

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11**

*дисциплина: Операционные системы*

Студент: Брамхачарья Хасана

Группа: НПИбд-01-20

**МОСКВА**

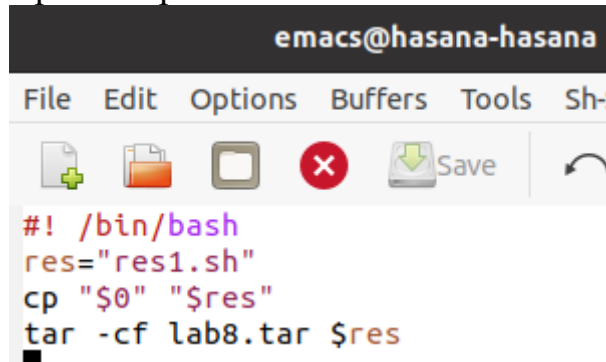
2021 г.

## Цель:

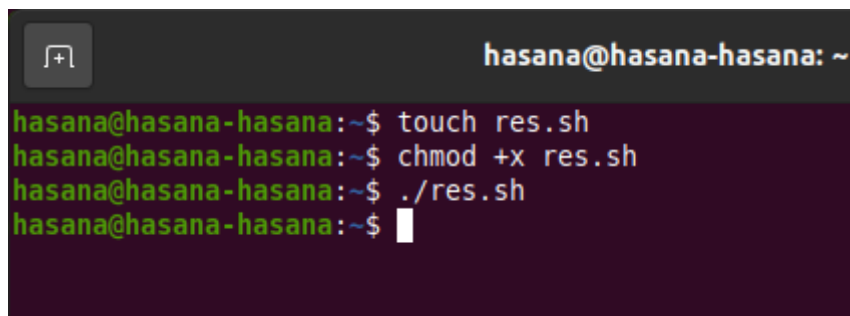
Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

## Ход работы:

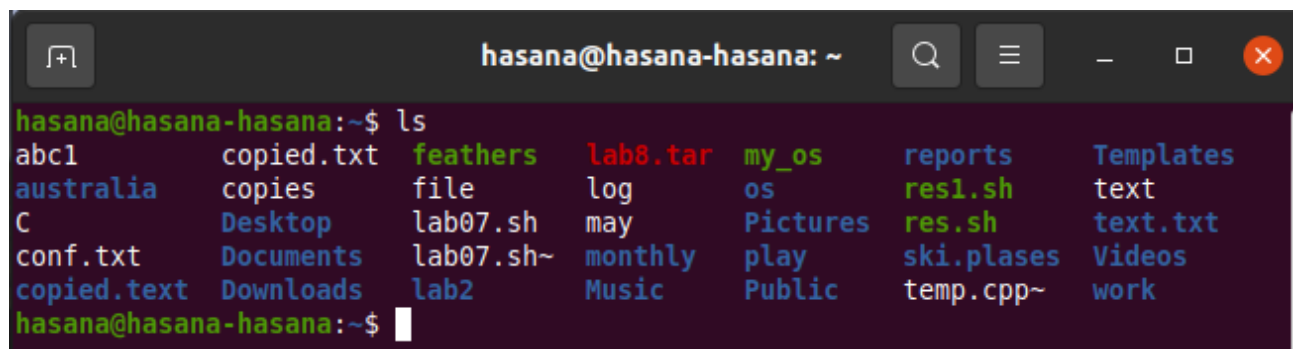
1. Напишем скрипт, который при запуске будет делать резервную копию самого себя в другую директорию backup в вашем домашнем каталоге. При этом файл будет архивироваться архиватором tar



```
#!/bin/bash
res="res1.sh"
cp "$0" "$res"
tar -cf lab8.tar $res
```



```
hasana@hasana-hasana:~$ touch res.sh
hasana@hasana-hasana:~$ chmod +x res.sh
hasana@hasana-hasana:~$ ./res.sh
hasana@hasana-hasana:~$
```



```
hasana@hasana-hasana:~$ ls
abcl      copied.txt  feathers   lab8.tar   my_os      reports    Templates
australia copies      file       log        os          res1.sh    text
C         Desktop    lab07.sh   may        Pictures    res.sh     text.txt
conf.txt  Documents  lab07.sh~  monthly    play        ski.plases Videos
copied.txt Downloads  lab2       Music      Public      temp.cpp~  work
hasana@hasana-hasana:~$
```

