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: Воронухин Н.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 10.12.24

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:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int *fd); – создает pipe, однонаправленный канал связи между процессами.
- ssize_t write(int fd, const void buf[.count], size_t count); - пишет count байт из буфера buf в файл, на который ссылается файловый дескриптор fd.
- ssize_t read(int fd, void buf[.count], size_t count); - считывает count байт в буфер buf из файла, на который ссылается файловый дескриптор fd
- int open(const char *pathname, int flags, /* mode_t mode */); - открывает файл описанный в pathname, параметр флагов flags определяет способ открытия файла (только для чтения, записи и тд), аргумент mode задает права доступа к файлу, в случае создания нового файла.
- int close(int fd); - закрывает файловый дескриптор.
- pid_t wait(int *status); - приостанавливает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится, или до появления сигнала, который либо завершает текущий процесс, либо требует вызвать функцию-обработчик.
- void exit(int status); - приводит к обычному завершению программы, и величина status возвращается процессу-родителю.
- int openat(int dirfd, const char *pathname, int flags, mode_t mode); - то же, что и open() но путь к фалуй относительный и в него передается дескриптор директории.
- exit_group – системный вызов завершающий все потоки процесса.

Программа получает из стандартного потока ввода имя файла, открывает его, создает 2 канала pipe и создает дочерний процесс с помощью fork. Далее происходит разделение на 2 подпрограммы с помощью if по pid процесса, дочерний процесс посылает в pipe2 сообщение о готовности к чтению, родительский процесс считывает его, и начинает считывать данные из потока ввода, отправляя их через pipe1 дочернему процессу. Дочерний процесс анализирует ввод, производя, если необходимо вычисления. В случае ошибки, он отправляет через pipe2 сообщение об этом. Результаты вычисления дочерний процесс записывает в файл.

В случае получения делителя 0, который служит маркером для остановки, дочерний процесс отправляет через pipe2 сообщение об остановке родителю и завершается, записав последний результат в файл.

Родитель производит итерации в цикле, каждый раз сначала считывая сообщение полученное от дочернего процесса, о его состоянии, и дальше либо считывает последующие данные из потока ввода, либо завершается с ошибкой (в случае возникновения ошибки в дочернем процессе), либо завершается нормально, перед этим получив код завершения дочернего процесса через wait(NULL), чтобы избежать ситуации, когда дочерний процесс становится зомби.

