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: Мустафаев А.Р.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 04.10.24

Москва, 2024

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:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int *fd); – создает однонаправленный канал.
- size_t read(int fd, void *buf, size_t count); - читает данные из файлового дескриптора в буфер.
- int dup2(int oldfd, int newfd); - дублирует файловые дескрипторы.
- int execlp(const char *file, const char *arg, ..., (char *) NULL); - выполнение другой программы в текущем процессе.
- size_t write(int fd, const void *buf, size_t count); - записывает данные в файловый дескриптор.
- int close(int fd); - закрывает файловый дескриптор.
- int open(const char *pathname, int flags); - открывает файлы.

Для начала создаем два файла parent.c и child.c для родительского и дочернего процессов соответственно. Далее в файле parent.c инициализируем pipe и делаем fork данного процесса. Делаем условие: pid = 0 (то есть айди процесса равно нулю, значит это дочерний процесс), тогда выполняем блок кода для дочернего процесса, иначе блок кода родительского процесса. В дочернем процессе закрываем дескриптор чтения из канала, перенаправляем stdout в канал, запускаем программу для чтения файла (child.c) с помощью команды execlp. В программе child открываем файл, путь до которого ввели в родительском процессе, перенаправляем stdin в файл, читаем файл и обрабатываем вещественные числа, командой write записываем в stdout полученный результат. В родительском процессе закрываем дескриптор записи в канал, читаем результат из канала и выводим его на экран, ожидаем завершения дочернего процесса с помощью wait.

Код программы

parent.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>

#define BUF_SIZE 1024
```

```c
int main() {
 int pipefd[2];
 pid_t pid;

 if (pipe(pipefd) == -1) {
      perror("pipe");
      exit(EXIT_FAILURE);
 }

 char filename[256];
 printf("Enter filename: ");
 scanf("%255s", filename);

 pid = fork();
 if (pid == -1) {
      perror("fork");
      exit(EXIT_FAILURE);
 }

 if (pid == 0) {

      close(pipefd[0]);

      dup2(pipefd[1], STDOUT_FILENO);
      close(pipefd[1]);

      execlp("./child", "child", filename, (char*)NULL);
      perror("execlp");
      exit(EXIT_FAILURE);
 } else {

      close(pipefd[1]);

      char buffer[BUF_SIZE];
      int n;
      while ((n = read(pipefd[0], buffer, sizeof(buffer))) > 0) {
            write(STDOUT_FILENO, buffer, n);
            }

      close(pipefd[0]);

      wait(NULL);
 }

 return 0;
}
```

child.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <ctype.h>

#define BUF_SIZE 1024

int main(int argc, char *argv[]) {
```

```
        if (argc != 2) {
            fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

        int filefd = open(argv[1], O_RDONLY);
        if (filefd == -1) {
            perror("open");
        exit(EXIT_FAILURE);
    }

    dup2(filefd, STDIN_FILENO);
    close(filefd);

        char buffer[BUF_SIZE];
        size_t bytes_read;
        char num_str[BUF_SIZE];
        int num_index = 0;
        int in_number = 0;
        double sum = 0;

        while ((bytes_read = read(STDIN_FILENO, buffer, BUF_SIZE)) > 0) {
            for (size_t i = 0; i < bytes_read; i++) {
                if (isdigit(buffer[i]) || buffer[i] == '.' || (buffer[i] == '-'
                      && !in_number)) {
                    num_str[num_index++] = buffer[i];
                    in_number = 1;
                } else if (in_number) {
                    num_str[num_index] = '\0';
                    double number = atof(num_str);
                    sum += number;
                    num_index = 0;
                    in_number = 0;
            }
        }
    }

        if (in_number) {
            num_str[num_index] = '\0';
            double number = atof(num_str);
            sum += number;
    }

        char str[BUF_SIZE];
        size_t n = sprintf(str, "Sum: %f\n", sum);
    write(STDOUT_FILENO, str, n);

        return 0;
}
```

Протокол работы программы

Тестирование:

```
$ ./parent.c
Enter filename: test.txt
Sum: 17.668000
```

```
$./parent.c
Enter filename: test2.txt
Sum: 10160.480044
```

