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:Заитова Е.А.

Преподаватель: Бахарев В.Д.

Оценка: ____________________

Дата: 13.10.24

Москва, 2024

Постановка задачи

Вариант 18.

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Общий метод и алгоритм решения

pid_t fork(void) — используется для создания дочернего процесса.

int pipe(int fd) — создает канал для однонаправленной связи между процессами. fd[0] используется для чтения из канала, а fd[1] — для записи в него.

ssize_t write(int fd, const void buf, size_t count) — записывает данные из буфера buf в файл, связанный с файловым дескриптором fd, в количестве байтов, указанном в count.

ssize_t read(int fd, void buf, size_t count) — читает данные из файла или канала, связанного с файловым дескриптором fd, в буфер buf в количестве байтов, указанном в count.

int execv(const char path, char const argv[]) — заменяет текущий процесс новым процессом, запускающим указанную программу.

int32_t open(const char*file, int oflag, …); – открывает файл и возвращает файловый дескриптор.

int close(int fd) – закрывает файл.

int dup2(int oldfd, int newfd) — дублирует файловый дескриптор oldfd, заменяя им дескриптор newfd. Перенаправление стандартного ввода дочернего процесса на канал.

int wait(int status) — приостанавливает выполнение родительского процесса до завершения дочернего процесса.

Алгоритм решения

Вначале программа инициализирует два канала для межпроцессной коммуникации и создает два дочерних процесса с помощью системного вызова fork. Эти дочерние процессы будут получать данные от родительского процесса через каналы, перенаправляя стандартный ввод (stdin) на соответствующий канал с помощью dup2.

Родительский процесс запрашивает у пользователя имена двух файлов и затем вводит текстовые данные через стандартный ввод. Программа считывает строки и в зависимости от номера строки (четная или нечетная) передает данные в соответствующий канал для одного из двух дочерних процессов. Данные передаются с помощью вызова write.

В дочерних процессах происходит запуск другой программы (./child), которая принимает данные через стандартный ввод, выполняет их обработку (удаление гласных из строк), а затем записывает результат в файл. После завершения всех операций родительский процесс ожидает завершения дочерних процессов с помощью вызова wait.