Код программы

Lab_1.c

```
#include <unistd.h>

#include <sys/types.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <stdlib.h>

#include <ctype.h>

#include <sys/wait.h>

#include <stdio.h> // sprintf() for cast float to char*



int strLen(char*);

int getFirstNumAndShift(float, float*, int*);

char* floatToStr(float);

void exitParent(int, int, int);

void exitChilde(int, int, int, int, char);

void stopChilde(float, int, int ,int, char);
```

```
const char STOP_MES = '1';

const char ERROR_MES = '2';

const char READY_TO_READ = '3';

const int FileNameBufSize = 15;

const int StrBufSize = 50;


typedef enum {

    CARRY_DIVIDED = 0b1,

    CARRY_DIVISOR = 0b10,

    SPACE_FOUND   = 0b100,

    ENDLINE_FOUND = 0b1000,

    CUR_VAL_DIGIT = 0b10000,

    CUR_VAL_SPACE = 0b100000,

    CUR_VAL_ENDL  = 0b1000000,

    CUR_VAL_SIGN  = 0b10000000,

    SIGN_FOUND    = 0b100000000,

    CUR_VAL_ZERO  = 0b1000000000,

    ZERO_FOUND    = 0b10000000000,

} FLAGSS;


typedef enum {

    READ_ERR = 1,

    WRONG_FILE_NAME_ERR,

    OPEN_FILE_ERR,

    CREATE_PIPE_ERR,

    CREATE_PROCESS_ERR,

    WRITE_ERR,

    WRONG_INSTRUCTION_ERR,

    CHILDE_ERROR,

} ERR_FLG;


int main() {

    char buff[FileNameBufSize];
```

```
        buff[FileNameBufSize - 1] = '\n';
        int nameFileLen = 0;
        if ((nameFileLen = read(STDIN_FILENO, buff, (int)sizeof(buff))) == -1)
exit(READ_ERR);
        else if (buff[nameFileLen - 1] != '\n') exit(WRONG_FILE_NAME_ERR);
        buff[nameFileLen - 1] = '\0';
        int fd = open(buff, O_WRONLY | O_CREAT | O_TRUNC, S_IRWXU | S_IRWXO);
        if (fd == -1) exit(OPEN_FILE_ERR);

        int pToChilde[2];
        if (pipe(pToChilde)) {
            close(fd);
            exit(CREATE_PIPE_ERR);
        }
        int pToParent[2];
        if (pipe(pToParent)) {
            close(fd);
            close(pToChilde[1]);
            close(pToChilde[0]);
            exit(CREATE_PIPE_ERR);
        }

        pid_t pid = fork();
        if (pid == -1) {
            close(fd);
            close(pToChilde[1]);
            close(pToChilde[0]);
            close(pToParent[1]);
            close(pToParent[0]);
            exit(CREATE_PROCESS_ERR);
        }
        else if (pid == 0) {
            close(pToChilde[1]);
```

```
            close(pToParent[0]);
            char strBuf[StrBufSize];
            float divided = 0;
            int divisor = 0;
            int parser_flag = CARRY_DIVIDED | ENDLINE_FOUND;
            int end_cycle_flag = 0;
            while (!end_cycle_flag) {
                int number = 0;
                if (write(pToParent[1], &READY_TO_READ, sizeof(READY_TO_READ)) == -1) {
                    exitChilde(WRITE_ERR, pToChilde[0], pToParent[1], fd, ERROR_MES);
                }
                if ((number = read(pToChilde[0], strBuf, (int)sizeof(strBuf))) == -1) {
                    exitChilde(READ_ERR, pToChilde[0], pToParent[1], fd, ERROR_MES);}
                else if (number == 0) {
                    if (parser_flag & SPACE_FOUND) {} // endl sont check
                }
                number /= sizeof(char);
                char* ptr = strBuf;
                for (; ptr < strBuf + number; ++ptr) {
                    if (*ptr == '0' && (parser_flag & (ENDLINE_FOUND | SPACE_FOUND |
SIGN_FOUND | ZERO_FOUND)))
                        parser_flag |= CUR_VAL_ZERO;
                    else if (isdigit(*ptr)) parser_flag |= CUR_VAL_DIGIT;
                    else if (*ptr == ' ') parser_flag |= CUR_VAL_SPACE;
                    else if (*ptr == '\n') parser_flag |= CUR_VAL_ENDL;
                    else if (*ptr == '-' || *ptr == '+') parser_flag |= CUR_VAL_SIGN;

                    switch (parser_flag) {
                        case CARRY_DIVIDED | CUR_VAL_DIGIT:
                        case CARRY_DIVIDED | CUR_VAL_DIGIT | ENDLINE_FOUND:
                            divided = divided * 10 + *ptr - '0';
                            parser_flag = CARRY_DIVIDED;
                            break;
```

```
