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-209БВ-24

Студент: Касаева Я.М.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 08.10.25

Москва, 2025

Постановка задачи

Вариант 5.

Родительский процесс создает дочерний процесс. Пользователь вводит имя файла, которое передается дочернему процессу. Родительский процесс передает команды пользователя через pipe1, который связан со стандартным входным потоком дочернего процесса. Дочерний процесс при необходимости передает данные в родительский процесс через pipe2. Пользователь вводит команды вида: «число<endline>». Дочерний процесс производит проверку на простоту. Если число составное, то это число записывается в файл. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются.

Общий метод и алгоритм решения

Использованные системные вызовы:

- `pid_t fork(void)` - создает дочерний процесс

- `int pipe(int fd[2])` - создает канал для межпроцессного взаимодействия

- `int dup2(int oldfd, int newfd)` - перенаправляет стандартные потоки ввода/вывода

- `execl(const char *path, const char *arg, ...)` - заменяет текущий процесс новым

- `waitpid(pid_t pid, int *status, int options)` - ожидает завершения дочернего процесса

- `read(int fd, void *buf, size_t count)` - чтение из файлового дескриптора

- `write(int fd, const void *buf, size_t count)` - запись в файловый дескрипторАлгоритм работы программы:

Алгоритм работы программы:

1. Родительский процесс:

   - Создает два канала (pipe1 и pipe2)

   - Получает от пользователя имя файла

   - Создает дочерний процесс с помощью fork()

   - В цикле принимает числа от пользователя и передает их через pipe1 дочернему процессу

   - Читает ответы дочернего процесса из pipe2

   - Завершает работу при получении сигнала "EXIT"

2. Дочерний процесс:

   - Перенаправляет стандартные потоки с помощью dup2()

   - Запускает программу обработки чисел с помощью execl()

   - Читает числа из pipe1 (перенаправленного stdin)

- Проверяет числа на простоту

- Записывает составные числа в файл

- Отправляет "OK" через pipe2 для составных чисел

- Отправляет "EXIT" и завершает работу при простом или отрицательном числе

Код программы

parent.c

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <ctype.h>
#include <limits.h>

// Функция проверки что строка - целое число в границах int
int is_integer(const char *str) {
    if (str == NULL || *str == '\0') return 0;

    int i = 0;

    if (str[0] == '-') {
        if (str[1] == '\0') return 0;
        i = 1;
    }

    for (; str[i] != '\0'; i++) {
        if (!isdigit(str[i])) return 0;
    }

    long num = atol(str);
    if (num < -2147483648 || num > 2147483647) {
        return 0;
    }

    return 1;
}

int main() {
    int pipe1[2], pipe2[2];
    char filename[100];
    char number[100];
    pid_t pid;

    pipe(pipe1);
    pipe(pipe2);

    printf("Введите имя файла: ");
    scanf("%s", filename);

    pid = fork();
    if (pid == 0) {
        // Дочерний процесс
        close(pipe1[1]);
        close(pipe2[0]);
        dup2(pipe1[0], 0);
        close(pipe1[0]);
        dup2(pipe2[1], 1);
```

```
        close(pipe2[1]);
        execl("./child", "child", filename, NULL);
        exit(1);
    } else {
        // Родительский процесс
        close(pipe1[0]);
        close(pipe2[1]);

        while (1) {
            printf("Введите число: ");
            scanf("%s", number);

            if (!is_integer(number)) {
                printf("Введенные данные не типа int, попробуйте еще раз:\n");
                continue;
            }

