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: Тульчинский Г.С.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 12.12.24

Москва, 2024

Постановка задачи

Вариант 1.

Пользователь вводит команды вида: «число число число». 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); – создает дочерний процесс.
- int pipe(int pipefd[2]) - создает наименованный канал для передачи данных между процессами
- void exit(int status) - завершает выполнение процесса и возвращение статуса
- int dup2(int oldfd, int newfd) - переназначает файловый дескриптор
- int open(const char* pathname, int flags, mode_t mode) - открывает\создает файл
- int close(int fd) - закртывает файл
- int execv(const char *file, char* const argv...) - заменет образ текущего процесса

на образ нового процесса file

- int write(int pipe, char* buffer, int size) - записывает данные в файл, связанный с файловым дескриптором
- int fgets(char* buffer, int size, stdin) - читает данные из файла(gjnjrf), связанного с файловым дескриптором
- pid_t wait(int status) - ожидает завершение дочернего процесса

В данной лабораторной работе я написал программу, состоящую из двух процессов: родительского и дочернего, которые взаимодействуют друг с другом с помощью канала (pipe). Родительский процесс запрашивает ввод чисел и передает их дочернему процессу для обработки. Дочерний процесс читает данные из канала, вычисляет сумму введенных чисел каждой новой строки, пока не встретит exit, и записывает результаты в указанный файл. Программа включает в себя обработку ошибок, таких как отсутствие аргументов командной строки и сбои при создании процессов и открытии файлов.

Код программы

parent.c:

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

#define BUFFER_SIZE 256

int main() {
    int pipe1[2];
    pid_t pid;
    char filename[BUFFER_SIZE];
    char buffer[BUFFER_SIZE];
    if (pipe(pipe1) == -1) {
        perror("Ошибка при создании pipe");
        exit(EXIT_FAILURE);
    }
    printf("Введите имя файла: ");
    if (fgets(filename, BUFFER_SIZE, stdin) == NULL) {
        perror("Ошибка ввода имени файла");
        exit(EXIT_FAILURE);
    }
    filename[strcspn(filename, "\n")] = '\0';
    pid = fork();
    if (pid == -1) {
        perror("Ошибка при создании процесса");
        exit(EXIT_FAILURE);
    }
    if (pid == 0) { // Дочерний процесс
        close(pipe1[1]); // Закрываем запись в pipe1
        dup2(pipe1[0], STDIN_FILENO);
        close(pipe1[1]);
        execlp("./child", "./child", filename, NULL);
        perror("Ошибка при запуске дочернего процесса");
        exit(EXIT_FAILURE);
    }
```

```c
    else { // Родительский процесс
        close(pipe1[0]); // Закрываем чтение из pipe1
        while (1) {
            printf("Введите числа через пробел (или 'exit' для выхода): ");
            if (fgets(buffer, BUFFER_SIZE, stdin) == NULL) {
                perror("Ошибка ввода команды");
                break;
            }
            if (strncmp(buffer, "exit", 4) == 0) {
                break;
            }
            write(pipe1[1], buffer, strlen(buffer));
        }
        close(pipe1[1]);
        wait(NULL);
    }
    return 0;
}
```

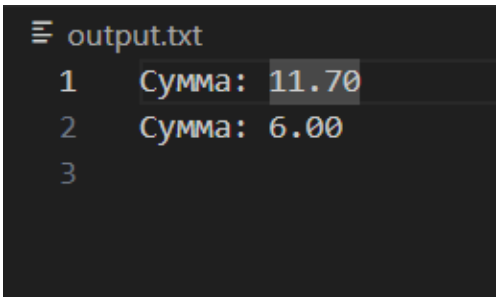
child.c:

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

#define BUFFER_S
#define stderr stderr
Expands to:
stderr

int main(int arg
    if (argc !=
        fprintf(stderr, "Использование: %s <имя файла>\n", argv[0]);
        exit(EXIT_FAILURE);
    }
    const char *filename = argv[1];
    FILE *file = fopen(filename, "w");
    if (!file) {
        perror("Ошибка при открытии файла");
        exit(EXIT_FAILURE);
    }
    char buffer[BUFFER_SIZE];
    while (fgets(buffer, BUFFER_SIZE, stdin) != NULL) {
        float sum = 0;
        char *token = strtok(buffer, " ");
        while (token != NULL) {
            sum += strtof(token, NULL);
            token = strtok(NULL, " ");
        }
        fprintf(file, "Сумма: %.2f\n", sum);
        fflush(file);
    }
    fclose(file);
    return 0;
}
```

Протокол работы программы

Тестирование

```
tulchinskij@LAPTOP-QIG5MTAH:~$ gcc os1_ch.c -g -o child -lm
tulchinskij@LAPTOP-QIG5MTAH:~$ gcc os1.c -g -o lab -lm
tulchinskij@LAPTOP-QIG5MTAH:~$ ./lab
Введите имя файла: output.txt
Введите числа через пробел (или 'exit' для выхода): 10 1.2 0.5
Введите числа через пробел (или 'exit' для выхода): 1 2 3
Введите числа через пробел (или 'exit' для выхода): exit
```



Strace

$ strace ./parent output.txt

```
execve("./lab", ["./lab", "output.txt"], 0x7ffe4baeeb18 /* 35 vars */) = 0
brk(NULL)                               = 0x5590fb53d000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd19df0c90) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f0865abc000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=16995, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 16995, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f0865ab7000
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) = 0x7f086588e000
mprotect(0x7f08658b6000, 2023424, PROT_NONE) = 0
mmap(0x7f08658b6000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x28000) = 0x7f08658b6000
mmap(0x7f0865a4b000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7f0865a4b000
mmap(0x7f0865aa4000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x215000) = 0x7f0865aa4000
mmap(0x7f0865aaa000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f0865aaa000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f086588b000
arch_prctl(ARCH_SET_FS, 0x7f086588b740) = 0
set_tid_address(0x7f086588ba10)         = 57294
set_robust_list(0x7f086588ba20, 24)     = 0
rseq(0x7f086588c0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f0865aa4000, 16384, PROT_READ) = 0
mprotect(0x5590e72c6000, 4096, PROT_READ) = 0
mprotect(0x7f0865af6000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f0865ab7000, 16995)           = 0
pipe2([3, 4], 0)                        = 0
```

```
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...}, AT_EMPTY_PATH) = 0
getrandom("\xbf\x9a\x59\x2c\xaf\x6b\x68\x00", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5590fb53d000
brk(0x5590fb55e000)                     = 0x5590fb55e000
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x4), ...}, 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"..., 34Введите имя файла: ) = 34
read(0,
```

Вывод

В ходе лабораторной работы я узнал о некоторых системных вызовах и научилася их использовать. Впервые мне пришлось создавать каналы, чтобы с их помощью обменивать данные между процессами. Сложность возникла в том, что мой компьютер не знает, что такое read().