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-216БВ-24

Студент: Настина М.О.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 11.10.24

Москва, 2024

Постановка задачи

Вариант 18.

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решения задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe).
Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы. 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.
Правило фильтрации: нечетные строки отправляются в pipe1, четные в pipe2. Дочерние процессы удаляют все гласные из строк.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork(void); – создает дочерний процесс.
- int pipe(int *fd); – создание неименованного канала для передачи данных между процессами
- int close(int fd) - закрыть файловый дескриптор
- int execl(const char *path, const char *arg, ...) - замена образа памяти процесса
- ssize_t write(int fd, const void *buf, size_t count) - запись данных в файловый дескриптор
- pid_t waitpid(pid_t pid, int *status, int options) - ожидание завершения конкретного дочернего процесса
- void exit(int status) - завершение выполнения процесса и возвращение статуса

В рамках лабораторной работы была разработана многопроцессная система для обработки текстовых данных с использованием межпроцессного взаимодействия через неименованные каналы. Программа состоит из двух частей: родительского процесса (parent.c), который управляет распределением данных, и дочернего процесса (child.c), выполняющего фильтрацию строк.

Родительский процесс начинает работу с запроса у пользователя имен двух файлов для записи результатов обработки. Первая введенная строка определяет имя файла для первого дочернего процесса (child1), вторая строка - для второго дочернего процесса (child2). После получения имен файлов родительский процесс создает два неименованных канала (pipe1 и pipe2) для организации взаимодействия с дочерними процессами.

Затем с помощью системных вызовов fork() создаются два дочерних процесса. Каждый дочерний процесс наследует открытые файловые дескрипторы каналов. В дочерних процессах происходит настройка дескрипторов: закрываются неиспользуемые концы каналов, и с помощью execl() происходит замена образа процесса на программу child, которой передаются параметры - имя файла для записи и файловый дескриптор для чтения данных.

Дочерние процессы представляют собой программу-фильтр, которая читает строки из переданного файлового дескриптора (читающего конца канала) и удаляет из них все гласные буквы английского алфавита. Обработанные строки выводятся на стандартный вывод с пометкой "Child processing", а также записываются в соответствующий файл. Для работы с текстом реализована функция is_vowel(), определяющая гласные буквы, и remove_vowels(), выполняющая их удаление из строки.

Родительский процесс тем временем читает строки произвольной длины из стандартного ввода и распределяет их между каналами согласно правилу фильтрации: нечетные по счету строки отправляются в pipe1 (для child1), четные - в pipe2 (для child2). После завершения ввода данных родительский процесс закрывает записывающие концы каналов и ожидает завершения работы дочерних процессов с помощью waitpid(). Программа корректно обрабатывает системные ошибки, которые могут возникнуть при работе с каналами, процессами и файлами.

Код программы

parent.c

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <errno.h>


int main() {

    char *file1 = NULL;
```

```
char *file2 = NULL;
size_t len = 0;
if (getline(&file1, &len, stdin) == -1) {
    if (errno != 0) perror("getline file1");
    exit(1);
}
file1[strcspn(file1, "\n")] = '\0';
len = 0;
if (getline(&file2, &len, stdin) == -1) {
    if (errno != 0) perror("getline file2");
    free(file1);
    exit(1);
}
file2[strcspn(file2, "\n")] = '\0';
int pipe1[2];
int pipe2[2];
if (pipe(pipe1) == -1) {
    perror("pipe1");
    free(file1);
    free(file2);
    exit(1);
}
if (pipe(pipe2) == -1) {
    perror("pipe2");
    free(file1);
    free(file2);
    exit(1);
}
pid_t pid1 = fork();
if (pid1 == -1) {
```

```
    perror("fork1");
    free(file1);
    free(file2);
    exit(1);
}
if (pid1 == 0) {
    close(pipe1[1]);
    close(pipe2[0]);
    close(pipe2[1]);
    char fd_str[10];
    sprintf(fd_str, "%d", pipe1[0]);
    execl("./child", "child", file1, fd_str, (char *)NULL);
    perror("execl child1");
    exit(1);
}

close(pipe1[0]);
pid_t pid2 = fork();
if (pid2 == -1) {
    perror("fork2");
    free(file1);
    free(file2);
    exit(1);
}
if (pid2 == 0) {
    close(pipe2[1]);
    close(pipe1[0]);
    close(pipe1[1]);
    char fd_str[10];
    sprintf(fd_str, "%d", pipe2[0]);
```

