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-215Б-23

Студент: Закарейшвили Г. М.

Преподаватель: Миронов Е.С. (ПМИ)

Оценка: _____

Дата: 03.03.24

Москва, 2024

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.
11 вариант) Child1 переводит строки в верхний регистр. Child2 превращает все пробельные символы в символ «_».

Общий метод и алгоритм решения

Использованные системные вызовы:

- **fork()** – создает дочерний процесс (копия родительского).
- **pipe()** – создает канал (pipe) для обмена данными между процессами.
- **exit()** – завершает выполнение процесса.
- **execl()** – заменяет текущий процесс новым (загрузка программы).
- **close()** – закрывает файловый дескриптор (например, конец канала).
- **dup2()** – перенаправляет дескриптор (например, pipe на stdin/stdout).
- **perror()** – выводит сообщение об ошибке (на основе errno).
- **wait()** – ожидает завершения дочернего процесса.

Алгоритм работы программы
Родительский процесс

1. Описание родительского процесса. Parent.c

1. Создаем два канала: pipe1 и pipe2. Обмен данными между процессами, описаными ниже, будет происходить через них.
2. Создаем дочерний процесс child1 с помощью fork().
   Описание работы дочернего процесса:
   1) Закрываем запись pipe1 и чтение pipe2 за ненадобностью.
   2) Перенаправляем с помощью dub2() чтение с pipe1 на stdin и stdout на запись с pipe2.
   3) Далее выполняется программа child1, которая читает из pipe1 и записывает в pipe2.
3. Создаем дочерний процесс child2 с помощью fork().
   Описание работы дочернего процесса:
   1) Закрываем чтение и запись pipe1 и запись pipe2 за ненадобностью.
   2) Перенаправляем с помощью dub2() чтение с pipe2 на stdin.
   3) Далее выполняется программа child2, которая читает из pipe2 и записывает в стандартный вывод stdout.
4. Продолжается родительский процесс Parent.c
   1) Закрываем pipe1 на чтение и pipe2 и на чтение, и на запись.
   2) Родительский процесс читает строки, которые вводит пользователь, и записывает их в pipe1.

3) Закрываем запись pipe1.Ожидаем завершения дочерних процессов с помощью wait().

2. Дочерний процесс child1

1. Считываем из stdin строку.
2. Преобразовываем строку в верхний регистр.
3. Отправляем строку в stdout (pipe2).

3. Дочерний процесс child2

1) Считываем из stdin строку.
2) Удаляем лишние пробелы.
3) Выводим измененную строку в stdout.

Код программы

parent.c

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

#define MAX_LINE 1000

int main() {
    int pipe1[2], pipe2[2];
    pid_t child1, child2;

    // Создаем два канала
    if (pipe(pipe1) == -1 || pipe(pipe2) == -1) {
        perror("Ошибка создания канала");
        exit(1);
    }

    // Создаем первый дочерний процесс
    if ((child1 = fork()) == -1) {
        perror("Ошибка создания первого дочернего процесса");
        exit(1);
    }

    if (child1 == 0) {
        // Код для первого дочернего процесса (child1)
        close(pipe1[1]);  // Закрываем конец записи pipe1
        close(pipe2[0]);  // Закрываем конец чтения pipe2
        dup2(pipe1[0], STDIN_FILENO);  // Перенаправляем чтение с pipe1 на stdin
        dup2(pipe2[1], STDOUT_FILENO);  // Перенаправляем вывод на pipe2
        close(pipe1[0]);
        close(pipe2[1]);
```

```
        execl("./child1", "child1", NULL);
        perror("Ошибка execl для child1");
        exit(1);
    }

    // Создаем второй дочерний процесс
    if ((child2 = fork()) == -1) {
        perror("Ошибка создания второго дочернего процесса");
        exit(1);
    }

    if (child2 == 0) {
        // Код для второго дочернего процесса (child2)
        close(pipe1[0]);
        close(pipe1[1]);  // Не используем pipe1
        close(pipe2[1]);  // Закрываем конец записи pipe2
        dup2(pipe2[0], STDIN_FILENO);  // Перенаправляем pipe2 на stdin
        close(pipe2[0]);

        execl("./child2", "child2", NULL);
        perror("Ошибка execl для child2");
        exit(1);
    }