            // Проверяем, жив ли дочерний процесс
            if (waitpid(pid, NULL, WNOHANG) == pid) {
                printf("Дочерний процесс завершен\n");
                break;
            }

            write(pipe1[1], number, strlen(number));
            write(pipe1[1], "\n", 1);

            char response[10];
            if (read(pipe2[0], response, 10) > 0) {
                if (strstr(response, "EXIT")) {
                    printf("Получен сигнал завершения\n");
                    break;
                }
            }
        }

        close(pipe1[1]);
        close(pipe2[0]);
        wait(NULL);
        printf("Родительский процесс завершен\n");
    }
    return 0;
}
```

child.c

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

int is_prime(int n) {
    if (n < 1) return 0;
    if (n == 1) return 1;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return 0;
    }
    return 1;
}

int main(int argc, char *argv[]) {
    FILE *file = fopen(argv[1], "w");
    char line[100];

    while (fgets(line, 100, stdin)) {
        int num = atoi(line);

        if (num < 0 || is_prime(num)) {
```

```
            fprintf(file, "EXIT: %d\n", num);
            fclose(file);
            printf("EXIT\n");
            fflush(stdout);
            return 0;
        } else {
            fprintf(file, "%d\n", num);
            fflush(file);
            printf("OK\n");
            fflush(stdout);
        }
    }
    fclose(file);
    return 0;
}
```

Протокол работы программы

Тестирование:

```
(base) yanakasaeva@MacBook-Air--YanaK src % gcc -o parent parent.c
gcc -o child child.c
./parent
Введите имя файла: res.txt
Введите число: 35
Введите число: 9
Введите число: 0
Введите число: -20
Получен сигнал завершения
Родительский процесс завершен
(base) yanakasaeva@MacBook-Air--YanaK src % cat res.txt
35
9
0
EXIT: -20
```

Strace:

1102 execve("./parent", ["./parent"], 0xfffffd166e18 /* 8 vars */) = 0**

1102 brk(NULL) = 0xaaaab3dfb000

1102 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffffaeb92000

1102 faccessat(AT_FDCWD, "/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

1102 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

1102 fstat(3, {st_mode=S_IFREG|0644, st_size=8467, ...}) = 0

1102 mmap(NULL, 8467, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffffaeb8f000

1102 close(3) = 0

1102 openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

1102 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\360\206\2\0\0\0\0"..., 832) = 832

1102 fstat(3, {st_mode=S_IFREG|0755, st_size=1722920, ...}) = 0

1102 mmap(NULL, 1892240, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_DENYWRITE, -1, 0) = 0xffffae98b000

1102 mmap(0xffffae990000, 1826704, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffffae990000

1102 munmap(0xffffae98b000, 20480) = 0

1102 munmap(0xffffaeb4e000, 44944) = 0

1102 mprotect(0xffffaeb29000, 81920, PROT_NONE) = 0

1102 mmap(0xffffaeb3d000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0xffffaeb3d000

1102 mmap(0xffffaeb42000, 49040, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffffaeb42000

1102 close(3) = 0

1102 set_tid_address(0xffffaeb92fb0) = 1102

1102 set_robust_list(0xffffaeb92fc0, 24) = 0

1102 rseq(0xffffaeb93600, 0x20, 0, 0xd428bc00) = 0

1102 mprotect(0xffffaeb3d000, 12288, PROT_READ) = 0

1102 mprotect(0xaaaab2e3f000, 4096, PROT_READ) = 0

1102 mprotect(0xffffaeb97000, 8192, PROT_READ) = 0

1102 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

1102 munmap(0xffffaeb8f000, 8467) = 0

1102 pipe2([3, 4], 0) = 0

1102 pipe2([5, 6], 0) = 0

1102 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

1102 getrandom("\x0f\x0a\x07\x01\x7d\xce\x26\x08", 8, GRND_NONBLOCK) = 8

1102 brk(NULL) = 0xaaaab3dfb000

1102 brk(0xaaaab3e1c000) = 0xaaaab3e1c000

1102 fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

1102 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"..., 34) = 34

1102 read(0, "res.txt\n", 1024) = 8

1102 clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0xffffaeb92fb0) = 1103

