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: Юрченко А.Н.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 20.11.24

Москва, 2024

Постановка задачи

Вариант 5.

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 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:

- execve - заменяет текущий процесс на новый, если нет ошибок, то не вернет ничего

- fork - создает новый дочерний процесс, для запуска child
- pipe - сосдает канал для межпроцессорного ваимодействия, через который происходит обмен данными
  pipe(pipe1) создает массив из 2 файловых дескрипторов
- execve - заменяет текущий процесс на новый, если нет ошибок, то не вернет ничего
- dup2 - перенаправляет файловые дискрипторы. Мы перенаправляем ввод и вывод дочернего процесса,
  чтоб он читал данные из pipe1 и писал в pipe2
- waitpid - чтоб родительский процесс не завершился до завершения дочернего

В данной лабораторной работе была написала программу, состоящую из двух процессов: родительского и дочернего, которые взаимодействуют друг с другом с помощью каналов (pipe). Родительский процесс запрашивает ввод числа и передает их дочернему процессу для обработки. Дочерний же процесс читает данные из канала, проверяет является ли число составным, если да то записывает его в указанный файл.

Код программы

parent.c:

```
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int main() {
    char filename[256];
    write(STDOUT_FILENO, "Enter the filename to store composite numbers: ", 47);
    int len = read(STDIN_FILENO, filename, sizeof(filename));
    if (len <= 1) {
        const char msg[] = "error: invalid filename\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    filename[len - 1] = '\0';

    int pipe1[2], pipe2[2]; //wr
    if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {
        const char msg[] = "error with create pipes\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    pid_t pid = fork();
    if (pid == -1) {
        const char msg[] = "error: fork failed\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {
        close(pipe1[1]);
        close(pipe2[0]);

        dup2(pipe1[0], STDIN_FILENO); // Перенаправляем pipe1 на стандартный ввод
        dup2(pipe2[1], STDOUT_FILENO);

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

        _exit(1);
    } else {
        close(pipe1[0]); //родительский отправляет числа
        close(pipe2[1]); //дочерний отправляет через это число в родительский

        int number;
        char buffer[256];
        while (1) {
            write(STDOUT_FILENO, "Enter a number (negative to exit): ", 35);
            int len = read(STDIN_FILENO, buffer, sizeof(buffer));
            if (len <= 1) break;

            buffer[len - 1] = '\0';
            number = atoi(buffer);

            // Отправляем число дочернему процессу
            write(pipe1[1], &number, sizeof(number));

            // Ожидаем ответ от дочернего процесса
            int child_response;
            if (read(pipe2[0], &child_response, sizeof(child_response)) > 0) { //считывает данные из pipe2
                if (child_response < 0) {
                    write(STDOUT_FILENO, "Child indicated to terminate\n", 29);
                    break;
                }
            }
            if (number < 0) break;
        }

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

        int status;
        waitpid(pid, &status, 0);
        write(STDOUT_FILENO, "Parent process exiting.\n", 24);
    }

    return 0;
}
```

child.c:

```
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>

bool is_prime(int num) {
    if (num <= 1) return false;
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) return false;
    }
    return true;
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        const char msg[] = "error: not enough arg\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        exit(EXIT_FAILURE);
    }

    const char *filename = argv[1];
    int pipe1 = STDIN_FILENO;   // Константа (0) указывает на stdin
    int pipe2 = STDOUT_FILENO;

    int file_fd = open(filename, O_WRONLY | O_CREAT | O_APPEND, 0644);  //O_WRONLY - открываем для записи,
    if (file_fd < 0) {                                                                                  //O_CREAT - создаем, если нет,
        const char msg[] = "error with open requested file\n";          //O_APPEND - добавляем в конец,
        write(STDERR_FILENO, msg, sizeof(msg));                         //0644 - права доступа на файл (r/w - владелец, r)
        exit(EXIT_FAILURE);
    }

    int number;
    while (1) {
        if (read(pipe1, &number, sizeof(number)) <= 0) break;

        if (number < 0) {
            write(pipe2, &number, sizeof(number));
            break;
        }

        if (is_prime(number) || number == 1 || number == 0) {
            write(pipe2, &number, sizeof(number));
        } else {
            char buffer[20];
            int len = snprintf(buffer, sizeof(buffer), "%d\n", number);
            //переделывает num в строку и записывает в buf

            write(file_fd, buffer, len);
            write(pipe2, &number, sizeof(number));
        }
    }

    close(file_fd);
    exit(0);
}
```

Протокол работы программы

Тестирование

```
ann@ann-ThinkPad-T460:~/Desktop/osi/1$ ./parent
Enter the filename to store composite numbers: output.txt
Enter a number (negative to exit): 4
Enter a number (negative to exit): 2
Enter a number (negative to exit): 1
Enter a number (negative to exit): 100
Enter a number (negative to exit):
Parent process exiting.
```

Содержимое файла:

4
100

Strace

```
$ strace ./parent
execve("./parent", ["./parent"], 0x7ffed1e2e560 /* 70 vars */) = 0
brk(NULL)                               = 0x6288514c3000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffca7614610) = -1 EINVAL (Недопустимый аргумент)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7be2be681000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=66559, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 66559, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7be2be670000
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\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
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\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\0"..., 784, 64) = 784
mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7be2be400000
mprotect(0x7be2be428000, 2023424, PROT_NONE) = 0
mmap(0x7be2be428000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7be2be428000
mmap(0x7be2be5bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7be2be5bd000
mmap(0x7be2be616000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7be2be616000
mmap(0x7be2be61c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7be2be61c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7be2be66d000
arch_prctl(ARCH_SET_FS, 0x7be2be66d740) = 0
set_tid_address(0x7be2be66da10)         = 7239
set_robust_list(0x7be2be66da20, 24)     = 0
rseq(0x7be2be66e0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7be2be616000, 16384, PROT_READ) = 0
mprotect(0x628850807000, 4096, PROT_READ) = 0
mprotect(0x7be2be6bb000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7be2be670000, 66559)           = 0
write(1, "Enter the filename to store comp"..., 47Enter the filename to store composite numbers: ) = 47
read(0, output.txt
"output.txt\n", 256)            = 11
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7be2be66da10) = 7379
close(3)                                = 0
close(6)                                = 0
write(1, "Enter a number (negative to exit"..., 35Enter a number (negative to exit): ) = 35
read(0, 4
"4\n", 256)                     = 2
write(4, "\4\0\0\0", 4)                 = 4
read(5, "\4\0\0\0", 4)                  = 4
write(1, "Enter a number (negative to exit"..., 35Enter a number (negative to exit): ) = 35
read(0, 1
"1\n", 256)                     = 2
write(4, "\1\0\0\0", 4)                 = 4
read(5, "\1\0\0\0", 4)                  = 4
write(1, "Enter a number (negative to exit"..., 35Enter a number (negative to exit): ) = 35
read(0, 100
"100\n", 256)                   = 4
write(4, "d\0\0\0", 4)                  = 4
read(5, "d\0\0\0", 4)                   = 4
write(1, "Enter a number (negative to exit"..., 35Enter a number (negative to exit): ) = 35
read(0,
"\n", 256)                      = 1
close(4)                                = 0
close(5)                                = 0
wait4(7379, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 7379
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=7379, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
write(1, "Parent process exiting.\n", 24Parent process exiting.
) = 24
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В ходе лабораторной работы я узнала о некоторых системных вызовах и научилась их использовать. Впервые мне пришлось создавать каналы, чтобы с их помощью обменивать данные между процессами. Сложность возникла в том, что было слишком много новой информации, и ушло долгое время на их понимание.