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: Павлов О.В.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 14.03.25

Москва, 2025

**Постановка задачи**

**Вариант 8.**

В файле записаны команды вида:«число число число<endline>». Дочерний процесс производит деление первого числа команда, на последующие числа в команде, а результат выводит в стандартный поток вывода. 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**

В данной лабораторной работе реализовано взаимодействие между родительским (parent.c) и дочерним (child.c) процессами с использованием механизма межпроцессного взаимодействия pipe(). Родительский процесс получает от пользователя имя файла и команду, передает их в дочерний процесс через канал. Дочерний процесс записывает данные в файл, выполняет арифметические операции и отправляет результат обратно родителю через другой канал.

Использованные системные вызовы:

* pid\_t fork(void); – создает дочерний процесс. Родительский и дочерний процессы выполняются независимо, но с общим начальным состоянием.
* int pipe(int pipefd[2]); – создает неименованный канал для передачи данных между процессами. Канал содержит два конца: pipefd[0] (чтение) и pipefd[1] (запись).
* int dup2(int oldfd, int newfd); – перенаправляет файловый дескриптор oldfd в newfd, что позволяет дочернему процессу читать из pipe1 и писать в pipe2.
* FILE \*fopen(const char \*pathname, const char \*mode); – открывает файл. В дочернем процессе файл создается для записи результата вычислений.
* size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream); – читает данные из потока stdin (в нашем случае, это перенаправленный pipe).
* size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb, FILE \*stream); – записывает данные в файл.
* int fclose(FILE \*stream); – закрывает открытый файловый поток.
* void perror(const char \*s); – выводит сообщение об ошибке, основанное на errno.
* int waitpid(pid\_t pid, int \*wstatus, int options); – позволяет родителю дождаться завершения дочернего процесса.
* int close(int fd); – закрывает файловый дескриптор, освобождая ресурсы.

#### ****Родительский процесс (****parent.c****):****

1. Создает два канала (pipe1 для передачи данных в child, pipe2 для получения результата от child).
2. Выполняет fork() для создания дочернего процесса.
3. В дочернем процессе:
   * Перенаправляет STDIN на pipe1[0] (чтение из канала).
   * Перенаправляет STDOUT на pipe2[1] (запись результата в канал).
   * Закрывает ненужные файловые дескрипторы.
   * Запускает execl() для выполнения child.c.
4. В родительском процессе:
   * Закрывает ненужные файловые дескрипторы.
   * Запрашивает у пользователя имя файла и команду (список чисел).
   * Передает эти данные в pipe1[1] (запись в канал).
   * Ждет завершения дочернего процесса (waitpid()).
   * Считывает результат из pipe2[0] и выводит на экран.

#### ****Дочерний процесс (****child.c****):****

1. Читает из stdin (канал pipe1) имя файла.
2. Открывает файл для записи (fopen()).
3. Читает из stdin команду (список чисел).
4. Разбирает команду, выполняет последовательное деление чисел.
   * Если встречается деление на ноль, записывает ошибку в файл и выводит сообщение об ошибке.
5. Записывает результат в файл и stdout (который перенаправлен в pipe2).
6. Закрывает файл и завершает работу.

**Код программы**

**parent.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/wait.h>

#include <errno.h>

#define BUFFER\_SIZE 1024

int main()

{

int pipe1[2], pipe2[2];

if (pipe(pipe1) == -1 || pipe(pipe2) == -1)

{

perror("Pipe creation failed");

return 1;

}

pid\_t pid = fork();

if (pid == -1)

{

perror("Fork failed");

return 1;

}

if (pid == 0)

{

close(pipe1[1]);

close(pipe2[0]);

dup2(pipe1[0], STDIN\_FILENO);

dup2(pipe2[1], STDOUT\_FILENO);

close(pipe1[0]);

close(pipe2[1]);

printf("Starting child process...\n"); // Отладочное сообщение

execl("./child", "child", (char \*)NULL);

perror("Execl failed");

exit(1); // Добавьте выход из дочернего процесса в случае ошибки

return 1;

}

else

{

close(pipe1[0]);

close(pipe2[1]);

char fileName[BUFFER\_SIZE];

printf("Enter filename: ");

scanf("%s", fileName);

write(pipe1[1], fileName, strlen(fileName));

write(pipe1[1], "\n", 1);

char command[BUFFER\_SIZE];

printf("Enter command (numbers separated by spaces): ");

getchar();

fgets(command, BUFFER\_SIZE, stdin);

write(pipe1[1], command, strlen(command));

write(pipe1[1], "\n", 1);

waitpid(pid, NULL, 0);

close(pipe1[1]);

char buffer[BUFFER\_SIZE];

read(pipe2[0], buffer, BUFFER\_SIZE);

printf("Child process output: %s\n", buffer);

close(pipe2[0]);

