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: Дехтеренко Д.С.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 01.11.24

Москва, 2024

Постановка задачи

Вариант 10.

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.

В файле записаны команды вида: «число». Дочерний процесс производит проверку этого числа на простоту. Если число составное, то дочерний процесс пишет это число в стандартный поток вывода. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются. Количество чисел может быть произвольным.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork() - создание дочернего процесса
- execl(const char *path, const char *arg, ...) – замена памяти процесса
- pid_t wait(int *status) - ожидание завершения дочернего процесса
- int pipe(int pipefd[2]) - создание неименованного канала для передачи данных между
- процессами
- int dup2(int oldfd, int newfd) - переназначение файлового дескриптора
- int open(const char *pathname, int flags, mode_t mode) - открытие\создание файла
- int close(int fd) - закрыть файл
- size_t read (int fd, void* buf, size_t cnt) – чтение из файла
- size_t write (int fd, void* buf, size_t cnt) – запись в файл

В программе родительского создаём массив символов, в который с клавиатуры пользователем считывается строка – имя файла. Далее этот файл открываем на чтение. Создаём канал. Создаём дочерний процесс. Закрываем в нём дескриптор канала на чтение. В дочернем процессе поток ввода переопределяем файлом, далее закрываем в неё дескриптор открытого файла. Переопределяем поток вывода каналом. Используем execl для замены памяти процесса на программу, написанную для дочернего процесса. В случае неудачи продолжит выполняться старый код, будет выведено сообщение об ошибке, и программа завершится.

В родительском процессе закрываем дескриптор открытого файла и дескриптор канала на запись. Считываем из канала символы, восстанавливаем из них числа и выводим их пользователю. Закрываем дескриптор на чтение и ожидаем завершение дочернего процесса.

В программе дочернего процесса считываем из входного потока числа и проверяем на простоту. В случае успех процесс завершается. В случае неуспеха числа выводятся в выходной поток.

При создании процесса, открытии файла, создания канала, переопределения дескрипторов, чтении из файла должны обрабатываться случаи ошибок: должно выводиться сообщение об ошибке, открытые дескрипторы должны явно закрываться, программа должна завершаться.

Код программы

parent.c

```
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include <fcntl.h>
#include "sys/wait.h"
#include <string.h>


int main()
{

    char file_name[101];
    scanf("%100s", file_name);
    int d = open(file_name, O_RDONLY);

    if (d == -1)
    {
        perror("open failed");
        return 1;
    }

    int pipe1[2];

    if (pipe(pipe1) == -1)
    {
        close(d);
        perror("pipe failed");
        return 2;
    }
```

```
pid_t pid = fork();

if (-1 == pid)
{
   close(d), close(pipe1[0]), close(pipe1[1]);
   perror("fork");
   return 6;
}


if (pid == 0)
{
   close(pipe1[0]);

   if (dup2(d, STDIN_FILENO) == -1)
   {
      close(d), close(pipe1[1]);
      perror("dup2 failed");
      return 3;
   }

   close(d);

   if (dup2(pipe1[1], STDOUT_FILENO) == -1)
   {
      close(pipe1[1]);
      perror("dup2 failed");
      return 3;
   }

   execl("child", "child", NULL);
   close(pipe1[1]);
   perror("execl() failed");
   return 4;
}
else
```

```
    {
        close(d);
        close(pipe1[1]);
        char number [1000];
        char symbol;
        int i = 0;
        size_t status;

        while ((status = read(pipe1[0], &symbol, sizeof(symbol))) > 0)
        {
            if (symbol == '\n')
            {
                number[i] = 0;
                write(STDOUT_FILENO, number, strlen(number));
                write(STDOUT_FILENO, "\n", 1);
                i = 0;
            }
            else
            {
                number[i++] = symbol;
            }
        }


        close(pipe1[0]);
        wait(NULL);

        if (status == -1)
        {
            perror("read failed");
            return 5;
        }
    }
    return 0;
}
```

child.c

```
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include <string.h>
#include <stdbool.h>
#define MAX_SIZE 1000


bool prime(int n)
{
    for(int i = 2; i * i <= n; i++)
        if(n % i == 0)
            return false;
    return true;
}

int main()
{
    char number[MAX_SIZE];
    char symbol;
    int i = 0;
    while ((read(STDIN_FILENO, &symbol, sizeof(symbol))) > 0)
    {
        if (symbol == '\n')
        {
            number[i] = 0;
            if (prime(atoi(number)))
            {
                return 0;
            }
            write(STDOUT_FILENO, number, strlen(number));
            write(STDOUT_FILENO, "\n", 1);
            i = 0;
        }
        else
        {
```

```
            number[i++] = symbol;
        }
    }
    return 0;
}
```

