

Лабораторная работа №11

Журавлев Георгий Иванович

Цель работы

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

Ход работы

1. Написал скрипт, который при запуске будет делать резервную копию самого себя в другую директорию backup в домашнем каталоге. При этом файл архивируется с помощью tar.

1.1. Прочитал мануал.

```

TAR(1)                                GNU TAR Manual

NAME
    tar - an archiving utility

SYNOPSIS
    Traditional usage
        tar {A|c|d|r|t|u|x}[GnSkUW0mpsMBiajJzZhPlRvwo] [ARG...]

    UNIX-style usage
        tar -A [OPTIONS] ARCHIVE ARCHIVE

        tar -c [-f ARCHIVE] [OPTIONS] [FILE...]
        tar -d [-f ARCHIVE] [OPTIONS] [FILE...]
        tar -t [-f ARCHIVE] [OPTIONS] [MEMBER...]
        tar -r [-f ARCHIVE] [OPTIONS] [FILE...]
        tar -u [-f ARCHIVE] [OPTIONS] [FILE...]
        tar -x [-f ARCHIVE] [OPTIONS] [MEMBER...]

    GNU-style usage
        tar {--catenate|--concatenate} [OPTIONS] ARCHIVE ARCHIVE

        tar --create [--file ARCHIVE] [OPTIONS] [FILE...]

        tar {--diff|--compare} [--file ARCHIVE] [OPTIONS] [FILE...]

        tar --delete [--file ARCHIVE] [OPTIONS] [MEMBER...]

        tar --append [-f ARCHIVE] [OPTIONS] [FILE...]

        tar --list [-f ARCHIVE] [OPTIONS] [MEMBER...]

        tar --test-label [--file ARCHIVE] [OPTIONS] [LABEL...]

        tar --update [--file ARCHIVE] [OPTIONS] [FILE...]

        tar --update [-f ARCHIVE] [OPTIONS] [FILE...]

        tar {--extract|--get} [-f ARCHIVE] [OPTIONS] [MEMBER...]

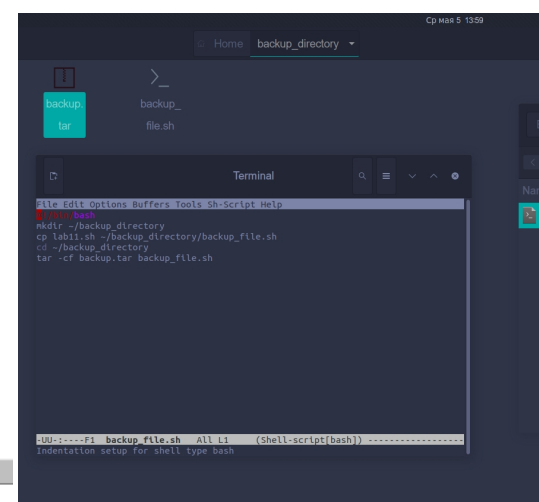
```

1.2. Написал скрипт и проверил его работу.

```
File Edit Options Buffers Tools Sh-Script Help
[Icons: New File, Open File, Save, Close, Undo, Redo, Copy, Paste, Find]

#!/bin/bash
mkdir ~/backup_directory
cp lab11.sh ~/backup_directory/backup_file.sh
cd ~/backup_directory
tar -cf backup.tar backup_file.sh
```

U: --- lab11.sh All L5 (Shell-script[sh])
Wrote /home/g zhuravlev/lab11.sh



```

g_zhuravlev@g ~ ~ chmod +x lab11.sh 815 13:56:02
g_zhuravlev@g ~ ~ ./lab11.sh 816 13:56:23
g_zhuravlev@g ~ ~ cd backup_directory/ 817 13:56:33
g_zhuravlev@g ~ /backup_directory ~ tree 818 13:56:57

  backup_file.sh
  backup.tar

directories, 2 files
g_zhuravlev@g ~ /backup_directory ~ cat backup.tar 819 13:57:01
backup_file.sh0000775000175000017500000000021414044474741015343 0ustar  g_zhuravlevg_zhuravlev#!/bin/
bash
mkdir ~/backup_directory
cp lab11.sh ~/backup_directory/backup_file.sh
cd ~/backup_directory
tar -cf backup.tar backup_file.sh
g_zhuravlev@g ~ /backup_directory ~ less backup.tar 820 13:57:43
backup.tar" may be a binary file. See it anyway?
g_zhuravlev@g ~ /backup_directory ~ gedit backup.tar & 821 13:58:15
1] 297266
g_zhuravlev@g ~ /backup_directory ~ 822 13:58:28
1] + 297266 done gedit backup.tar
g_zhuravlev@g ~ /backup_directory ~ cd 822 13:59:13
g_zhuravlev@g ~ ~ clear 823 14:02:41

