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: 20.10.23

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); - предоставляет 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.

Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение Далее происходит проверка поданного файла на чтение, и если прочитался успешно, создаётся pipe и дочерний процесс (с дальнейшими проверками их создания, конечно же). Потом происходит перераспределение файловых дескрипторов стандартного ввода (на файл) и вывода (на pipe) в дочернем процессе. Следующим шагом дочерний процесс запускает программу child.c и обрабатывает свой стандартный ввод. В то же время, родительский процесс читает pipe и выводит полученные результаты в стандартный поток вывода, а если встречает -1 (что значит завершение программы дочернего процесса неудачей), то выводит сообщение об ошибке (“Division by zero”) и завершает работу.

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[]){

    char* filename;
    if(!scanf("%s", filename)){
       perror("\nScan file problem!\n");
       _exit(EXIT_FAILURE);
    }

    int fd = open(filename, 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], STDOUT_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);
        int result = 0;
        while(read(pipe_fd[0], &result, sizeof(int))){
            if(result == -1){
                printf("Division by zero\n");
                _exit(EXIT_FAILURE);
            }
            else printf("%d\n", result);
        }
    }
    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{
        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){
                    res = -1;
                    write(STDOUT_FILENO, &res, sizeof(int));
                    _exit(EXIT_FAILURE);
                }
                tmp = 0;
            }
        }
        if(c == '\n' || c == EOF){
            write(STDOUT_FILENO, &res, sizeof(int));
            end_of_str = 0;
            res = 0;
        }
    }while(read(STDIN_FILENO, &c, sizeof(char)) > 0);

    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 ./parent

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


SYSCALL(args)           = return

munmap(0x111A3D000, 0x9C000)            = 0 0

munmap(0x111AD9000, 0x8000)             = 0 0

munmap(0x111AE1000, 0x4000)             = 0 0

munmap(0x111AE5000, 0x4000)             = 0 0

munmap(0x111AE9000, 0x54000)            = 0 0

open(".\0", 0x100000, 0x0)              = 3 0

fcntl(0x3, 0x32, 0x7FF7B3A0A250)                = 0 0

close(0x3)              = 0 0

fsgetpath(0x7FF7B3A0A260, 0x400, 0x7FF7B3A0A248)                = 58 0

fsgetpath(0x7FF7B3A0A260, 0x400, 0x7FF7B3A0A248)                = 14 0

csrctl(0x0, 0x7FF7B3A0A66C, 0x4)                = -1 1

__mac_syscall(0x7FF810C2E11B, 0x2, 0x7FF7B3A0A4E0)              = 0 0

csrctl(0x0, 0x7FF7B3A0A67C, 0x4)                = -1 1
```

```
__mac_syscall(0x7FF810C2B0A8, 0x5A, 0x7FF7B3A0A610)             = 0 0

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid
kernel access in action #10 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid
kernel access in action #10 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid
kernel access in action #10 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid
kernel access in action #10 at DIF offset 28

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

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

dup(0x4, 0x0, 0x0)              = 5 0

fstatat64(0x4, 0x7FF7B3A093B1, 0x7FF7B3A097B0)          = 0 0

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

fcntl(0x6, 0x32, 0x7FF7B3A09440)                = 0 0

dup(0x6, 0x0, 0x0)              = 7 0

dup(0x5, 0x0, 0x0)              = 8 0

close(0x3)              = 0 0

close(0x5)              = 0 0

close(0x4)              = 0 0

close(0x6)              = 0 0

shared_region_check_np(0x7FF7B3A09D38, 0x0, 0x0)                = 0 0

fsgetpath(0x7FF7B3A0A290, 0x400, 0x7FF7B3A0A1C8)                = 83 0

fcntl(0x8, 0x32, 0x7FF7B3A0A290)                = 0 0

close(0x8)              = 0 0

close(0x7)              = 0 0

getfsstat64(0x0, 0x0, 0x2)              = 8 0
```

```
getfsstat64(0x10C4FBA10, 0x43C0, 0x2)            = 8 0

getattrlist("/\0", 0x7FF7B3A0A120, 0x7FF7B3A0A090)               = 0 0

fsgetpath(0x7FF7B3A09F10, 0x400, 0x7FF7B3A09EF8)                 = 83 0

stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld_shared_cac
he_x86_64h\0", 0x7FF7B3A0A378, 0x0)              = 0 0

