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: М8О-211Б-23

Студент: Фоменко А.С. Преподаватель: Бахарев В.Д. Оценка:

Дата: 20.11.24

Москва, 2024

# Постановка задачи

## Вариант 8.

Цель работы

Приобретение практических навыков в:

* Управление процессами в ОС
* Обеспечение обмена данных между процессами посредством каналов

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы

программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в

результате работы.

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

Использованные системные вызовы:

* pid\_t fork(void)

Создает новый процесс, который является копией текущего процесса. Возвращает идентификатор процесса (PID): В родительском процессе — PID дочернего процесса.В дочернем процессе — 0. При ошибке возвращает -1. Используется для разделения программы на родительский и дочерний процесс.

* int pipe(int \*fd)

Создает канал (pipe) для межпроцессной связи, который представляет собой буфер в памяти. fd[0]: Конец для чтения. fd[1]: Конец для записи. Возвращает 0 при

успешном выполнении, -1 при ошибке.

* int dup2(int oldfd, int newfd)

Дублирует файловый дескриптор oldfd в указанный дескриптор newfd. Используется для перенаправления стандартного ввода или вывода. В данной работе используется для перенаправления конца канала pipe в стандартный ввод дочернего процесса.

* int execv(const char \*path, char \*const argv[])

запускает исполняемый файл в контексте уже существующего процесса, заменяя предыдущий исполняемый файл.

* int wait(int \*wstatus)

Ожидает завершения дочернего процесса. Возвращает PID завершившегося дочернего процесса.

* size\_t read(int fd, void \*buf, size\_t count) читает до count байт из файла.
* FILE \* open(const char \*pathname, const char \*mode)

Открывает файл в указанном режиме (r — только чтение). Используется для чтения входного файла.

* void exit(int status)

Завершение выполнения процесса с возвратом кода status.

* int close(int fd)

закрытие файла, связанного с файловым дескриптором fd.

В рамках выполнения данной лабораторной работы была написана программа для работы с процессами, использующая межпроцессорное взаимодействие через каналы.

# Код программы

# Протокол работы программы

**parent.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/wait.h>

#define MAX\_BUFFER 1024

int main() {

    int pipe1[2];

    if (pipe(pipe1) == -1) {

        perror("pipe failed");

        return 1;

    }

    pid\_t pid = fork();

    if (pid < 0) {

        perror("fork failed");

        return 1;

    }

    if (pid != 0) {

        close(pipe1[1]);

        char buffer[MAX\_BUFFER];

        while (read(pipe1[0], buffer, sizeof(buffer)) > 0) {

            printf("Результат из дочернего процесса: %s", buffer);

        }

        close(pipe1[0]);

        wait(NULL);

    } else {

        close(pipe1[0]);

        char filename[256];

        printf("Введите имя файла: ");

        fgets(filename, sizeof(filename), stdin);

        filename[strcspn(filename, "\n")] = 0;

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

        dup2(pipe1[1], STDOUT\_FILENO);

        close(pipe1[1]);

        execv(args[0], args);

        perror("execv failed");

        exit(EXIT\_FAILURE);

    }

    return 0;

}

# child.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <errno.h>

#include <limits.h>

#define MAX\_BUFFER 1024

int check\_overflow(const char \*str\_number) {

    char \*endptr;

    errno = 0;

    long result = strtol(str\_number, &endptr, 10);

    if ((result == LONG\_MAX || result == LONG\_MIN) && errno == ERANGE)

        return 0;

    else if (\*endptr != '\0' || result > INT\_MAX || result < INT\_MIN)

        return 0;

    return 1;

}

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

    if (argc < 2) {

        fprintf(stderr, "Не указано имя файла\n");

        return 1;

    }

    const char \*filename = argv[1];

    FILE \*file = fopen(filename, "r");

    if (file == NULL) {

        perror("Не удалось открыть файл");

        return 1;

