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-214БВ-24

Студент: Зайцев Т.И.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 26.10.24

Москва, 2025

Постановка задачи

Вариант 11.

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.

Использованные системные вызовы:

pid_t fork(void) - создает точную копию текущего процесса (дочерний процесс). Возвращает 0 в дочернем процессе и PID дочернего в родительском.

int pipe(int fd[2]) - создает неименованный канал для обмена данными между процессами. fd[0] - для чтения, fd[1] - для записи.

ssize_t write(int fd, const void *buf, size_t count) - записывает данные в файловый дескриптор.

ssize_t read(int fd, void *buf, size_t count) - читает данные из файлового дескриптора.

int open(const char *pathname, int flags) - открывает файл и возвращает файловый дескриптор.

int close(int fd) - закрывает файловый дескриптор.

int dup2(int oldfd, int newfd) - дублирует файловый дескриптор, перенаправляя потоки.

int execv(const char *path, const char *arg, ...) - заменяет текущий образ процесса новым выполняемым файлом.

pid_t wait(int *status) - ожидает завершения любого дочернего процесса.

void exit(int status) - завершает вызывающий процесс

Общий метод и алгоритм решения

Программа создает три pipe-канала для взаимодействия между процессами:
- parent_to_child1 - для передачи данных от родительского процесса к Child1
- child1_to_child2 - для передачи данных от Child1 к Child2
- child2_to_parent - для возврата результата от Child2 к родительскому процессу

Далее создаются два дочерних процесса через fork(). Каждый дочерний процесс настраивает свои стандартные потоки с помощью dup2() и выполняет соответствующую программу через execv().

Родительский процесс читает строки от пользователя и передает их по конвейеру. Child1 преобразует текст в верхний регистр, Child2 заменяет пробельные символы на подчеркивания. Конечный результат возвращается родительскому процессу и выводится в стандартный поток вывода.

Код программы

client.c

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

static char CHILD1_PROGRAM_NAME[] = "child1";
static char CHILD2_PROGRAM_NAME[] = "child2";


int main(int argc, char **argv) {
  char progpath[1024];
  {

    ssize_t len = readlink("/proc/self/exe", progpath, sizeof(progpath) - 1);
    if (len == -1) {
      const char msg[] = "error: failed to read full program path\n";
      write(STDERR_FILENO, msg, sizeof(msg));
      exit(EXIT_FAILURE);
    }

    while (progpath[len] != '/') --len;
    progpath[len] = '\0';
  }

    int parent_to_child1[2];
    int child1_to_child2[2];
    int child2_to_parent[2];

    if (pipe(parent_to_child1) == -1 || pipe(child1_to_child2) == -1 ||
pipe(child2_to_parent) == -1) {
```

```
      const char msg[] = "error: failed to create pipes\n";
      write(STDERR_FILENO, msg, sizeof(msg));
      exit(EXIT_FAILURE);
  }

pid_t child1 = fork();

switch(child1){
  case -1: { // NOTE: Kernel fails to create another process
    const char msg[] = "error: failed to spawn new process\n";
    write(STDERR_FILENO, msg, sizeof(msg));
    exit(EXIT_FAILURE);
  } break;

  case 0: { // Процесс child1
    // {
    //  pid_t pid = getpid(); // Получаем PID child1
    //  char msg[64];
    //  uint32_t length = snprintf(msg, sizeof(msg), "%d: I'm child1\n", pid);
    //  write(STDOUT_FILENO, msg, length);
    // }

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

    dup2(parent_to_child1[0], STDIN_FILENO);
    close(parent_to_child1[0]);

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


          char path[2048];
          strcpy(path, progpath);
          strcat(path, "/");
          strcat(path, CHILD1_PROGRAM_NAME);
    char *const args[] = {CHILD1_PROGRAM_NAME, NULL};

    int32_t status = execv(path, args);

    if (status == -1) {
      const char msg[] = "error: failed to exec child1\n";
      write(STDERR_FILENO, msg, sizeof(msg));
      exit(EXIT_FAILURE);
    }
  } break;
}

pid_t child2 = fork();

  switch (child2) {
  case -1: { // Ошибка создания процесса
    const char msg[] = "error: failed to spawn new process\n";
```