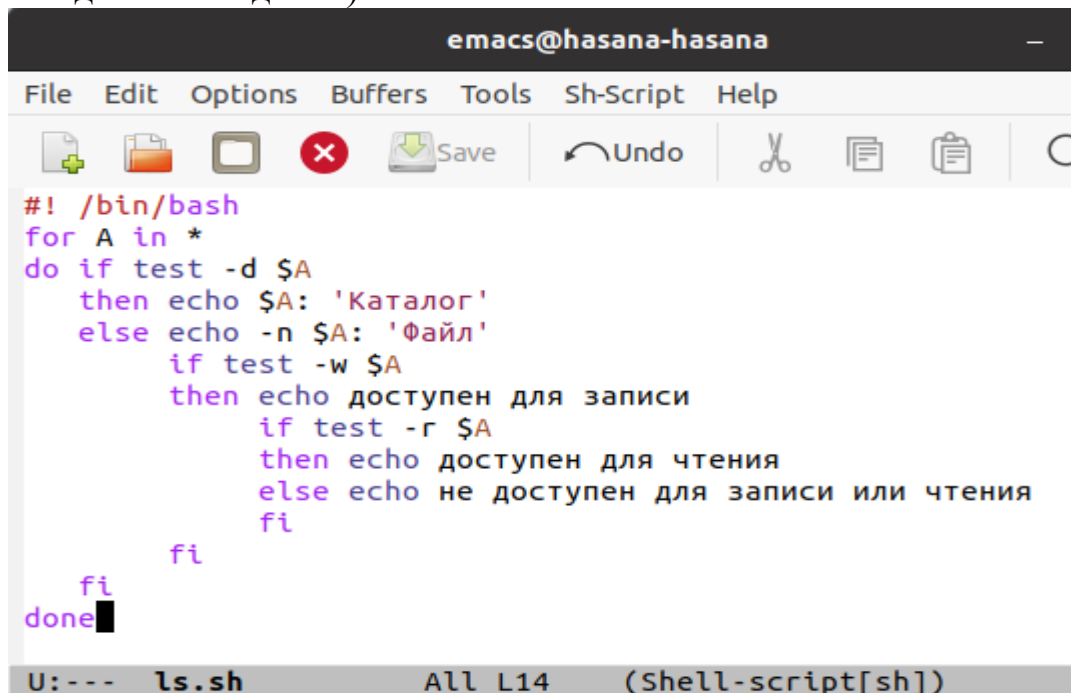
2. Напишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять.

```
emacs@hasana-hasana
File Edit Options Buffers Tools Sh-Script
+ X Save Und
#!/bin/bash
for A in *
do
echo "Аргументы мне запили"
head -1
done
U: --- cm.sh All L6 (St
```

