

Отчёт по лабораторной работе №10

**Программирование в командном процессоре ОС UNIX. Командные
файлы**

Киньябаева Аиша Иделевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	11
5	Выводы	14

Список иллюстраций

3.1	1 задание	7
3.2	Результат	7
3.3	Полученный архив	8
3.4	2 задание	8
3.5	Результат	8
3.6	3 задание	9
3.7	УРезультат	10
3.8	4 задание	10
3.9	Результат	10

Список таблиц

1 Цель работы

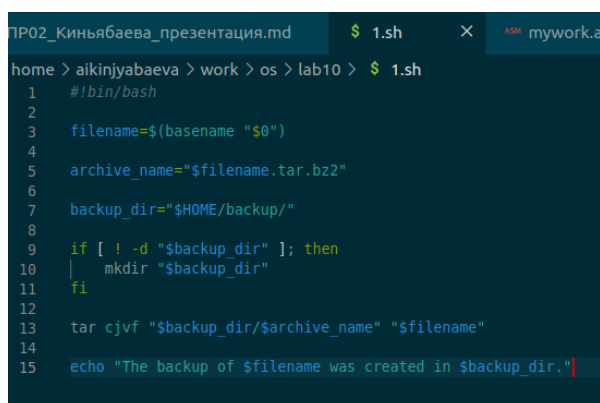
Целью данной работы является изучение основ программирования в оболочке ОС UNIX/Linux, научиться писать небольшие командные файлы.

2 Задание

Создание командных файлов

3 Выполнение лабораторной работы

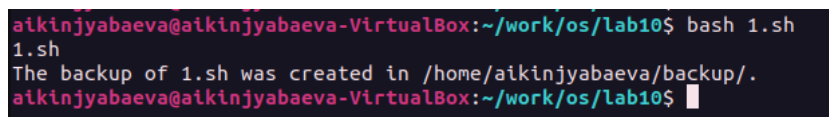
Создаю четыре скрипта для четырех заданий по лабораторной работе. И пишу первую программу, по которой скрипт должен копировать и архивировать себя в некоторый каталог (который при отсутствии создаем) (рис. 3.1)



```
1 #!/bin/bash
2
3 filename=$(basename "$0")
4
5 archive_name="$filename.tar.bz2"
6
7 backup_dir="$HOME/backup/"
8
9 if [ ! -d "$backup_dir" ]; then
10 |   mkdir "$backup_dir"
11 fi
12
13 tar cjvf "$backup_dir/$archive_name" "$filename"
14
15 echo "The backup of $filename was created in $backup_dir."
```

Рис. 3.1: 1 задание

Далее запускаю скрипт и смотрю на полученный результат (рис. 3.2), (рис. 3.3)



```
aikinjyabaeva@aikinjyabaeva-VirtualBox:~/work/os/lab10$ bash 1.sh
1.sh
The backup of 1.sh was created in /home/aikinjyabaeva/backup/.
aikinjyabaeva@aikinjyabaeva-VirtualBox:~/work/os/lab10$
```

Рис. 3.2: Результат

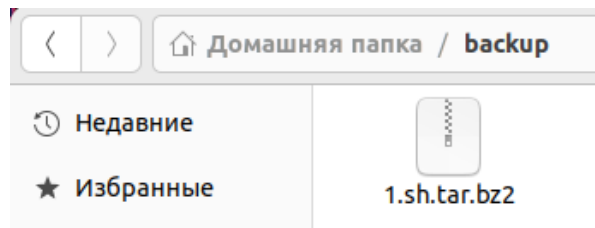


Рис. 3.3: Полученный архив

Содержимое скрипта второго задания, по которому нам необходимо произвести некоторое действие с введенными файлами (я их просто пересчитываю) (рис. 3.4), (рис. 3.5)

```

ПР02_Киньябаева_презентация.md  $ 1.sh

home > aikinjyabaeva > work > os > lab10 > $ 2.sh
1  #!/bin/bash
2
3  echo "Количество переданных аргументов: $#"
```

Рис. 3.4: 2 задание

```

aikinjyabaeva@aikinjyabaeva-VirtualBox:~/work/os/lab10$ bash 2.sh 1 2 3 4 5 6 6
6 6 6 6 6 6 6
Количество переданных аргументов: 14
Переданные аргументы:
1
2
3
4
5
6
6
6
6
6
6
6
6
6
aikinjyabaeva@aikinjyabaeva-VirtualBox:~/work/os/lab10$
```

Рис. 3.5: Результат

Следующая команда, заменяющая команду ls и результат ее работы(рис. 3.6),
(рис. 3.7)

```
ПР02_Киньябаева_презентация.md    $ 1.sh    $ 2.sh    $ 3.sh
home > aikinjabaeva > work > os > lab10 > $ 3.sh
1  #!/bin/bash
2
3  if [ $# -ne 1 ]; then
4      echo "Используйте команду: bash $0 <название директории>" >&2
5      exit 1
6  fi
7
8  directory="$1"
9
10 if [ ! -d "$directory" ]; then
11     echo "$directory is not a directory" >&2
12     exit 1
13 fi
14
15 for file in "$directory"/*; do
16     if [ -f "$file" ]; then
17         permissions=""
18
19         if [ -r "$file" ]; then
20             permissions="${permissions}--r-"
21         else
22             permissions="${permissions}-"
23         fi
24
25         if [ -w "$file" ]; then
26             permissions="${permissions}--w-"
27         else
28             permissions="${permissions}-"
29         fi
30
31         if [ -x "$file" ]; then
32             permissions="${permissions}--x-"
33         else
34             permissions="${permissions}-"
35         fi
36
37         echo -e "${permissions}\t$(basename "$file")"
38     elif [ -d "$file" ]; then
39         echo -e "d-----\t$(basename "$file")"
40     fi
41 done
```