    // Родительский процесс
    close(pipe1[0]);  // Закрываем конец чтения pipe1
    close(pipe2[0]);  // Закрываем конец чтения pipe2
    close(pipe2[1]);  // Закрываем конец записи pipe2 для родителя

    char line[MAX_LINE];
    printf("Введите строки (Ctrl+D для завершения):\n");

    // Чтение строк от пользователя и переслка их в child1
    while (fread(line, sizeof(char), MAX_LINE, stdin) != 0) {
        write(pipe1[1], line, strlen(line)); //записывает данные из line в канал
pipe1
    }

    close(pipe1[1]);  // Закрываем конец записи pipe1 после ввода

    // Ожидание завершения дочерних процессов
    wait(NULL);  // Ждем завершения child1
    wait(NULL);  // Ждем завершения child2

    printf("\nВсе процессы завершены.\n");
```

```
    return 0;
}
```

child1.c

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

#define MAX_LINE 1000

int main() {
    char line[MAX_LINE];

    // Чтение строк из stdin (перенаправленный pipe1)
    while (read(STDIN_FILENO, line, MAX_LINE) != 0) {
        // Преобразование строки в верхний регистр
        for (int i = 0; line[i]; i++) {
            line[i] = toupper(line[i]);
        }

        // Отправка строки в pipe2 (stdout был перенаправлен)
        write(STDOUT_FILENO, line, strlen(line)); // записываем в стандартный
поток вывода строку line длины strlen(line)
    }

    return 0;
}
```

child2.c

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

#define MAX_LINE 1000

void remove_extra_spaces(char *str) {
    int i = 0;
    while (str[i]) {
        if (str[i] == ' ') {
            str[i] = '_';
```

```
        }
        i++;
    }
}

int main() {
    char line[MAX_LINE];
```

Протокол работы программы

Тест:

Введите строки (Ctrl+D для завершения):
Papich says normaly
PAPICH___SAYS_NORMALY

Все процессы завершены.

Strace:

```
strace -f ./parent
execve("./parent", ["./parent"], 0x7fffe49d7ba8 /* 19 vars */) = 0
brk(NULL)                               = 0x7fffd5911000
arch_prctl(0x3001 /* ARCH_??? */, 0x7fffdcfd3240) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ffd30090000
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=16555, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 16555, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ffd3004b000
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"..., 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"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 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"..., 784, 64) = 784
mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ffd2fe20000
mprotect(0x7ffd2fe48000, 2023424, PROT_NONE) = 0
mmap(0x7ffd2fe48000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) =
0x7ffd2fe48000
mmap(0x7ffd2ffdd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7ffd2ffdd000
mmap(0x7ffd30036000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) =
0x7ffd30036000
mmap(0x7ffd3003c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7ffd3003c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ffd2fe10000
arch_prctl(ARCH_SET_FS, 0x7ffd2fe10740) = 0
set_tid_address(0x7ffd2fe10a10)         = 447
set_robust_list(0x7ffd2fe10a20, 24)     = 0
rseq(0x7ffd2fe110e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)
mprotect(0x7ffd30036000, 16384, PROT_READ) = 0
mprotect(0x7ffd30096000, 4096, PROT_READ) = 0
mprotect(0x7ffd30088000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
munmap(0x7ffd3004b000, 16555)           = 0
pipe2([3, 4], 0)                        = 0
pipe2([5, 6], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 448 attached
, child_tidptr=0x7ffd2fe10a10) = 448
[pid   448] set_robust_list(0x7ffd2fe10a20, 24 <unfinished ...>
[pid   447] clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD <unfinished ...>
[pid   448] <... set_robust_list resumed>) = 0
[pid   448] close(4)                    = 0
[pid   448] close(5)                    = 0
[pid   448] dup2(3, 0)                  = 0
[pid   448] dup2(6, 1)                  = 1
```

