Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине: «Разработка программных систем»

Студент	Израелян Ева Арамовна
Группа	РК6-64Б
Тип задания	Лабораторная работа №1
Тема лабораторной работы	«Многопроцессное программирование»

Студент _____ **Израелян Е.А.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Федорук В.Г.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2022 г.

Оглавление

Текст задания на лабораторную работу 3

Описание структуры программы и реализованных способов взаимодействия процессов 3

Описание основных используемых структур данных 4

Блок-схема программы 5

Пример результатов работы программы 6

Текст программы 6

Текст задания на лабораторную работу

Составить программу cpr, осуществляющую рекурсивное копирование поддерева файловой системы ОС UNIX. Программа cpr должна вызываться на выполнение в соответствие со следующим синтаксисом:

cpr *имя_дир_1 имя_дир_2*,

где *имя_дир_1* - имя директории, служащее корнем поддерева файлов, подлежащего копированию; *имя_дир_2* - имя директории, в которую должно быть скопировано поддерево.

Замечание. На языке оболочки ОС UNIX эта задача решается с помощью конвейера из трех команд:

tar cf - *имя_дир_1* | (cd *имя_дир_2* ; tar xf -)

Описание структуры программы и реализованных способов взаимодействия процессов

Производится проверка количества аргументов. В первом аргументе должно содержаться название директории, служащее корнем поддерева, подлежащего копированию, а во втором аргументе – название директории, в которую будет произведено копирование.

Программа функционирует в рамках двух процессов. С помощью системного вызова fork() создаётся копия родительского процесса. Процесс-родитель будет отвечать за создание архива с поддеревом, подлежащим копированию. Процесс-потомок – за извлечение содержимого архива в директорию, указанную во втором аргументе программы.

Взаимодействие между процессами осуществляется с помощью системного вызова pipe(), создающего канал передачи данных. Он потребуется для передачи содержимого указанной директории из вывода утилиты tar в режиме копирования на вход утилиты tar в режиме распаковки.

Для перенаправления ввода-вывода процессов при помощи системного вызова dup2() создаются дубликаты файловых дескрипторов. Стандартный вывод родителя (stdout) заменяется на запись в канал. Стандартный ввод (stdin) процесса-потомка заменяется на чтение из канала. В процессе-родителе выполняется системный вызов execl(), в результате которого создаётся архив,

содержащий поддерево, подлежащее копированию. В процессе-потомке с помощью вызова execl() происходит извлечение архива в указанную директорию. Для реализации копирования используется команда tar с ключами:

1) c – создание нового архива

2) f – задание имени архива

3) x – извлечение архива

4) C – указывает место распаковки архива

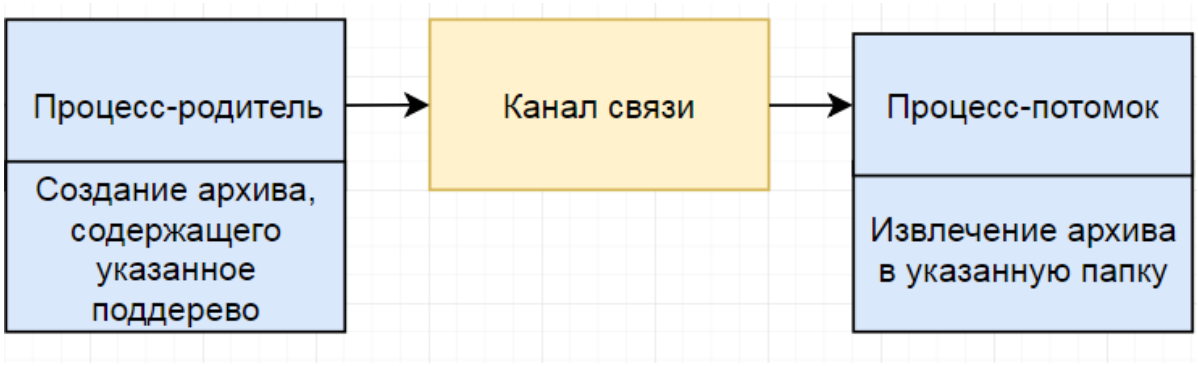

Рисунок 1. Схема взаимодействия между процессами

Описание основных используемых структур данных

Для реализации взаимодействия между процессами было использовано перенаправление ввода-вывода процессов. Для хранения двух целочисленных файловых дескрипторов, указывающих на концы канала, использовался целочисленный массив int fds[2].

