

Лабораторная работа №12

Операионные системы

Тойчубекова Асель Нурлановна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	17

Список иллюстраций

4.1	Создание и открытие файла	9
4.2	Редактирование файла	9
4.3	Запуск программы	10
4.4	Архив файла	10
4.5	Создание и открытие файла	10
4.6	Редактирование файла	11
4.7	Запуск программы	11
4.8	Создание и открытие файла	11
4.9	Редактирование файла	12
4.10	Запуск программы	13
4.11	Создание и открытие файла	13
4.12	Редактирование файла	14
4.13	Запуск программы	15
4.14	каталог по адресу	15
4.15	Запуск программы	15
4.16	каталог по адресу	16

Список таблиц

1 Цель работы

Целью данной лабораторной работы является изучение основы программирования в оболочке ОС LINUX. Также научиться писать небольшие командные файлы.

2 Задание

1. Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.
2. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.
3. Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.
4. Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки

3 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- C-оболочка (или csh) — надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода.

POSIX-совместимые оболочки разработаны на базе оболочки Корна.

Командные файлы, также известные как shell-скрипты, являются текстовыми файлами, содержащими последовательность команд командного процессо-

ра. Командные файлы позволяют пользователям автоматизировать задачи, выполняя несколько команд подряд без необходимости вводить их вручную.

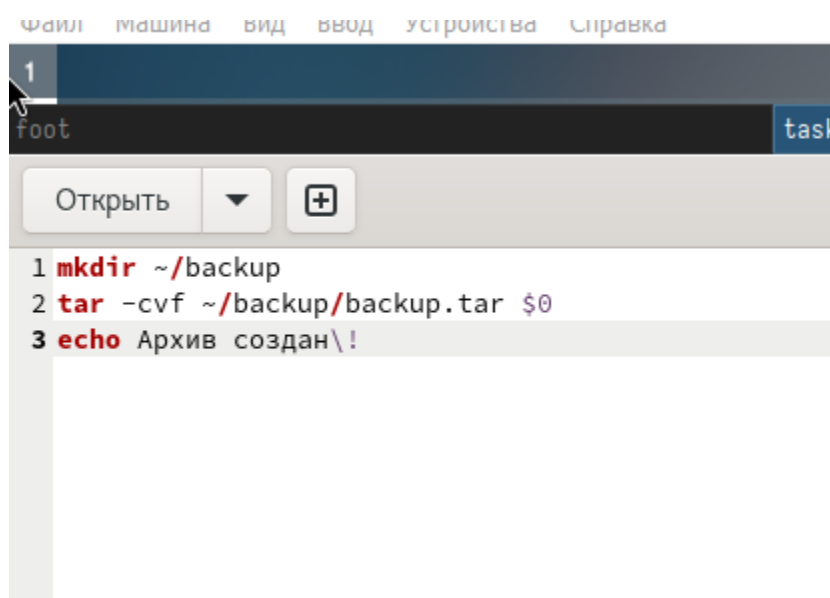
4 Выполнение лабораторной работы

Для начала я создаю файл task1, в котором буду писать программу и открою его в редакторе gedit (рис. 4.1).

```
[antoychubekova@antoychubekova ~]$ touch task1.sh  
[antoychubekova@antoychubekova ~]$ gedit task1.sh
```

Рис. 4.1: Создание и открытие файла

Редактирую файл, записывая код программы, которая будет делать резервную копию самого себя и при этом файл должен архивироваться. В этом коде я сперва создаю каталог backup, где будут храниться резервная копия и архив файла. С помощью команды tar -cvf создаю архив из резервной копии. С помощью команды echo вывожу сведения о созданном архиве. (рис. 4.2).