```
    write(STDERR_FILENO, msg, sizeof(msg));
    exit(EXIT_FAILURE);
  } break;

  case 0: { // Процесс child2
    // {
    //  pid_t pid = getpid();
    //  char msg[64];
    //  uint32_t length = snprintf(msg, sizeof(msg), "%d: I'm child2\n", pid);
    //  write(STDOUT_FILENO, msg, length);
    // }

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

    dup2(child1_to_child2[0], STDIN_FILENO);
    dup2(child2_to_parent[1], STDOUT_FILENO);

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

    char path[2048];
         strcpy(path, progpath);
         strcat(path, "/");
         strcat(path, CHILD2_PROGRAM_NAME);
    char *const args[] = {CHILD2_PROGRAM_NAME, NULL};

    int32_t status = execv(path, args);

    if (status == -1) {
      const char msg[] = "error: failed to exec child2\n";
      write(STDERR_FILENO, msg, sizeof(msg));
      exit(EXIT_FAILURE);
    }
  } break;

  default: { // Родительский процесс - ОБА ребенка созданы!
    // {
    //  pid_t pid = getpid();
    //  char msg[128];
    //  uint32_t length = snprintf(msg, sizeof(msg),
    //    "%d: I'm parent, my children have PIDs %d and %d\n",
    //    pid, child1, child2);
    //  write(STDOUT_FILENO, msg, length);
    // }

    // Закрываем ненужные концы pipe'ов в родителе
    close(parent_to_child1[0]);
    close(child1_to_child2[0]);
    close(child1_to_child2[1]);
    close(child2_to_parent[1]);

    char buf[4096];
```

```
      ssize_t bytes;

      const char msg[] = "Enter strings (empty line to exit):\n";
      write(STDOUT_FILENO, msg, sizeof(msg) - 1);

           while ((bytes = read(STDIN_FILENO, buf, sizeof(buf))) > 0) {
               if (buf[0] == '\n') {
                   break;
               }

        // Отправляем в child1 через pipe1
               ssize_t written = write(parent_to_child1[1], buf, bytes);
               if (written != bytes) {
                   const char msg[] = "error: failed to write to child1\n";
                   write(STDERR_FILENO, msg, sizeof(msg) - 1);
                   exit(EXIT_FAILURE);
               }

        // Получаем результат от child2 через pipe2
        bytes = read(child2_to_parent[0], buf, sizeof(buf));
        if (bytes > 0) {
          // write(STDOUT_FILENO, "Result: ", 9);
          write(STDOUT_FILENO, buf, bytes);
          write(STDOUT_FILENO, "\n", 1);
        }
      }
      close(parent_to_child1[1]);
      close(child2_to_parent[0]);
      waitpid(child1, NULL, 0);
      waitpid(child2, NULL, 0);
    } break;
  }

    return 0;



}
```

Child1.c

```
#include <stdint.h>
#include <stdbool.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

int is_space(char c) {
    return (c == ' ' || c == '\t' || c == '\n' || c == '\r' || c == '\v' || c == '\f');
}

int main(int argc, char **argv) {
    char buf[4096];
```

```
    ssize_t bytes;

    pid_t pid = getpid();

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

        // NOTE: Transform data - replace whitespace with '_'
        for (uint32_t i = 0; i < bytes; ++i) {
            if (is_space(buf[i])) {
                buf[i] = '_';
            }
        }

        {
            // NOTE: Send back to parent process
            int32_t written = write(STDOUT_FILENO, buf, bytes);
            if (written != bytes) {
                const char msg[] = "error: failed to send to parent\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(EXIT_FAILURE);
            }
        }
    }

    return 0;
}
```

Child2.c

```
#include <stdint.h>
#include <stdbool.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>

int is_space(char c) {
    return (c == ' ' || c == '\t' || c == '\n' || c == '\r' || c == '\v' || c == '\f');
}

int main(int argc, char **argv) {
    char buf[4096];
    ssize_t bytes;

    pid_t pid = getpid();

    while (bytes = read(STDIN_FILENO, buf, sizeof(buf))) {
        if (bytes < 0) {
            const char msg[] = "error: failed to read from stdin\n";
```

```
            write(STDERR_FILENO, msg, sizeof(msg));
            exit(EXIT_FAILURE);
        }

        // NOTE: Transform data - replace whitespace with '_'
        for (uint32_t i = 0; i < bytes; ++i) {
            if (is_space(buf[i])) {
                buf[i] = '_';
            }
        }