Код программы

main.c

```
#include <stdint.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/wait.h>

#define BUFSIZ 4096

void filter_data(int pipe1[],int pipe2[], char* buffer, int cnt_of_lines) {
    if (cnt_of_lines % 2 == 0) {
        write(pipe2[1], buffer, strlen(buffer) + 1);
    } else {
        write(pipe1[1], buffer, strlen(buffer) + 1);
    }
}



int main(int argc, char* argv[]) {
    if (argc > 1) {
        const char* msg = "Required console input\n";
        write(STDERR_FILENO, msg, strlen(msg) + 1);
        exit(EXIT_FAILURE);
    }
    char buffer[BUFSIZ];
    int pipe1[2], pipe2[2];
    pid_t child1_pid, child2_pid;
    ssize_t bytes;
    int cnt_of_lines = 1;

    char file1[BUFSIZ];
    char file2[BUFSIZ];

    const char* msg1 = "Enter filename for child1\n";
    write(STDOUT_FILENO, msg1, strlen(msg1));
    read(STDIN_FILENO, buffer, BUFSIZ);
    buffer[strcspn(buffer, "\n")] = '\0';
    strcpy(file1, buffer);
```

```
const char* msg2 = "Enter filename for child2\n";
write(STDOUT_FILENO, msg2, strlen(msg1));
read(STDIN_FILENO, buffer, BUFSIZ);
buffer[strcspn(buffer, "\n")] = '\0';
strcpy(file2, buffer);


if (pipe(pipe1) == -1) {
    const char* msg = "Error: failed to create pipe1\n";
    write(STDERR_FILENO, msg, strlen(msg) + 1);
    exit(EXIT_FAILURE);
}

if (pipe(pipe2) == -1) {
    const char* msg = "Error: failed to create pipe2\n";
    write(STDERR_FILENO, msg, strlen(msg) + 1);
    exit(EXIT_FAILURE);
}

child1_pid = fork();
if (child1_pid == -1) {
    const char* msg = "Error: failed to spawn new proccess\n";
    write(STDERR_FILENO, msg, strlen(msg) + 1);
    exit(EXIT_FAILURE);
}
else if (child1_pid == 0) {
    // in child proccess
    // dup channel - run child1 - handling errors
    // get data from channel not from stdin


    pid_t child1_pid = getpid();
    dup2(pipe1[0], STDIN_FILENO); // parent stdin connecting with child stdin
    // close(pipe1[1]);

    char* args[] = {"./child", file1, NULL};
    int status = execv("./child", args);

    if (status == -1) {
        const char* msg = "Failed to exec child1\n";
        write(STDERR_FILENO, msg, strlen(msg) + 1);
        exit(EXIT_FAILURE);
    }
}

child2_pid = fork();
if (child2_pid == -1) {
    const char* msg = "Error: failed to spawn new proccess\n";
    write(STDERR_FILENO, msg, strlen(msg) + 1);
    exit(EXIT_FAILURE);
} else if (child2_pid == 0) {
    dup2(pipe2[0], STDIN_FILENO);
    // close(pipe2[1]);

    char* args[] = {"./child", file2, NULL};
    int status = execv("./child", args);

    if (status == -1) {
        const char* msg = "Failed to exec child2\n";
        write(STDERR_FILENO, msg, strlen(msg) + 1);
        exit(EXIT_FAILURE);
    }
}

close(pipe1[0]); // close unuseful parents channels (reading)
close(pipe2[0]);



while(bytes = read(STDIN_FILENO, buffer, BUFSIZ)) {
    if (bytes < 0) {
        const char* msg = "error: failed to read from stdin\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(EXIT_FAILURE);
    }

    if (buffer[0] == '\n') {
        break;
    }

    buffer[bytes - 1] = '\0';
    filter_data(pipe1, pipe2, buffer, cnt_of_lines++);
}




close(pipe1[1]);
close(pipe2[1]);

int child1_status;
int child2_status;

wait(&child1_status);
wait(&child2_status);

return 0;
```

```
}
```

child.c

```
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>

#define BUFSIZ 4096

void handling(char* str, char* result) {
    char* p_str = str;
    char* p_result = result;

    while (*p_str) {
        if (*p_str != 'a' && *p_str != 'e' && *p_str != 'i' && *p_str != 'o' && *p_str
!= 'u' &&
            *p_str != 'A' && *p_str != 'E' && *p_str != 'I' && *p_str != 'O' && *p_str
!= 'U') {
            *p_result++ = *p_str;
        }
        ++p_str;
    }

    *p_result = '\0';
}

int main(int argc, char* argv[]) {
    if (argc != 2) {
        const char* msg = "Invalid amount parametres\n";
        write(STDERR_FILENO, msg, strlen(msg));
        exit(EXIT_FAILURE);
    }

    char buffer[BUFSIZ];
    ssize_t bytes;

    int file = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC | O_APPEND, 0600);
    if (file == -1) {
        const char* msg = "Error: failed to open file\n";
        write(STDERR_FILENO, msg, strlen(msg));
        exit(EXIT_FAILURE);
    }

    char result[BUFSIZ];
    while(bytes = read(STDIN_FILENO, buffer, BUFSIZ)) {
        buffer[bytes - 1] = '\0';
        handling(buffer, result);
        write(STDOUT_FILENO, result, strlen(result));
        write(STDOUT_FILENO, "\n", 1);

        write(file, result, strlen(result));
        write(file, "\n", 1);

    }

    close(file);

    return 0;
}
```

Протокол работы программы

Тестирование:

```
$ /a.out
Enter filename for child1
output1.txt
Enter filename for child2
output2.txt
hello
hll
world
wrld
im testing
m tstng
this
ths
program
prgrm
^C

$ cat < output1
wrld
ths

$ cat < output2
hll
m tstng
prgrm
```

