

Лабораторная работа 10

Программирование в командном процессоре ОС UNIX.

Командные файлы

Korshunova Polina

2023, 10 April

RUDN University, Moscow, Russian Federation

Изучить основы программирования в оболочке ОС UNIX/Linux.
Научиться писать небольшие командные файлы.

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.

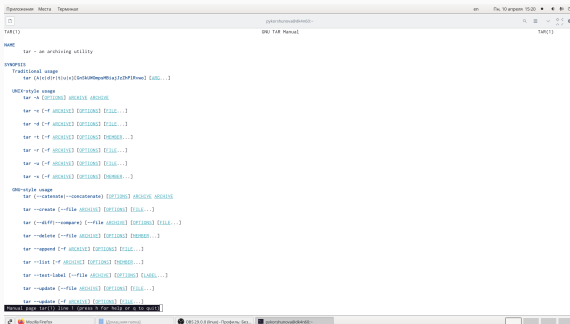
3. Написать командный файл - аналог команды `ls` (без использования самой этой команды и команды `dir`). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.
4. Написать командный файл, который получает в качестве аргумента командной 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.

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;
- С-оболочка (или csh) - надстройка над оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд;

- оболочка Корна (или ksh) - напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH - сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С 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.

1. Просматриваю справку tar. Создаю директорию backup.



The screenshot shows a terminal window with the title bar "Терминал" and a window icon. The terminal displays the output of the command `man tar`. The output is the manual page for `TAR(1)`, which describes the `tar` utility. The page is divided into sections: **NAME**, **SYNOPSIS**, and **GNU-style usage**. The **NAME** section states that `tar` is an archiving utility. The **SYNOPSIS** section lists traditional usage and GNU-style usage. The **GNU-style usage** section lists various options and their descriptions. The terminal window has a status bar at the bottom showing the system clock as "Сб, 10 апреля 19:20" and the window title "Терминал".

```
TAR(1)
NAME
tar - an archiving utility

SYNOPSIS
Traditional usage
tar [Xiddfilvz+]CcdkKOpqWt@iagIADPFHwml CUG...1

GNU-style usage
tar -h [OPTIONS] ARCHIVE ARCHIVE...1
tar -e [-f ARCHIVE] [OPTIONS] FILE...1
tar -d [-f ARCHIVE] [OPTIONS] FILE...1
tar -t [-f ARCHIVE] [OPTIONS] MEMBER...1
tar -r [-f ARCHIVE] [OPTIONS] FILE...1
tar -u [-f ARCHIVE] [OPTIONS] FILE...1
tar -x [-f ARCHIVE] [OPTIONS] MEMBER...1

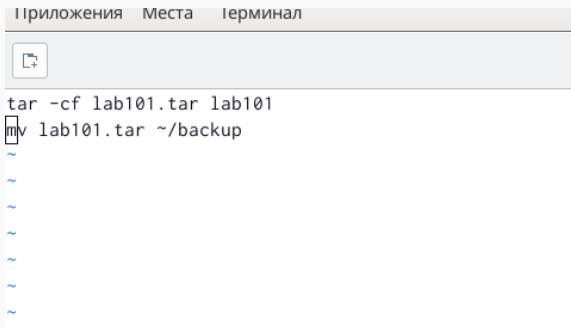
GNU-style usage
tar (--catenate|--concatenate) [OPTIONS] ARCHIVE ARCHIVE...1
tar --create [-f ARCHIVE] [OPTIONS] FILE...1
tar --diff [-f ARCHIVE] [OPTIONS] [FILE...1]
tar --delete [-f ARCHIVE] [OPTIONS] MEMBER...1
tar --append [-f ARCHIVE] [OPTIONS] FILE...1
tar --list [-f ARCHIVE] [OPTIONS] MEMBER...1
tar --test-label [-f ARCHIVE] [OPTIONS] LABEL...1
tar --update [-f ARCHIVE] [OPTIONS] FILE...1
tar --update [-f ARCHIVE] [OPTIONS] FILE...1

Manual page tar(1) line 1: create a file hold of a to disk
```

Рис. 1: Справка tar.

Выполнение работы

Пишу скрипт, который при запуске будет делать резервную копию самого себя в резервную директорию backup, созданную в домашнем каталоге заранее. При этом архивирую файл архиватором tar

A screenshot of a terminal window with a title bar containing 'Приложения', 'Места', and 'Терминал'. The terminal has a light gray background and a dark gray border. At the top left of the terminal area is a small icon of a document with a plus sign. The terminal displays two lines of text: 'tar -cf lab101.tar lab101' and 'mv lab101.tar ~/backup'. Below these lines are six blue tilde characters '~' stacked vertically. The cursor is positioned at the end of the second line of code.

```
tar -cf lab101.tar lab101
mv lab101.tar ~/backup
~
~
~
~
~
~
```

Рис. 2: Скрипт номер 1.

Добавляю право на исполнения файла, выполняю его и проверяю

Выполнение работы

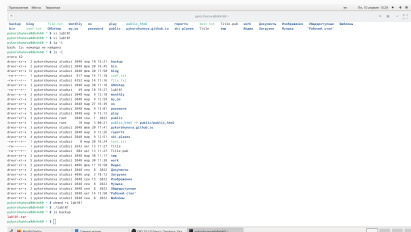
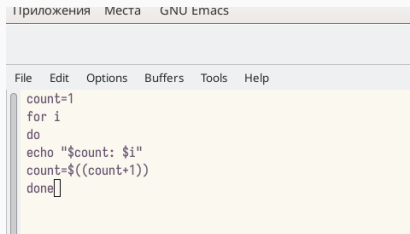


Рис. 3: Результат выполнения командного файла номер 1.