                        case CARRY_DIVIDED | CUR_VAL_DIGIT | SIGN_FOUND:
                            divided = divided * (*ptr - '0');
                            parser_flag = CARRY_DIVIDED;
                            break;


                        case CARRY_DIVIDED | CUR_VAL_SPACE:
                        case CARRY_DIVIDED | CUR_VAL_SPACE | ZERO_FOUND:
                            parser_flag = CARRY_DIVISOR | SPACE_FOUND;
                            break;


                        case CARRY_DIVIDED | CUR_VAL_ENDL:
                        case CARRY_DIVIDED | CUR_VAL_ENDL | ZERO_FOUND:
                            char* result = floatToStr(divided);
                            int len = strLen(result);
                            if (write(fd, result, len * sizeof(char)) == -1) {
                                free(result);
                                exitChilde(WRITE_ERR, pToChilde[0], pToParent[1], fd,
ERROR_MES);
                            }
                            free(result);
                            divided = 0;
                            parser_flag = CARRY_DIVIDED | ENDLINE_FOUND;
                            break;


                        case CARRY_DIVIDED | CUR_VAL_SIGN | ENDLINE_FOUND:
                            if (*ptr == '-') divided = -1;
                            else divided = 1;
                            parser_flag = CARRY_DIVIDED | SIGN_FOUND;
                            break;
```

```
                        case CARRY_DIVIDED | CUR_VAL_ZERO | ENDLINE_FOUND:
                        case CARRY_DIVIDED | CUR_VAL_ZERO | SIGN_FOUND:
                            divided = 0;
                            parser_flag = CARRY_DIVIDED | ZERO_FOUND;
                            break;



                        case CARRY_DIVISOR | CUR_VAL_DIGIT:
                        case CARRY_DIVISOR | CUR_VAL_DIGIT | SPACE_FOUND:
                            divisor = divisor * 10 + *ptr - '0';
                            parser_flag = CARRY_DIVISOR;
                            break;
                        case CARRY_DIVISOR | CUR_VAL_DIGIT | SIGN_FOUND:
                            divisor = divisor * (*ptr - '0');
                            parser_flag = CARRY_DIVISOR;
                            break;



                        case CARRY_DIVISOR | CUR_VAL_SPACE:
                        case CARRY_DIVISOR | CUR_VAL_SPACE | ZERO_FOUND:
                            if (divisor == 0) {
                                stopChilde(divided, pToChilde[0], pToParent[1], fd,
STOP_MES);

                                divided = 0;
                                end_cycle_flag = 1;
                            }
                            else {
                                divided /= divisor;
                                divisor = 0;
                                parser_flag = CARRY_DIVISOR | SPACE_FOUND;
                            }
```

```
                    break;



                case CARRY_DIVISOR | CUR_VAL_ENDL:
                case CARRY_DIVISOR | CUR_VAL_ENDL | ZERO_FOUND:
                    if (divisor == 0) {
                        stopChilde(divided, pToChilde[0], pToParent[1], fd,
STOP_MES);

                        divided = 0;
                        end_cycle_flag = 1;
                    }
                    else {
                        divided /= divisor;
                        divisor = 0;
                        char* result = floatToStr(divided);
                        int len = strLen(result);
                        if (write(fd, result, sizeof(char) * len) == -1) {
                            free(result);
                            exitChilde(WRITE_ERR, pToChilde[0], pToParent[1], fd,
ERROR_MES);
                        }
                        divided = 0;
                        free(result);
                        parser_flag = CARRY_DIVIDED | ENDLINE_FOUND;
                    }
                    break;



                case CARRY_DIVISOR | CUR_VAL_SIGN | SPACE_FOUND:
                    if (*ptr == '-') divisor = -1;
