МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 10**

по дисциплине:” Системное программирование”

на тему: “РАБОТА С КАНАЛАМИ”

Выполнил**:** студент группы 10701321

Грудинский К. А.

Принял**:** Кондратёнок Е.В.

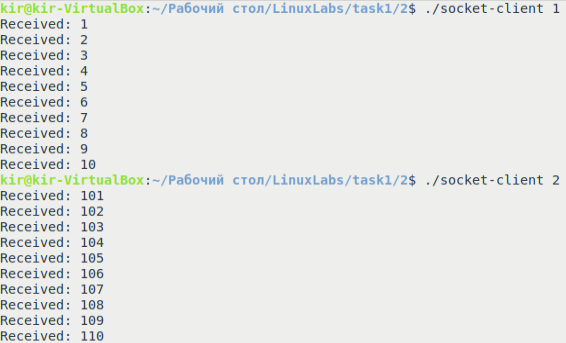
Минск 2023

# 

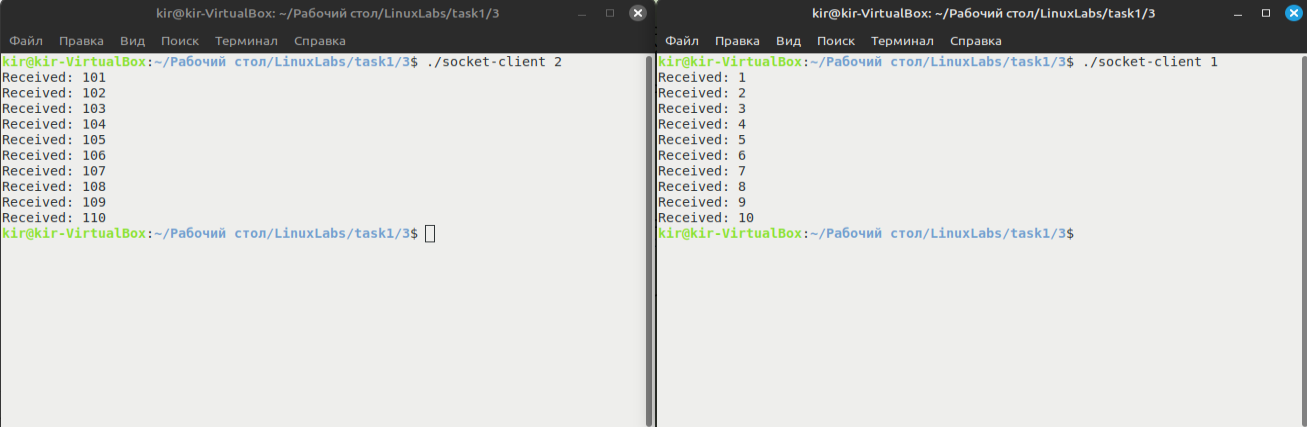
# 

1.1





1.2



1.3

# Лабораторная работа № 10. РАБОТА С КАНАЛАМИ

# Цель работы: Освоить механизм взаимодействия между процессами на основе каналов (pipes).Linux

**Задание 1**

Передайте через неименованный и именованный каналы следующую информацию связанную с периодами выполнения задания:

1-4 неделя семестра системное время и дату.

5-7 неделя семестра получить календарь, системное время и дату.

8-10 неделя семестра получить имя ядра и название операционной системы, а

также системную дату и время.

11-16 неделя семестра получить имя операционной системы, версии и реализацию ядра, системную дату и время, а также календарь.

$ Command –help

Бонус. Для автоматизации процесса напишите скрипт.**Решение**

#!/bin/bash

# Создаем неименованный канал

PIPE=$(mktemp -u)

mkfifo $PIPE

# Именованный канал

NAMED\_PIPE="named\_pipe"

mkfifo $NAMED\_PIPE

# Функция для получения информации об операционной системе

get\_os\_info() {

echo "Operating System: $(cat /etc/os-release | grep PRETTY\_NAME | cut -d= -f2)"

echo "Kernel Version: $(uname -r)"

}

# Функция для получения системной даты и времени

get\_date\_time() {

echo "Date and Time: $(date)"

