

РУДН. Операционные системы

Отчёт по лабораторной работе №9

Косинов Никита Андреевич, НПИМбв-02-20

Содержание

1	Цель работы	5
2	Ход работы	6
3	Написание простейшего скрипта	7
4	Написание командного файла, обрабатывающего аргументы	9
5	Написание командного файла - аналог команды <i>ls</i>	11
6	Написание командного файла, вычисляющего количество файлов по формату	13
7	Выводы	14

Список иллюстраций

3.1	Синхронизация с git	7
3.2	Создание файла	7
3.3	Запуск кода	7
3.4	Новые созданные документы и папки	8
3.5	Новые созданные документы и папки	8
3.6	Созданный архив	8
4.1	Создание документа	9
4.2	Результат выполнения командного файла	9
5.1	Код командного файла	11
5.2	Результат выполнения командного файла	12
5.3	Проверка результатов	12

Список таблиц

1 Цель работы

Основой пользования ЭВМ и его работы являются программы - блоки последовательно выполняемых простейших команд. Мы сталкивались ранее с написанием простейших команд и конвейеров в терминале. Но что делать, если нам нужно выполнить множество одинаковых, или зависящих от условия команд, или чтобы они выполнялись автоматически?

Оболочка ОС **Linux** позволяет базово программировать прямиком в терминале и даже сохранять блоки команд в текстовых, но исполняемых файлах. Цель данной работы - познакомиться с основами, предлагаемыми терминалом **Linux** для программирования.

2 Ход работы

Лабораторная работа выполнена в терминале **ОС Linux**, командная оболочка **bash** и хостинге хранения проектов **Github**. Действия по лабораторной работе представлены в следующем порядке:

1. написание простейшего скрипта;
2. написание командного файла, обрабатывающего аргументы;
3. написание командного файла - аналог команды *ls*;
4. написание командного файла, вычисляющего количество файлов по формату.

По завершении отчёта, вся рабочая папка отправляется на репозиторий на *github*.

3 Написание простейшего скрипта

Перед началом работы создадим новый рабочий каталог **lab09** и перейдём внутрь. Также не забываем синхронизироваться с нашим **Git**.

```
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro $ git pull
уже актуально.
```

Рис. 3.1: Синхронизация с git

Первое задание состоит в том, чтобы написать скрипт, копирующий архив себя в папку **backup**.

1. Создадим новый текстовый файл командой *touch*. Назовём его *saveself*. Откроем его с помощью *gedit* на запись.

```
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro $ touch saveself.txt
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro $ gedit saveself.txt
```

Рис. 3.2: Создание файла

2. Введём последовательность команд. Сначала скопируем файл *saveself* в нужную папку, затем скопированный файл заархивируем с помощью команды *tar*. Получим следующий код:

```
''' cp saveself backup tar -cf backup/saveself saveself '''
```

3. Сохраним файл, сделаем его исполняемым командой *chmod* и вызовем с помощью *bash*.

```
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ chmod +x saveself
nakosinov@dk8n59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ bash saveself
```

Рис. 3.3: Запуск кода

4. Получим требуемый результат.

```
nakosinov@dkan59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ ls
backup presentation report saveself
nakosinov@dkan59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ ls backup
saveself
nakosinov@dkan59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ cat backup/saveself
saveself000075500120540023420000000006414630356746013076 0ustar nakosinovstudscip saveself backup
tar -cf backup/saveself saveself
nakosinov@dkan59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $
```

Рис. 3.4: Новые созданные документы и папки

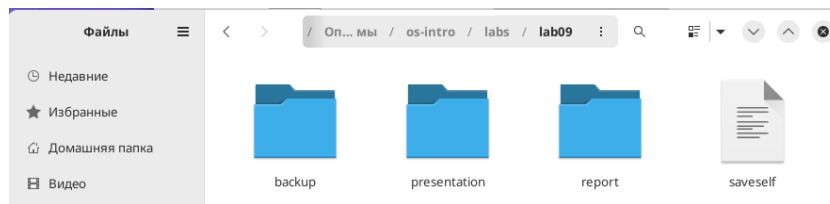


Рис. 3.5: Новые созданные документы и папки



Рис. 3.6: Созданный архив

4 Написание командного файла, обрабатывающего аргументы

Второе задание: написать командный файл, распечатывающий последовательно все введенные аргументы. Сложность задания в том, что аргументов может быть более 10