Strace:

```
execve("./parent", ["./parent"], 0x7fff8ae10168 /* 78 vars */) = 0
brk(NULL)                               = 0x5a9d30ed6000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f127c715000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или
каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=73607, ...}) = 0
mmap(NULL, 73607, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f127c703000
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\0\220\243\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
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f127c400000
mmap(0x7f127c428000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x28000) = 0x7f127c428000
mmap(0x7f127c5b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7f127c5b0000
mmap(0x7f127c5ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1fe000) = 0x7f127c5ff000
mmap(0x7f127c605000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f127c605000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f127c700000
arch_prctl(ARCH_SET_FS, 0x7f127c700740) = 0
set_tid_address(0x7f127c700a10)         = 40467
set_robust_list(0x7f127c700a20, 24)     = 0
rseq(0x7f127c701060, 0x20, 0, 0x53053053) = 0
mprotect(0x7f127c5ff000, 16384, PROT_READ) = 0
mprotect(0x5a9d2f745000, 4096, PROT_READ) = 0
mprotect(0x7f127c74d000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f127c703000, 73607)           = 0
pipe2([3, 4], 0)                        = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
```

```
getrandom("\x34\x09\x17\x41\x48\x77\x05\x10", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5a9d30ed6000
brk(0x5a9d30ef7000)                     = 0x5a9d30ef7000
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
write(1, "Enter filename: ", 16Enter filename: )        = 16
read(0, test.txt
"test.txt\n", 1024)             = 9


clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
SIGCHLDstrace: Process 40496 attached
, child_tidptr=0x7f127c700a10) = 40496
[pid 40467] close(4 <unfinished ...>
[pid 40496] set_robust_list(0x7f127c700a20, 24 <unfinished ...>
[pid 40467] <... close resumed>)        = 0
[pid 40496] <... set_robust_list resumed>) = 0
[pid 40467] read(3,  <unfinished ...>
[pid 40496] close(3)                    = 0
[pid 40496] dup2(4, 1)                  = 1
[pid 40496] close(4)                    = 0
[pid 40496] execve("./child", ["child", "test.txt"], 0x7ffecf59d878 /* 78 vars
*/) = 0
[pid 40496] brk(NULL)                   = 0x5665978dc000
[pid 40496] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x71af17534000
[pid 40496] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла
или каталога)
[pid 40496] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid 40496] fstat(3, {st_mode=S_IFREG|0644, st_size=73607, ...}) = 0
[pid 40496] mmap(NULL, 73607, PROT_READ, MAP_PRIVATE, 3, 0) = 0x71af17522000
[pid 40496] close(3)                    = 0
[pid 40496] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|
O_CLOEXEC) = 3
[pid 40496] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\
0\1\0\0\0\220\243\2\0\0\0\0"..., 832) = 832
[pid 40496] 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 40496] fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
[pid 40496] 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 40496] mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x71af17200000
[pid 40496] mmap(0x71af17228000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|
MAP_F
IXED|MAP_DENYWRITE, 3, 0x28000) = 0x71af17228000
[pid 40496] mmap(0x71af173b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_D
ENYWRITE, 3, 0x1b0000) = 0x71af173b0000
[pid 40496] mmap(0x71af173ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_F
IXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x71af173ff000
```

```
[pid 40496] mmap(0x71af17405000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|M
AP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x71af17405000
[pid 40496] close(3)                    = 0
[pid 40496] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
1, 0) = 0x71af1751f000
[pid 40496] arch_prctl(ARCH_SET_FS, 0x71af1751f740) = 0
[pid 40496] set_tid_address(0x71af1751fa10) = 40496
[pid 40496] set_robust_list(0x71af1751fa20, 24) = 0
[pid 40496] rseq(0x71af17520060, 0x20, 0, 0x53053053) = 0
[pid 40496] mprotect(0x71af173ff000, 16384, PROT_READ) = 0
[pid 40496] mprotect(0x566596884000, 4096, PROT_READ) = 0
[pid 40496] mprotect(0x71af1756c000, 8192, PROT_READ) = 0
[pid 40496] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid 40496] munmap(0x71af17522000, 73607) = 0
[pid 40496] openat(AT_FDCWD, "test.txt", O_RDONLY) = 3
[pid 40496] dup2(3, 0)                  = 0
[pid 40496] close(3)                    = 0
[pid 40496] read(0, "12.223 1.0\n\n1 0 3.445", 1024) = 21
[pid 40496] read(0, "", 1024)           = 0
[pid 40496] write(1, "Sum: 17.668000\n", 15) = 15
[pid 40467] <... read resumed>"Sum: 17.668000\n", 1024) = 15
[pid 40496] exit_group(0)               = ?
[pid 40467] write(1, "Sum: 17.668000\n", 15 <unfinished ...>
[pid 40496] +++ exited with 0 +++
<... write resumed>)                    = ? ERESTARTSYS (To be restarted if
SA_RESTART is set)
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=40496, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
write(1, "Sum: 17.668000\n", 15Sum: 17.668000
)        = 15
read(3, "", 1024)                       = 0
close(3)                                = 0
wait4(-1, NULL, 0, NULL)                = 40496
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Недопустимая операция
смещения)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Взаимодействие между родительским и дочерним процессами было успешно реализовано с помощью межпроцессного канала и системных вызовов. Результат обработки вещественных чисел, считанных из файла, был корректно передан от дочернего процесса к родительскому и выведен на экран. Приобретены практические навыки обеспечения обмена данных между процессами посредством каналов.