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-214Б-23

Студент: ЫсаевР. К.

Преподаватель: Бахарев В.Д. (ФИИТ)

Оценка: _____

Дата: 22.11.24

Москва, 2024

Постановка задачи

Вариант 3.

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 число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс производит деление первого числа, на последующие, а результат выводит в файл. Если происходит деление на 0, то тогда дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип int. Количество чисел может быть произвольным.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pipe: Создает канал для межпроцессного взаимодействия.
- close: Закрывает файловый дескриптор.
- dup2: Дублирует файловый дескриптор, перенаправляя его на другой дескриптор.
- execl: Заменяет текущий процесс новым процессом, выполняя указанную программу.
- write: Записывает данные в файловый дескриптор.
- read: Завершает процесс с указанным кодом выхода.

Я использовал форматирование строк для представления чисел и результатов деления с помощью функций format_result, itoa, write_to_buffer. Затем шла работа над parent.c. Вся суть заключается в том, чтобы считать имя файла, в который выведется результат, затем считать строку с числами. После этого все данные отправляются в child.c, который, в свою очередь, создает и открывает файл, а потом записывает в него через пробел результаты деления первого числа на остальные.

Код программы

child.c

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>

int write_to_buffer(char *buffer, size_t buffer_size, const char *str) {
    size_t len = strlen(str);
    if (len > buffer_size) {
        len = buffer_size;
    }
    memcpy(buffer, str, len);
    return len;
```

```
}

void itoa(int num, char *str) {
    int i = 0, isNegative = 0;
    if (num < 0) {
        isNegative = 1;
        num = -num;
    }
    do {
        str[i++] = (num % 10) + '0';
        num /= 10;
    } while (num > 0);
    if (isNegative) {
        str[i++] = '-';
    }
    str[i] = '\0';
    for (int j = 0, k = i - 1; j < k; j++, k--) {
        char temp = str[j];
        str[j] = str[k];
        str[k] = temp;
    }
}

int format_result(int numerator, int denominator, char *result, size_t size) {
    char num_str[32], denom_str[32], res_str[32];
    itoa(numerator, num_str);
    itoa(denominator, denom_str);
    float div_result = (float)numerator / denominator;
    int int_part = (int)div_result;
    itoa(int_part, res_str);
    int offset = 0;
    offset += write_to_buffer(result + offset, size - offset, num_str);
    offset += write_to_buffer(result + offset, size - offset, " / ");
    offset += write_to_buffer(result + offset, size - offset, denom_str);
    offset += write_to_buffer(result + offset, size - offset, " = ");
    offset += write_to_buffer(result + offset, size - offset, res_str);
    offset += write_to_buffer(result + offset, size - offset, "\n");
    return offset;
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        const char *msg = "Usage: ./child <filename>\n";
        write(STDERR_FILENO, msg, strlen(msg));
        exit(1);
    }

    char line[1024];
```

```
    int numbers[100];
    int count;
    int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);

    if (fd == -1) {
        const char *err = "Error opening file\n";
        write(STDERR_FILENO, err, strlen(err));
        exit(1);
    }

    ssize_t bytesRead;
    while ((bytesRead = read(STDIN_FILENO, line, sizeof(line) - 1)) > 0) {
        line[bytesRead] = '\0';
        count = 0;

        char *token = strtok(line, " ");
        while (token != NULL && count < 100) {
            numbers[count++] = atoi(token);
            token = strtok(NULL, " ");
        }

        if (count < 2) {
            const char *msg = "Недостаточно чисел в строке\n";
            write(STDERR_FILENO, msg, strlen(msg));
            continue;
        }

        int numerator = numbers[0];
        for (int i = 1; i < count; i++) {
            if (numbers[i] == 0) {
                const char *err = "Ошибка: деление на 0\n";
                write(STDERR_FILENO, err, strlen(err));
                write(fd, err, strlen(err));
                close(fd);
                exit(1);
            }

            char result[128];
            int length = format_result(numerator, numbers[i], result, sizeof(result));
            write(fd, result, length);
            write(STDOUT_FILENO, result, length);
        }
    }

