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-210Б-23

Студент: Горячев А. В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 19.11.24

Москва, 2024

Постановка задачи

Вариант 5.

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего 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, не перенаправляя стандартный поток вывода.

Пользователь вводит команды вида: «число<endline>». Далее это число передается от родительского процесса в дочерний. Дочерний процесс производит проверку на простоту. Если число составное, то в 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); – создает pipe – однонаправленный канал, используемый для передачи данных между процессами. Возвращает два файловых дескриптора: fd[0] для чтения и fd[1] для записи.
- int dup2(int oldfd, int newfd); – перенаправляет файловый дескриптор.
- ssize_t read(int fd, void *buf, size_t count); – читает count байтов из файла, на который ссылается файлового дескриптор fd, в буфер buf.
- ssize_t write(int fd, const void *buf, size_t count); – записывает count байтов из буфера buf в файл, на который ссылается файловый дескриптор fd.
- int open(const char *pathname, int flags, mode_t mode); – открывает файл с указанными флагами и правами доступа.
- int execv(const char *pathname, char *const argv[]); – заменяет текущий исполняемый образ процесса на новый.
- int close(int fd); – закрывает файловый дескриптор, освобождая связанные с ним ресурсы.

Программа реализует взаимодействие родительского и дочернего процессов через два пайпа. Сначала родительский процесс запрашивает у пользователя имя файла, после чего создаются два пайпа для обмена данными между процессами. Затем выполняется fork для создания дочернего процесса. Родительский процесс оставляет открытыми концы пайпов для записи данных в pipe1 и чтения из pipe2, а остальные закрывает. Дочерний процесс перенаправляет стандартный ввод на pipe1 и стандартный вывод на pipe2 с помощью dup2, после чего выполняет замену своего исполняемого образа с помощью execv, запуская child.out с переданным именем файла. Родительский процесс в цикле считывает числа, вводимые пользователем, передает их в дочерний процесс и получает обратно флаг завершения. Если флаг указывает на завершение (число простое или отрицательное), оба процесса закрывают дескрипторы пайпов и завершают работу. Дочерний процесс записывает составные числа в файл, а также возвращает родителю флаг, сигнализирующий о необходимости продолжения работы.

Код программы

parent.c

```
#include <stdint.h>
#include <stdbool.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>

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

static char CLIENT_PROGRAM_NAME[] = "./child.out";

int read_str(char *str, size_t size) {
    ssize_t status = read(STDIN_FILENO, str, size - 1);
    if (status > 0) {
        if (str[status - 1] == '\n') {
            str[status - 1] = '\0';
        } else {
            str[status] = '\0';
            char ch;
            while (read(STDIN_FILENO, &ch, 1) > 0 && ch != '\n');
            return 1;
        }
        return 0;
    }
    return 1;
}

int main() {
    char filename[130];
    int pipe1[2], pipe2[2];

    {
        const char msg[] = "Enter the name of file:\n";
        write(STDERR_FILENO, msg, sizeof(msg));
    }

    if (read_str(filename, sizeof(filename)) != 0) {
        const char msg[] = "Failed to read file\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {
        const char msg[] = "Failed to create pipe\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    const pid_t child = fork();

    if (child == -1) {
```

```
        const char msg[] = "Failed to spawn new process\n";
        write(STDOUT_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);

    } else if (child == 0) {
        close(pipe1[1]);
        close(pipe2[0]);
        dup2(pipe1[0], STDIN_FILENO);
        close(pipe1[0]);

        dup2(pipe2[1], STDOUT_FILENO);
        close(pipe2[1]);

        char *const args[] = {CLIENT_PROGRAM_NAME, filename, NULL};
        int status = execv(CLIENT_PROGRAM_NAME, args);

        if (status == -1) {
            const char msg[] = "Failed to exec into new exectuable image\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }

    } else {
        close(pipe2[1]);
        close(pipe1[0]);

        {
            const char msg[] = "Enter numbers. If you want to stop enter the prime number or Ctrl +
D:\n";
            write(STDERR_FILENO, msg, sizeof(msg));
        }

        while (1) {
            char buffer[256];
            ssize_t bytes_read = read(STDIN_FILENO, buffer, sizeof(buffer));

            if (bytes_read <= 0) break;

            write(pipe1[1], buffer, bytes_read);

            int res;
            ssize_t response = read(pipe2[0], &res, sizeof(res));
            if (response <= 0 || res == 1) {
                break;
            }
        }

        close(pipe1[1]);
        close(pipe2[0]);
    }

    return 0;
}
```

child.c

```
#include <stdint.h>
#include <stdbool.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

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

int main(int argc, char **argv) {
    if (argc < 2) {
        const char msg[] = "At least one argument is required\n";
        write(STDERR_FILENO, msg, sizeof(msg));
    }

    int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);

    while (1) {
        char buffer[256];
        ssize_t bytes_read = read(STDIN_FILENO, buffer, sizeof(buffer) - 1);
        if (bytes_read <= 0) break;

        buffer[bytes_read] = '\0';
        int number = atoi(buffer);
        int is_prime_ = is_prime(number);
        int is_negative = number < 0;
        int stop_flag = is_prime_ || is_negative;

        write(STDOUT_FILENO, &stop_flag, sizeof(stop_flag));

        if (stop_flag) break;
        if (number != 1 & number != 0)
            write(fd, buffer, strlen(buffer));
    }

    close(fd);
    return 0;

}
```

