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: Иванов И.П.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 06.10.25

Москва, 2025

Постановка задачи

Вариант 12.

Родительский процесс создает два дочерних процесса. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Child1 и Child2 можно «соединить» между собой дополнительным каналом. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child1 и child2 производят работу над строками. Child2 пересылает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода.

Child1 переводит строки в верхний регистр. Child2 убирает все задвоенные пробелы.

Общий метод и алгоритм решения

Использованные системные вызовы:

- *pid_t fork(void)*; – создает дочерний процесс.
- *int pipe(int* fd)*; – создает однонаправленный канал для межпроцессного взаимодействия.
- *int execl(const char* path, const char* arg, ...)*; – заменяет образ текущей программы, на указанную, принимая аргументы в качестве списка.
- *int dup2(int oldfd, int newfd)*; – создает копию файлового дескриптора *oldfd* в указанном дескрипторе *newfd*.
- *ssize_t write(int fd, const void* buf, size_t count)*; – записывает данные из буфера в файловый дескриптор.
- *int close(int fd)*; – закрывает файловый дескриптор.
- *pid_t waitpid(pid_t pid, int *status, int options)*; – ожидает изменения состояния указанного процесса.

Общий метод решения заключается в организации взаимодействия между процессами при помощи каналов, где родительский процесс передает строки первому дочернему процессу. Первый дочерний процесс преобразует полученные данные в верхний регистр и передает результат второму дочернему процессу. Второй дочерний процесс получает обработанные строки и удаляет из них все задвоенные пробелы, после чего возвращает финальные данные родительскому процессу. Алгоритм построен на последовательной обработке данных по конвейеру с контролируемым чтением и записью через пайпы, обеспечивающие обмен информацией.

Код программы

main.c

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

int main(int argc, char *argv[])
{
   if (argc != 1 && argv)
   {
      return 1;
   }

   int parent_to_child1[2];
   if (pipe(parent_to_child1) == -1)
   {
      const char error_msg[] = "Error: unable to create parent_to_child1 pipe\n";
      write(STDERR_FILENO, error_msg, sizeof(error_msg));
      exit(EXIT_FAILURE);
   }

   int child1_to_child2[2];
   if (pipe(child1_to_child2) == -1)
   {
      const char error_msg[] = "Error: unable to create child1_to_child2 pipe\n";
      write(STDERR_FILENO, error_msg, sizeof(error_msg));
      exit(EXIT_FAILURE);
   }

   int child2_to_parent[2];
   if (pipe(child2_to_parent) == -1)
   {
      const char error_msg[] = "Error: unable to create child2_to_parent pipe\n";
      write(STDERR_FILENO, error_msg, sizeof(error_msg));
      exit(EXIT_FAILURE);
   }

   pid_t child1_id = fork();
   if (child1_id == -1)
   {
      const char error_msg[] = "Error: unable to create child1\n";
      write(STDERR_FILENO, error_msg, sizeof(error_msg));
      exit(EXIT_FAILURE);
   }
   if (child1_id == 0)
   {
      close(parent_to_child1[1]);
      dup2(parent_to_child1[0], STDIN_FILENO);
      close(parent_to_child1[0]);

      close(child1_to_child2[0]);
      dup2(child1_to_child2[1], STDOUT_FILENO);
      close(child1_to_child2[1]);
```

```
    close(child2_to_parent[0]);
    close(child2_to_parent[1]);

    execl("./child1", "child1", NULL);

    const char error_msg[] = "Error: execl child1 failed\n";
    write(STDERR_FILENO, error_msg, sizeof(error_msg));
    exit(EXIT_FAILURE);
}

pid_t child2_id = fork();
if (child2_id == -1)
{
    const char error_msg[] = "Error: unable to create child2\n";
    write(STDERR_FILENO, error_msg, sizeof(error_msg));
    exit(EXIT_FAILURE);
}
if (child2_id == 0)
{
    close(child1_to_child2[1]);
    dup2(child1_to_child2[0], STDIN_FILENO);
    close(child1_to_child2[0]);

    close(child2_to_parent[0]);
    dup2(child2_to_parent[1], STDOUT_FILENO);
    close(child2_to_parent[1]);

    close(parent_to_child1[0]);
    close(parent_to_child1[1]);

    execl("./child2", "child2", NULL);
    const char error_msg[] = "Error: execl child2 failed\n";

    write(STDERR_FILENO, error_msg, sizeof(error_msg));
    exit(EXIT_FAILURE);
}

close(parent_to_child1[0]);
close(child1_to_child2[0]);
close(child1_to_child2[1]);
close(child2_to_parent[1]);