Strace:

```
$ strace -f ./a.out

execve("./a.out", ["./a.out"], 0x7ffd41ba7058 /* 26 vars */) = 0

brk(NULL)                               = 0x555f38257000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffddb580780) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f4b8fbd9000

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=17839, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 17839, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f4b8fbd4000

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) = 0x7f4b8f9ab000

mprotect(0x7f4b8f9d3000, 2023424, PROT_NONE) = 0

mmap(0x7f4b8f9d3000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7f4b8f9d3000

mmap(0x7f4b8fb68000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7f4b8fb68000

mmap(0x7f4b8fbc1000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x215000) = 0x7f4b8fbc1000

mmap(0x7f4b8fbc7000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f4b8fbc7000

close(3)                                = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f4b8f9a8000

arch_prctl(ARCH_SET_FS, 0x7f4b8f9a8740) = 0
```

```
set_tid_address(0x7f4b8f9a8a10)         = 67965

set_robust_list(0x7f4b8f9a8a20, 24)     = 0

rseq(0x7f4b8f9a90e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f4b8fbc1000, 16384, PROT_READ) = 0

mprotect(0x555f37a81000, 4096, PROT_READ) = 0

mprotect(0x7f4b8fc13000, 8192, PROT_READ) = 0

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

munmap(0x7f4b8fbd4000, 17839)           = 0

write(1, "Enter filename for child1\n", 26Enter filename for child1

) = 26

read(0, output1.txt

"output1.txt\n", 4096)          = 12

write(1, "Enter filename for child2\n", 26Enter filename for child2

) = 26

read(0, output2.txt

"output2.txt\n", 4096)          = 12

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

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

clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
Process 68052 attached

, child_tidptr=0x7f4b8f9a8a10) = 68052

[pid 67965] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>

[pid 68052] set_robust_list(0x7f4b8f9a8a20, 24) = 0

[pid 68052] getpid()                    = 68052

[pid 68052] dup2(3, 0)                  = 0

[pid 68052] execve("./child", ["./child", "output1.txt"], 0x7ffddb580958 /* 26 vars
*/strace: Process 68053 attached

 <unfinished ...>

[pid 67965] <... clone resumed>, child_tidptr=0x7f4b8f9a8a10) = 68053

[pid 68053] set_robust_list(0x7f4b8f9a8a20, 24 <unfinished ...>

[pid 67965] close(3 <unfinished ...>

[pid 68053] <... set_robust_list resumed>) = 0

[pid 67965] <... close resumed>)        = 0
```

```
    [pid 68053] dup2(5, 0 <unfinished ...>

    [pid 67965] close(5 <unfinished ...>

    [pid 68053] <... dup2 resumed>)        = 0

    [pid 67965] <... close resumed>)       = 0

    [pid 68053] execve("./child", ["./child", "output2.txt"], 0x7ffddb580958 /* 26 vars */
<unfinished ...>

    [pid 67965] read(0,  <unfinished ...>

    [pid 68053] <... execve resumed>)      = 0

    [pid 68052] <... execve resumed>)      = 0

    [pid 68053] brk(NULL)                  = 0x55e87e814000

    [pid 68052] brk(NULL)                  = 0x556d8a78b000

    [pid 68053] arch_prctl(0x3001 /* ARCH_??? */, 0x7fffa6756430 <unfinished ...>

    [pid 68052] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffda85243d0) = -1 EINVAL (Invalid
argument)

    [pid 68053] <... arch_prctl resumed>)  = -1 EINVAL (Invalid argument)

    [pid 68053] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>

    [pid 68052] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>

    [pid 68053] <... mmap resumed>)        = 0x7f3135edf000

    [pid 68052] <... mmap resumed>)        = 0x7ff59d323000

    [pid 68053] access("/etc/ld.so.preload", R_OK <unfinished ...>

    [pid 68052] access("/etc/ld.so.preload", R_OK <unfinished ...>

