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-214Б-23

Студент: Ткаченко Е.А.

Преподаватель:

Оценка: _____

Дата: 25.10.24

Москва, 2024

Постановка задачи

Вариант 19.

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 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 вероятности передает данные в соответствующий канал для одного из двух дочерних процессов (80% в первый, 20 во второй). 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

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

#define BUFSIZ 4096

void filter_data(int pipe1[], int pipe2[], char* buffer) {
    if ((rand() % 100) < 80)
    {
        write(pipe1[1], buffer, strlen(buffer) + 1);
    }
    else
    {
        write(pipe2[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);
    }



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

    int child1_status;
    int child2_status;

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

    return 0;
}
```

child1.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:

```
liza@NotebookLizaT:/mnt/c/Users/Лиза/CLionProjects/os$ ./parent
Enter filename for child1
file1.txt
Enter filename for child2
file2.txt
is aaadddd kljmklm
s dddd kljmklm
dddwsaklaaaaa
dddwskl
kjhjbiiiiaaaa
kjhjb
a lkj kjjasd
 lkj kjjsd
eadvds
dvds
^C
liza@NotebookLizaT:/mnt/c/Users/Лиза/CLionProjects/os$ cat file1.txt
kjhjb
 lkj kjjsd
liza@NotebookLizaT:/mnt/c/Users/Лиза/CLionProjects/os$ cat file2.txt
s dddd kljmklm
dddwskl
dvds
```

Strace:

```
liza@NotebookLizaT:/mnt/c/Users/Лиза/CLionProjects/os$ strace -f ./parent
execve("./parent", ["./parent"], 0x7fffc1478a58 /* 19 vars */) = 0
brk(NULL)                               = 0x7fffd42c3000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffdbac2be0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe371890000
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=16055, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 16055, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fe371894000
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"..., 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\226
\25\252\235\23<l\274\3731\3540\5\226\327"..., 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) = 0x7fe371620000
mprotect(0x7fe371648000, 2023424, PROT_NONE) = 0
mmap(0x7fe371648000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fe371648000
mmap(0x7fe3717dd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7fe3717dd000
mmap(0x7fe371836000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x215000) = 0x7fe371836000
mmap(0x7fe37183c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7fe37183c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe371610000
arch_prctl(ARCH_SET_FS, 0x7fe371610740) = 0
set_tid_address(0x7fe371610a10)         = 677
set_robust_list(0x7fe371610a20, 24)     = 0
rseq(0x7fe3716110e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)
mprotect(0x7fe371836000, 16384, PROT_READ) = 0
mprotect(0x7fe37189b000, 4096, PROT_READ) = 0
mprotect(0x7fe371888000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
munmap(0x7fe371894000, 16055)           = 0
write(1, "Enter filename for child1\n", 26Enter filename for child1
) = 26
read(0, 0x7fffdbabfc80, 4096)           = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---
read(0, 0x7fffdbabfc80, 4096)           = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---
read(0, 0x7fffdbabfc80, 4096)           = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---
```

```
read(0, 0x7fffdbabfc80, 4096)           = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---
read(0,
"\n", 4096)                     = 1
write(1, "Enter filename for child2\n", 26Enter filename for child2
) = 26
read(0,
"\n", 4096)                     = 1
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
Process 678 attached
, child_tidptr=0x7fe371610a10) = 678
[pid   678] set_robust_list(0x7fe371610a20, 24 <unfinished ...>
[pid   677] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
[pid   678] <... set_robust_list resumed>) = 0
[pid   678] getpid()                    = 678
[pid   678] dup2(3, 0)                  = 0
strace: Process 679 attached
[pid   678] execve("./child", ["./child", ""], 0x7fffdbac2db8 /* 19 vars */
<unfinished ...>
[pid   677] <... clone resumed>, child_tidptr=0x7fe371610a10) = 679
[pid   679] set_robust_list(0x7fe371610a20, 24 <unfinished ...>
[pid   677] close(3 <unfinished ...>
[pid   679] <... set_robust_list resumed>) = 0
[pid   677] <... close resumed>)        = 0
[pid   679] dup2(5, 0 <unfinished ...>
[pid   677] close(5 <unfinished ...>
[pid   679] <... dup2 resumed>)         = 0
[pid   677] <... close resumed>)        = 0
[pid   679] execve("./child", ["./child", ""], 0x7fffdbac2db8 /* 19 vars */
<unfinished ...>
[pid   677] read(0,  <unfinished ...>
[pid   678] <... execve resumed>)       = 0
[pid   678] brk(NULL)                   = 0x7fffd209e000
[pid   678] arch_prctl(0x3001 /* ARCH_??? */, 0x7fffda6eebb0) = -1 EINVAL (Invalid
argument)
[pid   678] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fb13b730000
[pid   678] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid   678] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 7
[pid   678] newfstatat(7, "", {st_mode=S_IFREG|0644, st_size=16055, ...},
AT_EMPTY_PATH) = 0
[pid   679] <... execve resumed>)       = 0
[pid   678] mmap(NULL, 16055, PROT_READ, MAP_PRIVATE, 7, 0 <unfinished ...>
[pid   679] brk(NULL <unfinished ...>
[pid   678] <... mmap resumed>)         = 0x7fb13b73c000
[pid   679] <... brk resumed>)          = 0x7fffedb22000
[pid   678] close(7 <unfinished ...>
[pid   679] arch_prctl(0x3001 /* ARCH_??? */, 0x7ffff59913e0 <unfinished ...>
[pid   678] <... close resumed>)        = 0
[pid   679] <... arch_prctl resumed>)   = -1 EINVAL (Invalid argument)
```