The image shows a screenshot of the gedit text editor. At the top, there is a menu bar with options: 'Файл', 'машинка', 'вид', 'ввод', 'устройства', and 'Справка'. Below the menu bar, there is a toolbar with a button labeled 'Открыть', a dropdown arrow, and a button with a plus sign. The main text area contains three lines of code:

```
1 mkdir ~/backup  
2 tar -cvf ~/backup/backup.tar $0  
3 echo Архив создан\!
```

Рис. 4.2: Редактирование файла

Меняю права доступа, включая права доступа на выполнение. Вызываю командный файл на выполнение и вижу сведения о том, что архив был создан, перейдя в домашний каталог в каталог backup видим, что архив удачно создан. (рис. 4.3 и рис. 4.4).

```
[antoyjchubekova@antoyjchubekova ~]$ chmod +x task1.sh
[antoyjchubekova@antoyjchubekova ~]$ bash task1.sh
task1.sh
Архив создан!
```

Рис. 4.3: Запуск программы

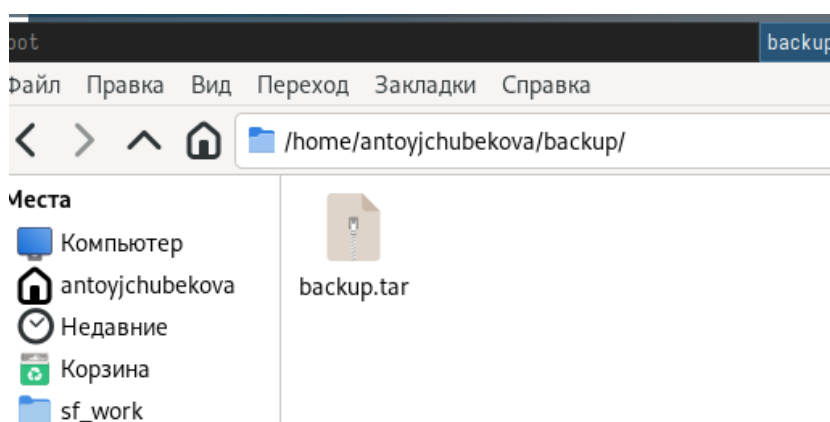


Рис. 4.4: Архив файла

Создаю файл task2, в котором буду писать программу и открою его в редакторе gedit (рис. 4.5).

```
[antoyjchubekova@antoyjchubekova ~]$ touch task2.sh
[antoyjchubekova@antoyjchubekova ~]$ gedit task2.sh
```

Рис. 4.5: Создание и открытие файла

Редактирую файл, записывая код программы, которая будет обрабатывать любое произвольное число аргументов командной строки. Для этого я создаю цикл который будет проходить по всей командной строке, по всем аргументам и буду выводить каждый из них с помощью команды echo, в конце добавляю done для указания, что конец цикла. (рис. 4.6).

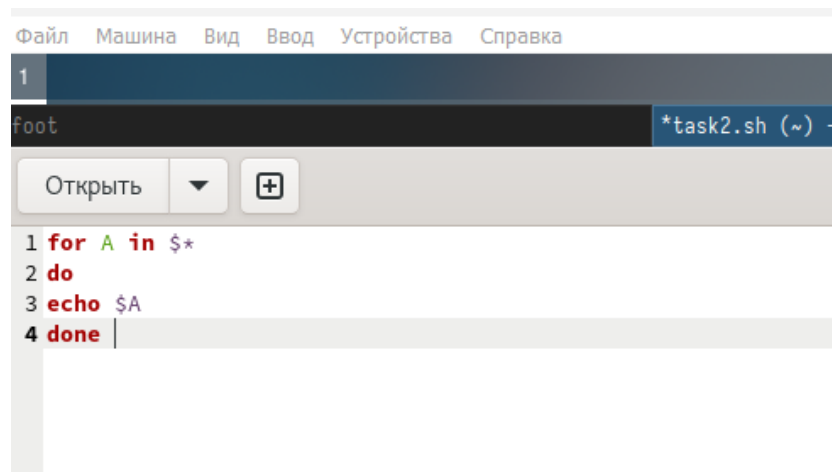


Рис. 4.6: Редактирование файла

Меняю права доступа, включая права на выполнение. Вызываю командный файл на выполнения в конце указывая аргументы, мы видим, что программа корректно работает и все аргументы выводятся на экран. (рис. 4.7).