                    else divisor = 1;
                    parser_flag = CARRY_DIVISOR | SIGN_FOUND;
                    break;
```

```
                        case CARRY_DIVISOR | CUR_VAL_ZERO | SPACE_FOUND:
                        case CARRY_DIVISOR | CUR_VAL_ZERO | SIGN_FOUND:
                            divisor = 0;
                            parser_flag = CARRY_DIVISOR | ZERO_FOUND;
                            break;

                        default:
                        /*
                            ERRORS:
                            case CARRY_DIVIDED | CUR_VAL_SPACE | ENDLINE_FOUND
                            case CARRY_DIVIDED | CUR_VAL_ENDL | ENDLINE_FOUND
                            case CARRY_DIVISOR | CUR_VAL_SPACE | SPACE_FOUND
                            case CARRY_DIVISOR | CUR_VAL_ENDL | SPACE_FOUND
                            case CARRY_DIVIDED | CUR_VAL_SIGN ...
                            IMPOSSIBLE:
                            case CARRY_DIVIDED | CUR_VAL_DIGIT | SPACE_FOUND : SPACE_FOUND
-> CARRY_DIVISOR
                            case CARRY_DIVIDED | CUR_VAL_SPACE | SPACE_FOUND
                            case CARRY_DIVIDED | CUR_VAL_ENDL | SPACE_FOUND
                            case CARRY_DIVISOR | CUR_VAL_DIGIT | ENDLINE_FOUND :
ENDLINE_FOUND -> CARRY_DIVIDED
                            case CARRY_DIVISOR | CUR_VAL_SPACE | ENDLINE_FOUND
                            case CARRY_DIVISOR | CUR_VAL_SPACE | ENDLINE_FOUND
                            case CARRY_DIVIDED | CUR_VAL_SIGN | SPACE_FOUND ...
                        */
                            exitChilde(WRONG_INSTRUCTION_ERR, pToChilde[0], pToParent[1],
fd, ERROR_MES);

                            break;
                    }
                    if (end_cycle_flag) break;
                }
            }
```

```
    }
    else {
        close(pToChilde[0]);
        close(pToParent[1]);
        close(fd);
        int end_cycle_flag = 0;
        char strBuf[StrBufSize];
        char sigHand[sizeof(STOP_MES) / sizeof(char)] = {'\0'};
        while (!end_cycle_flag) {
            if (read(pToParent[0], sigHand, sizeof(sigHand)) == -1) {
                exitParent(READ_ERR, pToChilde[1], pToParent[0]);
            }
            else if (sigHand[0] == READY_TO_READ) {
                int number = read(STDIN_FILENO, strBuf, sizeof(strBuf));
                if (number == -1) {
                    exitParent(READ_ERR, pToChilde[1], pToParent[0]);
                }
                else if (write(pToChilde[1], strBuf, number) == -1) {
                    exitParent(WRITE_ERR, pToChilde[1], pToParent[0]);
                }
            }
            else if (sigHand[0] == STOP_MES) {
                close(pToChilde[1]);
                close(pToParent[0]);
                end_cycle_flag = 1;
            }
            else exitParent(CHILDE_ERROR, pToChilde[1], pToParent[0]);
        }
        wait(NULL);
    }
    return 0;
}
```

```
void exitParent(int code, int pToChilde, int pToParent) {
    close(pToChilde);
    close(pToParent);
    wait(NULL);
    exit(code);
}

void exitChilde(int code, int pToChilde, int pToParent, int fd, char message) {
    write(pToParent, &message, sizeof(message));
    close(pToChilde);
    close(pToParent);
    close(fd);
    exit(code);
}

void stopChilde(float divided, int pToChilde, int pToParent, int fd, char message) {
    close(pToChilde);
    char* result = floatToStr(divided);
    int len = strLen(result);
    if (write(fd, result, sizeof(char) * len) == -1) {
        free(result);
        exitChilde(WRITE_ERR, pToChilde, pToParent, fd, ERROR_MES);
    }
    free(result);
    if (write(pToParent, &STOP_MES, sizeof(STOP_MES)) == -1)
        exitChilde(WRITE_ERR, pToChilde, pToParent, fd, ERROR_MES);
    close(pToParent);
    close(fd);
}

char* floatToStr(float num){
    char* str = (char*)calloc(15, sizeof(char));
    sprintf(str, "%g\n", num);
```

```
    return str;
}

int strLen(char* str){
    char* ptr = str;
    for (;*ptr != '\0'; ++ptr);
    return ptr - str;
}
```