        {
            // NOTE: Send back to parent process
            int32_t written = write(STDOUT_FILENO, buf, bytes);
            if (written != bytes) {
                const char msg[] = "error: failed to send to parent\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                exit(EXIT_FAILURE);
            }
        }
    }

    return 0;
}
```

Протокол работы программы:

```
tim@tim-potato-pc:~/OS_labs/lab_01$ ./client
Enter strings (empty line to exit):
        Hello, World!
_HELLO,_WORLD!_
Test String 123 && 321 !
TEST_STRING_123_&&_321_!_
a
A_
        a               bc      678ABc
R_A__BC_____678ABC_
Hello World #$! parent_to_child1
HELLO_WORLD_#$!_PARENT_TO_CHILD1_

tim@tim-potato-pc:~/OS_labs/lab_01$ ./client < test.txt
Enter strings (empty line to exit):
HELLO,_WORLD!_TEST_STRING____<-_FOUR_SPACES_____TABULATIONS!____&%^$_SYMBOLS
```

Strace:

```
10283 execve("./client", ["./client"], 0x7ffcc2b6c8d8 /* 79 vars */) = 0
10283 brk(NULL)                         = 0x5bf668c8f000
10283 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7593ef181000
10283 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)
10283 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
10283 fstat(3, {st_mode=S_IFREG|0644, st_size=69395, ...}) = 0
```

```
10283 mmap(NULL, 69395, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7593ef170000
10283 close(3)                          = 0
10283 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
10283 read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832
10283 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
10283 fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
10283 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
10283 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7593eee00000
10283 mmap(0x7593eee28000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7593eee28000
10283 mmap(0x7593eefb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1b0000) = 0x7593eefb0000
10283 mmap(0x7593eefff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7593eefff000
10283 mmap(0x7593ef005000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7593ef005000
10283 close(3)                          = 0
10283 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7593ef16d000
10283 arch_prctl(ARCH_SET_FS, 0x7593ef16d740) = 0
10283 set_tid_address(0x7593ef16da10)   = 10283
10283 set_robust_list(0x7593ef16da20, 24) = 0
10283 rseq(0x7593ef16e060, 0x20, 0, 0x53053053) = 0
10283 mprotect(0x7593eefff000, 16384, PROT_READ) = 0
10283 mprotect(0x5bf62a695000, 4096, PROT_READ) = 0
10283 mprotect(0x7593ef1bf000, 8192, PROT_READ) = 0
10283 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
10283 munmap(0x7593ef170000, 69395)     = 0
10283 readlink("/proc/self/exe", "/home/tim/OS_labs/lab_01/client", 1023) = 31
10283 pipe2([3, 4], 0)                  = 0
10283 pipe2([5, 6], 0)                  = 0
10283 pipe2([7, 8], 0)                  = 0
10283 clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
child_tidptr=0x7593ef16da10) = 10284
10283 clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD
<unfinished ...>
10284 set_robust_list(0x7593ef16da20, 24) = 0
10284 close(4)                          = 0
10283 <... clone resumed>, child_tidptr=0x7593ef16da10) = 10285
10284 close(5 <unfinished ...>
10283 close(3 <unfinished ...>
10284 <... close resumed>)              = 0
10283 <... close resumed>)              = 0
10284 close(7 <unfinished ...>
10283 close(5 <unfinished ...>
10284 <... close resumed>)              = 0
10283 <... close resumed>)              = 0
10285 set_robust_list(0x7593ef16da20, 24 <unfinished ...>
10284 close(8 <unfinished ...>
10283 close(6 <unfinished ...>
```