```
    execl("./child", "child", file2, fd_str, (char *)NULL);
    perror("execl child2");
    exit(1);
}

close(pipe2[0]);
char *line = NULL;
size_t line_len = 0;
int line_num = 1;
ssize_t read_bytes;
while ((read_bytes = getline(&line, &line_len, stdin)) != -1) {
    int pipe_fd = (line_num % 2 == 1) ? pipe1[1] : pipe2[1];
    ssize_t written = write(pipe_fd, line, read_bytes);
    if (written != read_bytes) {
        perror("write to pipe incomplete");
        free(line);
        free(file1);
        free(file2);
        exit(1);
    }
    line_num++;
}
if (errno != 0) {
    perror("getline input");
}
free(line);
close(pipe1[1]);
close(pipe2[1]);
waitpid(pid1, NULL, 0);
waitpid(pid2, NULL, 0);
```

```
    free(file1);
    free(file2);
    return 0;
}
```

child.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <ctype.h>

int is_vowel(char c) {
    c = tolower(c);
    return (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');
}
char *remove_vowels(const char *str) {
    size_t len = strlen(str);
    char *result = malloc(len + 1);
    if (!result) {
        return NULL;
    }
    char *p = result;
    while (*str) {
        if (!is_vowel(*str)) {
            *p++ = *str;
        }
        str++;
    }
    *p = '\0';
    return result;
```

```
}
int main(int argc, char **argv) {
    if (argc != 3) {
        fprintf(stderr, "Usage: child <filename> <fd>\n");
        return 1;
    }

    char *filename = argv[1];
    int fd = atoi(argv[2]);

    FILE *in = fdopen(fd, "r");
    if (!in) {
        perror("fdopen");
        return 1;
    }

    FILE *out = fopen(filename, "w");
    if (!out) {
        perror("fopen");
        fclose(in);
        return 1;
    }

    char *line = NULL;
    size_t len = 0;
    while (getline(&line, &len, in) != -1) {
        char *processed = remove_vowels(line);
        if (!processed) {
            perror("malloc");
            break;
```

```
        }
        printf("Child processing: %s", processed);
        fflush(stdout);
        fprintf(out, "%s", processed);
        fflush(out);
        free(processed);
    }
    free(line);
    fclose(in);
    fclose(out);
    return 0;
}
```

Протокол работы программы

Тестирование:

Ввод:

./parent

file1.txt

file2.txt

Hello World

Child processing: Hll Wrld

This is a test

Child processing: Ths s tst

Programming is fun

Child processing: Prgrmmng s fn

Lab work completed

Child processing: Lb wrk cmpltd

Содержимое file1.txt:

Hll Wrld

Prgrmmng s fn

Содержимое file2.txt:

Ths s tst

Lb wrk cmpltd

Strace:

4353 execve("./parent", ["./parent"], 0x7ffcff12e7b8 /* 37 vars */) = 0

4353 brk(NULL)   = 0x557bd4fe7000

4353 mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8195da3000

4353 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)

4353 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

4353 fstat(3, {st_mode=S_IFREG|0644, st_size=23175, ...}) = 0

4353 mmap(NULL, 23175, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8195d9d000

4353 close(3)   = 0

4353 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

4353 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0"..., 832) = 832

4353 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

4353 fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

4353 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

4353 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f8195b8b000

4353 mmap(0x7f8195bb3000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f8195bb3000

4353 mmap(0x7f8195d3b000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7f8195d3b000

