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-213Б-23

Студент: Гуляев А.П.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 13.10.24

Москва, 2024

Постановка задачи

Вариант 7.

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:

- pid_t fork(void); – создает дочерний процесс.
- int execve(const char *filename, char *const argv[], char *const envp[]) (и другие вариации
- exec) - замена образа памяти процесса
- ◦ pid_t wait() - Ожидание завершения дочерних процессов
- ◦ int pipe(int pipefd[2]) - создание неименованного канала для передачи данных между процессами
- ◦ int dup2(int oldfd, int newfd) - переназначение файлового дескриптора
- ◦ int open(const char *pathname, int flags, mode_t mode) - открытие\создание файла
- ◦ int close(int fd) - закрыть файл
-

Описание работы программы:

`main.c`:

1. **Открытие файла**:
   - В начале программы открывается файл, переданный через аргументы командной строки (`argv[1]`). Файл открывается для чтения (`O_RDONLY`), и дескриптор файла сохраняется в переменной `input`.
   - Если файл не удалось открыть, выводится ошибка в `STDERR`.
2. **Создание пайпа**:
   - Создается пайп с помощью вызова `pipe(pipefd)`, где `pipefd[0]` — это дескриптор для чтения из пайпа, а `pipefd[1]` — для записи.
3. **Форк процесса**:
   - Создается новый процесс с помощью `fork()`. Если это дочерний процесс (условие `if (child == 0)`), то происходит следующее:
     1. **Перенаправление потоков**:
        - Входной поток дочернего процесса (`STDIN_FILENO`) перенаправляется на дескриптор открытого файла (`input`).

- Выходной поток дочернего процесса (`STDOUT_FILENO`) перенаправляется на запись в пайп (`pipefd[1]`).
- Закрывается дескриптор для чтения из пайпа (`pipefd[0]`), так как дочерний процесс ничего не читает из пайпа.

2. **Выполнение дочернего процесса**:
   - В дочернем процессе выполняется программа `child` (с помощью `execve`), которая будет читать данные из файла и отправлять их через пайп.
   - Если выполнение не удалось, выводится ошибка в `STDERR`.

- Если это родительский процесс, то происходит следующее:

1. **Перенаправление потоков**:
   - Входной поток родительского процесса (`STDIN_FILENO`) перенаправляется на чтение из пайпа (`pipefd[0]`).
   - Закрывается дескриптор для записи в пайп (`pipefd[1]`), так как родительский процесс ничего не записывает в пайп.
2. **Ожидание завершения дочернего процесса**:
   - Родительский процесс ждет завершения дочернего процесса с помощью `wait(NULL)`.
3. **Чтение данных из пайпа**:
   - Родительский процесс читает данные из пайпа и выводит их в стандартный поток вывода (`STDOUT`), используя цикл с `read()`.

`child.c`:

1. **Чтение и обработка чисел**:
   - Основной задачей дочернего процесса является чтение чисел, переданных через стандартный ввод, их суммирование и вывод результата.
   - Дочерний процесс читает символы из входного потока, пока не встретит пробел или символ новой строки.
   - Когда слово заканчивается (число), оно преобразуется в тип `double` с помощью функции `read_double()`. Если преобразование прошло успешно, это число добавляется к сумме.
2. **Функция `read_double`**:
   - Эта функция принимает строку, представляющую число, и преобразует её в значение типа `double`.
   - Она корректно обрабатывает как целую, так и дробную часть числа. Также поддерживается знак числа (отрицательные значения).

3. **Вывод суммы**:
    - После обработки каждого числа, текущая сумма чисел выводится в стандартный поток вывода.
    - После каждой строки (обозначенной символом новой строки), сумма сбрасывается и начинается заново.
4. **Закрытие потоков**:
    - В конце дочернего процесса закрываются дескрипторы `STDOUT_FILENO` и `STDIN_FILENO`.

Основной поток взаимодействия:

- Родительский процесс передает дочернему процессу данные (числа).
- Дочерний процесс суммирует числа и отправляет их обратно в родительский процесс через пайп.
- Родительский процесс выводит результат на экран.