```
[pid   679] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid   678] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid   679] <... mmap resumed>)         = 0x7ffc53120000
[pid   678] <... openat resumed>)       = 7
[pid   679] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid   678] read(7,  <unfinished ...>
[pid   679] <... access resumed>)       = -1 ENOENT (No such file or directory)
[pid   678] <... 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"..., 832) = 832
[pid   679] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid   678] pread64(7,  <unfinished ...>
[pid   679] <... openat resumed>)       = 7
[pid   678] <... 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   679] newfstatat(7, "",  <unfinished ...>
[pid   678] pread64(7,  <unfinished ...>
[pid   679] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=16055, ...},
AT_EMPTY_PATH) = 0
[pid   678] <... 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   679] mmap(NULL, 16055, PROT_READ, MAP_PRIVATE, 7, 0 <unfinished ...>
[pid   678] pread64(7,  <unfinished ...>
[pid   679] <... mmap resumed>)         = 0x7ffc5311c000
[pid   678] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\226
\25\252\235\23<1\274\3731\3540\5\226\327"..., 68, 896) = 68
[pid   679] close(7 <unfinished ...>
[pid   678] newfstatat(7, "",  <unfinished ...>
[pid   679] <... close resumed>)        = 0
[pid   678] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...},
AT_EMPTY_PATH) = 0
[pid   679] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
[pid   678] pread64(7,  <unfinished ...>
[pid   679] <... openat resumed>)       = 7
[pid   678] <... 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   679] read(7,  <unfinished ...>
[pid   678] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 7, 0 <unfinished
...>
[pid   679] <... 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"..., 832) = 832
[pid   678] <... mmap resumed>)         = 0x7fb13b500000
[pid   679] pread64(7,  <unfinished ...>
[pid   678] mprotect(0x7fb13b528000, 2023424, PROT_NONE <unfinished ...>
[pid   679] <... 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   678] <... mprotect resumed>)     = 0
[pid   679] pread64(7,  <unfinished ...>
[pid   678] mmap(0x7fb13b528000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x28000 <unfinished ...>
[pid   679] <... 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   678] <... mmap resumed>)         = 0x7fb13b528000
[pid   679] pread64(7,  <unfinished ...>
[pid   678] mmap(0x7fb13b6bd000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1bd000 <unfinished ...>
```