const char msg[] = "Input text:\n";
write(STDOUT_FILENO, msg, sizeof(msg) - 1);

char buffer[128];
ssize_t bytes_read;

while ((bytes_read = read(STDIN_FILENO, buffer, sizeof(buffer) - 1)) > 0)
{
    write(parent_to_child1[1], buffer, bytes_read);

    ssize_t result_read = read(child2_to_parent[0], buffer, sizeof(buffer) - 1);
    if (result_read > 0)
    {
        buffer[result_read] = '\0';
        write(STDOUT_FILENO, "Processed string: ", 18);
        write(STDOUT_FILENO, buffer, result_read);
    }
```

```
    }

    close(parent_to_child1[1]);
    close(child2_to_parent[0]);

    int status;
    waitpid(child1_id, &status, 0);
    waitpid(child2_id, &status, 0);

    return 0;
}
```

child1.c

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

int main(int argc, char *argv[])
{
    if (argc != 1 && argv)
    {
        const char msg[] = "Error: invalid argument count\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        return 1;
    }

    char buffer[256];
    ssize_t bytes_read;

    while ((bytes_read = read(STDIN_FILENO, buffer, sizeof(buffer) - 1)) > 0)
    {
        buffer[bytes_read] = '\0';

        for (ssize_t i = 0; i < bytes_read; ++i)
        {
            buffer[i] = toupper((unsigned char)buffer[i]);
        }

        ssize_t bytes_written = write(STDOUT_FILENO, buffer, bytes_read);
        if (bytes_written != bytes_read)
        {
            const char msg[] = "Error: unable to write to pipe\n";
            write(STDERR_FILENO, msg, sizeof(msg) - 1);
            return 1;
        }
    }

    if (bytes_read < 0)
    {
        const char msg[] = "Error: unable to read from pipe\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        return 1;
    }
```

```
    return 0;
}
```

child2.c

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

int main(int argc, char *argv[])
{
        if (argc != 1 && argv)
        {
            const char msg[] = "Error: invalid argument count\n";
            write(STDERR_FILENO, msg, sizeof(msg) - 1);
            return 1;
        }

        char buffer[256];
        ssize_t bytes_read;

        while ((bytes_read = read(STDIN_FILENO, buffer, sizeof(buffer) - 1)) > 0)
        {
            buffer[bytes_read] = '\0';

            char result[256];
            int i = 0, j = 0;
            int space_found = 0;
            while (buffer[i] != '\0')
            {
                if (buffer[i] == ' ')
                {
                    if (!space_found)
                    {
                        result[j++] = ' ';
                        space_found = 1;
                    }
                }
                else
                {
                    result[j++] = buffer[i];
                    space_found = 0;
                }
                i++;
            }
            result[j] = '\0';

            ssize_t to_write = j;
            ssize_t bytes_written = write(STDOUT_FILENO, result, to_write);
            if (bytes_written != to_write)
            {
                const char msg[] = "Error: unable to write to pipe\n";
                write(STDERR_FILENO, msg, sizeof(msg) - 1);
                return 1;
            }
        }
```

```
    if (bytes_read < 0)
    {
        const char msg[] = "Error: unable to read from pipe\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        return 1;
    }

    return 0;
}
```

Протокол работы программы

Тестирование:

Strace:

```
strace -f ./main
execve("./main", ["./main"], 0x7fff2f910468 /* 65 vars */) = 0
brk(NULL)                               = 0x55fb2f1d5000
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
fstat(3, {st_mode=S_IFREG|0644, st_size=157675, ...}) = 0
mmap(NULL, 157675, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2b5aff0000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/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\0000x\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"..., 896, 64) = 896
fstat(3, {st_mode=S_IFREG|0755, st_size=2149728, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f2b5afee000
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"..., 896, 64) = 896
mmap(NULL, 2174000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f2b5ac00000
mmap(0x7f2b5ac24000, 1515520, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000) = 0x7f2b5ac24000
mmap(0x7f2b5ad96000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x196000) = 0x7f2b5ad96000
mmap(0x7f2b5ae05000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x204000) = 0x7f2b5ae05000
mmap(0x7f2b5ae0b000, 31792, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f2b5ae0b000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f2b5afeb000
arch_prctl(ARCH_SET_FS, 0x7f2b5afeb740) = 0
set_tid_address(0x7f2b5afeba10)         = 20542
set_robust_list(0x7f2b5afeba20, 24)     = 0
rseq(0x7f2b5afeb680, 0x20, 0, 0x53053053) = 0
mprotect(0x7f2b5ae05000, 16384, PROT_READ) = 0
mprotect(0x55fb02eb1000, 4096, PROT_READ) = 0
mprotect(0x7f2b5b056000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
getrandom("\x25\x86\x28\x52\xa4\x69\xf6\x76", 8, GRND_NONBLOCK) = 8
munmap(0x7f2b5aff0000, 157675)          = 0
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
pipe2([7, 8], 0)                        = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7f2b5afeba10) = 20543
strace: Process 20543 attached
[pid 20542] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
[pid 20543] set_robust_list(0x7f2b5afeba20, 24 <unfinished ...>
[pid 20542] rt_sigprocmask(SIG_BLOCK, ~[] <unfinished ...>
[pid 20543] <... set_robust_list resumed>) = 0
[pid 20542] <... rt_sigprocmask resumed>, [], 8) = 0
[pid 20542] clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
[pid 20543] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
[pid 20542] <... clone resumed>, child_tidptr=0x7f2b5afeba10) = 20544
strace: Process 20544 attached
[pid 20542] rt_sigprocmask(SIG_SETMASK, [] <unfinished ...>
```

[pid 20544] set_robust_list(0x7f2b5afeba20, 24 <unfinished ...>
[pid 20543] close(4 <unfinished ...>
[pid 20542] <... rt_sigprocmask resumed>, NULL, 8) = 0
[pid 20544] <... set_robust_list resumed>) = 0
[pid 20543] <... close resumed>) = 0
[pid 20542] close(3 <unfinished ...>
[pid 20543] dup2(3, 0 <unfinished ...>
[pid 20542] <... close resumed>) = 0
[pid 20544] rt_sigprocmask(SIG_SETMASK, [] <unfinished ...>
[pid 20542] close(5 <unfinished ...>
[pid 20543] <... dup2 resumed>) = 0
[pid 20542] <... close resumed>) = 0
[pid 20544] <... rt_sigprocmask resumed>, NULL, 8) = 0
[pid 20542] close(6 <unfinished ...>
[pid 20543] close(3 <unfinished ...>
[pid 20542] <... close resumed>) = 0
[pid 20543] <... close resumed>) = 0
[pid 20542] close(8 <unfinished ...>
[pid 20543] close(5 <unfinished ...>
[pid 20542] <... close resumed>) = 0
[pid 20544] close(6 <unfinished ...>
[pid 20542] write(1, "Input text:\n", 12 <unfinished ...>
Input text:
[pid 20543] <... close resumed>) = 0
[pid 20542] <... write resumed>) = 12
[pid 20544] <... close resumed>) = 0
[pid 20542] read(0 <unfinished ...>
[pid 20543] dup2(6, 1 <unfinished ...>
[pid 20544] dup2(5, 0 <unfinished ...>
[pid 20543] <... dup2 resumed>) = 1
[pid 20544] <... dup2 resumed>) = 0
[pid 20543] close(6 <unfinished ...>
[pid 20544] close(5 <unfinished ...>
[pid 20543] <... close resumed>) = 0
[pid 20544] <... close resumed>) = 0
[pid 20543] close(7 <unfinished ...>
[pid 20544] close(7 <unfinished ...>
[pid 20543] <... close resumed>) = 0
[pid 20544] <... close resumed>) = 0
[pid 20543] close(8 <unfinished ...>
[pid 20544] dup2(8, 1 <unfinished ...>
[pid 20543] <... close resumed>) = 0
[pid 20544] <... dup2 resumed>) = 1
[pid 20544] close(8 <unfinished ...>
[pid 20543] execve("./child1", ["child1"], 0x7fff21a28938 /* 65 vars */ <unfinished ...>
[pid 20544] <... close resumed>) = 0
[pid 20544] close(3) = 0
[pid 20544] close(4) = 0
[pid 20544] execve("./child2", ["child2"], 0x7fff21a28938 /* 65 vars */ <unfinished ...>
[pid 20543] <... execve resumed>) = 0
[pid 20543] brk(NULL) = 0x55d913bed000
[pid 20543] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid 20544] <... execve resumed>) = 0
[pid 20543] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 20544] brk(NULL <unfinished ...>
[pid 20543] <... openat resumed>) = 3
[pid 20544] <... brk resumed>) = 0x55c267827000
[pid 20543] fstat(3, {st_mode=S_IFREG|0644, st_size=157675, ...}) = 0

[pid 20543] mmap(NULL, 157675, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid 20544] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid 20543] <... mmap resumed>) = 0x7f29e04a0000
[pid 20544] <... access resumed>) = -1 ENOENT (No such file or directory)
[pid 20543] close(3 <unfinished ...>
[pid 20544] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 20543] <... close resumed>) = 0
[pid 20544] <... openat resumed>) = 3
[pid 20544] fstat(3 <unfinished ...>