Код программы

main.c:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <unistd.h>
#include <fcntl.h>



int main(int argc, char* argv[]) {
    int code;
    int input;
    input = open(argv[1], O_RDONLY);
    if (input < 0) {
        write(STDERR_FILENO, &input, sizeof(int));
    }
    int pipefd[2];
    pipe(pipefd);
```

```
    pid_t child;
    child = fork();

    if (child == 0) {
        dup2(input, STDIN_FILENO);
        dup2(pipefd[1], STDOUT_FILENO);
        close(pipefd[0]);

        if ((code = execve("child", argv, NULL)) != 0) {
            write(STDERR_FILENO, &code, sizeof(int));
        }
    } else {
        dup2(pipefd[0], STDIN_FILENO);
        close(pipefd[1]);
        wait(NULL);

        char c;
        while (read(STDIN_FILENO, &c, sizeof(char)) != 0) {
            write(STDOUT_FILENO, &c, sizeof(char));
        }
        close(STDOUT_FILENO);
        close(STDIN_FILENO);
    }

    return 0;
}
```

child.c:

```
#include <sys/types.h>
#include <sys/stat.h>
```

```
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>


int read_double(char* inp, double* number) {
    double int_part = 0;
    double frac_part = 0;
    double sign = 1;
    if (*inp == '-') {
        sign = -1;
        ++inp;
    }

    while (*inp != '.' && *inp != ',' && *inp != 0) {
        if (*inp < '0' || *inp > '9') {

            return 3;
        }
        int_part *= 10;
        int_part += (*inp - '0');
        ++inp;
    }
    if (*inp == 0) {
        *number = sign * int_part;
        return 0;
    }
    while (*inp != 0) {++inp;}
    --inp;
    while (*inp != '.' && *inp != ',') {
        if (*inp < '0' || *inp > '9' || *inp == 0) {
            return 3;
```

```
        }
        frac_part += (*inp - '0');
        frac_part /= 10;
        --inp;
    }
    *number = sign * (int_part + frac_part);
    return 0;
}


int main(int argc, char* argv[]) {
    char c;
    char word[64];
    char *ptr = word;
    double num;
    double sum = 0;
    int code;

    while (read(STDIN_FILENO, &c, sizeof(char)) != 0) {
        if (c != ' ' && c != '\n') {
            *ptr = c;
            ++ptr;
        } else {
            *ptr = 0;
            ptr = word;
            if ((code = read_double(ptr, &num)) != 0) {
                write(STDERR_FILENO, &code, sizeof(int));
            }
            sum += num;
            ptr = word;

            sprintf(word, "%g", sum);
```

```
            if (c == '\n') {
                char* dig = word;
                while (*dig != 0) {
                    write(STDOUT_FILENO, dig, sizeof(char));
                    ++dig;
                }
                write(STDOUT_FILENO, "\n", sizeof(char));
                sum = 0;
            }
        }
    }

    close(STDOUT_FILENO);
    close(STDIN_FILENO);

    return 0;
}
```

Протокол работы программы

```
npabwa@npabwa-Vivobook-ASUSLaptop-M6500QH-M6500QH:~/Desktop/MAI_OS/lab01/src$ cat < inp.txt
0 0 0
1.2 14 15.666
1 23 -16
-1.4 4 4
npabwa@npabwa-Vivobook-ASUSLaptop-M6500QH-M6500QH:~/Desktop/MAI_OS/lab01/src$ make
gcc main.c -o solution
gcc child.c -o child
npabwa@npabwa-Vivobook-ASUSLaptop-M6500QH-M6500QH:~/Desktop/MAI_OS/lab01/src$ make o
./solution inp.txt
0
30.866
8
6.6
npabwa@npabwa-Vivobook-ASUSLaptop-M6500QH-M6500QH:~/Desktop/MAI_OS/lab01/src$
```

Тестирование:

```
strace -f ./solution inp.txt
execve("./solution", ["./solution", "inp.txt"], 0x7ffd8b88fa80 /* 68 vars */) = 0
brk(NULL)                               = 0x648ddf8f2000
```