Рис. 3.6: 3 задание

```

aikinjabaeva@aikinjabaeva-VirtualBox:~/work/os/lab10$ bash 3.sh ~/work/site/
--r---w-- academic.Rproj
d----- assets
d----- config
d----- content
d----- data
--r---w-- go.mod
--r---w-- go.sum
d----- images
--r---w-- LICENSE.md
--r---w-- netlify.toml
--r---w-- preview.png
d----- public
--r---w-- README.md
d----- resources
d----- static
--r---w-- theme.toml
aikinjabaeva@aikinjabaeva-VirtualBox:~/work/os/lab10$

```

Рис. 3.7: УРезультат

И последнее задание, по которому вычисляем количество файлов определенного типа в каталоге(рис. 3.8), (рис. fig:009)

```

home > aikinjabaeva > work > os > lab10 > $ 4.sh
1  #!/bin/bash
2
3  if [ $# -ne 2 ]; then
4      echo "Используйте команду: bash $0 <Путь к директории> <Расширение файла без точки>"
5      exit 1
6  fi
7
8  directory_path=$1
9  file_extension=$2
10
11 if [ ! -d "$directory_path" ]; then
12     echo "Directory doesn't exist: $directory_path"
13     exit 1
14 fi
15
16 count=$(find "$directory_path" -type f -name ".*$file_extension" | wc -l)
17
18 echo "Number of .$file_extension files in $directory_path: $count"

```

Рис. 3.8: 4 задание

```

aikinjabaeva@aikinjabaeva-VirtualBox:~/work/os/lab10$ bash 4.sh ~/work/site/ m
d
Number of .md files in /home/aikinjabaeva/work/site/: 23
aikinjabaeva@aikinjabaeva-VirtualBox:~/work/os/lab10$

```

Рис. 3.9: Результат

Ну и в завершение работы выкладываю все на гит

4 Контрольные вопросы

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

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).

2. Что такое POSIX? POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ
3. Как определяются переменные и массивы в языке программирования bash?
К любому выбранному имени переменной через знак равно присваивается значение. Далее можно вновь обратиться к переменной через символы \${имя переменной} 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`? Оболочка `bash` поддерживает встроенные арифметические функции. Команда `let` является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Команда `read` позволяет читать значения переменных со стандартного ввода
5. Какие арифметические операции можно применять в языке программирования `bash`? Классические арифметические операции: сложение, вычитание, умножение, деление. А также множество других операций по типу: остаток от деления, побитное перемещение, отрицание и сравнение.
6. Что означает операция `(())`? Используется для облегчения программирования и обозначается как условие
7. Какие стандартные имена переменных Вам известны? • `HOME` — имя домашнего каталога пользователя. Если команда `cd` вводится без аргументов, то происходит переход в каталог, указанный в этой переменной. • `IFS` — последовательность символов, являющихся разделителями в командной строке, например, пробел, табуляция и перевод строки (`new line`). • `MAIL` — командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение `You have mail` (у Вас есть почта). • `TERM` — тип используемого терминала. • `LOGNAME` — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему. • В командном процессоре `Си` имеется ещё несколько стандартных переменных. Значение всех переменных можно просмотреть с помощью команды `set`.
8. Что такое метасимволы? Это различные символы (`| < > * » &`), которые

имеют особый смысл для командного процессора.

9. Как экранировать метасимволы? Экранирование может быть осуществлено с помощью предшествующего метасимволу символа , который, в свою очередь, является метасимволом.
10. Как создавать и запускать командные файлы? С помощью команды `bash` или же после изменения разрешений просто запускать
11. Как определяются функции в языке программирования `bash`? Группу команд можно объединить в функцию. Для этого существует ключевое слово `function`, после которого следует имя функции и список команд, заключённых в фигурные скобки. Удалить функцию можно с помощью команды `unset` с флагом `-f`
12. Каким образом можно выяснить, является файл каталогом или обычным файлом? `test -d file` — истина, если файл `file` является каталогом.
13. Каково назначение команд `set`, `typeset` и `unset`? Для создания массива используется команда `set` с флагом `-A`. Значение всех переменных можно просмотреть с помощью команды `set`. Если использовать `typeset -i` для объявления и присвоения переменной, то при последующем её применении она станет целой. Изъять переменную из программы можно с помощью команды `unset`. Удалить функцию можно с помощью команды `unset` с флагом `-f`.
14. Как передаются параметры в командные файлы? При вызове командного файла на выполнение параметры ему могут быть переданы точно таким же образом, как и выполняемой программе. С точки зрения командного файла эти параметры являются позиционными. Символ `$` является метасимволом командного процессора. Он используется, в частности, для ссылки на параметры, точнее, для получения их значений в командном файле.

5 Выводы

В ходе данной лабораторной работы были изучены основы программирования и созданы командные файлы.