Протокол работы программы

Тест 1:

```
luckyabatur@DESKTOP-92P4UEK:~/CLionProjects/OS_labs/lab1$ cat numbers.txt
212
32
43221242
34
42
18
8
98888888
124
3
7
88
156
11luckyabatur@DESKTOP-92P4UEK:~/CLionProjects/OS_labs/lab1$ ./parent
numbers.txt
212
32
43221242
34
42
18
8
98888888
124
```

Тест 2:

```
luckyabatur@DESKTOP-92P4UEK:~/CLionProjects/OS_labs/lab1$ cat numbers.txt
17
```

8742

35

978

23luckyabatur@DESKTOP-92P4UEK:~/CLionProjects/OS_labs/lab1$ ./parent

numbers.txt

Тест 3:

luckyabatur@DESKTOP-92P4UEK:~/CLionProjects/OS_labs/lab1$ cat numbers.txt

luckyabatur@DESKTOP-92P4UEK:~/CLionProjects/OS_labs/lab1$ ./parent

numbers.txt

Strace:

luckyabatur@DESKTOP-92P4UEK:~/CLionProjects/OS_labs/lab1$ echo numbers.txt | strace -f ./parent

execve("./parent", ["./parent"], 0x7ffea0b8fea8 /* 36 vars */) = 0

brk(NULL) = 0x559879d14000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff47dfec000

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=22955, ...}) = 0

mmap(NULL, 22955, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff47dfe6000

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) = 0x7ff47ddd4000

mmap(0x7ff47ddfc000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7ff47ddfc000

mmap(0x7ff47df84000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7ff47df84000

mmap(0x7ff47dfd3000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7ff47dfd3000

mmap(0x7ff47dfd9000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff47dfd9000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =

0x7ff47ddd1000
arch_prctl(ARCH_SET_FS, 0x7ff47ddd1740) = 0
set_tid_address(0x7ff47ddd1a10) = 23134
set_robust_list(0x7ff47ddd1a20, 24) = 0
rseq(0x7ff47ddd2060, 0x20, 0, 0x53053053) = 0
mprotect(0x7ff47dfd3000, 16384, PROT_READ) = 0
mprotect(0x5598782ad000, 4096, PROT_READ) = 0
mprotect(0x7ff47e024000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7ff47dfe6000, 22955) = 0
fstat(0, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
getrandom("\xb4\x90\x3c\xe3\x77\x69\x10\x67", 8, GRND_NONBLOCK) = 8
brk(NULL) = 0x559879d14000
brk(0x559879d35000) = 0x559879d35000
read(0, "numbers.txt\n", 4096) = 12
openat(AT_FDCWD, "numbers.txt", O_RDONLY) = 3
pipe2([4, 5], 0) = 0
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 23135 attached
, child_tidptr=0x7ff47ddd1a10) = 23135
[pid 23135] set_robust_list(0x7ff47ddd1a20, 24 <unfinished ...>
[pid 23134] **close**(3 <unfinished ...>
[pid 23135] <... set_robust_list resumed>) = 0
[pid 23134] <... close resumed>) = 0
[pid 23135] **close**(4 <unfinished ...>
[pid 23134] **close**(5 <unfinished ...>
[pid 23135] <... close resumed>) = 0
[pid 23134] <... close resumed>) = 0
[pid 23135] **dup2**(3, 0 <unfinished ...>
[pid 23134] **read**(4, <unfinished ...>
[pid 23135] <... dup2 resumed>) = 0
[pid 23135] **close**(3) = 0
[pid 23135] **dup2**(5, 1) = 1
[pid 23135] **execve**("child", ["child"], 0x7ffec79471a8 /* 36 vars */) = 0
[pid 23135] brk(NULL) = 0x55b3c4f3d000
[pid 23135] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x7f7786396000

