

# Лабораторная работа № 11.

---

Кальсин З. А.

<sup>1</sup>RUDN University, Moscow, Russian Federation

## Цель работы

---

# Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux, научиться писать небольшие командные файлы. # Выполнение лабораторной работы

Ход работы:

1. Написал скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации узнал, изучив справку.

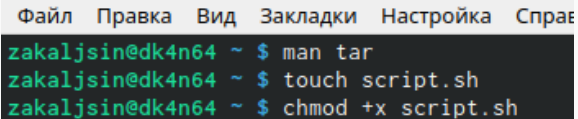
```
#!/bin/bash
# tar - an archiving utility

# optional usage
tar <[action]>[compression]<[format]> [file]...

# tar usage
tar -x -f backup.tar.gz file...
tar -x -f backup.tar.gz file...
tar -x -f backup.tar.gz file...
tar -x -f backup.tar.gz file...
tar -x -f backup.tar.gz file...
tar -x -f backup.tar.gz file...

# bzip2 usage
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...

# zip usage
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...
tar -czvf backup.tar.gz file...
```



Файл Правка Вид Закладки Настройка Справа

```
zakaljsin@dk4n64 ~ $ man tar
zakaljsin@dk4n64 ~ $ touch script.sh
zakaljsin@dk4n64 ~ $ chmod +x script.sh
```

```
#!/bin/bash  
backupname="ScriptBack.sh"  
cp "$0" "$backup_name"  
tar -cf laba.tar $backup_name
```

misk



script.sh

monthly



script2.sh

2. Написал пример командного файла, обрабатывающего любое произвольное

число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных

```
#!/bin/bash  
echo "Vvedite znachenie"  
head -1
```

```
zakaljsin@dk4n64 ~ $ ./script2.sh  
Vvedite znachenie  
9 8 7 6 5  
9 8 7 6 5  
zakaljsin@dk4n64 ~ $
```



3. Написал 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

```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
for A in *
do if test -d $A
then echo $A: is a directory
else echo -n $A: is a file and
    if test -w $A
    then echo writeable
    elif test -r $A
    then echo readable
    else echo neither readable nor writeable
    fi
fi
done
```

```
zakaljsin@dk4n64 ~ $ ./script3.sh
abc1: is a file andwriteable
abcl: is a directory
australia: is a directory
conf.txt: is a file andwriteable
etc: is a directory
feathers: is a file andwriteable
file.txt: is a file andwriteable
GNUstep: is a directory
hello: is a file andwriteable
hello.asm: is a file andwriteable
lab02.asm: is a file andwriteable
lab03a: is a directory
lab03b: is a directory
lab05: is a directory
lab06: is a directory
lab06.o: is a file andwriteable
lab07: is a directory
lab10.sh: is a file andwriteable
lab10.sh~: is a file andwriteable
Lab0S: is a directory
letters: is a directory
logfile: is a file andwriteable
may: is a file andwriteable
memos: is a directory
misk: is a directory
monthly: is a directory
mv os: is a directory
```

4. Написал 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.

```
le Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
format=""
direct=""
echo "point out format"
read format
echo "point out direct"
read direct
find "$direct" -name ".$format" -type f | wc -l
ls
```

```

zakaljsin@dk4n64 ~ $ ./script4.sh
point out format
.sh
point out direct
/home
      0      0      0 -
wc: 1: Нет такого файла или каталога
      0      0      0 итого
abc1      conf.txt  file.txt  hello.asm  lab03b
abc1      etc      GNUstep  lab02.asm  lab05
australia feathers  hello    lab03a    lab06
zakaljsin@dk4n64 ~ $

```

# Вывод 10

изучил основы программирования в оболочке ОС UNIX/Linux, научился писать небольшие командные файлы. # Ответы на контрольн## Слайд 1

собой мощный экраннй редактор текста, написанный на

1. Командный процессор (командная оболочка, интерпретатор команд shell) —

это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа

### 2. 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/Linuxподобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

3. Командный процессор `bash` обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем.

Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Например, команда `mark=/usr/andy/bin` присваивает значение строки символов `/usr/andy/bin` переменной `mark` типа строка символов.

Значение, присвоенное некоторой переменной, может быть впоследствии использовано. Для этого в соответствующем месте командной строки должно быть

употреблено имя этой переменной, которому предшествует метасимвол `$`. Например, команда `mv afile ${mark}` переместит файл `afile` из текущего каталога в каталог с абсолютным полным именем `/usr/andy/bin`.

Индексация массивов начинается с нулевого элемента.

4, 5, 6. Команда `let` является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение — это единичный терм (term), обычно целочисленный. Команда `let` берет два операнда и присваивает их переменной. Положительным моментом команды `let` можно считать то, что для идентификации переменной ей не нужен знак доллара; вы можете писать команды типа `let sum=x+7`, и `let` будет искать переменную `x` и добавлять к ней 7.

Команда `let` также расширяет другие выражения `let`, если они заключены в двойные круглые скобки. Таким способом вы можете создавать довольно сложные выражения.

Команда `let` не ограничена простыми арифметическими выражениями.

7. – 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



10. Последовательность команд может быть помещена в текстовый файл. Такой

файл называется командным. Далее этот файл можно выполнить по команде:

```
bash командный_файл [аргументы]
```

Чтобы не вводить каждый раз последовательности символов `bash`, необходимо

изменить код защиты этого командного файла, обеспечив доступ к этому файлу по

выполнению. Это может быть сделано с помощью команды

```
chmod +x имя_файла
```

Теперь можно вызывать свой командный файл на выполнение. просто

12. `ls -lrt` Если есть `d`, то является файл каталогом

13. Для создания массива используется команда `set` с флагом `-A`. За флагом следует

имя переменной, а затем список значений, разделённых пробелами.

Удалить функцию можно с помощью команды `unset` с флагом `-f`.

Команда `typeset` имеет четыре опции для работы с функциями:

- `-f` — перечисляет определённые на текущий момент функции;
- `-ft` — при последующем вызове функции иницирует её трассировку;
- `-fx` — экспортирует все перечисленные функции в любые дочерние программы

оболочек;

- 15. – \$\* — отображается вся командная строка или параметры оболочки;
- \$? — код завершения последней выполненной команды;
- \$\$ — уникальный идентификатор процесса, в рамках которого выполняется командный процессор;
- \$! — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда;
- \$- — значение флагов командного процессора;
- \${#} — возвращает целое число — количество слов, которые были результатом \$;
- \${#name} — возвращает целое значение длины строки в переменной