    }

    char line[MAX\_BUFFER];

    while (fgets(line, sizeof(line), file)) {

        int num1, num2;

        char \*token = strtok(line, " ");

        if (token != NULL) {

            int check\_of\_overflow = 0;

            if (check\_overflow(token)) check\_of\_overflow += 1;

            num1 = atoi(token);

            while ((token = strtok(NULL, " ")) != NULL) {

                if (check\_overflow(token)) check\_of\_overflow += 1;

                num2 = atoi(token);

                if (num2 == 0) {

                    fprintf(stderr, "Ошибка: деление на 0\n");

                    fclose(file);

                    exit(EXIT\_FAILURE);

                }

                int result = num1 / num2;

                if (check\_of\_overflow == 2) printf("%d / %d = %d\n", num1, num2, result);

                else printf("Ошибка: переполнение\n");

            }

        }

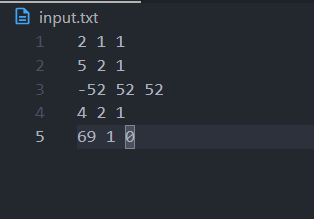
    }

    fclose(file);

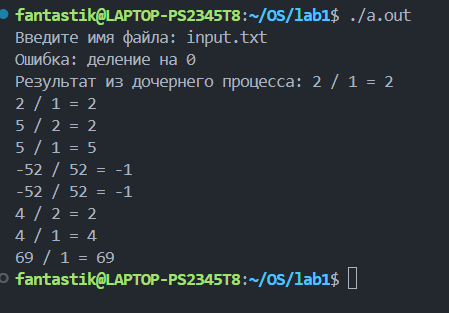
    return 0;

}

**Тест 1:**



**Тестирование:**



## Strace:

fantastik@LAPTOP-PS2345T8:~/OS/lab1$ strace ./a.out

execve("./a.out", ["./a.out"], 0x7fffae553a50 /\* 35 vars \*/) = 0

brk(NULL) = 0x55b7210ae000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f87062b9000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=19791, ...}) = 0

mmap(NULL, 19791, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f87062b4000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f87060a2000

mmap(0x7f87060ca000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f87060ca000

mmap(0x7f8706252000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7f8706252000

mmap(0x7f87062a1000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7f87062a1000

mmap(0x7f87062a7000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f87062a7000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f870609f000

arch\_prctl(ARCH\_SET\_FS, 0x7f870609f740) = 0

set\_tid\_address(0x7f870609fa10) = 222516

set\_robust\_list(0x7f870609fa20, 24) = 0

rseq(0x7f87060a0060, 0x20, 0, 0x53053053) = 0

mprotect(0x7f87062a1000, 16384, PROT\_READ) = 0

mprotect(0x55b71ff23000, 4096, PROT\_READ) = 0

mprotect(0x7f87062f1000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7f87062b4000, 19791) = 0

pipe2([3, 4], 0) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f870609fa10) = 222517

Введите имя файла: close(4) = 0

read(3, input.txt

Ошибка: деление на 0

"2 / 1 = 2\n2 / 1 = 2\n5 / 2 = 2\n5 "..., 1024) = 100

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=222517, si\_uid=1000, si\_status=1, si\_utime=0, si\_stime=0} ---

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x6), ...}) = 0

getrandom("\x5a\x00\xdf\xfc\x42\x72\xbf\x4c", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55b7210ae000

brk(0x55b7210cf000) = 0x55b7210cf000

write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202 \320\270\320\267 \320\264\320\276\321\207\320\265"..., 71Результат из дочернего процесса: 2 / 1 = 2