```
strace: Process 449 attached
[pid   448] close(3 <unfinished ...>
[pid   447] <... clone resumed>, child_tidptr=0x7ffd2fe10a10) = 449
[pid   449] set_robust_list(0x7ffd2fe10a20, 24 <unfinished ...>
[pid   447] close(3 <unfinished ...>
[pid   448] <... close resumed>)       = 0
[pid   447] <... close resumed>)       = 0
[pid   449] <... set_robust_list resumed>) = 0
[pid   447] close(5 <unfinished ...>
[pid   448] close(6 <unfinished ...>
[pid   447] <... close resumed>)       = 0
[pid   449] close(3 <unfinished ...>
[pid   447] close(6 <unfinished ...>
[pid   448] <... close resumed>)       = 0
[pid   447] <... close resumed>)       = 0
[pid   449] <... close resumed>)       = 0
[pid   447] newfstatat(1, "",  <unfinished ...>
[pid   448] execve("./child1", ["child1"], 0x7fffdcfd3418 /* 19 vars */ <unfinished ...>
[pid   447] <... newfstatat resumed>{st_mode=S_IFCHR|0640, st_rdev=makedev(0x4, 0x2), ...}, AT_EMPTY_PATH) = 0
[pid   449] close(4 <unfinished ...>
[pid   447] ioctl(1, TCGETS, {B38400 opost isig icanon echo ...}) = 0
[pid   449] <... close resumed>)       = 0
[pid   447] getrandom( <unfinished ...>
[pid   449] close(6 <unfinished ...>
[pid   447] <... getrandom resumed>"\x4f\xdd\x39\xb3\x07\xd4\x0a\x81", 8, GRND_NONBLOCK) = 8
[pid   449] <... close resumed>)       = 0
[pid   447] brk(NULL <unfinished ...>
[pid   449] dup2(5, 0 <unfinished ...>
[pid   447] <... brk resumed>)         = 0x7fffd5911000
[pid   449] <... dup2 resumed>)        = 0
[pid   447] brk(0x7fffd5932000 <unfinished ...>
[pid   449] close(5 <unfinished ...>
[pid   447] <... brk resumed>)         = 0x7fffd5932000
[pid   448] <... execve resumed>)      = 0
[pid   447] write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\201\321\202\321\200\320\276\320\272\320\270 (Ctr"..., 66Введите строки (Ctrl+D для завершения):
 <unfinished ...>
[pid   449] <... close resumed>)       = 0
[pid   447] <... write resumed>)       = 66
[pid   448] brk(NULL <unfinished ...>
[pid   447] newfstatat(0, "",  <unfinished ...>
[pid   449] execve("./child2", ["child2"], 0x7fffdcfd3418 /* 19 vars */ <unfinished ...>
[pid   447] <... newfstatat resumed>{st_mode=S_IFCHR|0640, st_rdev=makedev(0x4, 0x2), ...}, AT_EMPTY_PATH) = 0
[pid   448] <... brk resumed>)         = 0x7fffe5373000
[pid   447] ioctl(0, TCGETS <unfinished ...>
[pid   448] arch_prctl(0x3001 /* ARCH_??? */, 0x7fffec8864d0 <unfinished ...>
[pid   447] <... ioctl resumed>, {B38400 opost isig icanon echo ...}) = 0
[pid   448] <... arch_prctl resumed>)   = -1 EINVAL (Invalid argument)
[pid   447] read(0,  <unfinished ...>
[pid   448] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0c074d0000
[pid   448] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
[pid   448] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid   448] newfstatat(3, "",  <unfinished ...>
[pid   449] <... execve resumed>)      = 0
[pid   448] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=16555, ...}, AT_EMPTY_PATH) = 0
[pid   449] brk(NULL <unfinished ...>
[pid   448] mmap(NULL, 16555, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid   449] <... brk resumed>)         = 0x7fffdf532000
[pid   448] <... mmap resumed>)        = 0x7f0c074d4000
[pid   449] arch_prctl(0x3001 /* ARCH_??? */, 0x7fffe6c79550 <unfinished ...>
[pid   448] close(3 <unfinished ...>
[pid   449] <... arch_prctl resumed>)   = -1 EINVAL (Invalid argument)
[pid   448] <... close resumed>)       = 0
[pid   449] mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   448] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid   449] <... mmap resumed>)        = 0x7fb3c8480000
[pid   448] <... openat resumed>)      = 3
[pid   449] access("/etc/ld.so.preload", R_OK <unfinished ...>
[pid   448] read(3,  <unfinished ...>
[pid   449] <... access resumed>)      = -1 ENOENT (No such file or directory)
[pid   448] <... 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   449] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid   448] pread64(3,  <unfinished ...>
```

