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-209БВ-24

Студент: Котик М.Н.

Преподаватель: Миронов Е.С. (ПМИ)

Оценка: ____________________

Дата: 22.10.25

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.

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

Общий метод и алгоритм решения

Использованные системные вызовы:

- pid_t fork() - создание дочернего процесса

- int pipe(int pipefd[2]) - создание неименованного канала

- int dup2(int oldfd, int newfd) - переназначение файлового дескриптора

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

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

- pid_t waitpid(pid_t pid, int *status, int options) - ожидание завершения процесса

- int execl(const char *pathname, const char *arg, ...) - загрузка новой программы

Алгоритм работы программы:

1. Инициализация

- Создание трех каналов (pipe1, pipe2, pipe3) для межпроцессного взаимодействия

- Вывод информации о начале работы программы

2. Создание процессов

- Родительский процесс создает двух дочерних процессов через fork()

- Каждый дочерний процесс перенаправляет стандартные потоки с помощью dup2()

- Дочерние процессы запускают программы child1 и child2 через execl()

3. Организация конвейера

Parent → pipe1 → Child1 → pipe2 → Child2 → pipe3 → Parent

4. Обработка данных

- Пользователь вводит строки в родительском процессе

- Родитель отправляет строки в pipe1

- Child1 читает из pipe1, преобразует в верхний регистр, пишет в pipe2

- Child2 читает из pipe2, заменяет пробелы на '_', пишет в pipe3

- Родитель читает из pipe3 и выводит результат

5. Завершение работы

- При вводе "exit" родитель закрывает каналы

- Ожидает завершения дочерних процессов через waitpid()

- Программа корректно завершает работу

Код программы

parent.c

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <signal.h>

#include <errno.h>



#define BUFFER_SIZE 1024
```

```
void print_error(const char *msg)
{
    perror(msg);
    exit(EXIT_FAILURE);
}

int main()
{
    int pipe1[2];
    int pipe2[2];
    int pipe3[2];

    pid_t pid1, pid2;

    printf("=== Lab 1. Variant 11 ===\n");
    printf("Chain: Parent -> Child1 -> Child2 -> Parent\n\n");

    printf("=== Creating pipes ===\n");
    if (pipe(pipe1) == -1)
        print_error("pipe Parent->Child1");
    if (pipe(pipe2) == -1)
        print_error("pipe Child1->Child2");
    if (pipe(pipe3) == -1)
        print_error("pipe Child2->Parent");

    printf("=== Creating Child1 ===\n");
```

```
pid1 = fork();
if (pid1 == -1)
    print_error("fork Child1");

if (pid1 == 0)
{
    close(pipe1[1]);
    close(pipe2[0]);
    close(pipe3[0]);
    close(pipe3[1]);

    dup2(pipe1[0], STDIN_FILENO);
    close(pipe1[0]);

    dup2(pipe2[1], STDOUT_FILENO);
    close(pipe2[1]);

    execl("./child1", "child1", NULL);
    print_error("execl Child1");
}

printf("Child1 created with PID: %d\n", pid1);

printf("=== Creating Child2 ===\n");
pid2 = fork();
if (pid2 == -1)
    print_error("fork Child2");
```

```
if (pid2 == 0)
{

    close(pipe1[0]);
    close(pipe1[1]);
    close(pipe2[1]);
    close(pipe3[0]);

    dup2(pipe2[0], STDIN_FILENO);
    close(pipe2[0]);

    dup2(pipe3[1], STDOUT_FILENO);
    close(pipe3[1]);

    execl("./child2", "child2", NULL);
    print_error("execl Child2");
}

printf("Child2 created with PID: %d\n", pid2);

close(pipe1[0]);
close(pipe2[0]);
close(pipe2[1]);
close(pipe3[1]);

printf("\n=== Starting data processing ===\n");
printf("Enter strings for processing (type 'exit' to quit):\n");


char buffer[BUFFER_SIZE];

char result[BUFFER_SIZE];

int line_count = 0;


while (1)

{

    printf("> ");

    fflush(stdout);


    if (fgets(buffer, BUFFER_SIZE, stdin) == NULL)

    {

        break;

    }


    buffer[strcspn(buffer, "\n")] = 0;


    if (strcmp(buffer, "exit") == 0)

    {

        break;

    }


    line_count++;

    printf("[Parent] Sending string #%d to Child1: '%s'\n", line_count, buffer);


    strcat(buffer, "\n");
```

```
        ssize_t bytes_written = write(pipe1[1], buffer, strlen(buffer));

        if (bytes_written == -1)

