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-216Б-24

Студент: Ходаков Павел

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 23.10.2025

Москва, 2025

Постановка задачи

Вариант 7.

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода.

Стандартный поток вывода дочернего процесса перенаправляется в pipe1. Родительский процесс читает из pipe1 и прочитанное выводит в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами.

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

Общий метод и алгоритм решения

Использованные системные вызовы:

- int open(const char *pathname, int flags, mode_t mode) – возвращает файловый дескриптор
- ssize_t read(int fd, void *buf, size_t count) – возвращает число реально прочитанных байт
- ssize_t write(int fd, const void *buf, size_t count) – возвращает число реально записанных байт
- int close(int fd) – закрывает объект по файловому дескриптору
- int dup2(int oldfd, int newfd) – дублирует файловый дескриптор, потом newfd указывает туда же, куда и oldfd
- int pipe(pipefd[2]) – создаёт неименованный канал
- pid_t fork(void); – создает дочерний процесс.
- int execl(const char *path, const char *arg, (char *) NULL – заменяет код текущего процесса новой программой
- pid_t waitpid(pid, &status, options) – ожидает завершения процесса с данным pid
- void exit(status) – завершает процесс с очисткой
- void _exit(status) – завершает процесс без очистки

В рамках лабораторной работы я изучил теоретический материал (материалы лекций и примеры кода, материалы из интернета по системным вызовам).

Родительский процесс:

1. Посимвольно считывает из стандартного ввода имя файла до конца строки или конца ввода.
2. Открывает указанный файл на чтение.
3. Создаёт неименованный канал (pipe) для обмена данными с дочерним процессом.
4. Порождает дочерний процесс с помощью fork.
5. После fork:
   закрывает ненужный конец канала;
   читает данные, поступающие из канала от дочернего процесса;
   выводит их в собственный стандартный вывод.
6. Ожидает завершения дочернего процесса (waitpid) и корректно завершает работу.

Дочерний процесс:

1. Перенаправляет стандартный ввод на открытый файл, а стандартный вывод — на конец для записи в канал.

2. Закрывает лишние файловые дескрипторы.

3. Вызывает другую программу через execl.

4. Эта программа:

посимвольно считывает строку из стандартного ввода (теперь это файл);

парсит строку как неизвестное количество чисел с плавающей точкой, разделённых пробелами;

вычисляет их сумму;

записывает результат в стандартный вывод (который перенаправлен в канал).