}

# Функция для получения календаря

get\_calendar() {

echo "Calendar:"

cal

}

# Функция для записи информации в неименованный канал

write\_to\_pipe() {

get\_os\_info >$PIPE

get\_date\_time >$PIPE

get\_calendar >$PIPE

echo "END" >$PIPE

}

# Функция для записи информации в именованный канал

write\_to\_named\_pipe() {

get\_os\_info >$NAMED\_PIPE

get\_date\_time >$NAMED\_PIPE

get\_calendar >$NAMED\_PIPE

echo "END" >$NAMED\_PIPE

}

# Запускаем функцию записи в фоновом режиме

write\_to\_pipe &

# Запускаем функцию записи в именованный канал в фоновом режиме

write\_to\_named\_pipe &

# Читаем из неименованного канала

while read line; do

if [ "$line" == "END" ]; then

break

fi

echo $line

done <$PIPE

# Читаем из именованного канала

while read line; do

if [ "$line" == "END" ]; then

break

fi

echo $line

done <$NAMED\_PIPE

# Удаляем временные файлы

rm $PIPE

rm $NAMED\_PIPE

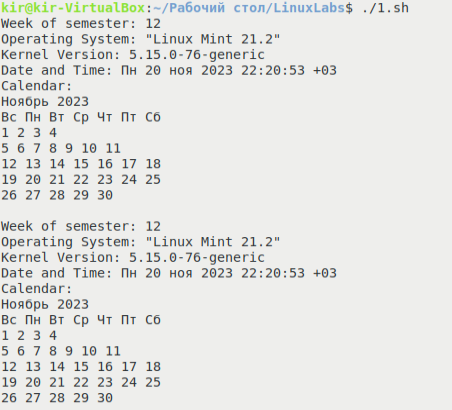


Рисунок 1 – задание 1

**Задание 2.**

Реализуйте взаимодействия двух процессов-братьев через канал, созданный в

родительском процессе, с использованием системных вызовов (см. упражнение 2, пример №1).

Разработайте универсальную программу перенаправления стандартного

ввода-вывода (см. упражнение 2, пример №2).

18

КАНАЛЫ И МЕЖПРОЦЕССНОЕ ВЗАИМОДЕЙСТВИЕ СЕРГЕЙ СТАНКЕВИЧ, МИНСК

Выполните программу, указанную в упражнении 2, пример №3 с выводом содержимого каталогов на выбор: proc, lib, mnt, usr, etc, home или dev.

Выполните программу, указанную в упражнении 2, пример №4 с выводом с

выводом данных, указанных в задании 1.

\* Во всех программах отобразить данные, указанные в задании 1, а также сведения,

касающиеся разработчиков, выполнивших данное задание.

**Решение**

**Задание 2.1**

*Pipe1-parent.c*

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

#define STR\_SIZE 32

int main (void)

{

int pf[2];

int pid1, pid2;

char spf [2][STR\_SIZE];

if (pipe (pf) == -1) { fprintf (stderr, "pipe() error\n"); return 1; }

sprintf (spf[0], "%d", pf[0]);

sprintf (spf[1], "%d", pf[1]);

if ((pid1 = fork ()) == 0) { close (pf[0]);

execl ("./pipe1-src", "pipe1-src", spf[1], NULL);

fprintf (stderr, "exec() [src] error\n"); return 1;

}

if ((pid2 = fork ()) == 0) { close (pf[1]);

execl ("./pipe1-dst", "pipe1-dst", spf[0], NULL);

fprintf (stderr, "exec() [dst] error\n"); return 1;

}

waitpid (pid1, NULL, 0);

close (pf[0]);

close (pf[1]);

waitpid (pid2, NULL, 0);

return 0;

*Pipe1-dst.c*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

int main (int argc, char \*\* argv)

{

int fd;

char ch;

if (argc < 2) {

fprintf (stderr, "dst: Too few arguments\n");

return 1;

}

fd = atoi (argv[1]);

while (read (fd, &ch, 1) > 0)

write (1, &ch, 1);

close (fd);

return 0;

}

*Pipe1-dst.src*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#define WAIT\_SECS 5

int main (int argc, char \*\* argv)

{

int i, fd;

if (argc < 2) {

fprintf (stderr, "src: Too few arguments\n");

return 1;

}

fd = atoi (argv[1]);

fprintf (stderr, "Wait please");

for (i = 0; i < WAIT\_SECS; i++, sleep (1))

fprintf (stderr, ".");

fprintf (stderr, "\n");

// Определение информации об операционной системе и ядре

char os\_info[256];

snprintf(os\_info, sizeof(os\_info), "OS: %s, Kernel: %s\n", getenv("OSTYPE"), getenv("KERNEL\_VERSION"));

// Получение системной даты и времени

FILE \*date\_pipe = popen("date", "r");

char date\_time[256];

fgets(date\_time, sizeof(date\_time), date\_pipe);

pclose(date\_pipe);

// Получение календаря

FILE \*cal\_pipe = popen("cal", "r");

char calendar[512];

fgets(calendar, sizeof(calendar), cal\_pipe);

pclose(cal\_pipe);

// Создание строки для передачи

char message[1024];

snprintf(message, sizeof(message), "%s%s%s\nWeek: 12\n", os\_info, date\_time, calendar);