- fds[0] - на конец канала для чтения
- fds[1] - на конец канала для записи

Блок-схема программы

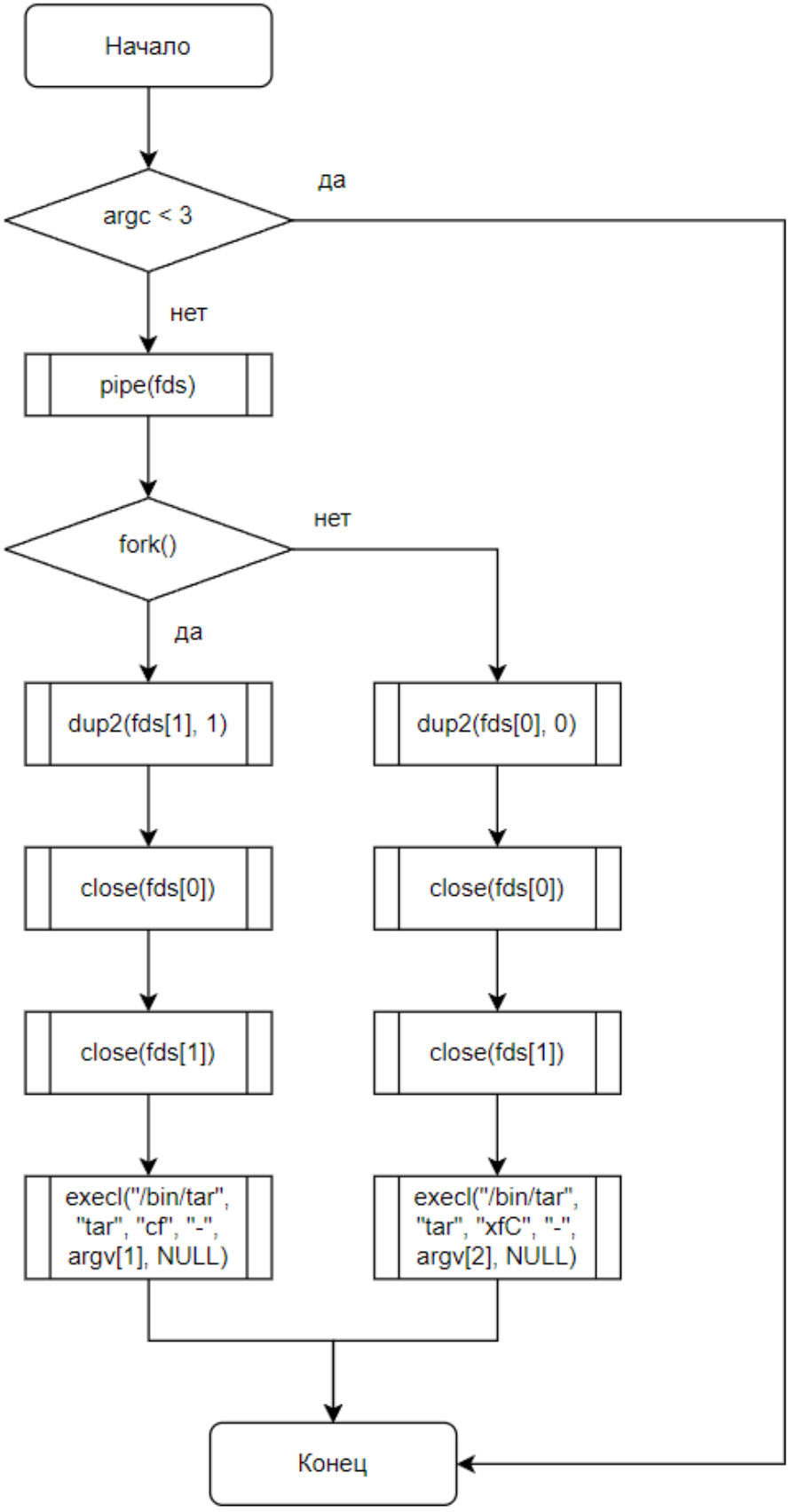

Рисунок 2. Блок-схема работы программы

Пример результатов работы программы

```
eva@eva-VirtualBox:~/lab1/old$ ls
test.txt
eva@eva-VirtualBox:~/lab1/old$ cd ..
eva@eva-VirtualBox:~/lab1$ ls
cpr  cpr.c  new  old
eva@eva-VirtualBox:~/lab1$ ./cpr new old
eva@eva-VirtualBox:~/lab1$ cd old
eva@eva-VirtualBox:~/lab1/old$ ls
new  test.txt
```

Рисунок 3. Пример результата работы программы

Текст программы

Файл cpr.c

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char** argv)
{
	if (argc < 3)
	{
		printf("Unacceptable number of arguments\n");
		printf("First - old_folder(input); second -
new_folder(output)\n");
		exit(1);
	}
	int fds[2];
	pipe(fds); // Создание канала
	if (fork()) // Создание процесса-потомка
	{
		dup2(fds[1], 1);
		close(fds[0]);
		close(fds[1]);
		execl("/bin/tar", "tar", "cf", "-", argv[1], NULL);
	}
	else
	{
		dup2(fds[0], 0);
		close(fds[0]);
		close(fds[1]);
		execl("/bin/tar", "tar", "xfC", "-", argv[2], NULL);
	}
	return 0;
}
```