    close(fd);
    return 0;
}
```

parent.c

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

int main(int argc, char *argv[]) {
    if (argc != 2) {
        const char *msg = "Usage: ./parent <filename>\n";
        write(STDERR_FILENO, msg, strlen(msg));
        exit(1);
    }

    int pipe1[2], pipe2[2];
    char buffer[1024];

    if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {
        const char *err = "Error creating pipe\n";
        write(STDERR_FILENO, err, strlen(err));
        exit(1);
    }

    pid_t pid = fork();

    if (pid < 0) {
        const char *err = "Error creating child process\n";
        write(STDERR_FILENO, err, strlen(err));
        exit(1);
    }

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

        dup2(pipe1[0], STDIN_FILENO);
        close(pipe1[0]);

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

        execl("./child", "./child", argv[1], NULL);
        const char *err = "execl: No such file or directory\n";
        write(STDERR_FILENO, err, strlen(err));
        exit(1);
    } else {
        close(pipe1[0]);
        close(pipe2[1]);

        const char *msg = "Введите команды (формат: число число число<endline>, для выхода
```

```
нажмите Ctrl+D):\n";
        write(STDOUT_FILENO, msg, strlen(msg));
        while ((read(STDIN_FILENO, buffer, sizeof(buffer))) > 0) {
            write(pipe1[1], buffer, strlen(buffer));
        }
        close(pipe1[1]);

        ssize_t bytesRead;
        while ((bytesRead = read(pipe2[0], buffer, sizeof(buffer))) > 0) {
            write(STDOUT_FILENO, buffer, bytesRead);
        }
        close(pipe2[0]);
    }

    return 0;
}
```

Протокол работы программы

```
admin1@admin1-VMware-Virtual-Platform:~/OSY_LABS/Laba1$ strace -f ./parent 1.txt

execve("./parent", ["./parent", "1.txt"], 0x7ffe225dab40 /* 79 vars */) = 0

brk(NULL)                               = 0x5bf717226000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7e9fbe748000

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=64511, ...}) = 0

mmap(NULL, 64511, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7e9fbe738000

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\3\0>\0\1\0\0\0\220\243\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\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) = 0x7e9fbe400000

mmap(0x7e9fbe428000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7e9fbe428000

mmap(0x7e9fbe5b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7e9fbe5b0000
```

mmap(0x7e9fbe5ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7e9fbe5ff000

mmap(0x7e9fbe605000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7e9fbe605000

close(3)   = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7e9fbe735000

arch_prctl(ARCH_SET_FS, 0x7e9fbe735740) = 0

set_tid_address(0x7e9fbe735a10)   = 9075

set_robust_list(0x7e9fbe735a20, 24)   = 0

rseq(0x7e9fbe736060, 0x20, 0, 0x53053053) = 0

mprotect(0x7e9fbe5ff000, 16384, PROT_READ) = 0

mprotect(0x5bf715bbd000, 4096, PROT_READ) = 0

mprotect(0x7e9fbe780000, 8192, PROT_READ) = 0

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

munmap(0x7e9fbe738000, 64511)   = 0

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

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

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
Process 9076 attached

, child_tidptr=0x7e9fbe735a10) = 9076

[pid 9076] set_robust_list(0x7e9fbe735a20, 24 <unfinished ...>

[pid 9075] close(3 <unfinished ...>

[pid 9076] <... set_robust_list resumed>) = 0

[pid 9075] <... close resumed>)   = 0

[pid 9075] close(6 <unfinished ...>

[pid 9076] close(4 <unfinished ...>

[pid 9075] <... close resumed>)   = 0

[pid 9076] <... close resumed>)   = 0

[pid 9075] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\213 (\321"..., 132 <unfinished ...>

[pid 9076] close(5Введите команды (формат: число число число<endline>, для выхода нажмите Ctrl+D):

)   = 0

[pid 9075] <... write resumed>)   = 132

[pid 9076] dup2(3, 0 <unfinished ...>

[pid 9075] read(0, <unfinished ...>

[pid 9076] <... dup2 resumed>)   = 0

[pid 9076] close(3)   = 0

[pid 9076] dup2(6, 1)   = 1

[pid 9076] close(6)   = 0

[pid 9076] execve("./child", ["./child", "1.txt"], 0x7fff72ec4730 /* 79 vars */) = 0

[pid 9076] brk(NULL)   = 0x5fadf10c2000

[pid 9076] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x77505d6d0000

[pid 9076] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

[pid 9076] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

[pid 9076] fstat(3, {st_mode=S_IFREG|0644, st_size=64511, ...}) = 0

[pid 9076] mmap(NULL, 64511, PROT_READ, MAP_PRIVATE, 3, 0) = 0x77505d6c0000

[pid 9076] close(3)   = 0

[pid 9076] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

[pid 9076] 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

[pid 9076] 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\0\0"..., 784, 64) = 784

[pid 9076] fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

[pid 9076] 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\0\0"..., 784, 64) = 784

[pid 9076] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x77505d400000

[pid 9076] mmap(0x77505d428000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x77505d428000

[pid 9076] mmap(0x77505d5b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x77505d5b0000

[pid 9076] mmap(0x77505d5ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x77505d5ff000

[pid 9076] mmap(0x77505d605000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x77505d605000