4353 mmap(0x7f8195d8a000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7f8195d8a000

```
4353  mmap(0x7f8195d90000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f8195d90000

4353  close(3)                  = 0

4353  mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f8195b88000

4353  arch_prctl(ARCH_SET_FS, 0x7f8195b88740) = 0

4353  set_tid_address(0x7f8195b88a10)   = 4353

4353  set_robust_list(0x7f8195b88a20, 24) = 0

4353  rseq(0x7f8195b89060, 0x20, 0, 0x53053053) = 0

4353  mprotect(0x7f8195d8a000, 16384, PROT_READ) = 0

4353  mprotect(0x557bc383e000, 4096, PROT_READ) = 0

4353  mprotect(0x7f8195ddb000, 8192, PROT_READ) = 0

4353  prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

4353  munmap(0x7f8195d9d000, 23175)     = 0

4353  getrandom("\xbb\x76\xe9\x33\x02\x6a\x6d\x42", 8, GRND_NONBLOCK) = 8

4353  brk(NULL)                  = 0x557bd4fe7000

4353  brk(0x557bd5008000)              = 0x557bd5008000

4353  fstat(0, {st_mode=S_IFREG|0644, st_size=151, ...}) = 0

4353  read(0, "file1.txt\nfile2.txt\nHello World,"..., 4096) = 151

4353  pipe2([3, 4], 0)               = 0

4353  pipe2([5, 6], 0)               = 0

4353  clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f8195b88a10) = 4354

4354  set_robust_list(0x7f8195b88a20, 24 <unfinished ...>

4353  close(3 <unfinished ...>

4354  <... set_robust_list resumed>)    = 0

4353  <... close resumed>)             = 0
```

```
4353  clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
4354  close(4)                  = 0
4354  close(5 <unfinished ...>
4353  <... clone resumed>, child_tidptr=0x7f8195b88a10) = 4355
4354  <... close resumed>)      = 0
4355  set_robust_list(0x7f8195b88a20, 24 <unfinished ...>
4353  close(5 <unfinished ...>
4355  <... set_robust_list resumed>)    = 0
4354  close(6 <unfinished ...>
4353  <... close resumed>)      = 0
4355  close(6 <unfinished ...>
4354  <... close resumed>)      = 0
4353  write(4, "Hello World, this is first line\n", 32 <unfinished ...>
4355  <... close resumed>)      = 0
4354  execve("./child", ["child", "file1.txt", "3"], 0x7fff79c7d618 /* 37 vars */ <unfinished ...>
4353  <... write resumed>)      = 32
4355  close(3 <unfinished ...>
4353  write(6, "This is the second test line\n", 29 <unfinished ...>
4355  <... close resumed>)      = -1 EBADF (Неправильный дескриптор файла)
4353  <... write resumed>)      = 29
4355  close(4 <unfinished ...>
4353  write(4, "Third line for processing\n", 26 <unfinished ...>
4355  <... close resumed>)      = 0
4353  <... write resumed>)      = 26
4353  write(6, "Another example string\n", 23 <unfinished ...>
4355  execve("./child", ["child", "file2.txt", "5"], 0x7fff79c7d618 /* 37 vars */ <unfinished ...>
4353  <... write resumed>)      = 23
```

```
4353  write(4, "Final test line here\n", 21) = 21
4353  read(0,  <unfinished ...>
4354  <... execve resumed>)            = 0
4353  <... read resumed>"", 4096)       = 0
4355  <... execve resumed>)            = 0
4353  close(4 <unfinished ...>
4354  brk(NULL <unfinished ...>
4353  <... close resumed>)             = 0
4355  brk(NULL <unfinished ...>
4353  close(6 <unfinished ...>
4354  <... brk resumed>)               = 0x559a09003000
4353  <... close resumed>)             = 0
4355  <... brk resumed>)               = 0x55b466f73000
4353  wait4(4354,  <unfinished ...>
4354  mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
4355  mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
4354  <... mmap resumed>)              = 0x7fe582e82000
4355  <... mmap resumed>)              = 0x7f36376c3000
4354  access("/etc/ld.so.preload", R_OK <unfinished ...>
4355  access("/etc/ld.so.preload", R_OK <unfinished ...>
4354  <... access resumed>)            = -1 ENOENT (Нет такого файла или каталога)
4355  <... access resumed>)            = -1 ENOENT (Нет такого файла или каталога)
4354  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
4355  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
4354  <... openat resumed>)            = 4
4355  <... openat resumed>)            = 3
```

```
4354  fstat(4,  <unfinished ...>

4355  fstat(3,  <unfinished ...>

4354  <... fstat resumed>{st_mode=S_IFREG|0644, st_size=23175, ...}) = 0

4355  <... fstat resumed>{st_mode=S_IFREG|0644, st_size=23175, ...}) = 0

4354  mmap(NULL, 23175, PROT_READ, MAP_PRIVATE, 4, 0 <unfinished ...>

4355  mmap(NULL, 23175, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>

4354  <... mmap resumed>)              = 0x7fe582e7c000

4355  <... mmap resumed>)              = 0x7f36376bd000

4354  close(4 <unfinished ...>

4355  close(3 <unfinished ...>

4354  <... close resumed>)             = 0

4355  <... close resumed>)             = 0

4355  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>

4354  openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>

4355  <... openat resumed>)            = 3

4354  <... openat resumed>)            = 4

4355  read(3,  <unfinished ...>

4354  read(4,  <unfinished ...>

4355  <... read resumed>"\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

4354  <... read resumed>"\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

4355  pread64(3,  <unfinished ...>

4354  pread64(4,  <unfinished ...>

4355  <... 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

4354  <... 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
```