}

return 0;

}

**child.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <errno.h>

#include <fcntl.h>

#define BUFFER\_SIZE 1024

int main()

{

char filename[BUFFER\_SIZE];

if (fgets(filename, BUFFER\_SIZE, stdin) == NULL)

{

perror("Failed to read filename");

return 1;

}

filename[strcspn(filename, "\n")] = '\0';

FILE \*file = fopen(filename, "w");

if (file == NULL)

{

perror("Failed to open file");

return 1;

}

char command[BUFFER\_SIZE];

if (fgets(command, BUFFER\_SIZE, stdin) == NULL)

{

perror("Failed to read command");

fclose(file);

return 1;

}

printf("Received command: %s\n", command); // Отладочное сообщение

int numbers[BUFFER\_SIZE], num\_count = 0;

char \*token = strtok(command, " ");

while (token != NULL)

{

numbers[num\_count++] = atoi(token);

token = strtok(NULL, " ");

}

if (num\_count < 2)

{

fprintf(file, "Error: Not enough numbers in command.\n");

fclose(file);

return 1;

}

int result = numbers[0];

for (int i = 1; i < num\_count; i++)

{

if (numbers[i] == 0)

{

fprintf(stderr, "Division by zero encountered. Terminating.\n");

fprintf(file, "Error: Division by zero.\n");

fclose(file);

return 1;

}

result /= numbers[i];

}

fprintf(file, "Result: %d\n", result);

fclose(file);

printf("Result: %d\n", result);

return 0;

}

**Протокол работы программы**

./parent

Enter filename: output.txt

Enter command (numbers separated by spaces): 500 2 5

Child process output: Received command: 500 2 5

Result: 50

./parent

Enter filename: output.txt

Enter command (numbers separated by spaces): 100 0 1

Division by zero encountered. Terminating.

Child process output: Received command: 100 0 1

./parent

Enter filename: output.txt

Enter command (numbers separated by spaces): 555 1000

Child process output: Received command: 555 1000

Result: 0

**STRACE**

execve("./parent", ["./parent"], 0xffffec9af5d8 /\* 32 vars \*/) = 0

brk(NULL) = 0xaaaac1a8a000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xffffbe146000

faccessat(AT\_FDCWD, "/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=15047, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 15047, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0xffffbe142000

close(3) = 0

openat(AT\_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\340u\2\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=1637400, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 1805928, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xffffbdf58000

mmap(0xffffbdf60000, 1740392, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0) = 0xffffbdf60000

munmap(0xffffbdf58000, 32768) = 0

munmap(0xffffbe109000, 32360) = 0

mprotect(0xffffbe0e8000, 61440, PROT\_NONE) = 0

mmap(0xffffbe0f7000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x187000) = 0xffffbe0f7000

mmap(0xffffbe0fd000, 48744, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0xffffbe0fd000

close(3) = 0

set\_tid\_address(0xffffbe146f50) = 71655

set\_robust\_list(0xffffbe146f60, 24) = 0

rseq(0xffffbe147620, 0x20, 0, 0xd428bc00) = 0

mprotect(0xffffbe0f7000, 16384, PROT\_READ) = 0

mprotect(0xaaaabdf41000, 4096, PROT\_READ) = 0

mprotect(0xffffbe14b000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0xffffbe142000, 15047) = 0

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

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

**clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 71656 attached**

**<unfinished ...>**

[pid 71656] set\_robust\_list(0xffffbe146f60, 24 <unfinished ...>

[pid 71655] <... clone resumed>, child\_tidptr=0xffffbe146f50) = 71656

[pid 71656] <... set\_robust\_list resumed>) = 0

[pid 71655] close(3) = 0

[pid 71656] close(4 <unfinished ...>

[pid 71655] close(6 <unfinished ...>

[pid 71656] <... close resumed>) = 0

[pid 71655] <... close resumed>) = 0

[pid 71656] close(5 <unfinished ...>

[pid 71655] newfstatat(1, "", <unfinished ...>

[pid 71656] <... close resumed>) = 0

[pid 71655] <... newfstatat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x1), ...}, AT\_EMPTY\_PATH) = 0