Протокол работы программы

Тестирование:

```
vnadez@dbnicknv:~/Рабочий стол/Projects/OS_LABS/LAB_1$ ./lab_1
txt.txt
10 10 10
12 334 -56 43 5654 3 3
123 324 3345
-234 2
32
-3
2
0
4 0
------------
in txt1.txt:
0.1
-3.73177e-10
0.000113492
-83
32
-3
2
0
```

```
vnadez@dbnicknv:~/Рабочий стол/Projects/OS_LABS/LAB_1$ ./lab_1

txt2.txt

67

-7 -7 -7 -7 -7 -7 -7 -7 -7 -7

0 0

vnadez@dbnicknv:~/Рабочий стол/Projects/OS_LABS/LAB_1$

--------

in txt2.txt:

67

1.73467e-07

0
```

Strace:

Ввод:

```
vnadez@dbnicknv:~/Рабочий стол/Projects/OS_LABS/LAB_1$ strace -f --
output=outStrace.txt ./lab_1

txt.txt

10 10 10 10

111 111 11 -1

-23 45 54 454 545 454

0 0
```

Вывод:

```
88999 execve("./lab_1", ["./lab_1"], 0x7ffe7090a9d0 /* 64 vars */) = 0

88999 brk(NULL)                           = 0x55d773428000

88999 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f2e3168e000

88999 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)

88999 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

88999 newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=92170, ...}, AT_EMPTY_PATH) = 0

88999 mmap(NULL, 92170, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2e31677000

88999 close(3)                            = 0

88999 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

88999 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"...,
832) = 832

88999 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
```

```
     88999 newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1922136, ...}, AT_EMPTY_PATH) =
0

     88999 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

     88999 mmap(NULL, 1970000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f2e31496000

     88999 mmap(0x7f2e314bc000, 1396736, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x26000) = 0x7f2e314bc000

     88999 mmap(0x7f2e31611000, 339968, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x17b000) = 0x7f2e31611000

     88999 mmap(0x7f2e31664000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ce000) = 0x7f2e31664000

     88999 mmap(0x7f2e3166a000, 53072, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f2e3166a000

     88999 close(3)                          = 0

     88999 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f2e31493000

     88999 arch_prctl(ARCH_SET_FS, 0x7f2e31493740) = 0

     88999 set_tid_address(0x7f2e31493a10)   = 88999

     88999 set_robust_list(0x7f2e31493a20, 24) = 0

     88999 rseq(0x7f2e31494060, 0x20, 0, 0x53053053) = 0

     88999 mprotect(0x7f2e31664000, 16384, PROT_READ) = 0

     88999 mprotect(0x55d773006000, 4096, PROT_READ) = 0

     88999 mprotect(0x7f2e316c0000, 8192, PROT_READ) = 0

     88999 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0

     88999 munmap(0x7f2e31677000, 92170)     = 0

     88999 read(0, "txt.txt\n", 15)          = 8

     88999 openat(AT_FDCWD, "txt.txt", O_WRONLY|O_CREAT|O_TRUNC, 0707) = 3

     88999 pipe2([4, 5], 0)                  = 0

     88999 pipe2([6, 7], 0)                  = 0

     88999 clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f2e31493a10) = 89002

     88999 close(4)                          = 0

     88999 close(7)                          = 0

     88999 close(3)                          = 0

     88999 read(6,  <unfinished ...>

     89002 set_robust_list(0x7f2e31493a20, 24) = 0
```

```
89002 close(5)                          = 0
89002 close(6)                          = 0
89002 write(7, "3", 1 <unfinished ...>
88999 <... read resumed>"3", 1)         = 1
89002 <... write resumed>)              = 1
88999 read(0,  <unfinished ...>
89002 read(4,  <unfinished ...>
88999 <... read resumed>"10 10 10 10\n", 50) = 12
88999 write(5, "10 10 10 10\n", 12)     = 12
89002 <... read resumed>"10 10 10 10\n", 50) = 12
88999 read(6,  <unfinished ...>
89002 getrandom("\x6b\x0e\x17\x24\xee\x0e\x0c\xb8", 8, GRND_NONBLOCK) = 8
89002 brk(NULL)                         = 0x55d773428000
89002 brk(0x55d773449000)               = 0x55d773449000
89002 write(3, "0.01\n", 5)              = 5
89002 write(7, "3", 1 <unfinished ...>
88999 <... read resumed>"3", 1)         = 1
89002 <... write resumed>)              = 1
88999 read(0,  <unfinished ...>
89002 read(4,  <unfinished ...>
88999 <... read resumed>"111 111 11 -1\n", 50) = 14
88999 write(5, "111 111 11 -1\n", 14)   = 14
89002 <... read resumed>"111 111 11 -1\n", 50) = 14
88999 read(6,  <unfinished ...>
89002 write(3, "-0.0909091\n", 11)      = 11
89002 write(7, "3", 1 <unfinished ...>
88999 <... read resumed>"3", 1)         = 1
89002 <... write resumed>)              = 1
88999 read(0,  <unfinished ...>
89002 read(4,  <unfinished ...>
88999 <... read resumed>"-23 45 54 454 545 454\n", 50) = 22
88999 write(5, "-23 45 54 454 545 454\n", 22) = 22
89002 <... read resumed>"-23 45 54 454 545 454\n", 50) = 22
```

```
     88999 read(6,  <unfinished ...>
     89002 write(3, "-6.2278e-11\n", 12)     = 12
     89002 write(7, "3", 1 <unfinished ...>
     88999 <... read resumed>"3", 1)         = 1
     89002 <... write resumed>)              = 1
     88999 read(0,  <unfinished ...>
     89002 read(4,  <unfinished ...>
     88999 <... read resumed>"0 0 \n", 50)   = 5
     88999 write(5, "0 0 \n", 5)             = 5
     89002 <... read resumed>"0 0 \n", 50)   = 5
     88999 read(6,  <unfinished ...>
     89002 close(4)                          = 0
     89002 write(3, "0\n", 2)                = 2
     89002 write(7, "1", 1 <unfinished ...>
     88999 <... read resumed>"1", 1)         = 1
     89002 <... write resumed>)              = 1
     88999 close(5 <unfinished ...>
     89002 close(7 <unfinished ...>
     88999 <... close resumed>)              = 0
     89002 <... close resumed>)              = 0
     88999 close(6 <unfinished ...>
     89002 close(3 <unfinished ...>
     88999 <... close resumed>)              = 0
     89002 <... close resumed>)              = 0
     88999 wait4(-1,  <unfinished ...>
     89002 exit_group(0)                     = ?
     89002 +++ exited with 0 +++
     88999 <... wait4 resumed>NULL, 0, NULL) = 89002
     88999 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=89002, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
     88999 exit_group(0)                     = ?
     88999 +++ exited with 0 +++
```

Вывод

При выполнении этой работы я получил практические навыки в управлении процессами посредством системных вызовов, и в организации межпроцессорного взаимодействия посредством каналов.