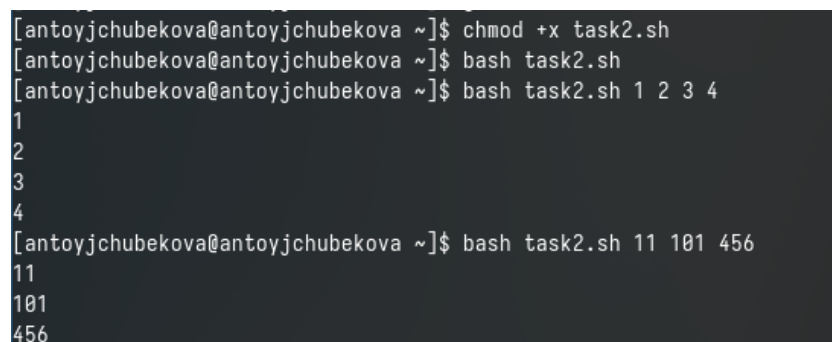


Рис. 4.7: Запуск программы

Создаю файл task3, в котором буду писать программу и открою его в редакторе gedit (рис. 4.8).

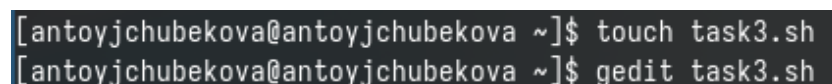
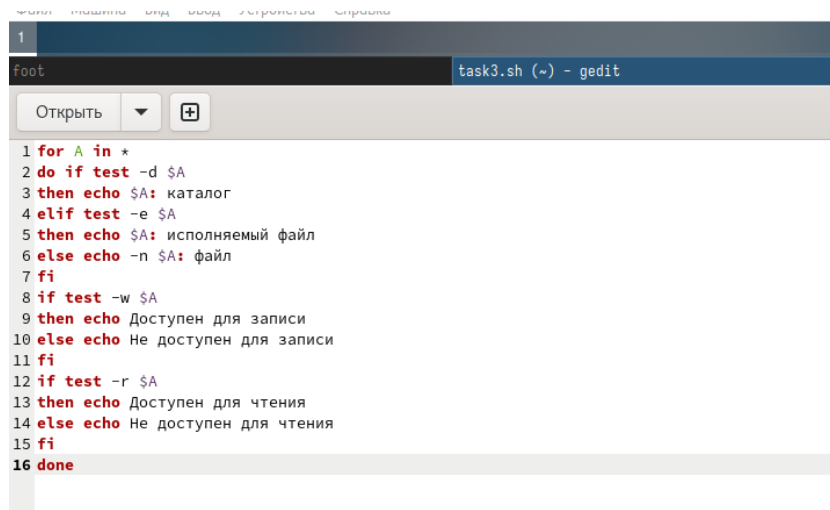


Рис. 4.8: Создание и открытие файла

Редактирую файл, который будет выводить данные о каталоге как команда ls, а также выводит информацию о правах доступа. Для этого я с помощью цик-

ла for прохожусь по асем у содержанию данного каталога, указывая * для этого. Паралельно проверяю если это каталог, исполняемый файл или просто файл с помощью команды test и его опций -d(для директорий) -e(для исполняемого файла) -f(для файла) и используя команду if/elif/else как и в других высокоуровневых языках программирования. Паралельно с проверкой если условие истина вывожу соответствующее сообщение о том каталог это или файл и о правах доступа. (рис. 4.9).



```
1 for A in *
2 do if test -d $A
3 then echo $A: каталог
4 elif test -e $A
5 then echo $A: исполняемый файл
6 else echo -n $A: файл
7 fi
8 if test -w $A
9 then echo Доступен для записи
10 else echo Не доступен для записи
11 fi
12 if test -r $A
13 then echo Доступен для чтения
14 else echo Не доступен для чтения
15 fi
16 done
```