```
10285 <... set_robust_list resumed>)    = 0
10283 <... close resumed>)              = 0
10284 <... close resumed>)              = 0
10283 close(8 <unfinished ...>
10284 dup2(3, 0 <unfinished ...>
10285 close(3 <unfinished ...>
10283 <... close resumed>)              = 0
10284 <... dup2 resumed>)               = 0
10283 write(1, "Enter strings (empty line to exi"..., 36 <unfinished ...>
10285 <... close resumed>)              = 0
10284 close(3 <unfinished ...>
10285 close(4 <unfinished ...>
10283 <... write resumed>)              = 36
10284 <... close resumed>)              = 0
10283 read(0,  <unfinished ...>
10285 <... close resumed>)              = 0
10284 dup2(6, 1 <unfinished ...>
10283 <... read resumed>"Hello, World!\nTest String    <- "..., 4096) = 76
10285 close(6 <unfinished ...>
10283 write(4, "Hello, World!\nTest String    <- "..., 76 <unfinished ...>
10284 <... dup2 resumed>)               = 1
10283 <... write resumed>)              = 76
10285 <... close resumed>)              = 0
10283 read(7,  <unfinished ...>
10284 close(6 <unfinished ...>
10285 close(7 <unfinished ...>
10284 <... close resumed>)              = 0
10285 <... close resumed>)              = 0
10285 dup2(5, 0 <unfinished ...>
10284 execve("/home/tim/OS_labs/lab_01/child1", ["child1"], 0x7ffe5426ae28 /* 79
vars */ <unfinished ...>
10285 <... dup2 resumed>)               = 0
10285 dup2(8, 1)                        = 1
10285 close(5)                          = 0
10285 close(8)                          = 0
10285 execve("/home/tim/OS_labs/lab_01/child2", ["child2"], 0x7ffe5426ae28 /* 79
vars */ <unfinished ...>
10284 <... execve resumed>)             = 0
10284 brk(NULL)                         = 0x5eb64c67c000
10285 <... execve resumed>)             = 0
10284 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7be0a149f000
10285 brk(NULL <unfinished ...>
10284 access("/etc/ld.so.preload", R_OK <unfinished ...>
10285 <... brk resumed>)                = 0x63d2345b9000
10284 <... access resumed>)             = -1 ENOENT (Нет такого файла или каталога)
10284 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
10285 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
10284 <... openat resumed>)             = 3
10284 fstat(3,  <unfinished ...>
10285 <... mmap resumed>)               = 0x7e28409f2000
10284 <... fstat resumed>{st_mode=S_IFREG|0644, st_size=69395, ...}) = 0
10285 access("/etc/ld.so.preload", R_OK <unfinished ...>
10284 mmap(NULL, 69395, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
```

```
10285 <... access resumed>)             = -1 ENOENT (Нет такого файла или каталога)
10284 <... mmap resumed>)               = 0x7be0a148e000
10285 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
10284 close(3 <unfinished ...>
10285 <... openat resumed>)             = 3
10284 <... close resumed>)              = 0
10285 fstat(3, {st_mode=S_IFREG|0644, st_size=69395, ...}) = 0
10284 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
10285 mmap(NULL, 69395, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
10284 <... openat resumed>)             = 3
10285 <... mmap resumed>)               = 0x7e28409e1000
10284 read(3,  <unfinished ...>
10285 close(3 <unfinished ...>
10284 <... 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
10285 <... close resumed>)              = 0
10284 pread64(3,  <unfinished ...>
10285 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC
<unfinished ...>
10284 <... 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
10285 <... openat resumed>)             = 3
10284 fstat(3,  <unfinished ...>
10285 read(3,  <unfinished ...>
10284 <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
10285 <... 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
10284 pread64(3,  <unfinished ...>
10285 pread64(3,  <unfinished ...>
10284 <... 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
10285 <... 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
10284 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished
...>
10285 fstat(3,  <unfinished ...>
10284 <... mmap resumed>)               = 0x7be0a1200000
10285 <... fstat resumed>{st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
10284 mmap(0x7be0a1228000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
10285 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
10284 <... mmap resumed>)               = 0x7be0a1228000
10285 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished
...>
10284 mmap(0x7be0a13b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1b0000 <unfinished ...>
10285 <... mmap resumed>)               = 0x7e2840600000
10284 <... mmap resumed>)               = 0x7be0a13b0000
```