Протокол работы программы

Тестирование:

```
alexandr@DESKTOP-U71DG2Q:~/sem3/osi/lab1$ ./a.out
Enter the name of file:
1.txt
Enter numbers. If you want to stop enter the prime number or Ctrl + D:
4
25
91
0
1
92
7
alexandr@DESKTOP-U71DG2Q:~/sem3/osi/lab1$ cat 1.txt
4
25
91
92
```

Strace

strace ./a.out
execve("./a.out", ["./a.out"], 0x7ffe88f65f60 /* 30 vars */) = 0
brk(NULL) = 0x55c579785000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc124800c0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f263ba90000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/glibc-hwcaps/x86-64-v3", 0x7ffc1247f2e0, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/glibc-hwcaps/x86-64-v2", 0x7ffc1247f2e0, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/tls/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/tls/x86_64/x86_64", 0x7ffc1247f2e0, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/tls/x86_64", 0x7ffc1247f2e0, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/tls/x86_64", 0x7ffc1247f2e0, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/tls", 0x7ffc1247f2e0, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/x86_64/x86_64", 0x7ffc1247f2e0, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/x86_64", 0x7ffc1247f2e0, 0) = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/x86_64", 0x7ffc1247f2e0, 0) = -1 ENOENT (No such file or directory)

```
	openat(AT_FDCWD, "/usr/local/cuda-11.0/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	newfstatat(AT_FDCWD, "/usr/local/cuda-11.0/lib64", {st_mode=S_IFDIR|0755, st_size=4096, ...}, 0) = 0
	openat(AT_FDCWD, "glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	openat(AT_FDCWD, "glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	openat(AT_FDCWD, "tls/x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	openat(AT_FDCWD, "tls/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	openat(AT_FDCWD, "tls/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	openat(AT_FDCWD, "x86_64/x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	openat(AT_FDCWD, "x86_64/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
	openat(AT_FDCWD, "libc.so.6", O_RDONLY|O_CLOEXEC) = -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=33235, ...}, AT_EMPTY_PATH) = 0
	mmap(NULL, 33235, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f263ba87000
	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\0P\237\2\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"..., 784, 64) = 784
	pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0"..., 48, 848) = 48
	pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68
	newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 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"..., 784, 64) = 784
	mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f263b85e000
	mprotect(0x7f263b886000, 2023424, PROT_NONE) = 0
	mmap(0x7f263b886000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f263b886000
	mmap(0x7f263ba1b000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f263ba1b000
	mmap(0x7f263ba74000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f263ba74000
	mmap(0x7f263ba7a000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f263ba7a000
	close(3)                                = 0
	mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f263b85b000
	arch_prctl(ARCH_SET_FS, 0x7f263b85b740) = 0
	set_tid_address(0x7f263b85ba10)         = 374332
	set_robust_list(0x7f263b85ba20, 24)     = 0
	rseq(0x7f263b85c0e0, 0x20, 0, 0x53053053) = 0
	mprotect(0x7f263ba74000, 16384, PROT_READ) = 0
	mprotect(0x55c548e00000, 4096, PROT_READ) = 0
	mprotect(0x7f263baca000, 8192, PROT_READ) = 0
	prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
	munmap(0x7f263ba87000, 33235)           = 0
	write(2, "Enter the name of file:\n\0", 25Enter the name of file:
	) = 25
	read(0, 1.txt
	"1.txt\n", 129)                 = 6
	pipe2([3, 4], 0)                        = 0
	pipe2([5, 6], 0)                        = 0
	clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7f263b85ba10) = 374405
	close(6)                                = 0
	close(3)                                = 0
	write(2, "Enter numbers. If you want to st"..., 72Enter numbers. If you want to stop enter the prime number or Ctrl + D:
	) = 72
	read(0, 4
	"4\n", 256)                     = 2
	write(4, "4\n", 2)                      = 2
	read(5, "\0\0\0\0", 4)                  = 4
```

```
read(0, 25
"25\n", 256)                = 3
write(4, "25\n", 3)                = 3
read(5, "\0\0\0\0", 4)              = 4
read(0, 1
"1\n", 256)                 = 2
write(4, "1\n", 2)                 = 2
read(5, "\0\0\0\0", 4)              = 4
read(0, 0
"0\n", 256)                 = 2
write(4, "0\n", 2)                 = 2
read(5, "\0\0\0\0", 4)              = 4
read(0, 7
"7\n", 256)                 = 2
write(4, "7\n", 2)                 = 2
read(5, "\1\0\0\0", 4)              = 4
close(4)                    = 0
close(5)                    = 0
exit_group(0)                   = ?
+++ exited with 0 +++
```

Вывод

В ходе данной работы я научился создавать процессы и настраивать общение между ними. Изначально я пробовал перенаправить stdin родительского процесса напрямую на вход pipe1, чтобы избавиться от ручной пересылки данных через write. Однако этот подход не сработал, поскольку пайп в таком случае работает только на уровне файловых дескрипторов, а stdin родителя ожидает интерактивный ввод. После этого я перешел к решению через чтение данных из stdin с помощью read в родительском процессе и write этих данных в дескриптор, созданный через pipe.