[pid 20543] openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 20544] <... fstat resumed>, {st_mode=S_IFREG|0644, st_size=157675, ...}) = 0
[pid 20543] <... openat resumed>) = 3
[pid 20544] mmap(NULL, 157675, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid 20543] read(3 <unfinished ...>
[pid 20544] <... mmap resumed>) = 0x7f4c8be0f000
[pid 20543] <... read resumed>, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000x\2\0\0\0\0\0"..., 832) = 832
[pid 20544] close(3 <unfinished ...>
[pid 20543] pread64(3 <unfinished ...>
[pid 20544] <... close resumed>) = 0
[pid 20543] <... 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"..., 896, 64) = 896
[pid 20544] openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid 20543] fstat(3 <unfinished ...>
[pid 20544] <... openat resumed>) = 3
[pid 20543] <... fstat resumed>, {st_mode=S_IFREG|0755, st_size=2149728, ...}) = 0
[pid 20544] read(3 <unfinished ...>
[pid 20543] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 20544] <... read resumed>, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000x\2\0\0\0\0\0"..., 832) = 832
[pid 20543] <... mmap resumed>) = 0x7f29e049e000
[pid 20544] 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"..., 896, 64) = 896
[pid 20543] pread64(3 <unfinished ...>
[pid 20544] fstat(3 <unfinished ...>
[pid 20543] <... 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"..., 896, 64) = 896
[pid 20544] <... fstat resumed>, {st_mode=S_IFREG|0755, st_size=2149728, ...}) = 0
[pid 20543] mmap(NULL, 2174000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid 20544] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 20543] <... mmap resumed>) = 0x7f29e0200000
[pid 20544] <... mmap resumed>) = 0x7f4c8be0d000
[pid 20543] mmap(0x7f29e0224000, 1515520, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000 <unfinished ...>
[pid 20544] pread64(3 <unfinished ...>
[pid 20543] <... mmap resumed>) = 0x7f29e0224000
[pid 20544] <... 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"..., 896, 64) = 896
[pid 20543] mmap(0x7f29e0396000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x196000 <unfinished ...>
[pid 20544] mmap(NULL, 2174000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid 20543] <... mmap resumed>) = 0x7f29e0396000
[pid 20544] <... mmap resumed>) = 0x7f4c8ba00000
[pid 20543] mmap(0x7f29e0405000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x204000 <unfinished ...>

[pid 20544] mmap(0x7f4c8ba24000, 1515520, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000 <unfinished ...>
[pid 20543] <... mmap resumed>) = 0x7f29e0405000
[pid 20544] <... mmap resumed>) = 0x7f4c8ba24000
[pid 20543] mmap(0x7f29e040b000, 31792, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 20544] mmap(0x7f4c8bb96000, 454656, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x196000 <unfinished ...>
[pid 20543] <... mmap resumed>) = 0x7f29e040b000
[pid 20544] <... mmap resumed>) = 0x7f4c8bb96000
[pid 20543] close(3 <unfinished ...>
[pid 20544] mmap(0x7f4c8bc05000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x204000 <unfinished ...>
[pid 20543] <... close resumed>) = 0
[pid 20544] <... mmap resumed>) = 0x7f4c8bc05000
[pid 20543] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0 <unfinished ...>
[pid 20544] mmap(0x7f4c8bc0b000, 31792, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid 20543] <... mmap resumed>) = 0x7f29e049b000
[pid 20544] <... mmap resumed>) = 0x7f4c8bc0b000
[pid 20543] arch_prctl(ARCH_SET_FS, 0x7f29e049b740 <unfinished ...>
[pid 20544] close(3 <unfinished ...>
[pid 20543] <... arch_prctl resumed>) = 0
[pid 20544] <... close resumed>) = 0
[pid 20543] set_tid_address(0x7f29e049ba10 <unfinished ...>
[pid 20544] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0 <unfinished ...>
[pid 20543] <... set_tid_address resumed>) = 20543
[pid 20544] <... mmap resumed>) = 0x7f4c8be0a000
[pid 20543] set_robust_list(0x7f29e049ba20, 24 <unfinished ...>
[pid 20544] arch_prctl(ARCH_SET_FS, 0x7f4c8be0a740 <unfinished ...>
[pid 20543] <... set_robust_list resumed>) = 0
[pid 20544] <... arch_prctl resumed>) = 0
[pid 20543] rseq(0x7f29e049b680, 0x20, 0, 0x53053053 <unfinished ...>
[pid 20544] set_tid_address(0x7f4c8be0aa10 <unfinished ...>
[pid 20543] <... rseq resumed>) = 0
[pid 20544] <... set_tid_address resumed>) = 20544
[pid 20544] set_robust_list(0x7f4c8be0aa20, 24) = 0
[pid 20544] rseq(0x7f4c8be0a680, 0x20, 0, 0x53053053) = 0
[pid 20543] mprotect(0x7f29e0405000, 16384, PROT_READ) = 0
[pid 20543] mprotect(0x55d9077f2000, 4096, PROT_READ <unfinished ...>
[pid 20544] mprotect(0x7f4c8bc05000, 16384, PROT_READ <unfinished ...>
[pid 20543] <... mprotect resumed>) = 0
[pid 20544] <... mprotect resumed>) = 0
[pid 20543] mprotect(0x7f29e0506000, 8192, PROT_READ <unfinished ...>
[pid 20544] mprotect(0x55c2327ee000, 4096, PROT_READ <unfinished ...>
[pid 20543] <... mprotect resumed>) = 0
[pid 20544] <... mprotect resumed>) = 0
[pid 20543] prlimit64(0, RLIMIT_STACK, NULL <unfinished ...>
[pid 20544] mprotect(0x7f4c8be75000, 8192, PROT_READ <unfinished ...>
[pid 20543] <... prlimit64 resumed>, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
[pid 20544] <... mprotect resumed>) = 0
[pid 20544] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) =
0
[pid 20543] getrandom(<unfinished ...>
[pid 20544] getrandom(<unfinished ...>
[pid 20543] <... getrandom resumed>"\x0d\xe7\xc0\x44\x73\xde\x92\x88", 8, GRND_NONBLOCK) = 8