// Запись в файловый дескриптор

if (write(fd, message, strlen(message)) == -1) {

fprintf(stderr, "src: write() error\n");

return 1;

}

close(fd);

return 0;

}

}

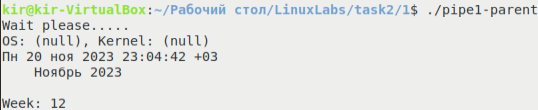


Рисунок 2 – задание 2.1

**Задание 2.1**

*dup01.c*

#include <unistd.h>

#include <stdio.h>

#include <fcntl.h>

#include <stdlib.h>

#define FILENAME "myfile"

int main(void) {

int fd = open(FILENAME, O\_WRONLY | O\_CREAT | O\_TRUNC, 0640);

if (fd == -1) {

fprintf(stderr, "open() error\n");

return 1;

}

// Определение информации об операционной системе и ядре

char os\_info[256];

snprintf(os\_info, sizeof(os\_info), "OS: %s, Kernel: %s\n", getenv("OSTYPE"), getenv("KERNEL\_VERSION"));

// Получение системной даты и времени

FILE \*date\_pipe = popen("date", "r");

char date\_time[256];

fgets(date\_time, sizeof(date\_time), date\_pipe);

pclose(date\_pipe);

// Получение календаря

FILE \*cal\_pipe = popen("cal", "r");

char calendar[512];

fgets(calendar, sizeof(calendar), cal\_pipe);

pclose(cal\_pipe);

// Создание строки для записи в файл

char message[1024];

snprintf(message, sizeof(message), "%s%s%s\nWeek: 12\n", os\_info, date\_time, calendar);

// Запись в файловый дескриптор

if (write(fd, message, strlen(message)) == -1) {

fprintf(stderr, "write() error\n");

close(fd);

return 1;

}

close(fd);

return 0;

}

*dup02.c*

#include <unistd.h>

#include <stdio.h>

#include <fcntl.h>

#define FILENAME "myfile"

int main (void)

{

char ch;

int fd = open (FILENAME, O\_RDONLY);

if (fd == -1) {

fprintf (stderr, "open() error\n"); return 1;

}

if (dup2 (fd, 0) == -1) {

fprintf (stderr, "dup2() error\n"); return 1;

}

while (read (0, &ch, 1) > 0)

write (1, &ch, 1);

close (fd);

return 0;

}



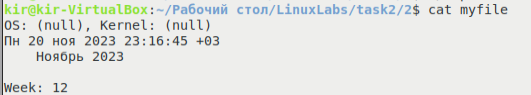




Рисунок 3 – задание 2.2

**Задание 2.3**

*pipe2.c*

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

int main (int argc, char \*\* argv)

{

int pf[2];

int pid1, pid2;

if (argc < 3) { fprintf (stderr, "Too few arguments\n"); return 1; }

if (pipe (pf) == -1) { fprintf (stderr, "pipe() error\n"); return 1; }

if ((pid1 = fork ()) == 0) {

dup2 (pf[1], 1); close (pf[0]);

execlp ("ls", "ls", argv[1], NULL);

fprintf (stderr, "exec() [1] error\n");

return 1;

}

if ((pid1 = fork ()) == 0) {

dup2 (pf[0], 0);

close (pf[1]);

execlp ("grep", "grep", "-i", argv[2], NULL);

fprintf (stderr, "exec() [2] error\n");

return 1;

}

close (pf[1]);

waitpid (pid1, NULL, 0);

close (pf[0]);

waitpid (pid2, NULL, 0);

return 0;

}

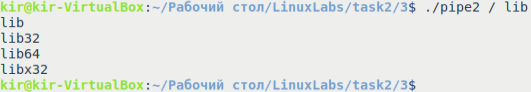


Рисунок 4 – задание 2.3

**Задание 2.4**

*fifo2-server.c*

#include <stdio.h>

#include <sys/stat.h>

#include <string.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#define FIFO\_NAME "myfifo"

#define BUF\_SIZE 1024

int main(void) {

FILE \*fifo;

char \*buf;

if (mkfifo(FIFO\_NAME, 0640) == -1) {

fprintf(stderr, "Can't create fifo\n");

return 1;

}

fifo = fopen(FIFO\_NAME, "r");

if (fifo == NULL) {

fprintf(stderr, "Cannot open fifo\n");

return 1;

}

buf = (char \*)malloc(BUF\_SIZE);

if (buf == NULL) {

fprintf(stderr, "malloc() error\n");

return 1;

}

// Чтение из FIFO и вывод в консоль

while (fgets(buf, BUF\_SIZE, fifo) != NULL) {

printf("%s", buf);