```

2. Написал пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. В качестве примера сделал вывод строки.

```

Terminal
File Edit Options Buffers Tools Sh-Script Help
~/bin/bash

head -1

-uu-:----F1 vvodl.sh All L1 (Shell-script[bash]) -----

g_zhuravlev@g ~ ~ chmod +x vvodl.sh 824 14:02:57
g_zhuravlev@g ~ ~ ./vvodl.sh 825 14:03:11
RUDN UNIVERSITY
RUDN UNIVERSITY

```

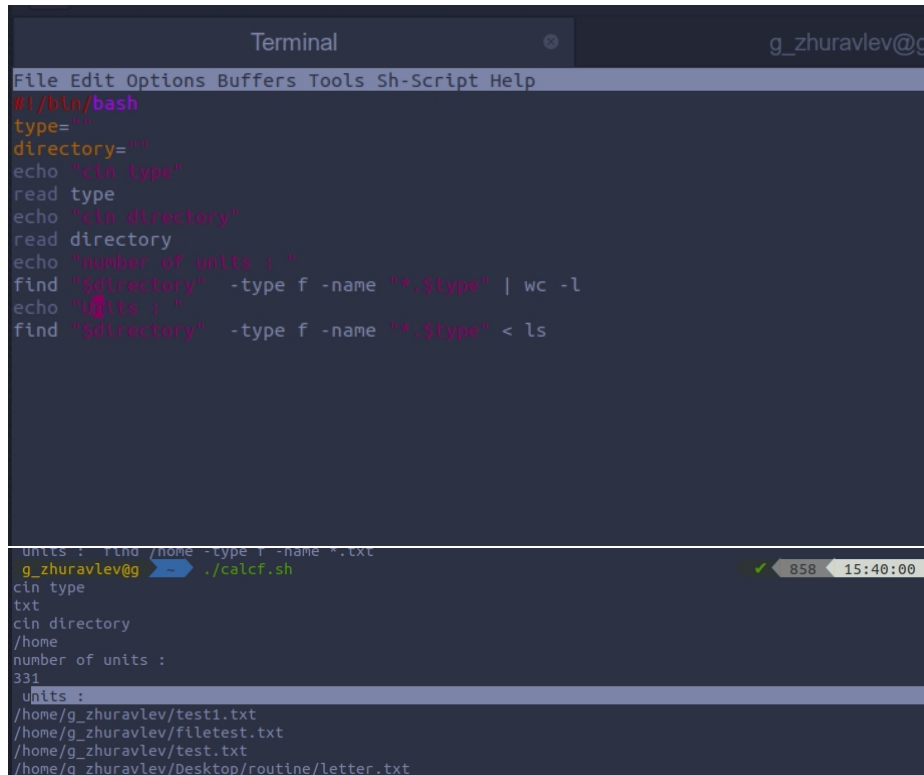
3. Написал командный файл — аналог команды ls (без использования самой этой команды и команды dir).

```
#!/bin/bash
for something in *
do if test -d $something
then echo $something - directory
else echo -n $something - file, you can:
    if test -r $something
    then echo read
    else echo -
    fi
    if test -w $something
    then echo write
    else echo -
    fi
    if test -x $something
    then echo execute
    else echo -
    fi
fi
done
```

```
U:--- ls2.sh All L5 (Shell-script[bash])
Wrote /home/g_zhuravlev/ls2.sh
mwheel-scroll: Beginning of buffer [3 times]
Saving file /home/g_zhuravlev/ls2.sh...
Wrote /home/g_zhuravlev/ls2.sh
U:%* - *Messages* Bot L52 (Messages)
Wrote /home/g_zhuravlev/ls2.sh 5
```

```
g_zhuravlev@g: ~/ls2.sh
abc1 - file, you can:read
write
-
abc2 - file, you can:read
write
-
apps - directory
backup_directory - directory
cat - file, you can:read
write
-
```