Рис. 4.9: Редактирование файла

Меняю права доступа, включая права на выполнение. Вызываю командный файл на выполнение, мы видим, что программа корректно работает и на экран выводится все содержимое домашнего каталога с правами доступа. (рис. 4.10).

```

[antoyjchubekova@antoyjchubekova ~]$ chmod +x task3.sh
[antoyjchubekova@antoyjchubekova ~]$ bash task3.sh
abc1: исполняемый файл
Доступен для записи
Доступен для чтения
a.sh: исполняемый файл
Доступен для записи
Доступен для чтения
australia: каталог
Доступен для записи
Доступен для чтения
backup: каталог
Доступен для записи
Доступен для чтения
b.sh: исполняемый файл
Доступен для записи
Доступен для чтения
crr.cpp: исполняемый файл
Доступен для записи
Доступен для чтения
c.sh: исполняемый файл
Доступен для записи
Доступен для чтения
directory: каталог
Доступен для записи
Доступен для чтения
Downloads: каталог
Доступен для записи
Доступен для чтения
d.sh: исполняемый файл
Доступен для записи
Доступен для чтения
feathers: исполняемый файл
Доступен для записи
Доступен для чтения
file.txt: исполняемый файл
Доступен для записи
Доступен для чтения

```

Рис. 4.10: Запуск программы

Создаю файл task4, в котором буду писать программу и открою его в редакторе gedit (рис. 4.11).

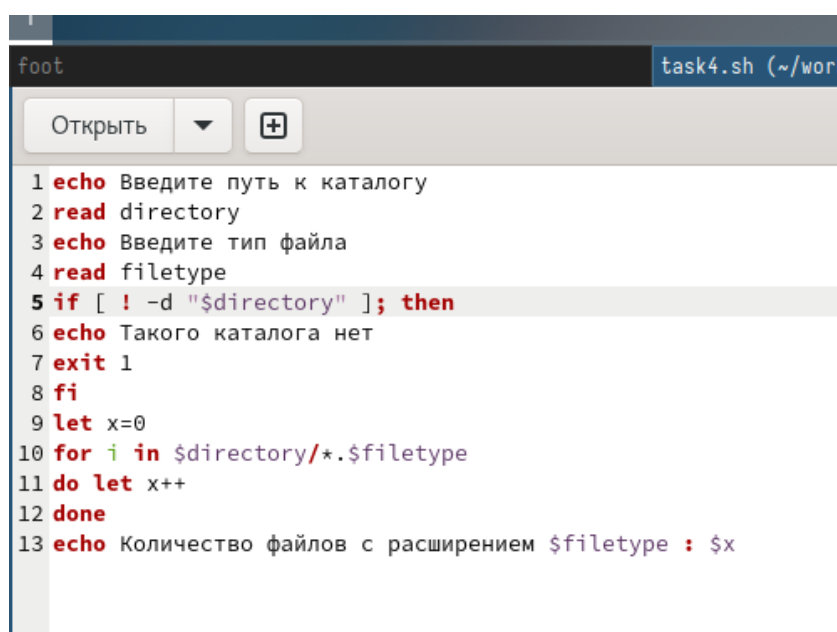
```

[antoyjchubekova@antoyjchubekova work]$ touch task4.sh
[antoyjchubekova@antoyjchubekova work]$ gedit task4.sh
[antoyjchubekova@antoyjchubekova work]$ chmod +x task4.sh

