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: Матвеев Семен Романович

Преподаватель: Миронов Евгений Сергеевич

Оценка:

Дата: 10.03.25

Москва, 2025

# Постановка задачи

## Вариант 6.

В файле записаны команды вида: «число число число». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип int. Количество чисел может быть произвольным.

Родительский процесс создает один дочерний процесс Child1. Перенаправление стандартных потоков ввода-вывода показано на картинке варианта: родительский процесс открывает файл, передает выполнение дочернему. Дочерний процесс получает на поток ввода данные из файла, обрабатывает эти данные, после чего по каналу передает результат своей работы на поток ввода родительского процесса.

# Общий метод и алгоритм решения

Использованные системные вызовы:

* **fork()** – создает дочерний процесс (копия родительского).
* **pipe()** – создает канал (pipe) для обмена данными между процессами.
* **exit()** – завершает выполнение процесса.
* **execl()** – заменяет текущий процесс новым (загрузка программы).
* **close()** – закрывает файловый дескриптор (например, конец канала).
* **dup2()** – перенаправляет дескриптор (например, pipe на stdin/stdout).
* **perror()** – выводит сообщение об ошибке (на основе errno).
* **wait()** – ожидает завершения дочернего процесса.

**Алгоритм работы программы Родительский процесс**

1. **Родительский процесс (parent.c)**
   1. **Создание каналов:**
      * Родительский процесс создает один канал (pipe1) для обмена данными между процессами.

## Создание первого дочернего процесса (child1):

* + - Родительский процесс создает дочерний процесс с помощью fork().
    - В дочернем процессе:
      * Закрываются ненужные концы каналов — конец записи pipe1.
      * Стандартный ввод (stdin) перенаправляется на чтение из fd, а стандартный вывод (stdout) — на запись в конец чтения pipe1.
      * Закрывает оставшиеся каналы — конец чтения pipe1 и fd.
      * Выполняется программа child1, которая будет читать данные из fd, обрабатывать их и записывать результат в pipe1.

## Родительский процесс продолжает работу:

* + - В дочернем процессе:
      * Закрываются ненужные концы каналов - fd и конца записи pipe1.
      * Стандартный ввод (stdin) перенаправляется на конец записи pipe1.
      * Закрывает оставшиеся каналы — конец записи pipe1.
      * Принимается на вход результат работы дочернего процесса и выводится в консоль через stdout

## Завершение работы:

* + - Родительский процесс ожидает завершения дочернего процесса с помощью

wait().

* + - После завершения всех процессов выводится сообщение о завершении работы.

## Первый дочерний процесс (child.c)

1. Читает строку из стандартного ввода (перенаправленный fd).
2. Считает сумму чисел в строке
3. Отправляет исходную строку и результат (сумму чисел) в стандартный вывод (перенаправленный pipe1).
4. Завершает работу после обработки всех строк.

# Код программы

**parent.c**

#include "fcntl.h"

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

#include "sys/wait.h"

#include "unistd.h"

#define MAX\_FILEPATH 256

#define BUFFER\_SIZE 4096

int main() {

char filepath[MAX\_FILEPATH];

printf("Введите путь к файлу:\n");

scanf("%s", filepath);

int fd = open(filepath, O\_RDONLY);

if (fd == -1) {

printf("Ошибка open\n");

exit(1);

}

int pipe1[2];

pid\_t child;

if (pipe(pipe1) == -1) {

perror("Ошибка при создании канала\n");

exit(1);

}

if ((child = fork()) == -1) {

perror("Ошибка при создании дочернего процесса\n");

exit(1);

}

if (child == 0) {

close(pipe1[0]);

dup2(fd, STDIN\_FILENO);

dup2(pipe1[1], STDOUT\_FILENO);

close(fd);

close(pipe1[1]);

execl("./child", "child", NULL);

perror("Ошибка execl для child\n");

exit(1);

}

close(fd);

close(pipe1[1]);

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

close(pipe1[0]);

char line[BUFFER\_SIZE];

while (fgets(line, BUFFER\_SIZE, stdin) != NULL) {

write(STDOUT\_FILENO, line, strlen(line));

}

wait(NULL);

return 0;

}

**child.c**

#include "stdio.h"

#include "string.h"

#include "unistd.h"

#define BUFFER\_SIZE 4096

long long get\_line\_sum(char\* line) {

// можно и strol использовать, если быть особо ленивым

long long line\_sum = 0;

int number = 0;

int digit\_power = 1;

int i = strlen(line);

while (i >= 0) {

if ((line[i] >= '0') && (line[i] <= '9')) {

number += (line[i] - '0') \* digit\_power;

digit\_power \*= 10;

} else {

if (line[i] == '-') {

number \*= -1;

}

line\_sum += number;

number = 0;

digit\_power = 1;

}

i--;

}

if (number != 0) {

line\_sum += number;

}

return line\_sum;