4355 fstat(3, <unfinished ...>

4354 fstat(4, <unfinished ...>

4355 <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

4354 <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

4355 pread64(3, <unfinished ...>

4354 pread64(4, <unfinished ...>

4355 <... pread64 resumed>"\6\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

4354 <... pread64 resumed>"\6\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

4355 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>

4354 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 4, 0 <unfinished ...>

4355 <... mmap resumed>) = 0x7f36374ab000

4354 <... mmap resumed>) = 0x7fe582c6a000

4355 mmap(0x7f36374d3000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>

4354 mmap(0x7fe582c92000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x28000 <unfinished ...>

4355 <... mmap resumed>) = 0x7f36374d3000

4355 mmap(0x7f363765b000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000 <unfinished ...>

4354 <... mmap resumed>) = 0x7fe582c92000

4355 <... mmap resumed>) = 0x7f363765b000

4354 mmap(0x7fe582e1a000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1b0000 <unfinished ...>

4355 mmap(0x7f36376aa000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000 <unfinished ...>

4354 <... mmap resumed>) = 0x7fe582e1a000

4355 <... mmap resumed>) = 0x7f36376aa000

```
4354  mmap(0x7fe582e69000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 4, 0x1fe000 <unfinished ...>

4355  mmap(0x7f36376b0000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

4354  <... mmap resumed>)             = 0x7fe582e69000

4355  <... mmap resumed>)             = 0x7f36376b0000

4354  mmap(0x7fe582e6f000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

4355  close(3 <unfinished ...>

4354  <... mmap resumed>)             = 0x7fe582e6f000

4355  <... close resumed>)            = 0

4355  mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

4354  close(4 <unfinished ...>

4355  <... mmap resumed>)             = 0x7f36374a8000

4354  <... close resumed>)            = 0

4355  arch_prctl(ARCH_SET_FS, 0x7f36374a8740 <unfinished ...>

4354  mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>

4355  <... arch_prctl resumed>)       = 0

4354  <... mmap resumed>)             = 0x7fe582c67000

4355  set_tid_address(0x7f36374a8a10)   = 4355

4354  arch_prctl(ARCH_SET_FS, 0x7fe582c67740 <unfinished ...>

4355  set_robust_list(0x7f36374a8a20, 24 <unfinished ...>

4354  <... arch_prctl resumed>)       = 0

4355  <... set_robust_list resumed>)    = 0

4354  set_tid_address(0x7fe582c67a10 <unfinished ...>

4355  rseq(0x7f36374a9060, 0x20, 0, 0x53053053 <unfinished ...>

4354  <... set_tid_address resumed>)    = 4354

4355  <... rseq resumed>)             = 0
```

```
4354  set_robust_list(0x7fe582c67a20, 24) = 0
4355  mprotect(0x7f36376aa000, 16384, PROT_READ <unfinished ...>
4354  rseq(0x7fe582c68060, 0x20, 0, 0x53053053 <unfinished ...>
4355  <... mprotect resumed>)          = 0
4354  <... rseq resumed>)              = 0
4355  mprotect(0x55b45959e000, 4096, PROT_READ) = 0
4354  mprotect(0x7fe582e69000, 16384, PROT_READ <unfinished ...>
4355  mprotect(0x7f36376fb000, 8192, PROT_READ <unfinished ...>
4354  <... mprotect resumed>)          = 0
4355  <... mprotect resumed>)          = 0
4354  mprotect(0x5599c9334000, 4096, PROT_READ <unfinished ...>
4355  prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
4354  <... mprotect resumed>)          = 0
4355  munmap(0x7f36376bd000, 23175 <unfinished ...>
4354  mprotect(0x7fe582eba000, 8192, PROT_READ <unfinished ...>
4355  <... munmap resumed>)            = 0
4354  <... mprotect resumed>)          = 0
4355  fcntl(5, F_GETFL <unfinished ...>
4354  prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
4355  <... fcntl resumed>)             = 0 (flags O_RDONLY)
4355  getrandom( <unfinished ...>
4354  <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
4355  <... getrandom resumed>"\x81\x53\x10\xde\x51\x7e\xc1\x6e", 8, GRND_NONBLOCK) = 8
4355  brk(NULL <unfinished ...>
4354  munmap(0x7fe582e7c000, 23175 <unfinished ...>
4355  <... brk resumed>)               = 0x55b466f73000
4355  brk(0x55b466f94000 <unfinished ...>