stat64("/Users/mishazhadnov/Desktop/wD/OS_labs_3t/lab1/scr/parent\0",
0x7FF7B3A099A0, 0x0)               = 0 0

open("/Users/mishazhadnov/Desktop/wD/OS_labs_3t/lab1/scr/parent\0", 0x0, 0x0)
= 3 0

mmap(0x0, 0x33C4, 0x1, 0x40002, 0x3, 0x0)                = 0x10C53A000 0

fcntl(0x3, 0x32, 0x7FF7B3A09AB0)                 = 0 0

close(0x3)               = 0 0

munmap(0x10C53A000, 0x33C4)              = 0 0

stat64("/Users/mishazhadnov/Desktop/wD/OS_labs_3t/lab1/scr/parent\0",
0x7FF7B3A09F00, 0x0)               = 0 0

stat64("/usr/lib/libSystem.B.dylib\0", 0x7FF7B3A08F50, 0x0)              = -1 2

stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0",
0x7FF7B3A08F00, 0x0)          = -1 2

stat64("/usr/lib/system/libdispatch.dylib\0", 0x7FF7B3A06B50, 0x0)
= -1 2

stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/system/libdispatch.dylib\0"
, 0x7FF7B3A06B00, 0x0)           = -1 2

stat64("/usr/lib/system/libdispatch.dylib\0", 0x7FF7B3A06B50, 0x0)
= -1 2

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

ioctl(0x3, 0x80086804, 0x7FF7B3A08B58)           = 0 0

close(0x3)               = 0 0

mprotect(0x10C4F6000, 0x1000, 0x1)               = 0 0
```

```
shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0)             = 0 0

mprotect(0x10C4F9000, 0x40000, 0x1)              = 0 0

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

bsdthread_register(0x7FF810EF5BC4, 0x7FF810EF5BB0, 0x2000)               =
1073742303 0

shm_open(0x7FF810DA0F5A, 0x0, 0x10D9F465)                = 3 0

fstat64(0x3, 0x7FF7B3A08DA0, 0x0)                = 0 0

mmap(0x0, 0x3000, 0x1, 0x40001, 0x3, 0x0)                = 0x10C53C000 0

close(0x3)               = 0 0

ioctl(0x2, 0x4004667A, 0x7FF7B3A08E54)           = 0 0

mprotect(0x10C544000, 0x1000, 0x0)               = 0 0

mprotect(0x10C54B000, 0x1000, 0x0)               = 0 0

mprotect(0x10C54C000, 0x1000, 0x0)               = 0 0

mprotect(0x10C553000, 0x1000, 0x0)               = 0 0

mprotect(0x10C53F000, 0x98, 0x1)                 = 0 0

mprotect(0x10C53F000, 0x98, 0x3)                 = 0 0

mprotect(0x10C53F000, 0x98, 0x1)                 = 0 0

mprotect(0x10C554000, 0x1000, 0x1)               = 0 0

mprotect(0x10C555000, 0x98, 0x1)                 = 0 0

mprotect(0x10C555000, 0x98, 0x3)                 = 0 0

mprotect(0x10C555000, 0x98, 0x1)                 = 0 0

mprotect(0x10C53F000, 0x98, 0x3)                 = 0 0

mprotect(0x10C53F000, 0x98, 0x1)                 = 0 0

mprotect(0x10C554000, 0x1000, 0x3)               = 0 0

mprotect(0x10C554000, 0x1000, 0x1)               = 0 0

mprotect(0x10C4F9000, 0x40000, 0x3)              = 0 0
```

```
mprotect(0x10C4F9000, 0x40000, 0x1)              = 0 0

issetugid(0x0, 0x0, 0x0)                 = 0 0

mprotect(0x10C4F9000, 0x40000, 0x3)              = 0 0

getentropy(0x7FF7B3A08900, 0x20, 0x0)            = 0 0

mprotect(0x10C4F9000, 0x40000, 0x1)              = 0 0

getpid(0x0, 0x0, 0x0)            = 9784 0

mprotect(0x10C4F9000, 0x40000, 0x3)              = 0 0

mprotect(0x10C4F9000, 0x40000, 0x1)              = 0 0

getattrlist("/Users/mishazhadnov/Desktop/wD/OS_labs_3t/lab1/scr/parent\0",
0x7FF7B3A08D80, 0x7FF7B3A08D98)               = 0 0

