled probe ID 1712 (ID 961: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 1712 (ID 961: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 1712 (ID 961: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 1712 (ID 961: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

46558/0x4b5196: wait4(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 46559 0

dtrace: error on enabled probe ID 1713 (ID 173: syscall::read:return): invalid kernel access in action #13 at DIF offset 68

46558/0x4b5196: getrlimit(0x1008, 0x7FF7B81633E0, 0x0) = 0 0

46558/0x4b5196: fstat64(0x1, 0x7FF7B81633C8, 0x0) = 0 0

46558/0x4b5196: ioctl(0x1, 0x4004667A, 0x7FF7B8163414) = 0 0

dtrace: error on enabled probe ID 1712 (ID 961: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 1712 (ID 961: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

**Вывод**

Благодаря данной лабораторной работе я на практике изучил принципы работы с неименованными каналами для межпроцессного взаимодействия, разобрался, как перенаправлять потоки ввода/вывода, а также научился использовать системные вызовы и обращаться с файловыми дескрипторами (которые важно вовремя и уместно закрывать).

Очевидно, что в реальных, “рабочих” программах используется большее количество неименованных каналов и процессов. Эта лабораторная работа научила базовому обращению с ними.