```
[pid   679] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\226
\25\252\235\23<1\274\3731\3540\5\226\327"..., 68, 896) = 68
[pid   678] <... mmap resumed>)         = 0x7fb13b6bd000
[pid   679] newfstatat(7, "",  <unfinished ...>
[pid   678] mmap(0x7fb13b716000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x215000 <unfinished ...>
[pid   679] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...},
AT_EMPTY_PATH) = 0
[pid   678] <... mmap resumed>)         = 0x7fb13b716000
[pid   679] pread64(7,
"\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   678] mmap(0x7fb13b71c000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   679] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 7, 0 <unfinished
...>
[pid   678] <... mmap resumed>)         = 0x7fb13b71c000
[pid   679] <... mmap resumed>)         = 0x7ffc52eb0000
[pid   678] close(7 <unfinished ...>
[pid   679] mprotect(0x7ffc52ed8000, 2023424, PROT_NONE <unfinished ...>
[pid   678] <... close resumed>)        = 0
[pid   679] <... mprotect resumed>)     = 0
[pid   678] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid   679] mmap(0x7ffc52ed8000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x28000 <unfinished ...>
[pid   678] <... mmap resumed>)         = 0x7fb13b4f0000
[pid   679] <... mmap resumed>)         = 0x7ffc52ed8000
[pid   678] arch_prctl(ARCH_SET_FS, 0x7fb13b4f0740 <unfinished ...>
[pid   679] mmap(0x7ffc5306d000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x1bd000 <unfinished ...>
[pid   678] <... arch_prctl resumed>)   = 0
[pid   679] <... mmap resumed>)         = 0x7ffc5306d000
[pid   678] set_tid_address(0x7fb13b4f0a10 <unfinished ...>
[pid   679] mmap(0x7ffc530c6000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 7, 0x215000 <unfinished ...>
[pid   678] <... set_tid_address resumed>) = 678
[pid   679] <... mmap resumed>)         = 0x7ffc530c6000
[pid   678] set_robust_list(0x7fb13b4f0a20, 24 <unfinished ...>
[pid   679] mmap(0x7ffc530cc000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   678] <... set_robust_list resumed>) = 0
[pid   679] <... mmap resumed>)         = 0x7ffc530cc000
[pid   678] rseq(0x7fb13b4f10e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid   679] close(7 <unfinished ...>
[pid   678] <... rseq resumed>)         = -1 ENOSYS (Function not implemented)
[pid   679] <... close resumed>)        = 0
[pid   679] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x7ffc52ea0000
[pid   678] mprotect(0x7fb13b716000, 16384, PROT_READ <unfinished ...>
[pid   679] arch_prctl(ARCH_SET_FS, 0x7ffc52ea0740 <unfinished ...>
[pid   678] <... mprotect resumed>)     = 0
[pid   679] <... arch_prctl resumed>)   = 0
[pid   678] mprotect(0x7fb13b77f000, 4096, PROT_READ <unfinished ...>
[pid   679] set_tid_address(0x7ffc52ea0a10 <unfinished ...>
[pid   678] <... mprotect resumed>)     = 0
[pid   679] <... set_tid_address resumed>) = 679
```

```
[pid   678] mprotect(0x7fb13b778000, 8192, PROT_READ <unfinished ...>
[pid   679] set_robust_list(0x7ffc52ea0a20, 24 <unfinished ...>
[pid   678] <... mprotect resumed>)     = 0
[pid   679] <... set_robust_list resumed>) = 0
[pid   678] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
[pid   679] rseq(0x7ffc52ea10e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid   678] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
[pid   679] <... rseq resumed>)         = -1 ENOSYS (Function not implemented)
[pid   678] munmap(0x7fb13b73c000, 16055) = 0
[pid   679] mprotect(0x7ffc530c6000, 16384, PROT_READ <unfinished ...>
[pid   678] openat(AT_FDCWD, "", O_WRONLY|O_CREAT|O_TRUNC|O_APPEND, 0600 <unfinished
...>
[pid   679] <... mprotect resumed>)     = 0
[pid   678] <... openat resumed>)       = -1 ENOENT (No such file or directory)
[pid   679] mprotect(0x7ffc53125000, 4096, PROT_READ <unfinished ...>
[pid   678] write(2, "Error: failed to open file\n", 27Error: failed to open file
 <unfinished ...>
[pid   679] <... mprotect resumed>)     = 0
[pid   678] <... write resumed>)        = 27
[pid   679] mprotect(0x7ffc53118000, 8192, PROT_READ <unfinished ...>
[pid   678] exit_group(1 <unfinished ...>
[pid   679] <... mprotect resumed>)     = 0
[pid   678] <... exit_group resumed>)   = ?
[pid   679] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
[pid   677] <... read resumed>0x7fffdbabfc80, 4096) = ? ERESTARTSYS (To be restarted
if SA_RESTART is set)
[pid   678] +++ exited with 1 +++
[pid   677] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=678,
si_uid=1000, si_status=1, si_utime=0, si_stime=0} ---
[pid   679] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
[pid   677] read(0,  <unfinished ...>
[pid   679] munmap(0x7ffc5311c000, 16055) = 0
[pid   679] openat(AT_FDCWD, "", O_WRONLY|O_CREAT|O_TRUNC|O_APPEND, 0600) = -1 ENOENT
(No such file or directory)
[pid   679] write(2, "Error: failed to open file\n", 27Error: failed to open file
) = 27
[pid   679] exit_group(1)               = ?
[pid   679] +++ exited with 1 +++
<... read resumed>0x7fffdbabfc80, 4096) = ? ERESTARTSYS (To be restarted if SA_RESTART
is set)
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=679, si_uid=1000,
si_status=1, si_utime=0, si_stime=0} ---
read(0,
"\n", 4096)                     = 1
close(4)                                = 0
close(6)                                = 0
wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 1}], 0, NULL) = 678
wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 1}], 0, NULL) = 679
exit_group(0)                           = ?
+++ exited with 0 +++
```

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.