}

fclose(fifo);

free(buf);

unlink(FIFO\_NAME);

return 0;

}

*fifo2-client.c*

#include <stdio.h>

#include <fcntl.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#define FIFO\_NAME "myfifo"

int main(void) {

int fifo;

fifo = open(FIFO\_NAME, O\_WRONLY);

if (fifo == -1) {

fprintf(stderr, "Cannot open fifo\n");

return 1;

}

// Определение информации об операционной системе и ядре

char os\_info[256];

snprintf(os\_info, sizeof(os\_info), "OS: %s, Kernel: %s\n", getenv("OSTYPE"), getenv("KERNEL\_VERSION"));

// Получение системной даты и времени

FILE \*date\_pipe = popen("date", "r");

char date\_time[256];

fgets(date\_time, sizeof(date\_time), date\_pipe);

pclose(date\_pipe);

// Получение календаря

FILE \*cal\_pipe = popen("cal", "r");

char calendar[512];

fgets(calendar, sizeof(calendar), cal\_pipe);

pclose(cal\_pipe);

// Создание строки для записи в FIFO

char message[1024];

snprintf(message, sizeof(message), "%s%s%s\nWeek: 12\n", os\_info, date\_time, calendar);

// Запись в FIFO

if (write(fifo, message, strlen(message)) == -1) {

fprintf(stderr, "write() error\n");

close(fifo);

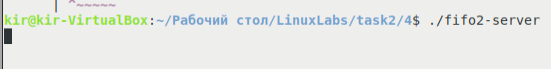
return 1;

}

close(fifo);

return 0;

}





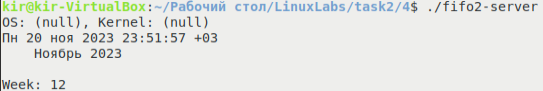


Рисунок 5 – задание 2.4

**Задание 3**

Разработайте универсальную программу передачи данных с помощью именованного канала с выводом данных о разработчике программы и параметров, указанных в задании №1.

**Решение**

*writer.c*

#include <stdio.h>

#include <fcntl.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#define FIFO\_NAME "myfifo"

#define BUF\_SIZE 1024

int main(void) {

int fifo;

fifo = open(FIFO\_NAME, O\_WRONLY);

if (fifo == -1) {

fprintf(stderr, "Cannot open fifo\n");

return 1;

}

// Определение информации об операционной системе и ядре

char os\_info[256];

snprintf(os\_info, sizeof(os\_info), "OS: %s, Kernel: %s\n", getenv("OSTYPE"), getenv("KERNEL\_VERSION"));

// Получение системной даты и времени

FILE \*date\_pipe = popen("date", "r");

char date\_time[256];

fgets(date\_time, sizeof(date\_time), date\_pipe);

pclose(date\_pipe);

// Получение календаря

FILE \*cal\_pipe = popen("cal", "r");

char calendar[512];

fgets(calendar, sizeof(calendar), cal\_pipe);

pclose(cal\_pipe);

// Создание строки для записи в FIFO

char message[1024];

snprintf(message, sizeof(message), "%s%s%s\nWeek: 12\nGrudinskiy K. A.\n", os\_info, date\_time, calendar);

// Запись в FIFO

if (write(fifo, message, strlen(message)) == -1) {

fprintf(stderr, "write() error\n");

close(fifo);

return 1;

}

close(fifo);

return 0;

}

*reader.c*

#include <stdio.h>

#include <sys/stat.h>

#include <string.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#define FIFO\_NAME "myfifo"

#define BUF\_SIZE 1024

int main(void) {

FILE \*fifo;

char \*buf;

if (mkfifo(FIFO\_NAME, 0640) == -1) {

fprintf(stderr, "Can't create fifo\n");

return 1;

}

fifo = fopen(FIFO\_NAME, "r");

if (fifo == NULL) {

fprintf(stderr, "Cannot open fifo\n");

return 1;

}

buf = (char \*)malloc(BUF\_SIZE);

if (buf == NULL) {

fprintf(stderr, "malloc() error\n");

return 1;

}

// Чтение из FIFO и вывод в консоль

while (fgets(buf, BUF\_SIZE, fifo) != NULL) {

printf("%s", buf);

}

fclose(fifo);

free(buf);

unlink(FIFO\_NAME);

return 0;

}

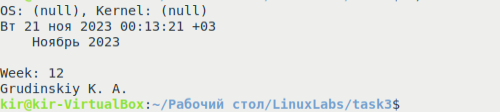


Рисунок 6 – задание 3

Вывод*:* В ходе выполнения лабораторной работы был основан механизм взаимодействия между процессами на основе каналов.