[pid 9076] close(3)   = 0

[pid 9076] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x77505d6bd000

[pid 9076] arch_prctl(ARCH_SET_FS, 0x77505d6bd740) = 0

[pid 9076] set_tid_address(0x77505d6bda10) = 9076

[pid 9076] set_robust_list(0x77505d6bda20, 24) = 0

[pid 9076] rseq(0x77505d6be060, 0x20, 0, 0x53053053) = 0

[pid 9076] mprotect(0x77505d5ff000, 16384, PROT_READ) = 0

[pid 9076] mprotect(0x5fadf0029000, 4096, PROT_READ) = 0

[pid 9076] mprotect(0x77505d708000, 8192, PROT_READ) = 0

[pid 9076] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

[pid 9076] munmap(0x77505d6c0000, 64511) = 0

[pid 9076] openat(AT_FDCWD, "1.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 3

[pid 9076] read(0, 100

 <unfinished ...>

[pid 9075] <... read resumed>"100\n", 1024) = 4

[pid 9075] write(4, "100\n\237~", 6)   = 6

[pid 9076] <... read resumed>"100\n\237~", 1023) = 6

[pid 9075] read(0, <unfinished ...>

[pid 9076] write(2,
"\320\235\320\265\320\264\320\276\321\201\321\202\320\260\321\202\320\276\321\207\320\275\320\276
\321\207\320\270\321\201\320"..., 52Недостаточно чисел в строке

) = 52

[pid 9076] read(0, 2

 <unfinished ...>

[pid 9075] <... read resumed>"2\n", 1024) = 2

[pid 9075] write(4, "2\n0\n\237~", 6) = 6

[pid 9076] <... read resumed>"2\n0\n\237~", 1023) = 6

[pid 9075] read(0, <unfinished ...>

[pid 9076] write(2,
"\320\235\320\265\320\264\320\276\321\201\321\202\320\260\321\202\320\276\321\207\320\275\320\276
\321\207\320\270\321\201\320"..., 52Недостаточно чисел в строке

) = 52

[pid 9076] read(0, 5

<unfinished ...>

[pid 9075] <... read resumed>"5\n", 1024) = 2

[pid 9075] write(4, "5\n0\n\237~", 6) = 6

[pid 9076] <... read resumed>"5\n0\n\237~", 1023) = 6

[pid 9075] read(0, <unfinished ...>

[pid 9076] write(2,
"\320\235\320\265\320\264\320\276\321\201\321\202\320\260\321\202\320\276\321\207\320\275\320\276
\321\207\320\270\321\201\320"..., 52Недостаточно чисел в строке

) = 52

[pid 9076] read(0, 20

 <unfinished ...>

[pid 9075] <... read resumed>"20\n", 1024) = 3

[pid 9075] write(4, "20\n\n\237~", 6) = 6

[pid 9076] <... read resumed>"20\n\n\237~", 1023) = 6

[pid 9075] read(0, <unfinished ...>

[pid 9076] write(2,
"\320\235\320\265\320\264\320\276\321\201\321\202\320\260\321\202\320\276\321\207\320\275\320\276
\321\207\320\270\321\201\320"..., 52Недостаточно чисел в строке

) = 52

[pid 9076] read(0, 50

 <unfinished ...>

[pid 9075] <... read resumed>"50\n", 1024) = 3

[pid 9075] write(4, "50\n\n\237~", 6) = 6

[pid 9076] <... read resumed>"50\n\n\237~", 1023) = 6

[pid 9075] read(0, <unfinished ...>

[pid 9076] write(2,
"\320\235\320\265\320\264\320\276\321\201\321\202\320\260\321\202\320\276\321\207\320\275\320\276
\321\207\320\270\321\201\320"..., 52Недостаточно чисел в строке

) = 52

[pid 9076] read(0, <unfinished ...>

[pid 9075] <... read resumed>"", 1024) = 0

[pid 9075] close(4)   = 0

[pid 9076] <... read resumed>"", 1023) = 0

[pid 9075] read(5, <unfinished ...>

[pid 9076] close(3)   = 0

[pid 9076] exit_group(0)   = ?

[pid 9075] <... read resumed>"", 1024) = 0

[pid 9076] +++ exited with 0 +++

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=9076, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---

close(5)   = 0

exit_group(0)   = ?

+++ exited with 0 +++

Вывод

В ходе лабораторной работы были изучены и применены методы межпроцессного взаимодействия через каналы (pipes) и системные вызовы для создания и управления процессами в Unix-подобных системах. Были освоены навыки перенаправления ввода/вывода, обработки ошибок и корректного освобождения ресурсов. Также были реализованы собственные функции для форматирования строк и обработки данных, что позволило успешно выполнить поставленные задачи.