access("/Users/mishazhadnov/Desktop/wD/OS_labs_3t/lab1/scr\0", 0x4, 0x0)
= 0 0

open("/Users/mishazhadnov/Desktop/wD/OS_labs_3t/lab1/scr\0", 0x0, 0x0)
= 3 0

fstat64(0x3, 0x7FADF2704500, 0x0)                = 0 0

csrctl(0x0, 0x7FF7B3A0900C, 0x4)                 = -1 1

fgetattrlist(0x3, 0x7FF7B3A09020, 0x7FF7B3A09040)                = 0 0

__mac_syscall(0x7FF81B4B2719, 0x2, 0x7FF7B3A09040)               = 0 0

fcntl(0x3, 0x32, 0x7FF7B3A08CB0)                 = 0 0

close(0x3)               = 0 0

open("/Users/mishazhadnov/Desktop/wD/OS_labs_3t/lab1/scr/Info.plist\0", 0x0,
0x0)                = -1 2

proc_info(0x2, 0x2638, 0xD)              = 64 0

csops_audittoken(0x2638, 0x10, 0x7FF7B3A08F90)           = -1 22

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid
kernel access in action #10 at DIF offset 28

dtrace: error on enabled probe ID 1741 (ID 571: syscall::sysctl:return): invalid
kernel access in action #10 at DIF offset 28
```

```
csops(0x2638, 0x0, 0x7FF7B3A093F4)              = 0 0

sysctlbyname(kern.system_version_compat, 0x1A, 0x0, 0x0, 0x7FF7B3A09424)
= 0 0

mprotect(0x10C4F9000, 0x40000, 0x3)             = 0 0

getrlimit(0x1008, 0x7FF7B3A09FE0, 0x0)          = 0 0

fstat64(0x0, 0x7FF7B3A09FC8, 0x0)               = 0 0

ioctl(0x0, 0x4004667A, 0x7FF7B3A0A014)          = 0 0


dtrace: error on enabled probe ID 1714 (ID 959: syscall::read_nocancel:return):
invalid kernel access in action #12 at DIF offset 68

test_ok.txt

0

6

2

0

dtrace: error on enabled probe ID 1714 (ID 959: syscall::read_nocancel:return):
invalid kernel access in action #12 at DIF offset 68

open("test_ok.txt\0", 0x0, 0x0)          = 3 0

pipe(0x0, 0x0, 0x0)              = 4 0

fork()           = 9789 0

close(0x5)              = 0 0

wait4(0xFFFFFFFFFFFFFFFF, 0x0, 0x0)             = 9789 0

dtrace: error on enabled probe ID 1713 (ID 173: syscall::read:return): invalid
kernel access in action #12 at DIF offset 68

fstat64(0x1, 0x7FF7B3A0A388, 0x0)               = 0 0

ioctl(0x1, 0x4004667A, 0x7FF7B3A0A3D4)          = 0 0
```

```
dtrace: error on enabled probe ID 1712 (ID 961: syscall::write_nocancel:return): invalid kernel access in action #12 at DIF offset 68

dtrace: error on enabled probe ID 1713 (ID 173: syscall::read:return): invalid kernel access in action #12 at DIF offset 68

dtrace: error on enabled probe ID 1712 (ID 961: syscall::write_nocancel:return): invalid kernel access in action #12 at DIF offset 68

dtrace: error on enabled probe ID 1713 (ID 173: syscall::read:return): invalid kernel access in action #12 at DIF offset 68

dtrace: error on enabled probe ID 1712 (ID 961: syscall::write_nocancel:return): invalid kernel access in action #12 at DIF offset 68

dtrace: error on enabled probe ID 1713 (ID 173: syscall::read:return): invalid kernel access in action #12 at DIF offset 68

dtrace: error on enabled probe ID 1712 (ID 961: syscall::write_nocancel:return): invalid kernel access in action #12 at DIF offset 68

dtrace: error on enabled probe ID 1713 (ID 173: syscall::read:return): invalid kernel access in action #12 at DIF offset 68

lseek(0x0, 0xFFFFFFFFFFFFFFFF, 0x1)              = 23771292 0
```

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.