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-215Б-23

Студент: Кобзев К. А.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 10.07.25

Москва, 2025

Постановка задачи

Вариант 2.

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочерн его процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс при необходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, неперенаправляя стандартный поток вывода.

2 вариант) Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип float. Количество чисел может быть произвольным.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); — создаёт дочерний процесс.
- int pipe(int *fd); — создаёт неименованный канал для взаимодействия процессов.
- int dup2(int fd, int fd2); — дублирует файловый дескриптор, перенаправляя ввод/вывод.
- int execlp(const char* file, const char* arg, ...); — заменяет текущий образ процесса новым, ищет исполняемый файл в путях PATH.
- int close(int fd); — закрывает файловый дескриптор.
- pid_t wait(int* stat_loc); — ожидает завершения любого дочернего процесса

Алгоритм решения

Родитель (parent.c)

1. Инициализирует один канал (pipe).
2. Получает от пользователя имя файла для вывода.
3. Создаёт дочерний процесс с помощью fork().
4. В родительском процессе закрывает дескриптор канала, предназначенный для чтения.
5. В цикле читает строки из стандартного ввода и пишет их в канал. Ввод прекращается по пустой строке.
6. После завершения ввода закрывает дескриптор канала для записи, чтобы дочерний процесс получил EOF.
7. Ожидает завершения дочернего процесса с помощью wait().

Ребёнок (child.c)

1. Код, выполняемый сразу после fork():
2. Закрывает дескриптор канала, предназначенный для записи.
3. Дублирует дескриптор канала для чтения на стандартный ввод (STDIN_FILENO).
4. Закрывает исходный дескриптор канала для чтения (так как он уже скопирован).
5. Запускает новую программу (./child), передавая ей имя файла.

Код в отдельной программе child.c:

1. Открывает файл для записи, имя которого было получено как аргумент.
2. В цикле считывает строки из стандартного ввода (который теперь является каналом).
3. Для каждой строки вычисляет сумму всех чисел.
4. Записывает результат в файл.
5. После получения EOF из канала, закрывает файл и завершает работу.

Код программы

parent.c

```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/wait.h>


int main()

{

    char filename[256], buffer[1024];

    int pipefd[2];


    printf("Введите имя файла для вывода: ");

    if (!fgets(filename, sizeof(filename), stdin))

    {

        perror("fgets");

        exit(1);

    }


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

    if (pipe(pipefd) == -1)

    {

        perror("pipe");

        exit(1);
```

```
}

pid_t pid = fork();
if (pid < 0)
{
    perror("fork");
    exit(1);
}

if (pid == 0)
{ // Дочерний процесс
    close(pipefd[1]);
    dup2(pipefd[0], STDIN_FILENO);
    close(pipefd[0]);
    execlp("./child", "child", filename, NULL);
    perror("execlp");
    exit(1);
}
else
{ // Родительский процесс
    close(pipefd[0]);
    printf("Введите строки с числами (float). Пустая строка — завершение.\n");
    while (fgets(buffer, sizeof(buffer), stdin))
    {
        if (buffer[0] == '\n')
            break;
        if (write(pipefd[1], buffer, strlen(buffer)) == -1)
        {
            perror("write");
            break;
```

```
            }
        }
        close(pipefd[1]);
        wait(NULL);
    }

    return 0;
}
```

child.c

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        fprintf(stderr, "Использование: %s <имя_файла_для_вывода>\n", argv[0]);
        return 1;
    }
    FILE *outFile = fopen(argv[1], "w");
    if (!outFile)
    {
        perror("Ошибка открытия файла для записи");
        return 1;
    }
    char line[1024];
    while (fgets(line, sizeof(line), stdin))
    {
        double sum = 0.0f;
        char *ptr = line;
```