        {

            perror("write to Child1");

            break;

        }


        ssize_t bytes_read = read(pipe3[0], result, BUFFER_SIZE - 1);

        if (bytes_read > 0)

        {

            result[bytes_read] = '\0';


            result[strcspn(result, "\n")] = '\0';

            printf("[Parent] Final result from Child2: '%s'\n\n", result);

        }

        else if (bytes_read == -1)

        {

            perror("read from Child2");

            break;

        }

        else

        {

            printf("[Parent] No result received from Child2\n\n");

        }

    }


    printf("\n=== Shutting down ===\n");
    printf("Closing pipes to signal children to exit...\n");


    close(pipe1[1]);

    close(pipe3[0]);


    printf("Waiting for children to exit...\n");


    int status1, status2;

    waitpid(pid1, &status1, 0);

    waitpid(pid2, &status2, 0);


    printf("Child processes exited with codes: Child1=%d, Child2=%d\n",

           WEXITSTATUS(status1), WEXITSTATUS(status2));


    printf("Program finished successfully.\n");

    return 0;

}
```

child1.c

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>


#define BUFFER_SIZE 1024


int main()
```

```
{
    char buffer[BUFFER_SIZE];


    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);


    while (fgets(buffer, BUFFER_SIZE, stdin) != NULL)
    {
        for (int i = 0; buffer[i] != '\0' && buffer[i] != '\n'; i++)
        {
            buffer[i] = toupper(buffer[i]);
        }


        printf("%s", buffer);
        fflush(stdout);
    }


    return 0;
}
```

child2.c

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>


#define BUFFER_SIZE 1024


int main()

{

    char buffer[BUFFER_SIZE];


    setvbuf(stdin, NULL, _IONBF, 0);

    setvbuf(stdout, NULL, _IONBF, 0);


    while (fgets(buffer, BUFFER_SIZE, stdin) != NULL)

    {


        for (int i = 0; buffer[i] != '\0' && buffer[i] != '\n'; i++)

        {

            if (isspace(buffer[i]))

            {

                buffer[i] = '_';

            }

        }


        printf("%s", buffer);
        fflush(stdout);
    }

    return 0;
}
```

Протокол работы программы

Здесь нужно показать тесты программы (текст или скриншоты), а затем показать полный вывод утилиты strace (или какой-либо другой утилиты на Windows, если вы выполняете лабы на этой операционной системе).

В strace нужно обязательно выделить, где происходят системные вызовы, которые вы использовали в лабораторной работе (например, где в первой лабораторной работе был вызван fork и другие вызовы). Полный список вызовов, которые нужно будет выделить в выводе strace, будет указан при выдаче лабы в нашем канале.

```
execve("./parent", ["./parent"], 0x7ffd770f5708 /* 4 vars */) = 0

arch_prctl(ARCH_SET_FS, 0x7fe562941b48) = 0

set_tid_address(0x7fe562941fb8)         = 10

brk(NULL)                               = 0x55fb85b1d000

brk(0x55fb85b1f000)                     = 0x55fb85b1f000

mmap(0x55fb85b1d000, 4096, PROT_NONE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x55fb85b1d000

mprotect(0x7fe56293e000, 4096, PROT_READ) = 0

mprotect(0x55fb5bdd7000, 4096, PROT_READ) = 0

ioctl(1, TIOCGWINSZ, {ws_row=27, ws_col=129, ws_xpixel=0, ws_ypixel=0}) = 0

writev(1, [{iov_base="=== Lab 1. Variant 11 ===", iov_len=25}, {iov_base="\n", iov_len=1}], 2=== Lab 1. Variant 11 ===

) = 26
```

```
writev(1, [{iov_base="", iov_len=0}, {iov_base="Chain: Parent -> Child1 -> Child"..., iov_len=44}], 2Chain: Parent -> Child1 -> Child2 -> Parent

) = 44

writev(1, [{iov_base="", iov_len=0}, {iov_base="\n", iov_len=1}], 2

) = 1

writev(1, [{iov_base="=== Creating pipes ===", iov_len=22}, {iov_base="\n", iov_len=1}], 2=== Creating pipes ===

) = 23

pipe([3, 4])                    = 0

pipe([5, 6])                    = 0

pipe([7, 8])                    = 0

writev(1, [{iov_base="=== Creating Child1 ===", iov_len=23}, {iov_base="\n", iov_len=1}], 2=== Creating Child1 ===