1. Создаём новый текстовый файл *replіc* и открываем его на редактирование.

```
nakosinov@dkn59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ touch replіc
nakosinov@dkn59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ gedit replіc
```

Рис. 4.1: Создание документа

2. Идея решения состоит в том, чтобы пройтись циклом *for* по всей строке аргументов, которая воспринимается циклом как массив. Напишем следующий код:

```
''' for s in $* do echo $s done '''
```

3. Запустим скрипт, введя много переменных и 0 переменных.

```
nakosinov@dkn59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ bash replіc 1 2 3 4 5 6 7 8
9 10 a b c
1
2
3
4
5
6
7
8
9
10
a
b
c
nakosinov@dkn59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ bash replіc
nakosinov@dkn59 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $
```

Рис. 4.2: Результат выполнения командного файла

4. Результат удовлетворительный.

5 Написание командного файла - аналог команды *ls*

Третье задание состоит в том, чтобы написать командный файл, выводящий список файлов и каталогов данного с указанием прав доступа.

1. Создаём файл *new_ls*.
2. Записываем в него следующий код. Идея состоит в том, чтобы сохранить список файлов, посчитать их количество, а далее вывести только названия и права доступа.

```
1 list=$(ls $*)
2 pr=$(ls -l $*)
3 n=${#list[*]}
4 echo 'Files and papkas:' $n
5 x=0
6 t=2
7 while
8     ((x<n))
9 do
10     echo $((x+1)) ':' ${pr[t]} ${list[$x]}
11     x=$((x+1))
12     t=$((t+9))
13 done
```

Рис. 5.1: Код командного файла

3. Проверяем полученный результат.

```

nakosinov@dk4n62 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ bash new_ls report
Files and pakpas: 8
1 : -rw-r--r-- CHANGELOG.md
2 : -rw-r--r-- LICENSE
3 : -rw-r--r-- package.json
4 : -rw-r--r-- README.git-flow.md
5 : -rw-r--r-- README.md
6 : -rw-r--r-- README.ru.md
7 : drwxr-xr-x report
8 : drwxr-xr-x scripts

```

Рис. 5.2: Результат выполнения командного файла

4. Сравниваем его со стандартной функцией `ls -l`

```

nakosinov@dk4n62 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab09 $ ls -l report
итого 35
-rw-r--r-- 1 nakosinov studsci 2457 map 19 20:39 CHANGELOG.md
-rw-r--r-- 1 nakosinov studsci 18657 map 19 20:39 LICENSE
-rw-r--r-- 1 nakosinov studsci 398 map 19 20:39 package.json
-rw-r--r-- 1 nakosinov studsci 5653 map 19 20:39 README.git-flow.md
-rw-r--r-- 1 nakosinov studsci 36 map 19 20:39 README.md
-rw-r--r-- 1 nakosinov studsci 71 map 19 20:39 README.ru.md
drwxr-xr-x 5 nakosinov studsci 2048 июн 7 19:17 report
drwxr-xr-x 2 nakosinov studsci 2048 map 19 20:39 scripts

```

Рис. 5.3: Проверка результатов

6 Написание командного файла, вычисляющего количество файлов по формату

Четвёртая задача заключается в написании файла, принимающему на вход 2 аргумента: путь к директории и формат файлов, количество которых нужно посчитать.

1. Создаём файл *format*

2. Записываем туда код

```
''' list=$(find $1 -name ".*$2" -print) echo listn ={list[*]} echo 'Files of format'  
.$2 '=' $n '''
```

7 Выводы

Для работы с операционной и файловой системой очень помогает оболочка командной строки *bash*. При этом, оболочка позволяет сохранять блоки команд в единый программный файл, что сильно упрощает работу с ним.