```
    arch_prctl(0x3001 /* ARCH_??? */, 0x7ffd388c1160) = -1 EINVAL (Invalid
argument)
    mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x780a5ef70000
    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=76971, ...},
AT_EMPTY_PATH) = 0
    mmap(NULL, 76971, PROT_READ, MAP_PRIVATE, 3, 0) = 0x780a5ef5d000
    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\0I\17\357\204\3$\f\221\2039x\
324\224\323\236S"..., 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) =
0x780a5ec00000
    mprotect(0x780a5ec28000, 2023424, PROT_NONE) = 0
    mmap(0x780a5ec28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x780a5ec28000
    mmap(0x780a5edbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1bd000) = 0x780a5edbd000
    mmap(0x780a5ee16000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x215000) = 0x780a5ee16000
    mmap(0x780a5ee1c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x780a5ee1c000
    close(3)                                = 0
    mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x780a5ef5a000
    arch_prctl(ARCH_SET_FS, 0x780a5ef5a740) = 0
    set_tid_address(0x780a5ef5aa10)         = 7752
    set_robust_list(0x780a5ef5aa20, 24)     = 0
    rseq(0x780a5ef5b0e0, 0x20, 0, 0x53053053) = 0
    mprotect(0x780a5ee16000, 16384, PROT_READ) = 0
    mprotect(0x648ddf1f7000, 4096, PROT_READ) = 0
    mprotect(0x780a5efaa000, 8192, PROT_READ) = 0
    prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
    munmap(0x780a5ef5d000, 76971)           = 0
    openat(AT_FDCWD, "inp.txt", O_RDONLY)   = 3
    pipe2([4, 5], 0)                        = 0
    clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLD, child_tidptr=0x780a5ef5aa10) = 7753
    strace: Process 7753 attached
```

```
[pid  7752] dup2(4, 0)                  = 0
[pid  7752] close(5 <unfinished ...>
[pid  7753] set_robust_list(0x780a5ef5aa20, 24 <unfinished ...>
[pid  7752] <... close resumed>)        = 0
[pid  7753] <... set_robust_list resumed>) = 0
[pid  7752] wait4(-1,  <unfinished ...>
[pid  7753] dup2(3, 0)                  = 0
[pid  7753] dup2(5, 1)                  = 1
[pid  7753] close(4)                    = 0
[pid  7753] execve("child", ["./solution", "inp.txt"], NULL) = 0
[pid  7753] brk(NULL)                   = 0x5d9e64d4f000
[pid  7753] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffca5991160) = -1
EINVAL (Invalid argument)
[pid  7753] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7222e0946000
[pid  7753] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file
or directory)
[pid  7753] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 4
[pid  7753] newfstatat(4, "", {st_mode=S_IFREG|0644,
st_size=76971, ...}, AT_EMPTY_PATH) = 0
[pid  7753] mmap(NULL, 76971, PROT_READ, MAP_PRIVATE, 4, 0) =
0x7222e0933000
[pid  7753] close(4)                    = 0
[pid  7753] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6",
O_RDONLY|O_CLOEXEC) = 4
[pid  7753] read(4, "\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  7753] pread64(4, "\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  7753] pread64(4, "\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  7753] pread64(4, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\
221\2039x\324\224\323\236S"..., 68, 896) = 68
[pid  7753] newfstatat(4, "", {st_mode=S_IFREG|0755,
st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid  7753] pread64(4, "\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  7753] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4,
0) = 0x7222e0600000
[pid  7753] mprotect(0x7222e0628000, 2023424, PROT_NONE) = 0
[pid  7753] mmap(0x7222e0628000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x28000) = 0x7222e0628000
[pid  7753] mmap(0x7222e07bd000, 360448, PROT_READ, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 4, 0x1bd000) = 0x7222e07bd000
[pid  7753] mmap(0x7222e0816000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x215000) = 0x7222e0816000
[pid  7753] mmap(0x7222e081c000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7222e081c000
[pid  7753] close(4)                    = 0
[pid  7753] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_ANONYMOUS, -1, 0) = 0x7222e0930000
[pid  7753] arch_prctl(ARCH_SET_FS, 0x7222e0930740) = 0
[pid  7753] set_tid_address(0x7222e0930a10) = 7753
[pid  7753] set_robust_list(0x7222e0930a20, 24) = 0
```