}

int main() {

char line[BUFFER\_SIZE];

while (fgets(line, BUFFER\_SIZE, stdin) != NULL) {

long long line\_sum = get\_line\_sum(line);

char answer[32];

sprintf(answer, " -> %lld\n", line\_sum);

write(STDOUT\_FILENO, line, strlen(line) - 1);

write(STDOUT\_FILENO, answer, strlen(answer));

}

return 0;

}

# Протокол работы программы

## Тест:

**Strace:**

/parent

Введите путь к файлу:

file.txt

17 21 67 -> 105

12 50 0 -> 62

7 -21 314 -> 300

-201 77 -> -124

strace -f ./parent

execve("./parent", ["./parent"], 0x7fff7238ba18 /\* 77 vars \*/) = 0

brk(NULL) = 0x55e1b9805000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffbf01ed80) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f9b1f537000

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=166595, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 166595, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f9b1f50e000

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\0\315A\vq\17\17\tLh2\355\331Y1\0m"..., 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) = 0x7f9b1f2e5000

mprotect(0x7f9b1f30d000, 2023424, PROT\_NONE) = 0

mmap(0x7f9b1f30d000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f9b1f30d000

mmap(0x7f9b1f4a2000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f9b1f4a2000

mmap(0x7f9b1f4fb000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7f9b1f4fb000

mmap(0x7f9b1f501000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f9b1f501000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f9b1f2e2000

arch\_prctl(ARCH\_SET\_FS, 0x7f9b1f2e2740) = 0

set\_tid\_address(0x7f9b1f2e2a10) = 455489

set\_robust\_list(0x7f9b1f2e2a20, 24) = 0

rseq(0x7f9b1f2e30e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f9b1f4fb000, 16384, PROT\_READ) = 0

mprotect(0x55e179bc7000, 4096, PROT\_READ) = 0

mprotect(0x7f9b1f571000, 8192, PROT\_READ) = 0

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

munmap(0x7f9b1f50e000, 166595) = 0

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x5), ...}, AT\_EMPTY\_PATH) = 0

getrandom("\xba\x60\xf6\x4b\x83\xdc\xd2\x76", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55e1b9805000

brk(0x55e1b9826000) = 0x55e1b9826000

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\277\321\203\321\202\321\214 \320\272 \321\204\320\260\320"..., 39Введите путь к файлу:

) = 39

newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x5), ...}, AT\_EMPTY\_PATH) = 0

read(0, file.txt

"file.txt\n", 1024) = 9

openat(AT\_FDCWD, "file.txt", O\_RDONLY) = 3

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

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 455630 attached

, child\_tidptr=0x7f9b1f2e2a10) = 455630

[pid 455489] close(3 <unfinished ...>

[pid 455630] set\_robust\_list(0x7f9b1f2e2a20, 24 <unfinished ...>

[pid 455489] <... close resumed>) = 0

[pid 455489] close(5 <unfinished ...>

[pid 455630] <... set\_robust\_list resumed>) = 0

[pid 455489] <... close resumed>) = 0

[pid 455489] dup2(4, 0) = 0

[pid 455630] close(4 <unfinished ...>

[pid 455489] close(4) = 0

[pid 455630] <... close resumed>) = 0

[pid 455489] write(1, "\n", 1 <unfinished ...>

[pid 455630] dup2(3, 0

<unfinished ...>

[pid 455489] <... write resumed>) = 1

[pid 455630] <... dup2 resumed>) = 0

[pid 455489] read(0, <unfinished ...>

[pid 455630] dup2(5, 1) = 1

[pid 455630] close(3) = 0

[pid 455630] close(5) = 0

[pid 455630] execve("./child", ["child"], 0x7fffbf01ef58 /\* 77 vars \*/) = 0

[pid 455630] brk(NULL) = 0x55988f0a3000

[pid 455630] arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffedc7b0620) = -1 EINVAL (Invalid argument)

[pid 455630] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe6e831d000

[pid 455630] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 455630] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 455630] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=166595, ...}, AT\_EMPTY\_PATH) = 0

[pid 455630] mmap(NULL, 166595, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fe6e82f4000

[pid 455630] close(3) = 0

[pid 455630] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 455630] 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

[pid 455630] 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

[pid 455630] 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

[pid 455630] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\315A\vq\17\17\tLh2\355\331Y1\0m"..., 68, 896) = 68

[pid 455630] newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2220400, ...}, AT\_EMPTY\_PATH) = 0

[pid 455630] 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

[pid 455630] mmap(NULL, 2264656, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe6e80cb000

[pid 455630] mprotect(0x7fe6e80f3000, 2023424, PROT\_NONE) = 0

[pid 455630] mmap(0x7fe6e80f3000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fe6e80f3000

[pid 455630] mmap(0x7fe6e8288000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fe6e8288000

[pid 455630] mmap(0x7fe6e82e1000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7fe6e82e1000