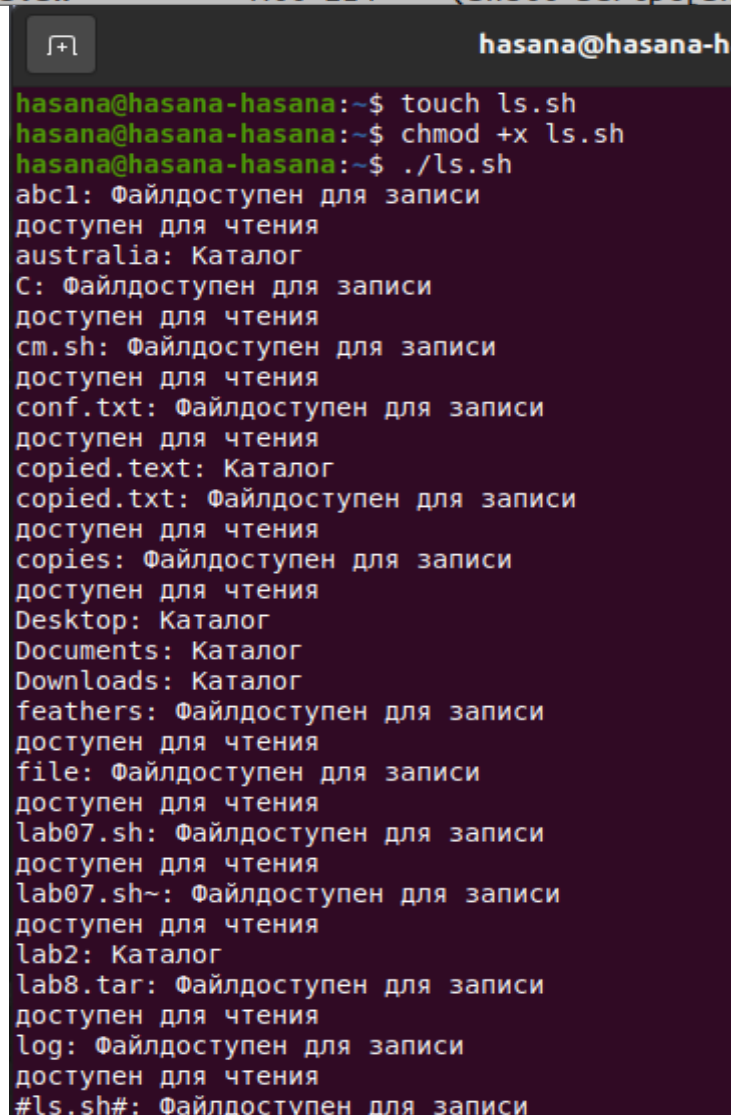
```
hasana@hasana-hasana
hasana@hasana-hasana:~$ touch cm.sh
hasana@hasana-hasana:~$ chmod +x cm.sh
hasana@hasana-hasana:~$ ./cm.sh
```

```
hasana@hasana-hasana: ~
hasana@hasana-hasana:~$ ./cm.sh
Аргументы мне запили
1
1
Аргументы мне запили
2
2
Аргументы мне запили
3
3
Аргументы мне запили
4
4
Аргументы мне запили
5
5
Аргументы мне запили
6
6
Аргументы мне запили
7
7
Аргументы мне запили
8
8
Аргументы мне запили
9
9
Аргументы мне запили
10
10
Аргументы мне запили
```

3. Напишем командный файл — аналог команды ls (без использования самой этой команды и команды dir).



```
emacs@hasana-hasana
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/bin/bash
for A in *
do if test -d $A
  then echo $A: 'Каталог'
  else echo -n $A: 'Файл'
    if test -w $A
    then echo доступен для записи
      if test -r $A
      then echo доступен для чтения
      else echo не доступен для записи или чтения
      fi
    fi
  fi
done
U: --- ls.sh All L14 (Shell-script[sh])
```



```
hasana@hasana-h
hasana@hasana-hasana:~$ touch ls.sh
hasana@hasana-hasana:~$ chmod +x ls.sh
hasana@hasana-hasana:~$ ./ls.sh
abcl: Файлдоступен для записи
доступен для чтения
australia: Каталог
C: Файлдоступен для записи
доступен для чтения
cm.sh: Файлдоступен для записи
доступен для чтения
conf.txt: Файлдоступен для записи
доступен для чтения
copied.txt: Каталог
copied.txt: Файлдоступен для записи
доступен для чтения
copies: Файлдоступен для записи
доступен для чтения
Desktop: Каталог
Documents: Каталог
Downloads: Каталог
feathers: Файлдоступен для записи
доступен для чтения
file: Файлдоступен для записи
доступен для чтения
lab07.sh: Файлдоступен для записи
доступен для чтения
lab07.sh~: Файлдоступен для записи
доступен для чтения
lab2: Каталог
lab8.tar: Файлдоступен для записи
доступен для чтения
log: Файлдоступен для записи
доступен для чтения
#ls.sh#: Файлдоступен для записи
```

```
hasana@hasana-hasana
lab2: Каталог
lab8.tar: Файлдоступен для записи
доступен для чтения
log: Файлдоступен для записи
доступен для чтения
#ls.sh#: Файлдоступен для записи
доступен для чтения
ls.sh: Файлдоступен для записи
доступен для чтения
may: Файлдоступен для записи
доступен для чтения
monthly: Каталог
Music: Каталог
my_os: Файлдоступен для записи
доступен для чтения
os: Каталог
Pictures: Каталог
play: Каталог
Public: Каталог
reports: Каталог
res1.sh: Файлдоступен для записи
доступен для чтения
res.sh: Файлдоступен для записи
доступен для чтения
ski.places: Каталог
temp.cpp~: Файлдоступен для записи
доступен для чтения
Templates: Каталог
text: Файлдоступен для записи
доступен для чтения
text.txt: Каталог
Videos: Каталог
work: Каталог
hasana@hasana-hasana:~$
```