```
[pid   449] <... openat resumed>)      = 3
[pid   448] <... 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   449] newfstatat(3, "",  <unfinished ...>
[pid   448] pread64(3,  <unfinished ...>
[pid   449] <... newfstatat resumed>{st_mode=S_IFREG|0644, st_size=16555, ...}, AT_EMPTY_PATH) = 0
[pid   448] <... 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   449] mmap(NULL, 16555, PROT_READ, MAP_PRIVATE, 3, 0 <unfinished ...>
[pid   448] pread64(3,  <unfinished ...>
[pid   449] <... mmap resumed>)        = 0x7fb3c8484000
[pid   448] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68,
896) = 68
[pid   449] close(3 <unfinished ...>
[pid   448] newfstatat(3, "",  <unfinished ...>
[pid   449] <... close resumed>)       = 0
[pid   448] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid   449] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC <unfinished ...>
[pid   448] pread64(3,  <unfinished ...>
[pid   449] <... openat resumed>)      = 3
[pid   448] <... 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   449] read(3,  <unfinished ...>
[pid   448] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid   449] <... read resumed>"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
[pid   448] <... mmap resumed>)        = 0x7f0c07260000
[pid   449] pread64(3,  <unfinished ...>
[pid   448] mprotect(0x7f0c07288000, 2023424, PROT_NONE <unfinished ...>
[pid   449] <... 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   448] <... mprotect resumed>)    = 0
[pid   449] pread64(3,  <unfinished ...>
[pid   448] mmap(0x7f0c07288000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000 <unfinished ...>
[pid   449] <... 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   448] <... mmap resumed>)        = 0x7f0c07288000
[pid   449] pread64(3,  <unfinished ...>
[pid   448] mmap(0x7f0c0741d000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000
<unfinished ...>
[pid   449] <... pread64 resumed>"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68,
896) = 68
[pid   448] <... mmap resumed>)        = 0x7f0c0741d000
[pid   449] newfstatat(3, "",  <unfinished ...>
[pid   448] mmap(0x7f0c07476000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000 <unfinished ...>
[pid   449] <... newfstatat resumed>{st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 0
[pid   448] <... mmap resumed>)        = 0x7f0c07476000
[pid   449] pread64(3,  <unfinished ...>
[pid   448] mmap(0x7f0c0747c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0
<unfinished ...>
[pid   449] <... 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   448] <... mmap resumed>)        = 0x7f0c0747c000
[pid   449] mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0 <unfinished ...>
[pid   448] close(3 <unfinished ...>
[pid   449] <... mmap resumed>)        = 0x7fb3c8210000
[pid   448] <... close resumed>)       = 0
[pid   449] mprotect(0x7fb3c8238000, 2023424, PROT_NONE <unfinished ...>
[pid   448] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   449] <... mprotect resumed>)    = 0
[pid   448] <... mmap resumed>)        = 0x7f0c07250000
[pid   449] mmap(0x7fb3c8238000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000 <unfinished ...>
[pid   448] arch_prctl(ARCH_SET_FS, 0x7f0c07250740 <unfinished ...>
[pid   449] <... mmap resumed>)        = 0x7fb3c8238000
[pid   448] <... arch_prctl resumed>)  = 0
[pid   449] mmap(0x7fb3c83cd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000
<unfinished ...>
[pid   448] set_tid_address(0x7f0c07250a10 <unfinished ...>
[pid   449] <... mmap resumed>)        = 0x7fb3c83cd000
[pid   448] <... set_tid_address resumed>) = 448
[pid   449] mmap(0x7fb3c8426000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000 <unfinished ...>
[pid   448] set_robust_list(0x7f0c07250a20, 24 <unfinished ...>
[pid   449] <... mmap resumed>)        = 0x7fb3c8426000
[pid   448] <... set_robust_list resumed>) = 0
[pid   449] mmap(0x7fb3c842c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0
<unfinished ...>
```