[pid 71656] dup3(3, 0, 0 <unfinished ...>

[pid 71655] getrandom( <unfinished ...>

[pid 71656] <... dup3 resumed>) = 0

[pid 71655] <... getrandom resumed>"\x0b\xfe\xe6\x4d\xe7\xaa\xcc\x05", 8, GRND\_NONBLOCK) = 8

[pid 71656] dup3(6, 1, 0 <unfinished ...>

[pid 71655] brk(NULL <unfinished ...>

[pid 71656] <... dup3 resumed>) = 1

[pid 71655] <... brk resumed>) = 0xaaaac1a8a000

[pid 71656] close(3 <unfinished ...>

[pid 71655] brk(0xaaaac1aab000 <unfinished ...>

[pid 71656] <... close resumed>) = 0

[pid 71655] <... brk resumed>) = 0xaaaac1aab000

[pid 71656] close(6 <unfinished ...>

[pid 71655] newfstatat(0, "", <unfinished ...>

[pid 71656] <... close resumed>) = 0

[pid 71655] <... newfstatat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x1), ...}, AT\_EMPTY\_PATH) = 0

[pid 71656] newfstatat(1, "", <unfinished ...>

[pid 71655] write(1, "Enter filename: ", 16 <unfinished ...>

[pid 71656] <... newfstatat resumed>{st\_mode=S\_IFIFO|0600, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

Enter filename: [pid 71655] <... write resumed>) = 16

[pid 71656] getrandom( <unfinished ...>

[pid 71655] read(0, <unfinished ...>

[pid 71656] <... getrandom resumed>"\x1f\x6e\xb0\x45\x7c\x04\x2d\x89", 8, GRND\_NONBLOCK) = 8

[pid 71656] brk(NULL) = 0xaaaac1a8a000

[pid 71656] brk(0xaaaac1aab000) = 0xaaaac1aab000

[pid 71656] execve("./child", ["child"], 0xfffff229fa68 /\* 32 vars \*/) = 0

[pid 71656] brk(NULL) = 0xaaaadc275000

[pid 71656] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xffff98351000

[pid 71656] faccessat(AT\_FDCWD, "/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 71656] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 71656] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=15047, ...}, AT\_EMPTY\_PATH) = 0

[pid 71656] mmap(NULL, 15047, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0xffff9834d000

[pid 71656] close(3) = 0

[pid 71656] openat(AT\_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 71656] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\340u\2\0\0\0\0\0"..., 832) = 832

[pid 71656] newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=1637400, ...}, AT\_EMPTY\_PATH) = 0

[pid 71656] mmap(NULL, 1805928, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xffff98163000

[pid 71656] mmap(0xffff98170000, 1740392, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0) = 0xffff98170000

[pid 71656] munmap(0xffff98163000, 53248) = 0

[pid 71656] munmap(0xffff98319000, 11880) = 0

[pid 71656] mprotect(0xffff982f8000, 61440, PROT\_NONE) = 0

[pid 71656] mmap(0xffff98307000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x187000) = 0xffff98307000

[pid 71656] mmap(0xffff9830d000, 48744, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0xffff9830d000

[pid 71656] close(3) = 0

[pid 71656] set\_tid\_address(0xffff98351f50) = 71656

[pid 71656] set\_robust\_list(0xffff98351f60, 24) = 0

[pid 71656] rseq(0xffff98352620, 0x20, 0, 0xd428bc00) = 0

[pid 71656] mprotect(0xffff98307000, 16384, PROT\_READ) = 0

[pid 71656] mprotect(0xaaaabc841000, 4096, PROT\_READ) = 0

[pid 71656] mprotect(0xffff98356000, 8192, PROT\_READ) = 0

[pid 71656] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 71656] munmap(0xffff9834d000, 15047) = 0

[pid 71656] newfstatat(0, "", {st\_mode=S\_IFIFO|0600, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

[pid 71656] getrandom("\x1c\x37\xfc\x65\xb4\x3b\x42\x9b", 8, GRND\_NONBLOCK) = 8

[pid 71656] brk(NULL) = 0xaaaadc275000

[pid 71656] brk(0xaaaadc296000) = 0xaaaadc296000

[pid 71656] read(0, <unfinished ...>

[pid 71655] <... read resumed>0xaaaac1a8a6b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71656] <... read resumed>0xaaaadc2752a0, 4096) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71655] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71656] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71655] read(0, <unfinished ...>