```
[pid 20544] <... getrandom resumed>"\x3d\x80\x78\x8d\x2a\xf8\xa6\x95", 8, GRND_NONBLOCK) = 8
[pid 20543] munmap(0x7f29e04a0000, 157675 <unfinished ...>
[pid 20544] munmap(0x7f4c8be0f000, 157675 <unfinished ...>
[pid 20543] <... munmap resumed>)       = 0
[pid 20544] <... munmap resumed>)       = 0
[pid 20544] read(0 <unfinished ...>
[pid 20543] read(0asdads    asdas d
 <unfinished ...>
[pid 20542] <... read resumed>, "asdads    asdas d\n", 127) = 18
[pid 20542] write(4, "asdads    asdas d\n", 18) = 18
[pid 20543] <... read resumed>, "asdads    asdas d\n", 255) = 18
[pid 20542] read(7 <unfinished ...>
[pid 20543] write(1, "ASDADS    ASDAS D\n", 18 <unfinished ...>
[pid 20544] <... read resumed>, "ASDADS    ASDAS D\n", 255) = 18
[pid 20543] <... write resumed>)        = 18
[pid 20543] read(0 <unfinished ...>
[pid 20544] write(1, "ASDADS ASDAS D\n", 15 <unfinished ...>
[pid 20542] <... read resumed>, "ASDADS ASDAS D\n", 127) = 15
[pid 20544] <... write resumed>)        = 15
[pid 20542] write(1, "Processed string: ", 18 <unfinished ...>
[pid 20544] read(0Processed string:  <unfinished ...>
[pid 20542] <... write resumed>)        = 18
[pid 20542] write(1, "ASDADS ASDAS D\n", 15ASDADS ASDAS D
) = 15
[pid 20542] read(0, "", 127)            = 0
[pid 20542] close(4)                    = 0
[pid 20543] <... read resumed>, "", 255) = 0
[pid 20542] close(7)                    = 0
[pid 20542] wait4(20543 <unfinished ...>
[pid 20543] exit_group(0)               = ?
[pid 20544] <... read resumed>, "", 255) = 0
[pid 20543] +++ exited with 0 +++
[pid 20544] exit_group(0)               = ?
[pid 20542] <... wait4 resumed>, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 20543
[pid 20544] +++ exited with 0 +++
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=20543, si_uid=1000, si_status=0,
si_utime=0, si_stime=0} ---
wait4(20544, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 20544
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Во время выполнения лабораторной работы были изучены и использованы основные системные вызовы для работы с процессами и межпроцессным взаимодействием в Linux. Была создана программа, демонстрирующая создание процессов, создание каналов связи между ними и перенаправление стандартных потоков ввода-вывода.