```

```
4354  <... munmap resumed>)             = 0
4355  <... brk resumed>)                = 0x55b466f94000
4355  openat(AT_FDCWD, "file2.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666 <unfinished
...>
4354  fcntl(3, F_GETFL)                 = 0 (flags O_RDONLY)
4354  getrandom("\x7b\x8d\x3e\x17\xa5\xd9\xb0\xba", 8, GRND_NONBLOCK) = 8
4354  brk(NULL <unfinished ...>
4355  <... openat resumed>)             = 3
4354  <... brk resumed>)                = 0x559a09003000
4355  fstat(5,  <unfinished ...>
4354  brk(0x559a09024000 <unfinished ...>
4355  <... fstat resumed>{st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
4354  <... brk resumed>)                = 0x559a09024000
4355  read(5,  <unfinished ...>
4354  openat(AT_FDCWD, "file1.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666 <unfinished
...>
4355  <... read resumed>"This is the second test line\nAno"..., 4096) = 52
4355  fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}) = 0
4355  write(1, "Child processing: Ths s th scnd "..., 39) = 39
4355  fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
4355  write(3, "Ths s th scnd tst ln\n", 21 <unfinished ...>
4354  <... openat resumed>)             = 4
4355  <... write resumed>)              = 21
4354  fstat(3,  <unfinished ...>
4355  write(1, "Child processing: nthr xmpl strn"..., 34) = 34
4354  <... fstat resumed>{st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
4355  write(3, "nthr xmpl strng\n", 16 <unfinished ...>
4354  read(3,  <unfinished ...>
```

```
4355  <... write resumed>)            = 16

4354  <... read resumed>"Hello World, this is first line\n"..., 4096) = 79

4355  read(5, "", 4096)               = 0

4354  fstat(1,  <unfinished ...>

4355  close(5)                        = 0

4354  <... fstat resumed>{st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x5), ...}) = 0

4355  close(3 <unfinished ...>

4354  write(1, "Child processing: Hll Wrld, ths "..., 42) = 42

4354  fstat(4,  <unfinished ...>

4355  <... close resumed>)            = 0

4354  <... fstat resumed>{st_mode=S_IFREG|0644, st_size=0, ...}) = 0

4355  exit_group(0 <unfinished ...>

4354  write(4, "Hll Wrld, ths s frst ln\n", 24 <unfinished ...>

4355  <... exit_group resumed>)        = ?

4354  <... write resumed>)            = 24

4355  +++ exited with 0 +++

4354  write(1, "Child processing: Thrd ln fr prc"..., 37 <unfinished ...>

4353  <... wait4 resumed>NULL, 0, NULL) = ? ERESTARTSYS (To be restarted if
SA_RESTART is set)

4354  <... write resumed>)            = 37

4353  --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=4355, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---

4354  write(4, "Thrd ln fr prcssng\n", 19 <unfinished ...>

4353  wait4(4354,  <unfinished ...>

4354  <... write resumed>)            = 19

4354  write(1, "Child processing: Fnl tst ln hr\n", 32) = 32

4354  write(4, "Fnl tst ln hr\n", 14)   = 14

4354  read(3, "", 4096)               = 0
```

```
4354  close(3)                  = 0

4354  close(4)                  = 0

4354  exit_group(0)             = ?

4354  +++ exited with 0 +++

4353  <... wait4 resumed>NULL, 0, NULL) = 4354

4353  --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=4354, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---

4353  wait4(4355, NULL, 0, NULL)      = 4355

4353  exit_group(0)             = ?

4353  +++ exited with 0 +++
```

Вывод

Программа успешно создает два дочерних процесса и организует межпроцессное взаимодействие через неименованные каналы pipe. Родительский процесс корректно распределяет вводимые строки между процессами согласно правилу фильтрации: нечетные строки направляются в первый дочерний процесс, четные - во второй. Оба дочерних процесса успешно удаляют все гласные буквы из полученных строк и записывают результаты как в стандартный вывод, так и в соответствующие файлы.

Основные трудности возникли с правильной организацией закрытия файловых дескрипторов pipe в дочерних процессах, чтобы избежать утечек ресурсов. Также потребовалось тщательно проработать обработку ошибок при системных вызовах fork(), pipe() и exec(), чтобы программа устойчиво работала в различных сценариях.