) = 24

rt_sigprocmask(SIG_BLOCK, ~[RTMIN RT_1 RT_2], [], 8) = 0

rt_sigprocmask(SIG_BLOCK, ~[], ~[KILL STOP RTMIN RT_1 RT_2], 8) = 0

fork(strace: Process 11 attached

)                               = 11

[pid    10] rt_sigprocmask(SIG_SETMASK, ~[KILL STOP RTMIN RT_1 RT_2],  <unfinished ...>

[pid    11] set_tid_address(0x7fe562941fb8 <unfinished ...>

[pid    10] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid    10] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid    11] <... set_tid_address resumed>) = 11

[pid    10] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid    10] writev(1, [{iov_base="Child1 created with PID: 11", iov_len=27}, {iov_base="\n", iov_len=1}], 2 Child1 created with PID: 11

<unfinished ...>

[pid    11] rt_sigprocmask(SIG_SETMASK, ~[KILL STOP RTMIN RT_1 RT_2],  <unfinished ...>

[pid    10] <... writev resumed>)       = 28

[pid    11] <... rt_sigprocmask resumed>NULL, 8) = 0
```

```
[pid    10] writev(1, [{iov_base="=== Creating Child2 ===", iov_len=23},
{iov_base="\n", iov_len=1}], 2 <unfinished ...>

[pid    11] rt_sigprocmask(SIG_SETMASK, === Creating Child2 ===

[],  <unfinished ...>

[pid    10] <... writev resumed>)       = 24

[pid    11] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid    10] rt_sigprocmask(SIG_BLOCK, ~[RTMIN RT_1 RT_2],  <unfinished
...>

[pid    11] close(4 <unfinished ...>

[pid    10] <... rt_sigprocmask resumed>[], 8) = 0

[pid    11] <... close resumed>)        = 0

[pid    10] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>

[pid    11] close(5 <unfinished ...>

[pid    10] <... rt_sigprocmask resumed>~[KILL STOP RTMIN RT_1 RT_2],
8) = 0

[pid    11] <... close resumed>)        = 0

[pid    11] close(7)                    = 0

[pid    11] close(8)                    = 0

[pid    11] dup2(3, 0 <unfinished ...>

[pid    10] fork( <unfinished ...>

[pid    11] <... dup2 resumed>)         = 0

[pid    11] close(3strace: Process 12 attached

)                   = 0

[pid    10] <... fork resumed>)         = 12

[pid    12] set_tid_address(0x7fe562941fb8 <unfinished ...>

[pid    11] dup2(6, 1 <unfinished ...>

[pid    10] rt_sigprocmask(SIG_SETMASK, ~[KILL STOP RTMIN RT_1
RT_2],  <unfinished ...>

[pid    12] <... set_tid_address resumed>) = 12

[pid    11] <... dup2 resumed>)         = 1

[pid    10] <... rt_sigprocmask resumed>NULL, 8) = 0
```

```
[pid    12] rt_sigprocmask(SIG_SETMASK, ~[KILL STOP RTMIN RT_1
RT_2],  <unfinished ...>

[pid    11] close(6 <unfinished ...>

[pid    10] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid    12] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid    10] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid    11] <... close resumed>)        = 0

[pid    10] writev(1, [{iov_base="Child2 created with PID: 12", iov_len=27},
{iov_base="\n", iov_len=1}], 2 <unfinished ...>

[pid    12] rt_sigprocmask(SIG_SETMASK, Child2 created with PID: 12

[],  <unfinished ...>

[pid    11] execve("./child1", ["child1"], 0x7fffe5036758 /* 4 vars */ <unfinished
...>

[pid    10] <... writev resumed>)       = 28

[pid    12] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid    10] close(3)                    = 0

[pid    12] close(3 <unfinished ...>

[pid    10] close(5 <unfinished ...>

[pid    12] <... close resumed>)        = 0

[pid    10] <... close resumed>)        = 0

[pid    12] close(4 <unfinished ...>

[pid    11] <... execve resumed>)       = 0

[pid    10] close(6 <unfinished ...>

[pid    12] <... close resumed>)        = 0

[pid    10] <... close resumed>)        = 0

[pid    12] close(6 <unfinished ...>

[pid    11] arch_prctl(ARCH_SET_FS, 0x7ff1ee1d5b48 <unfinished ...>

[pid    10] close(8)                    = 0

[pid    11] <... arch_prctl resumed>)   = 0

[pid    10] writev(1, [{iov_base="", iov_len=0}, {iov_base="\n", iov_len=1}], 2
<unfinished ...>