```
[pid  7753] rseq(0x7222e09310e0, 0x20, 0, 0x53053053) = 0
[pid  7753] mprotect(0x7222e0816000, 16384, PROT_READ) = 0
[pid  7753] mprotect(0x5d9e63c3f000, 4096, PROT_READ) = 0
[pid  7753] mprotect(0x7222e0980000, 8192, PROT_READ) = 0
[pid  7753] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid  7753] munmap(0x7222e0933000, 76971) = 0
[pid  7753] read(0, "0", 1)             = 1
[pid  7753] read(0, " ", 1)             = 1
[pid  7753] read(0, "0", 1)             = 1
[pid  7753] read(0, " ", 1)             = 1
[pid  7753] read(0, "0", 1)             = 1
[pid  7753] read(0, "\n", 1)            = 1
[pid  7753] write(1, "0", 1)            = 1
[pid  7753] write(1, "\n", 1)           = 1
[pid  7753] read(0, "1", 1)             = 1
[pid  7753] read(0, ".", 1)             = 1
[pid  7753] read(0, "2", 1)             = 1
[pid  7753] read(0, " ", 1)             = 1
[pid  7753] read(0, "1", 1)             = 1
[pid  7753] read(0, "4", 1)             = 1
[pid  7753] read(0, " ", 1)             = 1
[pid  7753] read(0, "1", 1)             = 1
[pid  7753] read(0, "5", 1)             = 1
[pid  7753] read(0, ".", 1)             = 1
[pid  7753] read(0, "6", 1)             = 1
[pid  7753] read(0, "6", 1)             = 1
[pid  7753] read(0, "6", 1)             = 1
[pid  7753] read(0, "\n", 1)            = 1
[pid  7753] write(1, "3", 1)            = 1
[pid  7753] write(1, "0", 1)            = 1
[pid  7753] write(1, ".", 1)            = 1
[pid  7753] write(1, "8", 1)            = 1
[pid  7753] write(1, "6", 1)            = 1
[pid  7753] write(1, "6", 1)            = 1
[pid  7753] write(1, "\n", 1)           = 1
[pid  7753] read(0, "1", 1)             = 1
[pid  7753] read(0, " ", 1)             = 1
[pid  7753] read(0, "2", 1)             = 1
[pid  7753] read(0, "3", 1)             = 1
[pid  7753] read(0, " ", 1)             = 1
[pid  7753] read(0, "-", 1)             = 1
[pid  7753] read(0, "1", 1)             = 1
[pid  7753] read(0, "6", 1)             = 1
```

```
[pid  7753] read(0, "\n", 1)            = 1
[pid  7753] write(1, "8", 1)            = 1
[pid  7753] write(1, "\n", 1)           = 1
[pid  7753] read(0, "-", 1)             = 1
[pid  7753] read(0, "1", 1)             = 1
[pid  7753] read(0, ".", 1)             = 1
[pid  7753] read(0, "4", 1)             = 1
[pid  7753] read(0, " ", 1)             = 1
[pid  7753] read(0, "4", 1)             = 1
[pid  7753] read(0, " ", 1)             = 1
[pid  7753] read(0, "4", 1)             = 1
[pid  7753] read(0, "\n", 1)            = 1
[pid  7753] write(1, "6", 1)            = 1
[pid  7753] write(1, ".", 1)            = 1
[pid  7753] write(1, "6", 1)            = 1
[pid  7753] write(1, "\n", 1)           = 1
[pid  7753] read(0, "", 1)              = 0
[pid  7753] exit_group(0)               = ?
[pid  7753] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)       = 7753
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=7753,
si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
read(0, "0", 1)                         = 1
write(1, "0", 10)                        = 1
read(0, "\n", 1)                        = 1
write(1, "\n", 1
)                       = 1
read(0, "3", 1)                         = 1
write(1, "3", 13)                        = 1
read(0, "0", 1)                         = 1
write(1, "0", 10)                        = 1
read(0, ".", 1)                         = 1
write(1, ".", 1.)                        = 1
read(0, "8", 1)                         = 1
write(1, "8", 18)                        = 1
read(0, "6", 1)                         = 1
write(1, "6", 16)                        = 1
read(0, "6", 1)                         = 1
write(1, "6", 16)                        = 1
read(0, "\n", 1)                        = 1
write(1, "\n", 1
)                       = 1
read(0, "8", 1)                         = 1
write(1, "8", 18)                        = 1
```

```
read(0, "\n", 1)                        = 1
write(1, "\n", 1
)                       = 1
read(0, "6", 1)                         = 1
write(1, "6", 16)                        = 1
read(0, ".", 1)                         = 1
write(1, ".", 1.)                        = 1
read(0, "6", 1)                         = 1
write(1, "6", 16)                        = 1
read(0, "\n", 1)                        = 1
write(1, "\n", 1
)                       = 1
read(0, "", 1)                          = 0
close(1)                                = 0
close(0)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В результате обработки файла с командами, состоящими из произвольного количества чисел типа float, дочерний процесс успешно суммирует все указанные значения и выводит полученную сумму в стандартный поток вывода. Это решение позволяет эффективно обрабатывать входные данные, независимо от их объема, благодаря динамическому подходу к чтению и суммированию чисел.