

Отчёт по лабораторной работе №10

Программирование в командном процессоре ОС UNIX. Командные файлы.

Грузинова Елизавета Константиновна; НКНбд-02-21

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Контрольные вопросы	12
6	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Создание файла pr1.sh	7
4.2	Программный код файла pr1.sh	7
4.3	Файл в действии	7
4.4	Созданный программой архив файла	8
4.5	Программный код файла pr2.sh	8
4.6	Скрипт в действии, в данном случае, он записывает необходимые аргументы и распечатывает их обратно	9
4.7	Программный код файла pr3.sh	9
4.8	Вывод информации о каждом файле или каталоге	10
4.9	Программный код файла pr4.sh	10
4.10	Вывод количества файлов в /home/gruzinova формата .txt	11

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux и 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;
- С-оболочка (или csh) — надстройка на 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]

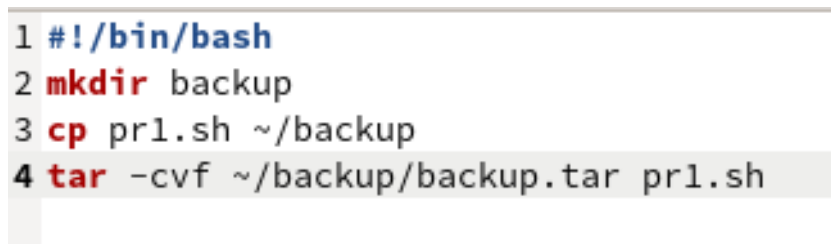
4 Выполнение лабораторной работы

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. (рис. 4.1, 4.2, 4.3, 4.4)



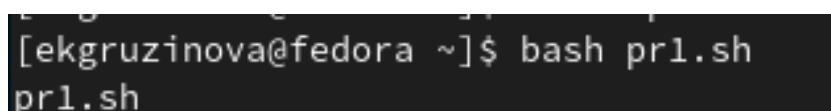
```
$ touch pr1.sh
```

Рис. 4.1: Создание файла pr1.sh



```
1 #!/bin/bash
2 mkdir backup
3 cp pr1.sh ~/backup
4 tar -cvf ~/backup/backup.tar pr1.sh
```

Рис. 4.2: Программный код файла pr1.sh



```
[ekgruzinova@fedora ~]$ bash pr1.sh
pr1.sh
```

Рис. 4.3: Файл в действии

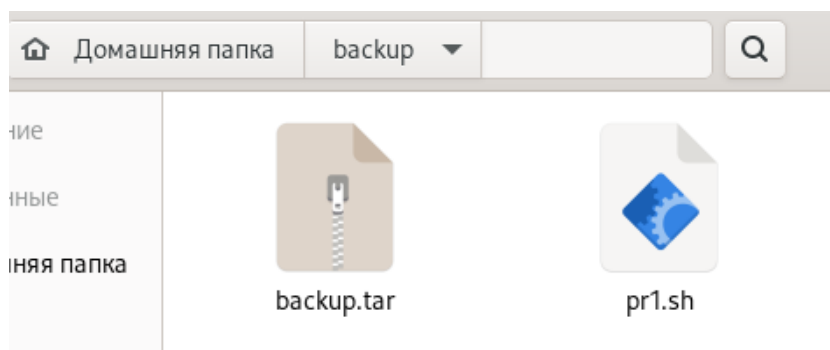


Рис. 4.4: Созданный программой архив файла

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. (рис. 4.5, 4.6)

```
#!/bin/bash
for A in $*
do echo $A
done
```

Рис. 4.5: Программный код файла pr2.sh

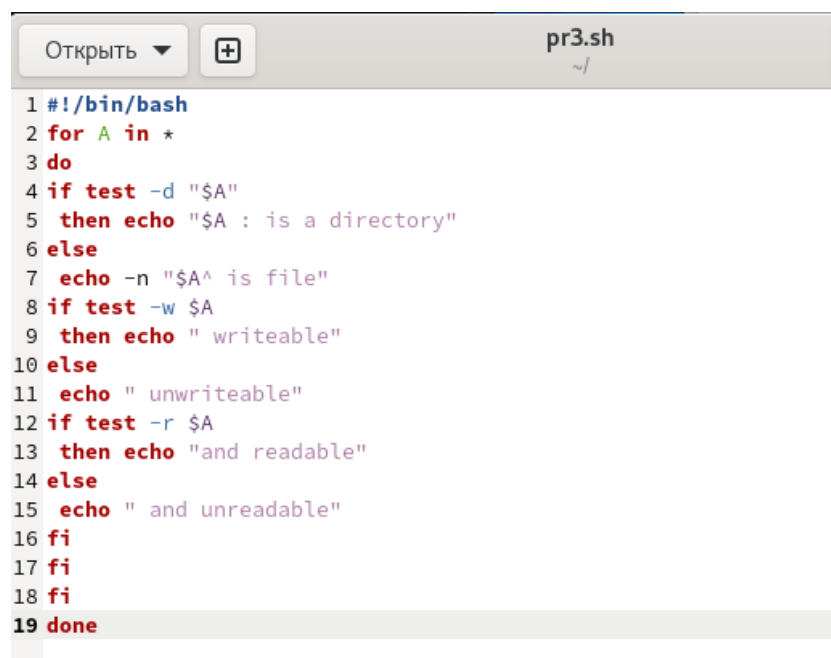

```

[ekgruzinova@fedora ~]$ bash pr2.sh 123 5543 3398 65 2 3
123
5543
3398
65
2
3

```

Рис. 4.6: Скрипт в действии, в данном случае, он записывает необходимые аргументы и распечатывает их обратно