[pid    12] <... close resumed>)        = 0

[pid    10] <... writev resumed>)       = 1

[pid    11] set_tid_address(0x7ff1ee1d5fb8 <unfinished ...>

[pid    10] writev(1, [{iov_base="=== Starting data processing ===",
iov_len=32}, {iov_base="\n", iov_len=1}], 2 <unfinished ...>

[pid    12=== Starting data processing ===

] close(7 <unfinished ...>

[pid    10] <... writev resumed>)       = 33

[pid    11] <... set_tid_address resumed>) = 11

[pid    12] <... close resumed>)        = 0

[pid    10] writev(1, [{iov_base="Enter strings for processing (ty"...,
iov_len=51}, {iov_base="\n", iov_len=1}], 2Enter strings for processing (type 'exit' to
quit):

) = 52

[pid    12] dup2(5, 0 <unfinished ...>

[pid    11] brk(NULL <unfinished ...>

[pid    10] writev(1, [{iov_base="> ", iov_len=2}, {iov_base=NULL, iov_len=0}],
2 <unfinished ...>

[pid    12] <... dup2 resumed>)         = 0

[pid    11] <... brk resumed>)          = 0x55dbd363c000

[pid    12] close(5>  <unfinished ...>

[pid    10] <... writev resumed>)       = 2

[pid    12] <... close resumed>)        = 0

[pid    11] brk(0x55dbd363e000 <unfinished ...>

[pid    12] dup2(8, 1)                  = 1

[pid    12] close(8 <unfinished ...>

[pid    10] read(0,  <unfinished ...>

[pid    12] <... close resumed>)        = 0

[pid    12] execve("./child2", ["child2"], 0x7fffe5036758 /* 4 vars */ <unfinished
...>
[pid    11] <... brk resumed>)         = 0x55dbd363e000

[pid    12] <... execve resumed>)      = 0

[pid    12] arch_prctl(ARCH_SET_FS, 0x7f8a94dddb48 <unfinished ...>

[pid    11] mmap(0x55dbd363c000, 4096, PROT_NONE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid    12] <... arch_prctl resumed>)  = 0

[pid    12] set_tid_address(0x7f8a94dddfb8 <unfinished ...>

[pid    11] <... mmap resumed>)        = 0x55dbd363c000

[pid    12] <... set_tid_address resumed>) = 12

[pid    11] mprotect(0x7ff1ee1d2000, 4096, PROT_READ <unfinished ...>

[pid    12] brk(NULL <unfinished ...>

[pid    11] <... mprotect resumed>)    = 0

[pid    12] <... brk resumed>)         = 0x55a5ba5d4000

[pid    12] brk(0x55a5ba5d6000 <unfinished ...>

[pid    11] mprotect(0x55dbc26d1000, 4096, PROT_READ <unfinished ...>

[pid    12] <... brk resumed>)         = 0x55a5ba5d6000

[pid    11] <... mprotect resumed>)    = 0

[pid    12] mmap(0x55a5ba5d4000, 4096, PROT_NONE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0 <unfinished ...>

[pid    11] readv(0,  <unfinished ...>

[pid    12] <... mmap resumed>)        = 0x55a5ba5d4000

[pid    12] mprotect(0x7f8a94dda000, 4096, PROT_READ) = 0

[pid    12] mprotect(0x55a58b782000, 4096, PROT_READ) = 0

[pid    12] readv(0, helllo world)

 <unfinished ...>

[pid    10] <... read resumed>"helllo world)\n", 1024) = 14

[pid    10] writev(1, [{iov_base="[Parent] Sending string #1 to Ch"...,
iov_len=52}, {iov_base="\n", iov_len=2}], 2[Parent] Sending string #1 to Child1:
'helllo world)'

) = 54

[pid    10] write(4, "helllo world)\n", 14) = 14
```

```
[pid    11] <... readv resumed>[{iov_base="h", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    10] read(7,  <unfinished ...>

[pid    11] readv(0, [{iov_base="e", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="l", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="l", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="l", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="o", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base=" ", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="w", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="o", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="r", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="l", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="d", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base=")", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] readv(0, [{iov_base="\n", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] writev(1, [{iov_base="HELLLO WORLD)\n", iov_len=14}, {iov_base=NULL, iov_len=0}], 2 <unfinished ...>