    [pid 68053] <... access resumed>)      = -1 ENOENT (No such file or directory)

    [pid 68052] <... access resumed>)      = -1 ENOENT (No such file or directory)

    [pid 68053] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

    [pid 68052] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>

    [pid 68053] <... openat resumed>)      = 7

    [pid 68052] <... openat resumed>)      = 7

    [pid 68053] newfstatat(7, "",  <unfinished ...>

    [pid 68052] newfstatat(7, "",  <unfinished ...>

    [pid 68053] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=17839, ...},
AT_EMPTY_PATH) = 0

    [pid 68052] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=17839, ...},
AT_EMPTY_PATH) = 0
```

```
     [pid 68053] mmap(NULL, 17839, PROT_READ, MAP_PRIVATE, 7, 0 <unfinished ...>

     [pid 68052] mmap(NULL, 17839, PROT_READ, MAP_PRIVATE, 7, 0 <unfinished ...>

     [pid 68053] <... mmap resumed>)         = 0x7f3135eda000

     [pid 68052] <... mmap resumed>)         = 0x7ff59d31e000

     [pid 68053] close(7 <unfinished ...>

     [pid 68052] close(7 <unfinished ...>

     [pid 68053] <... close resumed>)        = 0

     [pid 68052] <... close resumed>)        = 0

     [pid 68053] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>

     [pid 68052] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>

     [pid 68053] <... openat resumed>)       = 7

     [pid 68052] <... openat resumed>)       = 7

     [pid 68053] read(7,  <unfinished ...>

     [pid 68052] read(7,  <unfinished ...>

     [pid 68053] <... read
resumed>"\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 68052] <... read
resumed>"\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 68053] pread64(7,  <unfinished ...>

     [pid 68052] pread64(7,  <unfinished ...>

     [pid 68053] <... pread64
resumed>"\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 68052] <... pread64
resumed>"\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 68053] pread64(7,  <unfinished ...>

     [pid 68052] pread64(7,  <unfinished ...>

     [pid 68053] <... pread64 resumed>"\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 68052] <... pread64 resumed>"\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 68053] pread64(7,  <unfinished ...>

     [pid 68052] pread64(7,  <unfinished ...>

     [pid 68053] <... pread64
resumed>"\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 68052] <... pread64
resumed>"\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 68053] newfstatat(7, "",  <unfinished ...>

     [pid 68052] newfstatat(7, "",  <unfinished ...>

     [pid 68053] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...},
AT_EMPTY_PATH) = 0

     [pid 68052] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...},
AT_EMPTY_PATH) = 0

     [pid 68053] pread64(7,  <unfinished ...>

     [pid 68052] pread64(7,  <unfinished ...>

     [pid 68053] <... pread64
resumed>"\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 68052] <... pread64
resumed>"\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 68053] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 7, 0 <unfinished
...>

     [pid 68052] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 7, 0 <unfinished
...>

     [pid 68053] <... mmap resumed>)         = 0x7f3135cb1000

     [pid 68052] <... mmap resumed>)         = 0x7ff59d0f5000

     [pid 68053] mprotect(0x7f3135cd9000, 2023424, PROT_NONE <unfinished ...>

     [pid 68052] mprotect(0x7ff59d11d000, 2023424, PROT_NONE <unfinished ...>

     [pid 68053] <... mprotect resumed>)     = 0

     [pid 68052] <... mprotect resumed>)     = 0

     [pid 68053] mmap(0x7f3135cd9000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x28000 <unfinished ...>

     [pid 68052] mmap(0x7ff59d11d000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x28000 <unfinished ...>