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. (рис. 4.7, 4.8)



```

1 #!/bin/bash
2 for A in *
3 do
4 if test -d "$A"
5 then echo "$A : is a directory"
6 else
7 echo -n "$A^ is file"
8 if test -w $A
9 then echo " writeable"
10 else
11 echo " unwriteable"
12 if test -r $A
13 then echo "and readable"
14 else
15 echo " and unreadable"
16 fi
17 fi
18 fi
19 done

```

Рис. 4.7: Программный код файла `pr3.sh`

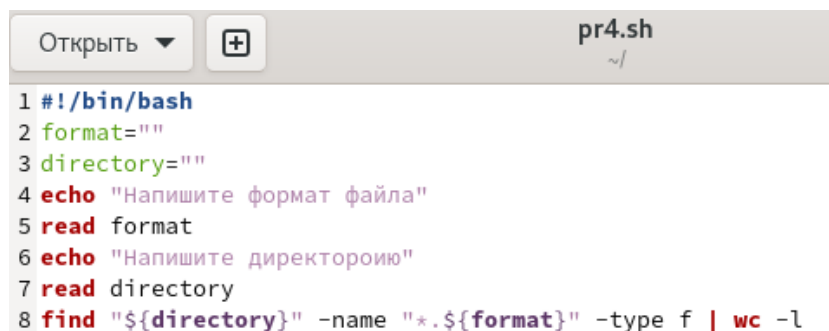
```

lab07.sh^ is file writeable
lab07.sh~^ is file writeable
lab07.zip^ is file writeable
Lab08.zip^ is file writeable
labo71^ is file writeable
monthly : is a directory
monthly1 : is a directory
my_os^ is file unwriteabl
and readable
play : is a directory
pr1.sh^ is file writeable
pr2.sh^ is file writeable
pr3.sh^ is file writeable

```

Рис. 4.8: Вывод информации о каждом файле или каталоге

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. (рис. 4.9, 4.10)



```

1 #!/bin/bash
2 format=""
3 directory=""
4 echo "Напишите формат файла"
5 read format
6 echo "Напишите директорию"
7 read directory
8 find "${directory}" -name "*.${format}" -type f | wc -l

```

Рис. 4.9: Программный код файла pr4.sh

```
ekguzinova@fedora ~]$ bash pr4.sh
Напишите формат файла
txt
Напишите директорию
/home/ekguzinova
11
australia      feathers      lab074      monthly1    README.md    Изображения
backup         file.txt     lab07.sh   my_os      reports      лаба7
bin            homet       lab07.sh~  play       ski.places   лаба8
```

Рис. 4.10: Вывод количества файлов в /home/guzinova формата .txt

5 Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

Командная оболочка - это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.

Примеры: MC, BASH, VI и т.д.

Они отличаются своей реализацией командных оболочек, т.е. некоторые из них построены на оболочке `bash`, `Борнаб Корна` и т.д.

2. Что такое POSIX?

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Разработан для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода.

3. Как определяются переменные и массивы в языке программирования `bash`?

Для определения переменной используют имя переменной, присваиваемое значение и знак “=” между ними.

Для создания массива используется команда `set` с флагом `-A`. За флагом следует имя переменной, а затем список значений, разделённых пробелами.

4. Каково назначение операторов `let` и `read`?

Команда `let` берет два операнда и присваивает их переменной.

Команда `read` присваивает значение, введенное пользователем, той переменной, указанной после команды.

5. Какие арифметические операции можно применять в языке программирования `bash`?

Сложение (+), вычитание (-), умножение (*), целочисленное деление (/) и целочисленный остаток от деления (%) и т.д.

6. Что означает операция (())?

Вычисление арифметических выражений.

7. Какие стандартные имена переменных Вам известны?

X, Y, Z, `trash`.

8. Что такое метасимволы?

Метасимволы - это символы вида `< > * ? | " &`, имеющие определенный смысл для командного процессора.

9. Как экранировать метасимволы?

Для этого необходимо снять специальный символ с метасимвола, то есть заключить в одинарные кавычки. Это не работает с символами `$, ', , "`.

10. Как создавать и запускать командные файлы?

Для создания файла используем `touch` имя файла. Для запуска пишем `bash командный_файл [аргументы]`.

11. Как определяются функции в языке программирования `bash`?

Функция - это группа команд.

12. Каким образом можно выяснить, является файл каталогом или обычным файлом?

С помощью команды `test -f` имя _файла.

13. Каково назначение команд `set`, `typeset` и `unset`?

Команда `typeset` задаёт и/или накладывает ограничения на переменные.

Команда `set` изменяет значения внутренних переменных сценария.

Команда `unset` удаляет переменную, фактически – устанавливает ее значение в `null`.

14. Как передаются параметры в командные файлы?

Часто сценарий написан так, что аргументы могут быть переданы в любом порядке с использованием флагов.

15. Назовите специальные переменные языка `bash` и их назначение

Переменные - `?` `#` `$` `!`. Они позволяют с помощью команды `echo` получить следующую информацию:

- – текущие флаги интерпретатора (установка флагов может быть изменена командой `set`);

`2 #` – число аргументов, которое было сохранено интерпретатором при выполнении какой-либо команды;

`?` – код возврата последней выполняемой команды;

6 Выводы

В процессе выполнения лабораторной работы изучила основы программирования в оболочке ОС UNIX/Linux и научилась писать небольшие командные файлы.

Список литературы

1. Лабораторная работа №10. Программирование в командном процессоре ОС UNIX. Командные файлы [Электронный ресурс]. URL: <https://esystem.ru/dn.ru/pluginfile.php/1383185>.