[pid    12] <... readv resumed>[{iov_base="H", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    11] <... writev resumed>)       = 14

[pid    12] readv(0,  <unfinished ...>

[pid    11] readv(0,  <unfinished ...>
[pid    12] <... readv resumed>[{iov_base="E", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="L", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="L", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="L", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="O", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base=" ", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="W", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="O", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="R", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="L", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="D", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base=")", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] readv(0, [{iov_base="\n", iov_len=1}, {iov_base="", iov_len=0}], 2) = 1

[pid    12] writev(1, [{iov_base="HELLLO_WORLD)\n", iov_len=14}, {iov_base=NULL, iov_len=0}], 2 <unfinished ...>

[pid    10] <... read resumed>"HELLLO_WORLD)\n", 1023) = 14

[pid    12] <... writev resumed>)       = 14

[pid    10] writev(1, [{iov_base="[Parent] Final result from Child"..., iov_len=49}, {iov_base="'\n\n", iov_len=3}], 2 <unfinished ...>

[Parent] Final result from Child2: 'HELLLO_WORLD)'



[pid    12] readv(0,  <unfinished ...>

[pid    10] <... writev resumed>)       = 52
```

```
[pid    10] writev(1, [{iov_base="> ", iov_len=2}, {iov_base=NULL, iov_len=0}],
2> ) = 2

[pid    10] read(0, mew mew mew)

"mew mew mew)\n", 1024) = 13

[pid    10] writev(1, [{iov_base="[Parent] Sending string #2 to Ch"...,
iov_len=51}, {iov_base="\n", iov_len=2}], 2[Parent] Sending string #2 to Child1:
'mew mew mew)'

) = 53

[pid    10] write(4, "mew mew mew)\n", 13) = 13

[pid    11] <... readv resumed>[{iov_base="m", iov_len=1}, {iov_base="",
iov_len=0}], 2) = 1

[pid    10] read(7,  <unfinished ...>

[pid    11] readv(0, [{iov_base="e", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base="w", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base=" ", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base="m", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base="e", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base="w", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base=" ", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base="m", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base="e", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base="w", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base=")", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

[pid    11] readv(0, [{iov_base="\n", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1
```

```
    [pid    11] writev(1, [{iov_base="MEW MEW MEW)\n", iov_len=13},
{iov_base=NULL, iov_len=0}], 2 <unfinished ...>

    [pid    12] <... readv resumed>[{iov_base="M", iov_len=1}, {iov_base="",
iov_len=0}], 2) = 1

    [pid    11] <... writev resumed>)       = 13

    [pid    12] readv(0,  <unfinished ...>

    [pid    11] readv(0,  <unfinished ...>

    [pid    12] <... readv resumed>[{iov_base="E", iov_len=1}, {iov_base="",
iov_len=0}], 2) = 1

    [pid    12] readv(0, [{iov_base="W", iov_len=1}, {iov_base="", iov_len=0}], 2)
= 1

    [pid    12] readv(0, [{iov_base=" ", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

    [pid    12] readv(0, [{iov_base="M", iov_len=1}, {iov_base="", iov_len=0}], 2)
= 1

    [pid    12] readv(0, [{iov_base="E", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

    [pid    12] readv(0, [{iov_base="W", iov_len=1}, {iov_base="", iov_len=0}], 2)
= 1

    [pid    12] readv(0, [{iov_base=" ", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

    [pid    12] readv(0, [{iov_base="M", iov_len=1}, {iov_base="", iov_len=0}], 2)
= 1

    [pid    12] readv(0, [{iov_base="E", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

    [pid    12] readv(0, [{iov_base="W", iov_len=1}, {iov_base="", iov_len=0}], 2)
= 1

    [pid    12] readv(0, [{iov_base=")", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

    [pid    12] readv(0, [{iov_base="\n", iov_len=1}, {iov_base="", iov_len=0}], 2) =
1

    [pid    12] writev(1, [{iov_base="MEW_MEW_MEW)\n", iov_len=13},
{iov_base=NULL, iov_len=0}], 2) = 13

    [pid    10] <... read resumed>"MEW_MEW_MEW)\n", 1023) = 13

    [pid    12] readv(0,  <unfinished ...>
    [pid    10] writev(1, [{iov_base="[Parent] Final result from Child"...,
iov_len=48}, {iov_base="\n\n", iov_len=3}], 2[Parent] Final result from Child2:
'MEW_MEW_MEW)'