[pid 23135] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid 23135] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid 23135] fstat(3, {st_mode=S_IFREG|0644, st_size=22955, ...}) = 0
[pid 23135] mmap(NULL, 22955, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7786390000
[pid 23135] **close**(3) = 0
[pid 23135] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[pid 23135] **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"..., 832) = 832
[pid 23135] 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
[pid 23135] fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
[pid 23135] 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
[pid 23135] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f778617e000
[pid 23135] mmap(0x7f77861a6000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f77861a6000
[pid 23135] mmap(0x7f778632e000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7f778632e000
[pid 23135] mmap(0x7f778637d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7f778637d000
[pid 23135] mmap(0x7f7786383000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f7786383000
[pid 23135] **close**(3) = 0
[pid 23135] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f778617b000
[pid 23135] arch_prctl(ARCH_SET_FS, 0x7f778617b740) = 0
[pid 23135] set_tid_address(0x7f778617ba10) = 23135
[pid 23135] set_robust_list(0x7f778617ba20, 24) = 0
[pid 23135] rseq(0x7f778617c060, 0x20, 0, 0x53053053) = 0
[pid 23135] mprotect(0x7f778637d000, 16384, PROT_READ) = 0
[pid 23135] mprotect(0x55b3c49e9000, 4096, PROT_READ) = 0
[pid 23135] mprotect(0x7f77863ce000, 8192, PROT_READ) = 0
[pid 23135] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 23135] munmap(0x7f7786390000, 22955) = 0
[pid 23135] fstat(0, {st_mode=S_IFREG|0644, st_size=53, ...}) = 0
[pid 23135] getrandom("\x03\xd8\x9b\xa8\xa6\xc2\x65\x49", 8, GRND_NONBLOCK) = 8
[pid 23135] brk(NULL) = 0x55b3c4f3d000
[pid 23135] brk(0x55b3c4f5e000) = 0x55b3c4f5e000

[pid 23135] read(0, "212\n32\n43221242\n34\n42\n18\n8\n98888"..., 4096) = 53
[pid 23135] fstat(1, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
[pid 23135] **write**(1, "212\n32\n43221242\n34\n42\n18\n8\n98888"..., 40) = 40
[pid 23134] <... read resumed>"2", 1) = 1
[pid 23135] lseek(0, -12, SEEK_CUR <unfinished ...>
[pid 23134] **read**(4, <unfinished ...>
[pid 23135] <... lseek resumed>) = 41
[pid 23134] <... read resumed>"1", 1) = 1
[pid 23135] exit_group(0 <unfinished ...>
[pid 23134] **read**(4, <unfinished ...>
[pid 23135] <... exit_group resumed>) = ?
[pid 23134] <... read resumed>"2", 1) = 1
[pid 23134] **read**(4, "\n", 1) = 1
[pid 23135] +++ exited with 0 +++
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=23135, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...}) = 0
write(1, "212\n", 4212
) = 4
read(4, "3", 1) = 1
read(4, "2", 1) = 1
read(4, "\n", 1) = 1
write(1, "32\n", 332
) = 3
read(4, "4", 1) = 1
read(4, "3", 1) = 1
read(4, "2", 1) = 1
read(4, "2", 1) = 1
read(4, "1", 1) = 1
read(4, "2", 1) = 1
read(4, "4", 1) = 1
read(4, "2", 1) = 1
read(4, "\n", 1) = 1
write(1, "43221242\n", 943221242
) = 9
read(4, "3", 1) = 1
read(4, "4", 1) = 1

read(4, "\n", 1) = 1
write(1, "34\n", 334
) = 3
read(4, "4", 1) = 1
read(4, "2", 1) = 1
read(4, "\n", 1) = 1
write(1, "42\n", 342
) = 3
read(4, "1", 1) = 1
read(4, "8", 1) = 1
read(4, "\n", 1) = 1
write(1, "18\n", 318
) = 3
read(4, "8", 1) = 1
read(4, "\n", 1) = 1
write(1, "8\n", 28
) = 2
read(4, "9", 1) = 1
read(4, "8", 1) = 1
read(4, "8", 1) = 1
read(4, "8", 1) = 1
read(4, "8", 1) = 1
read(4, "8", 1) = 1
read(4, "8", 1) = 1
read(4, "8", 1) = 1
read(4, "\n", 1) = 1
write(1, "98888888\n", 998888888
) = 9
read(4, "1", 1) = 1
read(4, "2", 1) = 1
read(4, "4", 1) = 1
read(4, "\n", 1) = 1
write(1, "124\n", 4124
) = 4
read(4, "", 1) = 0
close(4) = 0
wait4(-1, NULL, 0, NULL) = 23135

```
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

При выполнении работы познакомился с новыми системными вызовами и научился с ними работать. Возникли проблемы при считывании чисел из пайпа. Потребовалось проявить внимательность, чтобы сделать все проверки и закрыть все дескрипторы.