```
10285 mmap(0x7e2840628000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000 <unfinished ...>
10284 mmap(0x7be0a13ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000 <unfinished ...>
10285 <... mmap resumed>)              = 0x7e2840628000
10284 <... mmap resumed>)              = 0x7be0a13ff000
10285 mmap(0x7e28407b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1b0000 <unfinished ...>
10284 mmap(0x7be0a1405000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
10285 <... mmap resumed>)              = 0x7e28407b0000
10284 <... mmap resumed>)              = 0x7be0a1405000
10285 mmap(0x7e28407ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000 <unfinished ...>
10284 close(3 <unfinished ...>
10285 <... mmap resumed>)              = 0x7e28407ff000
10284 <... close resumed>)             = 0
10285 mmap(0x7e2840805000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>
10284 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
10285 <... mmap resumed>)              = 0x7e2840805000
10284 <... mmap resumed>)              = 0x7be0a148b000
10285 close(3 <unfinished ...>
10284 arch_prctl(ARCH_SET_FS, 0x7be0a148b740 <unfinished ...>
10285 <... close resumed>)             = 0
10284 <... arch_prctl resumed>)        = 0
10285 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>
10284 set_tid_address(0x7be0a148ba10 <unfinished ...>
10285 <... mmap resumed>)              = 0x7e28409de000
10284 <... set_tid_address resumed>)   = 10284
10285 arch_prctl(ARCH_SET_FS, 0x7e28409de740 <unfinished ...>
10284 set_robust_list(0x7be0a148ba20, 24 <unfinished ...>
10285 <... arch_prctl resumed>)        = 0
10284 <... set_robust_list resumed>)   = 0
10285 set_tid_address(0x7e28409dea10 <unfinished ...>
10284 rseq(0x7be0a148c060, 0x20, 0, 0x53053053 <unfinished ...>
10285 <... set_tid_address resumed>)   = 10285
10284 <... rseq resumed>)              = 0
10285 set_robust_list(0x7e28409dea20, 24) = 0
10285 rseq(0x7e28409df060, 0x20, 0, 0x53053053) = 0
10284 mprotect(0x7be0a13ff000, 16384, PROT_READ) = 0
10284 mprotect(0x5eb630233000, 4096, PROT_READ) = 0
10285 mprotect(0x7e28407ff000, 16384, PROT_READ <unfinished ...>
10284 mprotect(0x7be0a14dd000, 8192, PROT_READ <unfinished ...>
10285 <... mprotect resumed>)          = 0
10284 <... mprotect resumed>)          = 0
10285 mprotect(0x63d20027d000, 4096, PROT_READ <unfinished ...>
10284 prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
10285 <... mprotect resumed>)          = 0
10284 <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
10285 mprotect(0x7e2840a30000, 8192, PROT_READ <unfinished ...>
10284 munmap(0x7be0a148e000, 69395 <unfinished ...>
10285 <... mprotect resumed>)          = 0
```

```
10284 <... munmap resumed>)             = 0
10285 prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
10284 getpid( <unfinished ...>
10285 <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
10284 <... getpid resumed>)             = 10284
10284 read(0,  <unfinished ...>
10285 munmap(0x7e28409e1000, 69395 <unfinished ...>
10284 <... read resumed>"Hello, World!\nTest String    <- "..., 4096) = 76
10285 <... munmap resumed>)             = 0
10284 write(1, "HELLO, WORLD!\nTEST STRING    <- "..., 76) = 76
10285 getpid( <unfinished ...>
10284 read(0,  <unfinished ...>
10285 <... getpid resumed>)             = 10285
10285 read(0, "HELLO, WORLD!\nTEST STRING    <- "..., 4096) = 76
10285 write(1, "HELLO,_WORLD!_TEST_STRING____<-_"..., 76) = 76
10283 <... read resumed>"HELLO,_WORLD!_TEST_STRING____<-_"..., 4096) = 76
10285 read(0,  <unfinished ...>
10283 write(1, "Result: \0", 9)         = 9
10283 write(1, "HELLO,_WORLD!_TEST_STRING____<-_"..., 76) = 76
10283 write(1, "\n", 1)                 = 1
10283 read(0, "", 4096)                 = 0
10283 close(4)                          = 0
10284 <... read resumed>"", 4096)       = 0
10283 close(7)                          = 0
10284 exit_group(0 <unfinished ...>
10283 wait4(10284,  <unfinished ...>
10284 <... exit_group resumed>)         = ?
10285 <... read resumed>"", 4096)       = 0
10284 +++ exited with 0 +++
10283 <... wait4 resumed>NULL, 0, NULL) = 10284
10283 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=10284, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
10285 exit_group(0 <unfinished ...>
10283 wait4(10285,  <unfinished ...>
10285 <... exit_group resumed>)         = ?
10285 +++ exited with 0 +++
10283 <... wait4 resumed>NULL, 0, NULL) = 10285
10283 --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=10285, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
10283 exit_group(0)                     = ?
10283 +++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы были успешно изучены и применены на практике механизмы межпроцессного взаимодействия.