1102 close(3) = 0

1103 set_robust_list(0xffffaeb92fc0, 24 <unfinished ...>

1102 close(6 <unfinished ...>

1103 <... set_robust_list resumed>) = 0

1102 <... close resumed>) = 0

1103 close(4 <unfinished ...>

1102 write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\207\320\270\321\201\320\273\320\276: ", 27 <unfinished ...>

1103 <... close resumed>) = 0

1102 <... write resumed>) = 27

1103 close(5 <unfinished ...>

1102 read(0, <unfinished ...>

1103 <... close resumed>) = 0

1103 dup3(3, 0, 0) = 0

1103 close(3) = 0

1103 dup3(6, 1, 0) = 1

1103 close(6) = 0

1103 execve("./child", ["child", "res.txt"], 0xffffc6fcd168 /* 8 vars */) = 0

1103 brk(NULL) = 0xaaaae35e8000

1103 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffffba760000

1103 faccessat(AT_FDCWD, "/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

1103 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

1103 fstat(3, {st_mode=S_IFREG|0644, st_size=8467, ...}) = 0

1103 mmap(NULL, 8467, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffffba75d000

1103 close(3) = 0

1103 openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

```
1103  read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\360\206\2\0\0\0\0"..., 832) =
832

1103  fstat(3, {st_mode=S_IFREG|0755, st_size=1722920, ...}) = 0

1103  mmap(NULL, 1892240, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_DENYWRITE, -1, 0) = 0xffffba559000

1103  mmap(0xffffba560000, 1826704, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffffba560000

1103  munmap(0xffffba559000, 28672)     = 0

1103  munmap(0xffffba71e000, 36752)     = 0

1103  mprotect(0xffffba6f9000, 81920, PROT_NONE) = 0

1103  mmap(0xffffba70d000, 20480, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0xffffba70d000

1103  mmap(0xffffba712000, 49040, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffffba712000

1103  close(3)                          = 0

1103  set_tid_address(0xffffba760fb0)   = 1103

1103  set_robust_list(0xffffba760fc0, 24) = 0

1103  rseq(0xffffba761600, 0x20, 0, 0xd428bc00) = 0

1103  mprotect(0xffffba70d000, 12288, PROT_READ) = 0

1103  mprotect(0xaaaac0dff000, 4096, PROT_READ) = 0

1103  mprotect(0xffffba765000, 8192, PROT_READ) = 0

1103  prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

1103  munmap(0xffffba75d000, 8467)      = 0

1103  getrandom("\x31\x1a\x56\x04\x8a\xa3\x13\x6d", 8, GRND_NONBLOCK) = 8

1103  brk(NULL)                         = 0xaaaae35e8000

1103  brk(0xaaaae3609000)               = 0xaaaae3609000

1103  openat(AT_FDCWD, "res.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3

1103  fstat(0, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0

1103  read(0,  <unfinished ...>

1102  <... read resumed>"35\n", 1024)   = 3

1102  wait4(1103, NULL, WNOHANG, NULL)  = 0

**1102  write(4, "35", 2)                = 2**

1103  <... read resumed>"35", 4096)     = 2

1102  write(4, "\n", 1 <unfinished ...>
```

1103 read(0, <unfinished ...>

1102 <... write resumed>) = 1

1103 <... read resumed>"\n", 4096) = 1

1102 read(5, <unfinished ...>

1103 fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0

1103 write(3, "35\n", 3) = 3

1103 fstat(1, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0

1103 write(1, "OK\n", 3 <unfinished ...>

1102 <... read resumed>"OK\n", 10) = 3

1103 <... write resumed>) = 3

1102 write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\207\320\270\321\201\320\273\320\276: ", 27 <unfinished ...>