     [pid 68053] <... mmap resumed>)         = 0x7f3135cd9000

     [pid 68052] <... mmap resumed>)         = 0x7ff59d11d000

     [pid 68053] mmap(0x7f3135e6e000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1bd000 <unfinished ...>

     [pid 68052] mmap(0x7ff59d2b2000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1bd000 <unfinished ...>

     [pid 68053] <... mmap resumed>)         = 0x7f3135e6e000

     [pid 68052] <... mmap resumed>)         = 0x7ff59d2b2000
     [pid 68053] mmap(0x7f3135ec7000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x215000 <unfinished ...>

     [pid 68052] mmap(0x7ff59d30b000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x215000 <unfinished ...>

     [pid 68053] <... mmap resumed>)         = 0x7f3135ec7000

     [pid 68052] <... mmap resumed>)         = 0x7ff59d30b000

     [pid 68053] mmap(0x7f3135ecd000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

     [pid 68052] mmap(0x7ff59d311000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

     [pid 68053] <... mmap resumed>)         = 0x7f3135ecd000

     [pid 68052] <... mmap resumed>)         = 0x7ff59d311000

     [pid 68053] close(7 <unfinished ...>

     [pid 68052] close(7 <unfinished ...>

     [pid 68053] <... close resumed>)        = 0

     [pid 68052] <... close resumed>)        = 0

     [pid 68053] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>

     [pid 68052] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>

     [pid 68053] <... mmap resumed>)         = 0x7f3135cae000

     [pid 68052] <... mmap resumed>)         = 0x7ff59d0f2000

     [pid 68053] arch_prctl(ARCH_SET_FS, 0x7f3135cae740 <unfinished ...>

     [pid 68052] arch_prctl(ARCH_SET_FS, 0x7ff59d0f2740 <unfinished ...>

     [pid 68053] <... arch_prctl resumed>)   = 0

     [pid 68052] <... arch_prctl resumed>)   = 0

     [pid 68053] set_tid_address(0x7f3135caea10 <unfinished ...>

     [pid 68052] set_tid_address(0x7ff59d0f2a10 <unfinished ...>

     [pid 68053] <... set_tid_address resumed>) = 68053

     [pid 68052] <... set_tid_address resumed>) = 68052

     [pid 68053] set_robust_list(0x7f3135caea20, 24 <unfinished ...>

     [pid 68052] set_robust_list(0x7ff59d0f2a20, 24 <unfinished ...>

     [pid 68053] <... set_robust_list resumed>) = 0

     [pid 68052] <... set_robust_list resumed>) = 0

     [pid 68053] rseq(0x7f3135caf0e0, 0x20, 0, 0x53053053 <unfinished ...>
     [pid 68052] rseq(0x7ff59d0f30e0, 0x20, 0, 0x53053053 <unfinished ...>

     [pid 68053] <... rseq resumed>)         = 0

     [pid 68052] <... rseq resumed>)         = 0

     [pid 68053] mprotect(0x7f3135ec7000, 16384, PROT_READ <unfinished ...>

     [pid 68052] mprotect(0x7ff59d30b000, 16384, PROT_READ <unfinished ...>

     [pid 68053] <... mprotect resumed>)     = 0

     [pid 68052] <... mprotect resumed>)     = 0

     [pid 68053] mprotect(0x55e87d9a6000, 4096, PROT_READ <unfinished ...>

     [pid 68052] mprotect(0x556d89533000, 4096, PROT_READ <unfinished ...>

     [pid 68053] <... mprotect resumed>)     = 0

     [pid 68052] <... mprotect resumed>)     = 0

     [pid 68053] mprotect(0x7f3135f19000, 8192, PROT_READ <unfinished ...>

     [pid 68052] mprotect(0x7ff59d35d000, 8192, PROT_READ <unfinished ...>

     [pid 68053] <... mprotect resumed>)     = 0

     [pid 68052] <... mprotect resumed>)     = 0

     [pid 68052] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>

     [pid 68053] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>

     [pid 68052] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

     [pid 68053] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

     [pid 68052] munmap(0x7ff59d31e000, 17839 <unfinished ...>

     [pid 68053] munmap(0x7f3135eda000, 17839 <unfinished ...>

     [pid 68052] <... munmap resumed>)       = 0