4. Напишем командный файл, который получает в качестве аргумента командной строки формат файла и вычисляет количество таких файлов в указанной директории.

```
emacs@hasana-hasana
File Edit Options Buffers Tools Sh-Script Help
[Icons] Save Undo [Icons]
#!/bin/bash
format=""
direct=""
echo "Формат"
read format;
echo "Директория"
read direct;
find "$direct" -name "*.$format" -type f | wc -l
ls
U: --- arg.sh All L9 (Shell-script[sh])
```

```
hasana@hasana-hasana: ~  
hasana@hasana-hasana:~$ touch arg.sh  
hasana@hasana-hasana:~$ chmod +x arg.sh  
hasana@hasana-hasana:~$ ./arg.sh  
Формат  
sh  
Директория  
/home/hasana  
0  
abc1      copied.text  file        ls.sh       play        Templates  
'#arg.sh#' copied.txt   lab07.sh    may         Public      text  
arg.sh    copies      lab07.sh~   monthly    reports     text.txt  
australia Desktop     lab2        Music       res1.sh     Videos  
C         Documents  lab8.tar    my_os       res.sh      work  
cm.sh     Downloads  log         os          ski.plases  
conf.txt  feathers   '#ls.sh#'  Pictures    temp.cpp~  
hasana@hasana-hasana:~$
```

## ВЫВОД:

В ходе работы я изучила основы программирования в оболочке ОС UNIX/Linux. Научилась писать небольшие командные файлы.

## Контрольные вопросы:

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

Программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.

Оболочка Борна - стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций.

С-оболочка - надстройка над оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд.

Оболочка Корна - напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна.

BASH - сокращение от Bourne Again Shell, в основе своей совмещает свойства оболочек С и Корна.

2. Что такое POSIX?

Набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

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

Переменная/=значение.

set -А переменная список значений.

4. Каково назначение операторов let и read?

let - берет два операнда и присваивает их переменной.

read - чтение значения переменных со стандартного ввода.

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

Операции логики, умножение, деление, сложение, вычитание.

6. Что означает операция (( ))?

Условия оболочки bash.

7. Какие стандартные имена переменных Вам известны?

PATH, IFS, MAIL, TERM, LOGNAME.

8. Что такое метасимволы?

Символы ' < > \* ? | \ " &, являются метасимволами и имеют для командного процессора отличный от обычных символом смысл.

9. Как экранировать метасимволы?

Экранирование может быть осуществлено с помощью предшествующего метасимволу символа \, который, в свою очередь, является метасимволом. Для экранирования группы метасимволов нужно заключить её в одинарные кавычки. Строка, заключённая в двойные кавычки, экранирует все метасимволы, кроме \$, ', \, ".

10. Как создавать и запускать командные файлы?

```
bash <командный_файл> [аргументы]
```

```
chmod +x <командный_файл>
```

```
./командный_файл
```

11. Как определяются функции в языке программирования bash?

Ключевое слово function <fun\_name> {тело функции}.

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?

Оболочка bash позволяет работать с массивами. Для создания массива используется команда set с флагом -A

typeset является встроенной инструкцией и предназначена для наложения ограничений на переменные

С помощью команды unset можно изъять переменную из программы

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. В командный файл можно передать до девяти параметров. При использовании где-либо в командном файле комбинации символов \$i, где  $0 < i < 10$ , вместо неё будет осуществлена подстановка значения параметра с порядковым номером i, т.е. аргумента командного файла с порядковым номером i. Использование комбинации символов \$0 приводит к подстановке вместо неё имени данного командного файла.

15. Назовите специальные переменные языка bash и их назначение.

- \$\* — 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 name;
- \${name[n]} — обращение к n-му элементу массива;
- \${name[\*]} — перечисляет все элементы массива, разделённые пробелом;
- \${name[@]} — то же самое, но позволяет учитывать символы пробелы в самих переменных;
- \${name:-value} — если значение переменной name не определено, то оно будет заменено на указанное value;
- \${name:value} — проверяется факт существования переменной;
- \${name=value} — если name не определено, то ему присваивается значение value;
- \${name?value} — останавливает выполнение, если имя переменной не определено, и выводит value как сообщение об ошибке;
- \${name+value} — это выражение работает противоположно \${name-value}. Если переменная определена, то подставляется value;
- \${name#pattern} — представляет значение переменной name с удалённым самым коротким левым образцом (pattern);
- \${#name[\*]} и \${#name[@]} — эти выражения возвращают количество элементов в массиве name.