Код программы

parent.cpp

```
#include <cstdlib>

#include <fcntl.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <string.h>

#include <stdio.h>


#include <string>


int main() {

    const size_t BUF_SIZE = 4096;

    std::string filename;

    char buf[BUF_SIZE];


    while (true) {

        ssize_t n = read(STDIN_FILENO, buf, 1);

        if (n < 0) {

            if (errno == EINTR) {

                continue;

            }
```

```
        perror("read");
        exit(EXIT_FAILURE);
    }
    if (n == 0) {
        break; // got EOF
    }

    if (buf[0] == '\n') {
        break;
    }
    filename.push_back(buf[0]);
    if (filename.size() >= BUF_SIZE) {
        const char *msg = "Filename is too long.\n";
        write(STDERR_FILENO, msg, strlen(msg));
        exit(EXIT_FAILURE);
    }
}

if (filename.empty()) {
    const char *msg = "Please provide filename.\n";
    write(STDERR_FILENO, msg, strlen(msg));
    exit(EXIT_SUCCESS);
}

int fd_in = open(filename.c_str(), O_RDONLY);
if (fd_in < 0) {
    perror("open");
    exit(EXIT_FAILURE);
}

int pipefd[2];
if (pipe(pipefd) != 0) {
    perror("pipe");
```

```
    close(fd_in);
    exit(EXIT_FAILURE);
}

const pid_t pid = fork();
switch (pid) {
case -1: {
    perror("fork");
    close(fd_in);
    close(pipefd[0]);
    close(pipefd[1]);
    exit(EXIT_FAILURE);
} break;

case 0: {
    // child
    if (dup2(fd_in, STDIN_FILENO) < 0) {
        _exit(EXIT_FAILURE);
    }
    if (dup2(pipefd[1], STDOUT_FILENO) < 0) {
        _exit(EXIT_FAILURE);
    }
    if (dup2(pipefd[1], STDERR_FILENO) < 0) {
        _exit(EXIT_FAILURE);
    }

    close(fd_in);
    close(pipefd[0]);
    close(pipefd[1]);

    execl("./child", "./child", (char*)NULL);
    perror("execl");
    _exit(EXIT_FAILURE);
```

```
} break;


default: {
    // parent
    close(fd_in);
    close(pipefd[1]);

    const size_t RBUF = 4096;
    char buf[RBUF];
    bool had_error = false;
    while (true) {
        ssize_t r = read(pipefd[0], buf, RBUF);
        if (r < 0) {
            if (errno == EINTR) {
                continue;
            }
            perror("read");
            had_error = true;
            break;
        }
        if (r == 0) {
            break; // got EOF
        }

        ssize_t woff = 0;
        while (woff < r) {
            ssize_t w = write(STDOUT_FILENO, buf + woff, r - woff);
            if (w < 0) {
                if (errno == EINTR) {
                    continue;
                }
                perror("write");
                had_error = true;
```

```
                    break;
                }
                woff += w;
            }
            if (had_error) {
                break;
            }
        }

        close(pipefd[0]);

        int status = 0;
        if (waitpid(pid, &status, 0) < 0) {
            perror("waitpid");
            exit(EXIT_FAILURE);
        }

        if (had_error) {
            exit(EXIT_FAILURE);
        }
        exit(EXIT_SUCCESS);
    } break;
    }
}
```

child.cpp

```
#include <unistd.h>

#include <string>
#include <vector>
#include <cstring>
#include <cstdlib>

int main() {
```

```
std::string line;
char ch;
while (true) {
    ssize_t n = read(STDIN_FILENO, &ch, 1);
    if (n < 0) {
        if (errno == EINTR) {
            continue;
        }
        perror("read");
        exit(EXIT_FAILURE);
    }
    if (n == 0) {
        // EOF
        if (line.empty()) {
            break;
        }
    } else {
        if (ch != '\n') {
            line.push_back(ch);
            continue;
        }
    }

    std::vector<float> nums;
    std::string cur;
    for (size_t i = 0; i < line.size(); ++i) {
        char c = line[i];
        if (c == ' ') {
            if (!cur.empty()) {
                nums.push_back(std::atof(cur.c_str()));
                cur.clear();
            }
        } else {
```

```
        cur.push_back(c);
    }
}
if (!cur.empty()) {
    nums.push_back(std::atof(cur.c_str()));
    cur.clear();
}

if (!nums.empty()) {
    float sum = 0;
    for (size_t i = 0; i < nums.size(); ++i) {
        sum += nums[i];
    }
    std::string out = std::to_string(sum) + "\n";
    size_t woff = 0;
    while (woff < out.size()) {
        ssize_t w = write(STDOUT_FILENO, out.c_str() + woff,
                          out.size() - woff);
        if (w < 0) {
            if (errno == EINTR) {
                continue;
            }
            perror("write");
            return EXIT_FAILURE;
        }
        woff += w;
    }
}

line.clear();
if (n == 0) {
    break;
}
```

```
    }


    exit(EXIT_SUCCESS);
}
```