    ) = 51

    [pid    10] writev(1, [{iov_base="> ", iov_len=2}, {iov_base=NULL, iov_len=0}],
2> ) = 2

    [pid    10] read(0, exit

    "exit\n", 1024)     = 5

    [pid    10] writev(1, [{iov_base="", iov_len=0}, {iov_base="\n", iov_len=1}], 2

    ) = 1

    [pid    10] writev(1, [{iov_base="=== Shutting down ===", iov_len=21},
{iov_base="\n", iov_len=1}], 2=== Shutting down ===

    ) = 22

    [pid    10] writev(1, [{iov_base="Closing pipes to signal children"...,
iov_len=43}, {iov_base="\n", iov_len=1}], 2Closing pipes to signal children to exit...

    ) = 44

    [pid    10] close(4)                    = 0

    [pid    11] <... readv resumed>[{iov_base="", iov_len=1}, {iov_base="",
iov_len=0}], 2) = 0

    [pid    11] exit_group(0 <unfinished ...>

    [pid    10] close(7 <unfinished ...>

    [pid    11] <... exit_group resumed>)   = ?

    [pid    10] <... close resumed>)        = 0

    [pid    10] writev(1, [{iov_base="Waiting for children to exit...", iov_len=31},
{iov_base="\n", iov_len=1}], 2Waiting for children to exit...

    ) = 32

    [pid    10] wait4(11,  <unfinished ...>

    [pid    11] +++ exited with 0 +++

    [pid    12] <... readv resumed>[{iov_base="", iov_len=1}, {iov_base="",
iov_len=0}], 2) = 0

    [pid    10] <... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) ==
0}], 0, NULL) = 11
```

```
[pid    12] exit_group(0 <unfinished ...>

[pid    10] --- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED,
si_pid=11, si_uid=0, si_status=0, si_utime=0, si_stime=0} ---

[pid    12] <... exit_group resumed>)   = ?

[pid    10] wait4(12,  <unfinished ...>

[pid    12] +++ exited with 0 +++

<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0,
NULL) = 12

--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=12,
si_uid=0, si_status=0, si_utime=0, si_stime=0} ---

writev(1, [{iov_base="Child processes exited with code"..., iov_len=53},
{iov_base="\n", iov_len=1}], 2Child processes exited with codes: Child1=0, Child2=0

) = 54

writev(1, [{iov_base="Program finished successfully.", iov_len=30},
{iov_base="\n", iov_len=1}], 2Program finished successfully.

) = 31

exit_group(0)                           = ?

+++ exited with 0 +++Strace
```

Выделенные системные вызовы

```
pipe([3, 4]) = 0

pipe([5, 6]) = 0

pipe([7, 8]) = 0


fork() = 10

fork() = 11


[pid 10] dup2(3, 0) = 0

[pid 10] dup2(6, 1) = 1

[pid 11] dup2(5, 0) = 0

[pid 11] dup2(8, 1) = 1
```

```
[pid 10] close(4) = 0
[pid 10] close(5) = 0
[pid 10] close(7) = 0
[pid 10] close(8) = 0
[pid 10] close(6) = 0
[pid 11] close(3) = 0
[pid 11] close(4) = 0
[pid 11] close(6) = 0
[pid 11] close(7) = 0
[pid 11] close(8) = 0

[pid 10] execve("./child1", ["child1"], 0x7ffe44d484a8 /* 3 vars */) = 0
[pid 11] execve("./child2", ["child2"], 0x7ffe44d484a8 /* 3 vars */) = 0

write(4, "hello world\n", 12) = 12
read(7, "HELLO_WORLD\n", 1024) = 12

close(4) = 0
close(7) = 0

wait4(10, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 10
wait4(11, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 11
```

Вывод

В ходе выполнения лабораторной работы были успешно освоены принципы управления процессами и организации межпроцессного взаимодействия через неименованные каналы в операционных системах. Основные сложности возникли при отладке корректного закрытия файловых дескрипторов и обеспечении синхронизации между процессами. В процессе работы научилась использовать системные вызовы fork(), pipe(), dup2(), execve() и другие, что позволило глубже понять механизмы работы операционной системы с процессами и межпроцессным взаимодействием. Полученные навыки будут полезны для выполнения последующих лабораторных работ по данной дисциплине и понимания фундаментальных принципов построения Unix-подобных систем.