1103 read(0, <unfinished ...>

1102 <... write resumed>) = 27

1102 read(0, "9\n", 1024) = 2

1102 wait4(1103, NULL, WNOHANG, NULL) = 0

1102 write(4, "9", 1) = 1

1103 <... read resumed>"9", 4096) = 1

1102 write(4, "\n", 1 <unfinished ...>

1103 read(0, <unfinished ...>

1102 <... write resumed>) = 1

1103 <... read resumed>"\n", 4096) = 1

1102 read(5, <unfinished ...>

1103 write(3, "9\n", 2) = 2

1103 write(1, "OK\n", 3 <unfinished ...>

1102 <... read resumed>"OK\n", 10) = 3

1103 <... write resumed>) = 3

1102 write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\207\320\270\321\201\320\273\320\276: ", 27 <unfinished ...>

1103 read(0, <unfinished ...>

1102 <... write resumed>) = 27

1102 read(0, "0\n", 1024) = 2

1102 wait4(1103, NULL, WNOHANG, NULL) = 0

1102 write(4, "0", 1)   = 1

1103 <... read resumed>"0", 4096)   = 1

1102 write(4, "\n", 1 <unfinished ...>

1103 read(0, <unfinished ...>

1102 <... write resumed>)   = 1

1103 <... read resumed>"\n", 4096)   = 1

1102 read(5, <unfinished ...>

1103 write(3, "0\n", 2)   = 2

1103 write(1, "OK\n", 3 <unfinished ...>

1102 <... read resumed>"OK\n", 10)   = 3

1103 <... write resumed>)   = 3

1102 write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\207\320\270\321\201\320\273\320\276: ", 27 <unfinished ...>

1103 read(0, <unfinished ...>

1102 <... write resumed>)   = 27

1102 read(0, "-20\n", 1024)   = 4

1102 wait4(1103, NULL, WNOHANG, NULL) = 0

1102 write(4, "-20", 3)   = 3

1103 <... read resumed>"-20", 4096)   = 3

1102 write(4, "\n", 1 <unfinished ...>

1103 read(0, <unfinished ...>

1102 <... write resumed>)   = 1

1103 <... read resumed>"\n", 4096)   = 1

1102 read(5, <unfinished ...>

1103 write(3, "EXIT: -20\n", 10)   = 10

1103 close(3)   = 0

1103 write(1, "EXIT\n", 5 <unfinished ...>

1102 <... read resumed>"EXIT\n", 10)   = 5

1103 <... write resumed>)   = 5

1102 write(1, "\320\237\320\276\320\273\321\203\321\207\320\265\320\275
\321\201\320\270\320\263\320\275\320\260\320\273 \320\267\320\260"..., 49 <unfinished ...>

1103 exit_group(0 <unfinished ...>

1102 <... write resumed>)   = 49

1103 <... exit_group resumed>) = ?

1102 close(4) = 0

1102 close(5 <unfinished ...>

1103 +++ exited with 0 +++

1102 <... close resumed>) = 0

1102 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=1103, si_uid=0, si_status=0, si_utime=0, si_stime=0} ---

1102 wait4(-1, NULL, 0, NULL) = 1103

1102 write(1, "\320\240\320\276\320\264\320\270\321\202\320\265\320\273\321\214\321\201\320\272\320\270\320\271 \320\277\321\200\320\276\321"..., 57) = 57

1102 lseek(0, -1, SEEK_CUR) = -1 ESPIPE (Illegal seek)

1102 exit_group(0) = ?

1102 +++ exited with 0 +++

Вывод

В ходе лабораторной работы были успешно реализованы механизмы межпроцессного взаимодействия через каналы (pipes). Программа корректно обрабатывает ввод пользователя, проверяет числа на простоту и завершает работу при выполнении условий. Основные сложности возникли при организации корректного обмена данными между процессами - требовалось обеспечить синхронную работу и избежать "зависаний" процессов. Особое внимание уделено обработке граничных случаев: проверке ввода на целочисленный тип, защите от переполнения и корректному завершению обоих процессов. Работа позволила получить практические навыки управления процессами, использования системных вызовов fork(), pipe(), dup2() и организации межпроцессной коммуникации в операционных системах.