Протокол работы программы

Вывод strace:

```
        execve("./parent", ["./parent"], 0x7ffd71f21730 /* 33 vars */) = 0
             brk(NULL)                                  = 0x5642297ca000
  arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcf36d6310) = -1 EINVAL (Invalid argument)
     mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
                                    0x7f6dddd47000
  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=25596, ...}, AT_EMPTY_PATH) = 0
          mmap(NULL, 25596, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f6dddd40000
                    close(3)                                = 0
  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.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\0\0\0\0\0\0\0\0\0"..., 832) = 832
  newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) = 0
  mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f6dddb14000
                  mprotect(0x7f6dddbae000, 1576960, PROT_NONE) = 0
mmap(0x7f6dddbae000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
                                  3, 0x9a000) = 0x7f6dddbae000
   mmap(0x7f6dddcbf000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
```

```
0x1ab000) = 0x7f6dddcbf000

mmap(0x7f6dddd2f000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x21a000) = 0x7f6dddd2f000

mmap(0x7f6dddd3d000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -
1, 0) = 0x7f6dddd3d000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3

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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f6dddaf4000

mmap(0x7f6dddaf7000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x3000) = 0x7f6dddaf7000

mmap(0x7f6dddb0e000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000)
= 0x7f6dddb0e000

mmap(0x7f6dddb12000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1d000) = 0x7f6dddb12000

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\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 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) = 0x7f6ddd8cb000

mprotect(0x7f6ddd8f3000, 2023424, PROT_NONE) = 0

mmap(0x7f6ddd8f3000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7f6ddd8f3000

mmap(0x7f6ddda88000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7f6ddda88000

mmap(0x7f6dddae1000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x215000) = 0x7f6dddae1000

mmap(0x7f6dddae7000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -
1, 0) = 0x7f6dddae7000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.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\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f6ddd7e4000
```

```
mmap(0x7f6ddd7f2000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe000) = 0x7f6ddd7f2000

mmap(0x7f6ddd86e000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000)
= 0x7f6ddd86e000

mmap(0x7f6ddd8c9000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xe4000) = 0x7f6ddd8c9000

close(3)                                = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f6ddd7e2000

arch_prctl(ARCH_SET_FS, 0x7f6ddd7e33c0) = 0

set_tid_address(0x7f6ddd7e3690)         = 713

set_robust_list(0x7f6ddd7e36a0, 24)     = 0

rseq(0x7f6ddd7e3d60, 0x20, 0, 0x53053053) = 0

mprotect(0x7f6dddae1000, 16384, PROT_READ) = 0

mprotect(0x7f6ddd8c9000, 4096, PROT_READ) = 0

mprotect(0x7f6dddb12000, 4096, PROT_READ) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f6ddd7e0000

mprotect(0x7f6dddd2f000, 45056, PROT_READ) = 0

mprotect(0x5642062be000, 4096, PROT_READ) = 0

mprotect(0x7f6dddd81000, 8192, PROT_READ) = 0

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

munmap(0x7f6dddd40000, 25596)           = 0

getrandom("\xa6\x50\x4c\xd5\x75\x78\x8d\xf0", 8, GRND_NONBLOCK) = 8

brk(NULL)                               = 0x5642297ca000

brk(0x5642297eb000)                     = 0x5642297eb000

read(0, "t", 1)                         = 1

read(0, "e", 1)                         = 1

read(0, "s", 1)                         = 1

read(0, "t", 1)                         = 1

read(0, ".", 1)                         = 1

read(0, "t", 1)                         = 1

read(0, "x", 1)                         = 1

read(0, "t", 1)                         = 1

read(0, "\n", 1)                        = 1

openat(AT_FDCWD, "test.txt", O_RDONLY)  = 3

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

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f6ddd7e3690) = 714

close(3)                                = 0
```

```
                     close(5)                                = 0
                     read(4, "7.250000\n2.650000\n", 4096)   = 18
 write(1, "7.250000\n2.650000\n", 18)    = ? ERESTARTSYS (To be restarted if SA_RESTART
                                          is set)
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=714, si_uid=1000, si_status=0,
                                 si_utime=0, si_stime=0} ---
                     write(1, "7.250000\n2.650000\n", 18)    = 18
                  read(4, "19.320000\n1500.000000\n600.799988"..., 4096) = 45
                  write(1, "19.320000\n1500.000000\n600.799988"..., 45) = 45
                     read(4, "", 4096)                       = 0
                     close(4)                                = 0
               wait4(714, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 714
                     exit_group(0)                           = ?
                               +++ exited with 0 +++
```

Вывод

У меня успешно получилось реализовать программу на языке C++, которая использует межпроцессорное взаимодействие через pipe. Все системные вызовы работают корректно, программа завершается без ошибок. Трудностей в работе не возникло.