```

Рис. 4.11: Создание и открытие файла

Редактирую файл, который будет получать в качестве аргумента командной строки формат файла и вычитать количество таких файлов в указанной директории. Путь к директории также передается в виде аргумента командной строки. Для этого я сперва с помощью команды `echo` и `read` вывожу на экран запрос пользователю, чтобы он ввел значения пути каталога и тип файла. Далее я проверяю существует ли такой каталог, если нет то вывожу об этом текст. Завожу переменную `x` и с помощью цикла ищу файла соответствующие требуемой, если нахожу прибавляю к `x + 1` который изначально был равен 0. Завершаю цикл с `done` и вывожу на экран `x`. (рис. 4.12).



```
1 echo Введите путь к каталогу
2 read directory
3 echo Введите тип файла
4 read filetype
5 if [ ! -d "$directory" ]; then
6 echo Такого каталога нет
7 exit 1
8 fi
9 let x=0
10 for i in $directory/*. $filetype
11 do let x++
12 done
13 echo Количество файлов с расширением $filetype : $x
```

Рис. 4.12: Редактирование файла

Меняю права доступа, включая права на выполнение. Вызываю командный файл на выполнение, сперва проверяю каталог по адресу `/home/antoychubekova/fun` на количество файлов с `sh` и получаю ответ 3, перейдя по данному адресу мы видим, что программа сработала верно. рис. 4.13 и рис. 4.14).

```

[antoyjchubekova@antoyjchubekova work]$ chmod +x task4.sh
[antoyjchubekova@antoyjchubekova work]$ bash task4.sh
Введите путь к каталогу
/home/antoyjchubekova/fun
Введите тип файла
sh
Количество файлов с расширением sh : 3
[antoyjchubekova@antoyjchubekova work]$

```

Рис. 4.13: Запуск программы

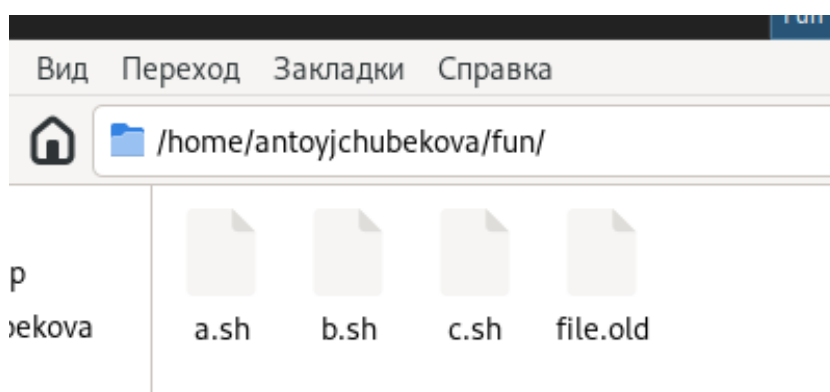


Рис. 4.14: каталог по адресу

Далее проверяю каталог по адресу /home/antoyjchubekova/ на количество файлов с txt и получаю ответ 2, перейдя по данному адресу мы видим, что программа сработала верно. рис. 4.15 и рис. 4.16).

```

[antoyjchubekova@antoyjchubekova work]$ bash task4.sh
Введите путь к каталогу
/home/antoyjchubekova
Введите тип файла
txt
Количество файлов с расширением txt : 2
[antoyjchubekova@antoyjchubekova work]$

```

Рис. 4.15: Запуск программы

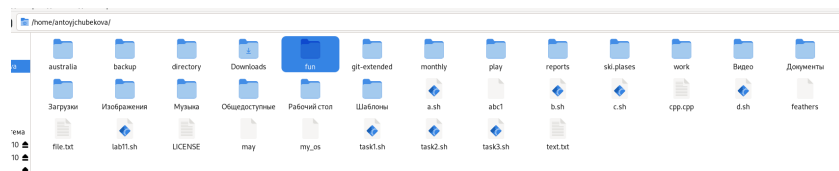


Рис. 4.16: каталог по адресу

5 Выводы

В ходе выполнения лабораторной работы №12 я изучила основы программирования в оболочке ОС LINUX. Также я научилась писать небольшие командные файлы.