```
        char *endptr;
        while (*ptr)
        {
            // strtof преобразует строку в float и передвигает endptr.
            float num = strtof(ptr, &endptr);
            // Если strtof ничего не считал, указатели останутся равны.
            if (ptr == endptr)
            {
                if (*ptr == '\0' || *ptr == '\n')
                {
                    break;
                }
                ptr++;
            }
            else
            {
                sum += num;
                ptr = endptr;
            }
        }
        fprintf(outFile, "%f\n", sum);
    }
    fclose(outFile);
    return 0;
}
```

Протокол работы программы

Тестирование:

→ src ./parent

Введите имя файла для вывода: result.txt

Введите строки с числами (float). Пустая строка — завершение.

1.2 3.0

0.0 1.2 3.0

→ src cat result.txt

4.200000

4.200000

→ src ./parent

Введите имя файла для вывода: result.txt

Введите строки с числами (float). Пустая строка — завершение.

1 2.3 100

0 2.323 1212

→ src cat result.txt

103.300000

1214.323000

strace

root@2273a6f3c6af:/workspace/lab1/src# strace -f ./parent

execve("./parent", ["./parent"], 0xffffd4e144b8 /* 12 vars */) = 0

brk(NULL) = 0x1eb96000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffffb003e000

faccessat(AT_FDCWD, "/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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=25959, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 25959, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffffb0037000

close(3) = 0

openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0000y\2\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1651408, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 1826912, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffffafe46000

mmap(0xffffafe50000, 1761376, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffffafe50000

munmap(0xffffafe46000, 40960) = 0

munmap(0xffffaffff000, 20576) = 0

mprotect(0xffffaffd7000, 86016, PROT_NONE) = 0

mmap(0xffffaffec000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18c000) = 0xffffaffec000

mmap(0xffffafff2000, 49248, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffffafff2000

close(3) = 0

set_tid_address(0xffffb003f050) = 38

set_robust_list(0xffffb003f060, 24) = 0

rseq(0xffffb003f6a0, 0x20, 0, 0xd428bc00) = 0

mprotect(0xffffaffec000, 16384, PROT_READ) = 0

mprotect(0x41f000, 4096, PROT_READ) = 0

mprotect(0xffffb0043000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0xffffb0037000, 25959) = 0

newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0

getrandom("\x98\xbf\xce\x16\x9b\x3b\x43\xe0", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x1eb96000

brk(0x1ebb7000) = 0x1ebb7000

newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\270\320\274\321\217 \321\204\320\260\320\271\320\273\320\260"..., 54Введите имя файла для вывода:) = 54

read(0, result.txt

"result.txt\n", 1024) = 11

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

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 39 attached

, child_tidptr=0xffffb003f050) = 39

[pid 39] set_robust_list(0xffffb003f060, 24 <unfinished ...>

[pid 38] close(3 <unfinished ...>

[pid 39] <... set_robust_list resumed>) = 0

[pid 38] <... close resumed>) = 0

[pid 39] close(4 <unfinished ...>

[pid 38] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\201\321\202\321\200\320\276\320\272\320\270 \321\201 \321"..., 107 <unfinished ...>

[pid 39] <... close resumed>) = 0

Введите строки с числами (float). Пустая строка — завершение.

[pid 38] <... write resumed>) = 107

[pid 39] dup3(3, 0, 0 <unfinished ...>

[pid 38] read(0, <unfinished ...>

[pid 39] <... dup3 resumed>) = 0

[pid 39] close(3) = 0

[pid 39] execve("./child", ["child", "result.txt"], 0xffffdacef928 /* 12 vars */) = 0

[pid 39] brk(NULL) = 0x74f3000

[pid 39] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff8c093000

[pid 39] faccessat(AT_FDCWD, "/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

[pid 39] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

[pid 39] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=25959, ...}, AT_EMPTY_PATH) = 0

[pid 39] mmap(NULL, 25959, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffff8c08c000

[pid 39] close(3) = 0

[pid 39] openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

[pid 39] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0000y\2\0\0\0\0"..., 832) = 832

[pid 39] newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1651408, ...}, AT_EMPTY_PATH) = 0

[pid 39] mmap(NULL, 1826912, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffff8be9b000

[pid 39] mmap(0xffff8bea0000, 1761376, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff8bea0000

[pid 39] munmap(0xffff8be9b000, 20480) = 0

[pid 39] munmap(0xffff8c04f000, 41056) = 0

[pid 39] mprotect(0xffff8c027000, 86016, PROT_NONE) = 0

[pid 39] mmap(0xffff8c03c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x18c000) = 0xffff8c03c000

[pid 39] mmap(0xffff8c042000, 49248, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffff8c042000

[pid 39] close(3) = 0

[pid 39] set_tid_address(0xffff8c094050) = 39

[pid 39] set_robust_list(0xffff8c094060, 24) = 0

[pid 39] rseq(0xffff8c0946a0, 0x20, 0, 0xd428bc00) = 0

[pid 39] mprotect(0xffff8c03c000, 16384, PROT_READ) = 0

[pid 39] mprotect(0x41f000, 4096, PROT_READ) = 0

[pid 39] mprotect(0xffff8c098000, 8192, PROT_READ) = 0

[pid 39] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

[pid 39] munmap(0xffff8c08c000, 25959) = 0

[pid 39] getrandom("\x23\x09\xba\xfa\xf8\x1d\x16\xd1", 8, GRND_NONBLOCK) = 8

[pid 39] brk(NULL) = 0x74f3000

[pid 39] brk(0x7514000) = 0x7514000

[pid 39] openat(AT_FDCWD, "result.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3

[pid 39] newfstatat(0, "", {st_mode=S_IFIFO|0600, st_size=0, ...}, AT_EMPTY_PATH) = 0

[pid 39] read(0, 1.0 2.0

<unfinished ...>

[pid 38] <... read resumed>"1.0 2.0\n", 1024) = 8

[pid 38] write(4, "1.0 2.0\n", 8) = 8

[pid 39] <... read resumed>"1.0 2.0\n", 4096) = 8

[pid 38] read(0, <unfinished ...>

[pid 39] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=0, ...}, AT_EMPTY_PATH) = 0

[pid 39] read(0, 2.0

<unfinished ...>

[pid 38] <... read resumed>"2.0\n", 1024) = 4

[pid 38] write(4, "2.0\n", 4) = 4

[pid 39] <... read resumed>"2.0\n", 4096) = 4

[pid 38] read(0, <unfinished ...>

[pid 39] read(0,

 <unfinished ...>

[pid 38] <... read resumed>"\n", 1024) = 1

[pid 38] close(4)   = 0

[pid 39] <... read resumed>"", 4096) = 0

[pid 38] wait4(-1, <unfinished ...>

[pid 39] write(3, "3.000000\n2.000000\n", 18) = 18

[pid 39] close(3)   = 0

[pid 39] exit_group(0)   = ?

[pid 39] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL)   = 39

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=39, si_uid=0, si_status=0, si_utime=0, si_stime=0} ---

exit_group(0)   = ?

+++ exited with 0 +++

Вывод

В результате выполнения лабораторной работы была разработана и отлажена программа, демонстрирующая взаимодействие процессов с помощью неименованных каналов. Были применены системные вызовы fork() для создания дочернего процесса и pipe() для организации однонаправленной передачи данных. Успешное использование вызова dup2() для перенаправления стандартного потока ввода позволило реализовать заданную схему и сохранить результат вычислений в файл.