     [pid 68053] <... munmap resumed>)       = 0

     [pid 68052] openat(AT_FDCWD, "output1.txt", O_WRONLY|O_CREAT|O_TRUNC|O_APPEND, 0600
<unfinished ...>

     [pid 68053] openat(AT_FDCWD, "output2.txt", O_WRONLY|O_CREAT|O_TRUNC|O_APPEND, 0600
<unfinished ...>

     [pid 68052] <... openat resumed>)       = 7

     [pid 68052] read(0,  <unfinished ...>

     [pid 68053] <... openat resumed>)       = 7

     [pid 68053] read(0, hello

      <unfinished ...>

     [pid 67965] <... read resumed>"hello\n", 4096) = 6

     [pid 67965] write(4, "hello\0", 6)      = 6
```

```
[pid 68052] <... read resumed>"hello\0", 4096) = 6

[pid 67965] read(0,  <unfinished ...>

[pid 68052] write(1, "hll", 3hll)          = 3

[pid 68052] write(1, "\n", 1

)           = 1

[pid 68052] write(7, "hll", 3)          = 3

[pid 68052] write(7, "\n", 1)           = 1

[pid 68052] read(0, world

 <unfinished ...>

[pid 67965] <... read resumed>"world\n", 4096) = 6

[pid 67965] write(6, "world\0", 6)      = 6

[pid 68053] <... read resumed>"world\0", 4096) = 6

[pid 67965] read(0,  <unfinished ...>

[pid 68053] write(1, "wrld", 4wrld)         = 4

[pid 68053] write(1, "\n", 1

)           = 1

[pid 68053] write(7, "wrld", 4)         = 4

[pid 68053] write(7, "\n", 1)           = 1

[pid 68053] read(0, im

 <unfinished ...>

[pid 67965] <... read resumed>"im\n", 4096) = 3

[pid 67965] write(4, "im\0", 3)         = 3

[pid 68052] <... read resumed>"im\0", 4096) = 3

[pid 67965] read(0,  <unfinished ...>

[pid 68052] write(1, "m", 1m)            = 1

[pid 68052] write(1, "\n", 1

)           = 1

[pid 68052] write(7, "m", 1)            = 1

[pid 68052] write(7, "\n", 1)           = 1

[pid 68052] read(0, testing

 <unfinished ...>

[pid 67965] <... read resumed>"testing\n", 4096) = 8

[pid 67965] write(6, "testing\0", 8)    = 8
```

```
    [pid 68053] <... read resumed>"testing\0", 4096) = 8

    [pid 67965] read(0,  <unfinished ...>

    [pid 68053] write(1, "tstng", 5tstng)        = 5

    [pid 68053] write(1, "\n", 1

    )           = 1

    [pid 68053] write(7, "tstng", 5)        = 5

    [pid 68053] write(7, "\n", 1)           = 1

    [pid 68053] read(0, this

     <unfinished ...>

    [pid 67965] <... read resumed>"this\n", 4096) = 5

    [pid 67965] write(4, "this\0", 5)       = 5

    [pid 68052] <... read resumed>"this\0", 4096) = 5

    [pid 67965] read(0,  <unfinished ...>

    [pid 68052] write(1, "ths", 3ths)          = 3

    [pid 68052] write(1, "\n", 1

    )           = 1

    [pid 68052] write(7, "ths", 3)          = 3

    [pid 68052] write(7, "\n", 1)           = 1

    [pid 68052] read(0, program

     <unfinished ...>

    [pid 67965] <... read resumed>"program\n", 4096) = 8

    [pid 67965] write(6, "program\0", 8)    = 8

    [pid 68053] <... read resumed>"program\0", 4096) = 8

    [pid 67965] read(0,  <unfinished ...>

    [pid 68053] write(1, "prgrm", 5prgrm)        = 5

    [pid 68053] write(1, "\n", 1

    )           = 1

    [pid 68053] write(7, "prgrm", 5)        = 5

    [pid 68053] write(7, "\n", 1)           = 1

    [pid 68053] read(0, ^C0x7fffa67544d0, 4096) = ? ERESTARTSYS (To be restarted if
SA_RESTART is set)

    strace: Process 67965 detached

    strace: Process 68052 detached
```

```
strace: Process 68053 detached
```

Вывод

В ходе лабораторной работы была создана программа, использующая каналы и системные вызовы для межпроцессного взаимодействия. Родительский процесс передает данные дочерним процессам через каналы, которые обрабатывают их параллельно. Программа продемонстрировала эффективное использование системных вызовов fork, pipe, dup2, read и write, а также правильную обработку ошибок и закрытие дескрипторов. В результате была разработана стабильная система для распределения и обработки данных между процессами.