```
[pid   448] rseq(0x7f0c072510e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid   449] <... mmap resumed>)         = 0x7fb3c842c000
[pid   448] <... rseq resumed>)         = -1 ENOSYS (Function not implemented)
[pid   449] close(3)                    = 0
[pid   448] mprotect(0x7f0c07476000, 16384, PROT_READ <unfinished ...>
[pid   449] mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0 <unfinished ...>
[pid   448] <... mprotect resumed>)     = 0
[pid   449] <... mmap resumed>)         = 0x7fb3c8200000
[pid   448] mprotect(0x7f0c074dc000, 4096, PROT_READ <unfinished ...>
[pid   449] arch_prctl(ARCH_SET_FS, 0x7fb3c8200740 <unfinished ...>
[pid   448] <... mprotect resumed>)     = 0
[pid   449] <... arch_prctl resumed>)   = 0
[pid   448] mprotect(0x7f0c074c8000, 8192, PROT_READ <unfinished ...>
[pid   449] set_tid_address(0x7fb3c8200a10 <unfinished ...>
[pid   448] <... mprotect resumed>)     = 0
[pid   449] <... set_tid_address resumed>) = 449
[pid   448] prlimit64(0, RLIMIT_STACK, NULL,  <unfinished ...>
[pid   449] set_robust_list(0x7fb3c8200a20, 24 <unfinished ...>
[pid   448] <... prlimit64 resumed>{rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
[pid   449] <... set_robust_list resumed>) = 0
[pid   448] munmap(0x7f0c074d4000, 16555 <unfinished ...>
[pid   449] rseq(0x7fb3c82010e0, 0x20, 0, 0x53053053 <unfinished ...>
[pid   448] <... munmap resumed>)       = 0
[pid   449] <... rseq resumed>)         = -1 ENOSYS (Function not implemented)
[pid   448] read(0,  <unfinished ...>
[pid   449] mprotect(0x7fb3c8426000, 16384, PROT_READ) = 0
[pid   449] mprotect(0x7fb3c848c000, 4096, PROT_READ) = 0
[pid   449] mprotect(0x7fb3c8478000, 8192, PROT_READ) = 0
[pid   449] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=8192*1024}) = 0
[pid   449] munmap(0x7fb3c8484000, 16555) = 0
[pid   449] read(0, text    prob <unfinished ...>
[pid   447] <... read resumed>"text    prob", 4096) = 12
[pid   447] read(0, "", 4096)           = 0
[pid   447] write(4, "text    prob\375\177", 14) = 14
[pid   448] <... read resumed>"text    prob\375\177", 1000) = 14
[pid   447] close(4 <unfinished ...>
[pid   448] write(1, "TEXT    PROB\375\177", 14 <unfinished ...>
[pid   447] <... close resumed>)        = 0
[pid   448] <... write resumed>)        = 14
[pid   449] <... read resumed>"TEXT    PROB\375\177", 1000) = 14
[pid   447] wait4(-1,  <unfinished ...>
[pid   448] read(0,  <unfinished ...>
[pid   449] write(1, "TEXT____PROB\375\177", 14TEXT____PROB <unfinished ...>
[pid   448] <... read resumed>"", 1000) = 0
[pid   449] <... write resumed>)        = 14
[pid   448] exit_group(0 <unfinished ...>
[pid   449] read(0,  <unfinished ...>
[pid   448] <... exit_group resumed>)   = ?
[pid   448] +++ exited with 0 +++
[pid   449] <... read resumed>"", 1000) = 0
[pid   447] <... wait4 resumed>NULL, 0, NULL) = 448
[pid   449] exit_group(0 <unfinished ...>
[pid   447] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=448, si_uid=0, si_status=0, si_utime=0, si_stime=0}
---
[pid   449] <... exit_group resumed>)   = ?
[pid   447] wait4(-1,  <unfinished ...>
[pid   449] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL)       = 449
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=449, si_uid=0, si_status=0, si_utime=0, si_stime=0} ---
write(1, "\n", 1
)                       = 1
write(1, "\320\222\321\201\320\265 \320\277\321\200\320\276\321\206\320\265\321\201\321\201\321\213
\320\267\320\260\320\262\320\265"..., 44Все процессы завершены.
) = 44
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Данная лабораторная работа помогла разобраться, как процессы в ОС обмениваются

данными — например, через каналы (pipes) и перенаправление ввода-вывода. Я научился создавать процессы через fork(), связывать их каналами (это важно для программ, где задачи выполняются одновременно), а также работать с функциями системы вроде pipe(), dup2() и execl(). Работа была полезной и интересной, хотя отладка требовала внимания: процессы должны четко взаимодействовать друг с другом.