[pid 71656] read(0, <unfinished ...>

[pid 71655] <... read resumed>0xaaaac1a8a6b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71656] <... read resumed>0xaaaadc2752a0, 4096) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71655] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71656] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71655] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71656] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71655] read(0, <unfinished ...>

[pid 71656] read(0, <unfinished ...>

[pid 71655] <... read resumed>0xaaaac1a8a6b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71656] <... read resumed>0xaaaadc2752a0, 4096) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71655] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71655] read(0, <unfinished ...>

[pid 71656] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71655] <... read resumed>0xaaaac1a8a6b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71656] read(0, <unfinished ...>

[pid 71655] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71656] <... read resumed>0xaaaadc2752a0, 4096) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71655] read(0, <unfinished ...>

[pid 71656] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71656] read(0, <unfinished ...>

[pid 71655] <... read resumed>0xaaaac1a8a6b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71656] <... read resumed>0xaaaadc2752a0, 4096) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71655] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71656] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71655] read(0, <unfinished ...>

[pid 71656] read(0, <unfinished ...>

[pid 71655] <... read resumed>0xaaaac1a8a6b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71656] <... read resumed>0xaaaadc2752a0, 4096) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71655] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71656] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71655] read(0, <unfinished ...>

[pid 71656] read(0, <unfinished ...>

[pid 71655] <... read resumed>0xaaaac1a8a6b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71656] <... read resumed>0xaaaadc2752a0, 4096) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71655] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71656] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71655] read(0, <unfinished ...>

[pid 71656] read(0, <unfinished ...>

[pid 71655] <... read resumed>0xaaaac1a8a6b0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71656] <... read resumed>0xaaaadc2752a0, 4096) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

[pid 71655] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71656] --- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

[pid 71655] read(0, <unfinished ...>

[pid 71656] read(0, output.txt

<unfinished ...>

[pid 71655] <... read resumed>"output.txt\n", 1024) = 11

[pid 71655] write(4, "output.txt", 10) = 10

[pid 71656] <... read resumed>"output.txt", 4096) = 10

[pid 71655] write(4, "\n", 1 <unfinished ...>

[pid 71656] read(0, <unfinished ...>

[pid 71655] <... write resumed>) = 1

[pid 71656] <... read resumed>"\n", 4096) = 1

[pid 71655] write(1, "Enter command (numbers separated"..., 45 <unfinished ...>

[pid 71656] openat(AT\_FDCWD, "output.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0666Enter command (numbers separated by spaces): <unfinished ...>

[pid 71655] <... write resumed>) = 45

[pid 71655] read(0, <unfinished ...>

[pid 71656] <... openat resumed>) = 3

[pid 71656] read(0, 500 2 5

<unfinished ...>

[pid 71655] <... read resumed>"500 2 5\n", 1024) = 8

[pid 71655] write(4, "500 2 5\n", 8) = 8

[pid 71656] <... read resumed>"500 2 5\n", 4096) = 8

[pid 71655] write(4, "\n", 1 <unfinished ...>

[pid 71656] newfstatat(1, "", <unfinished ...>

[pid 71655] <... write resumed>) = 1

[pid 71656] <... newfstatat resumed>{st\_mode=S\_IFIFO|0600, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

[pid 71655] wait4(71656, <unfinished ...>

[pid 71656] newfstatat(3, "", {st\_mode=S\_IFREG|0664, st\_size=0, ...}, AT\_EMPTY\_PATH) = 0

[pid 71656] write(3, "Result: 50\n", 11) = 11

[pid 71656] close(3) = 0

[pid 71656] write(1, "Received command: 500 2 5\n\nResul"..., 38) = 38

[pid 71656] exit\_group(0) = ?

[pid 71656] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 71656

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=71656, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

close(4) = 0

read(5, "Received command: 500 2 5\n\nResul"..., 1024) = 38

write(1, "Child process output: Received c"..., 60Child process output: Received command: 500 2 5

Result: 50

) = 60

write(1, "\n", 1

) = 1

close(5) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В ходе данной лабораторной работы были изучены и применены механизмы межпроцессного взаимодействия с использованием каналов (pipe) и создания процессов (fork). Работа показала важность обработки ошибок и корректного завершения процессов для предотвращения утечек ресурсов. Также была продемонстрирована необходимость проверки входных данных, особенно при выполнении арифметических операций, таких как деление. Данный опыт позволяет глубже понять основы работы операционной системы и взаимодействие процессов, что является важным навыком при разработке системного и прикладного программного обеспечения.