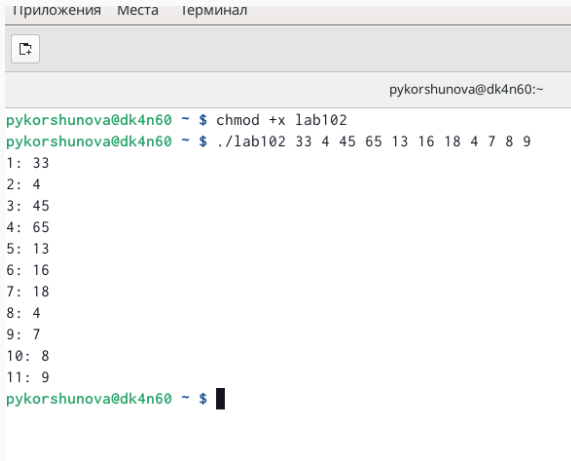
2. Пишу скрипт, обрабатывающий произвольное число аргументов командной строки. В моем случае скрипт последовательно выводит все значения переданных аргументов.

A screenshot of a GNU Emacs editor window. The title bar at the top shows 'Приложения', 'Места', and 'GNU Emacs'. Below the title bar is a menu bar with 'File', 'Edit', 'Options', 'Buffers', 'Tools', and 'Help'. The main editing area has a yellow background and contains the following shell script code:

```
count=1
for i
do
echo "$count: $i"
count=$((count+1))
done
```

Рис. 4: Скрипт номер 2.

Добавляю права на исполнение файла, выполняю его и проверяю корректность выполнения.

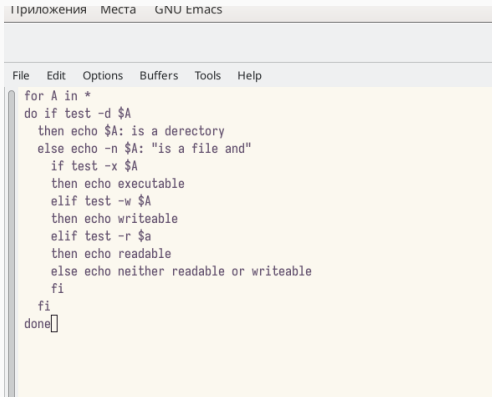


The screenshot shows a terminal window with a title bar containing 'Приложения', 'Места', and 'Терминал'. The terminal text is as follows:

```
pykorshunova@dk4n60:~  
pykorshunova@dk4n60 ~ $ chmod +x lab102  
pykorshunova@dk4n60 ~ $ ./lab102 33 4 45 65 13 16 18 4 7 8 9  
1: 33  
2: 4  
3: 45  
4: 65  
5: 13  
6: 16  
7: 18  
8: 4  
9: 7  
10: 8  
11: 9  
pykorshunova@dk4n60 ~ $
```

Рис. 5: Результат выполнения командного файла номер 2.

- Пишу командный файл, аналог команды `ls`, который выводит информацию о нужном каталоге: о правах доступа к файлам этого каталога. Скрипт определяет подкаталог или файл и выводит сообщение о правах доступа к файлам.



The image shows a screenshot of a GNU Emacs editor window. The title bar at the top reads "Приложения Места GNU Emacs". Below the title bar is a menu bar with the following items: "File", "Edit", "Options", "Buffers", "Tools", and "Help". The main editing area contains a shell script with the following code:

```
for A in *
do if test -d $A
  then echo $A: is a derectory
  else echo -n $A: "is a file and"
    if test -x $A
    then echo executable
    elif test -w $A
    then echo writeable
    elif test -r $a
    then echo readable
    else echo neither readable or writeable
    fi
  fi
done
```

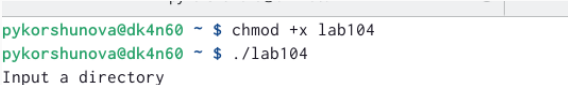
Рис. 6: Скрипт номер 3

4. Пишу скрипт, который считывает тип файлов (.txt, .doc, .jpg, .pdf и т.д.), а также путь к некоторой директории. И определяет количество файлов данного типа в заданной директории. Я указываю опцию -maxdepth 1, чтобы файлы необходимого типа искали только в заданном каталоге, а в его подкаталогах нет.

```
echo Input a directory
read dir
echo Input a file format
read format
find $dir -maxdepth 1 -name "$format" -type f | wc -l
```

Рис. 8: Скрипт номер 4.

Добавляю право на исполнения файла, выполняю его и проверяю корректность выполнения.

A screenshot of a terminal window with a light gray background. It shows a user named 'pykorshunova' at a host 'dk4n60' in a shell '~'. The first command is 'chmod +x lab104' and the second is './lab104'. The prompt 'Input a directory' is visible at the bottom.

```
pykorshunova@dk4n60 ~ $ chmod +x lab104
pykorshunova@dk4n60 ~ $ ./lab104
Input a directory
```

Рис. 9: Результат выполнения командного файла номер 4.

В ходе лабораторной работы я изучила основы программирования в оболочке ОС Linux, а также научилась писать небольшие командные файлы.