[pid 455630] mmap(0x7fe6e82e7000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fe6e82e7000

[pid 455630] close(3) = 0

[pid 455630] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe6e80c8000

[pid 455630] arch\_prctl(ARCH\_SET\_FS, 0x7fe6e80c8740) = 0

[pid 455630] set\_tid\_address(0x7fe6e80c8a10) = 455630

[pid 455630] set\_robust\_list(0x7fe6e80c8a20, 24) = 0

[pid 455630] rseq(0x7fe6e80c90e0, 0x20, 0, 0x53053053) = 0

[pid 455630] mprotect(0x7fe6e82e1000, 16384, PROT\_READ) = 0

[pid 455630] mprotect(0x559860ca5000, 4096, PROT\_READ) = 0

[pid 455630] mprotect(0x7fe6e8357000, 8192, PROT\_READ) = 0

[pid 455630] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 455630] munmap(0x7fe6e82f4000, 166595) = 0

[pid 455630] newfstatat(0, "", {st\_mode=S\_IFREG|0664, st\_size=35, ...}, AT\_EMPTY\_PATH) = 0

[pid 455630] getrandom("\xdc\xc0\x28\xb5\xef\x7f\xb1\x43", 8, GRND\_NONBLOCK) = 8

[pid 455630] brk(NULL) = 0x55988f0a3000

[pid 455630] brk(0x55988f0c4000) = 0x55988f0c4000

[pid 455630] read(0, "17 21 67\n12 50 0\n7 -21 314\n-201 "..., 4096) = 35

[pid 455630] write(1, "17 21 67", 8) = 8

[pid 455630] write(1, " -> 105\n", 8 <unfinished ...>

[pid 455489] <... read resumed>"17 21 67", 1024) = 8

[pid 455630] <... write resumed>) = 8

[pid 455489] read(0, <unfinished ...>

[pid 455630] write(1, "12 50 0", 7 <unfinished ...>

[pid 455489] <... read resumed>" -> 105\n", 1024) = 8

[pid 455630] <... write resumed>) = 7

[pid 455489] write(1, "17 21 67 -> 105\n", 16 <unfinished ...>

[pid 455630] write(1, " -> 62\n", 717 21 67 -> 105

<unfinished ...>

[pid 455489] <... write resumed>) = 16

[pid 455630] <... write resumed>) = 7

[pid 455489] read(0, <unfinished ...>

[pid 455630] write(1, "7 -21 314", 9 <unfinished ...>

[pid 455489] <... read resumed>"12 50 0 -> 62\n", 1024) = 14

[pid 455630] <... write resumed>) = 9

[pid 455489] write(1, "12 50 0 -> 62\n", 14 <unfinished ...>

[pid 455630] write(1, " -> 300\n", 812 50 0 -> 62

<unfinished ...>

[pid 455489] <... write resumed>) = 14

[pid 455630] <... write resumed>) = 8

[pid 455489] read(0, <unfinished ...>

[pid 455630] write(1, "-201 77", 7 <unfinished ...>

[pid 455489] <... read resumed>"7 -21 314 -> 300\n", 1024) = 17

[pid 455630] <... write resumed>) = 7

[pid 455489] write(1, "7 -21 314 -> 300\n", 17 <unfinished ...>

7 -21 314 -> 300

[pid 455630] write(1, " -> -124\n", 9 <unfinished ...>

[pid 455489] <... write resumed>) = 17

[pid 455630] <... write resumed>) = 9

[pid 455489] read(0, <unfinished ...>

[pid 455630] read(0, <unfinished ...>

[pid 455489] <... read resumed>"-201 77 -> -124\n", 1024) = 16

[pid 455630] <... read resumed>"", 4096) = 0

[pid 455489] write(1, "-201 77 -> -124\n", 16 <unfinished ...>

[pid 455630] exit\_group(0-201 77 -> -124

<unfinished ...>

[pid 455489] <... write resumed>) = 16

[pid 455630] <... exit\_group resumed>) = ?

[pid 455489] read(0, "", 1024) = 0

[pid 455630] +++ exited with 0 +++

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=455630, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

wait4(-1, NULL, 0, NULL) = 455630

exit\_group(0) = ?

+++ exited with 0 +++

# Вывод

Эта лабораторная работа позволила мне лучше понять механизмы межпроцессного взаимодействия в операционных системах, в частности, работу с каналами (pipes) и

перенаправлением ввода-вывода. Я научился создавать дочерние процессы с помощью `fork()` и управлять их взаимодействием через каналы, что является важным аспектом разработки

многозадачных приложений. Работа также помогла закрепить навыки работы с системными

вызовами, такими как `pipe()`, `dup2()`, и `execl()`, что будет полезно для дальнейшего изучения операционных систем. В целом, лабораторная работа была интересной и познавательной, хотя потребовала внимательности при отладке взаимодействия процессов.