4. Написал командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.), вычисляет количество таких файлов в указанной директории и выводит их на экран.



```
Terminal
File Edit Options Buffers Tools Sh-Script Help
g_zhuravlev@g

#!/bin/bash
type=""
directory=""
echo "cin type"
read type
echo "cin directory"
read directory
echo "number of units : "
find "$directory" -type f -name "*. $type" | wc -l
echo "units : "
find "$directory" -type f -name "*. $type" < ls

units : find /home -type f -name *.txt
g_zhuravlev@g ➤ ./calcf.sh
cin type
txt
cin directory
/home
number of units :
331
units :
/home/g_zhuravlev/test1.txt
/home/g_zhuravlev/filetest.txt
/home/g_zhuravlev/test.txt
/home/g_zhuravlev/Desktop/routine/letter.txt
```

Вывод.

Благодаря этой лабораторной работе, я изучил некоторые команды языка bash; научился писать небольшие командные файлы.

Контрольные вопросы.

1. Командная оболочка— это программа, позволяющая взаимодействовать с операционной системой .В U используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;

2. POSIX — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.
3. bash обеспечивает возможность использования переменных типа строка символов. Имена переменных начинаются с буквы.
 - А. За флагом следует имя переменной, а затем список значений, разделённых пробелами.
- 4,5. Команда let является показателем того, что последующие аргументы представляют собой выражения, подлежащие вычислению.
6. (())- запись условия в оболочке bash.
7.
 - 7.1. HOME — имя домашнего каталога пользователя.
 - 7.2. IFS — последовательность символов, являющихся разделителями в командной строке.
 - 7.3. MAIL — командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла \$MAIL.
 - 7.4. TERM — тип используемого терминала.
 - 7.5. LOGNAME — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему.
8. Такие символы, как ' < > * ? | \ " &, являются метасимволами и имеют для командного процессора специальное значение.
9. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа \, который указывает на то, что следующий символ должен быть использован в том смысле, в каком он встречается в файле.
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 умолчанию. Это может быть сделано с помощью команды


```
chmod +x имя_файла
```

 Теперь можно вызывать свой командный файл на выполнение, просто вводя его имя с терминала так, как будто он является выполняемой программой. Командный процессор распознает, что в Вашем файле на самом деле хранится не выполняемая программа, а программа, которая осуществит её интерпретацию.
11. Группу команд можно объединить в функцию. Для этого существует ключевое слово function, после которого следует команда.
12. ls -lrt Если есть d, то файл является каталогом.
- 13.1. Вы можете использовать команду set для вывода списка переменных окружения. В системах Ubuntu команда set выводит список переменных окружения в виде:


```
set -- $(env)
```
- 13.2. Команду unset Следует использовать для удаления переменной из вашего окружения командной оболочкой.
 - 13.3. Команда typeset имеет четыре опции для работы с функциями:
 - f — перечисляет определённые на текущий момент функции;
 - ft — при последующем вызове функции инициализирует её трассировку;
 - fx — экспортирует все перечисленные функции в любые дочерние программы оболочки;
 - fu — обозначает указанные функции как автоматически загружаемые.

14. В командный файл можно передать до девяти параметров. При использовании где-либо в командном файле комбинации символов $\$i$, где $0 < i < 10$, вместо нее будет осуществлена подстановка

- 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 $name$;
 - $\#{name[n]}$ — 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 обра-
зом, соответствующим $pattern$;
 - $\#{name[*]}$ и $\#{name[@]}$ — эти выражения возвращают количество элементов в массиве $name$.