) = 71

write(1, "2 / 1 = 2\n", 102 / 1 = 2

) = 10

write(1, "5 / 2 = 2\n", 105 / 2 = 2

) = 10

write(1, "5 / 1 = 5\n", 105 / 1 = 5

) = 10

write(1, "-52 / 52 = -1\n", 14-52 / 52 = -1

) = 14

write(1, "-52 / 52 = -1\n4 / 2 = 2\n4 / 1 = "..., 46-52 / 52 = -1

4 / 2 = 2

4 / 1 = 4

69 / 1 = 69

) = 46

read(3, "", 1024) = 0

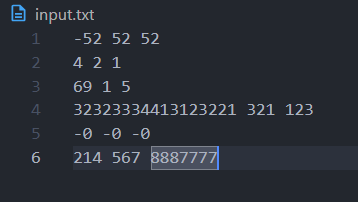
close(3) = 0

wait4(-1, NULL, 0, NULL) = 222517

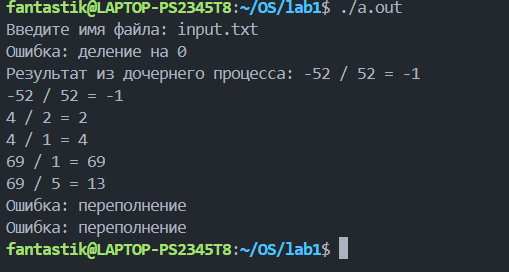
exit\_group(0) = ?

+++ exited with 0 +++

**Тест 2:**



**Тестирование:**



**Strace:**

fantastik@LAPTOP-PS2345T8:~/OS/lab1$ strace ./a.out

execve("./a.out", ["./a.out"], 0x7ffdff8e5d90 /\* 35 vars \*/) = 0

brk(NULL) = 0x55697f294000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f49f0b18000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=19791, ...}) = 0

mmap(NULL, 19791, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f49f0b13000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f49f0901000

mmap(0x7f49f0929000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f49f0929000

mmap(0x7f49f0ab1000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7f49f0ab1000

mmap(0x7f49f0b00000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7f49f0b00000

mmap(0x7f49f0b06000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f49f0b06000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f49f08fe000

arch\_prctl(ARCH\_SET\_FS, 0x7f49f08fe740) = 0

set\_tid\_address(0x7f49f08fea10) = 237667

set\_robust\_list(0x7f49f08fea20, 24) = 0

rseq(0x7f49f08ff060, 0x20, 0, 0x53053053) = 0

mprotect(0x7f49f0b00000, 16384, PROT\_READ) = 0

mprotect(0x55697e260000, 4096, PROT\_READ) = 0

mprotect(0x7f49f0b50000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7f49f0b13000, 19791) = 0

pipe2([3, 4], 0) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f49f08fea10) = 237668

Введите имя файла: close(4) = 0

read(3, input.txt

Ошибка: деление на 0

"-52 / 52 = -1\n-52 / 52 = -1\n4 / "..., 1024) = 150

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=237668, si\_uid=1000, si\_status=1, si\_utime=1 /\* 0.01 s \*/, si\_stime=0} ---

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x6), ...}) = 0

getrandom("\xf8\x5c\x9b\x13\x2e\x0a\x79\xf0", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55697f294000

brk(0x55697f2b5000) = 0x55697f2b5000

write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202 \320\270\320\267 \320\264\320\276\321\207\320\265"..., 75Результат из дочернего процесса: -52 / 52 = -1

) = 75

write(1, "-52 / 52 = -1\n", 14-52 / 52 = -1

) = 14

write(1, "4 / 2 = 2\n", 104 / 2 = 2

) = 10

write(1, "4 / 1 = 4\n", 104 / 1 = 4

) = 10

write(1, "69 / 1 = 69\n", 1269 / 1 = 69

) = 12

write(1, "69 / 5 = 13\n\320\236\321\210\320\270\320\261\320\272\320\260: \320\277\320\265\321\200"..., 9069 / 5 = 13

Ошибка: переполнение

Ошибка: переполнение

) = 90

read(3, "", 1024) = 0

close(3) = 0

wait4(-1, NULL, 0, NULL) = 237668

exit\_group(0) = ?

+++ exited with 0 +++

Вывод

В процессе выполнения лабораторной работы я научился управлять процессами в

операционной системе и реализовывать обмен данными между ними с использованием каналов. В рамках работы была создана и отлажена программа на языке Си, обеспечивающая эффективное взаимодействие процессов путем передачи данных через pipe.