Отчет по 11 лабораторной работе

Лукашов Никита Александрович

Содержание

List of Tables

List of Figures

1 Цель работы

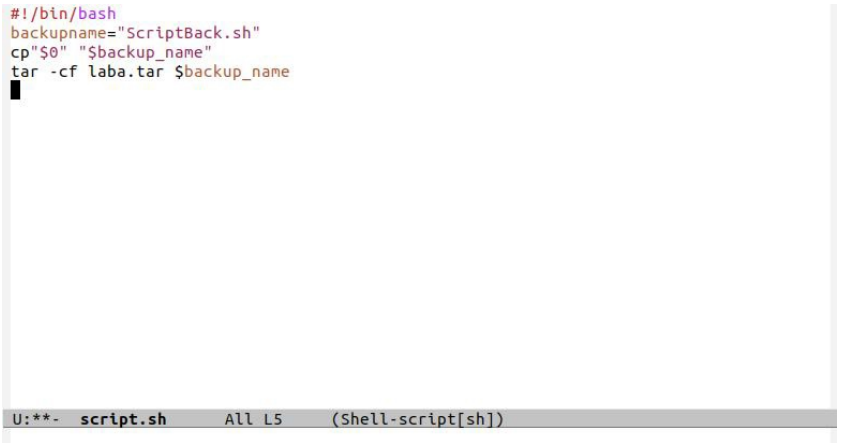
изучить основы программирования в оболочке ОС UNIX/Linux,научиться писать небольшие командные файлы.

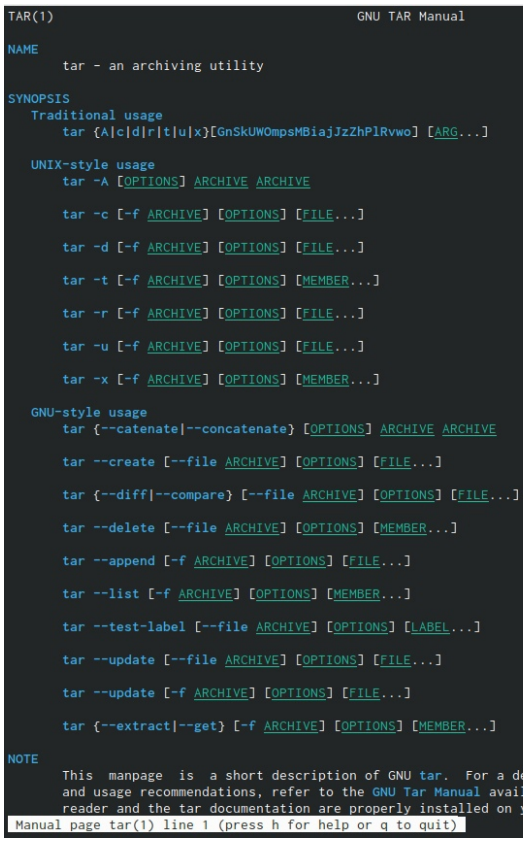
2 Задание

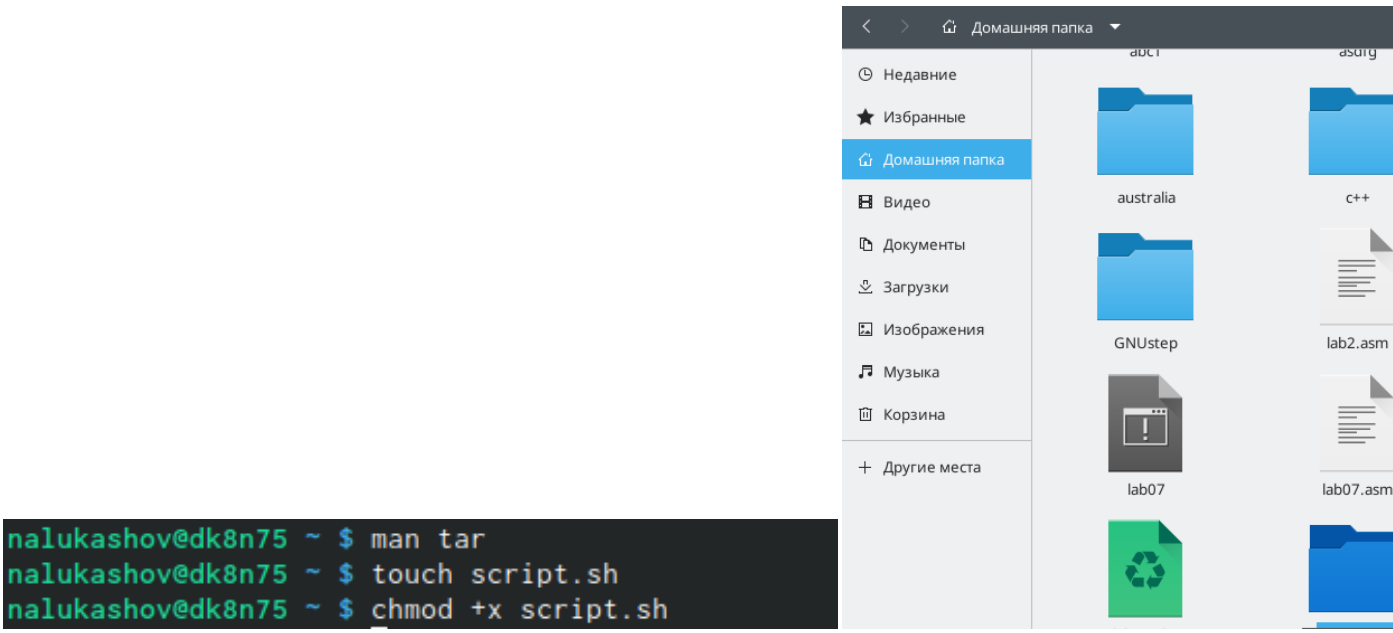
1. Ознакомиться с теоретическим материалом.
2. Ознакомиться с редактором emacs.
3. Выполнить упражнения.
4. Ответить на контрольные вопросы.

3 Выполнение лабораторной работы

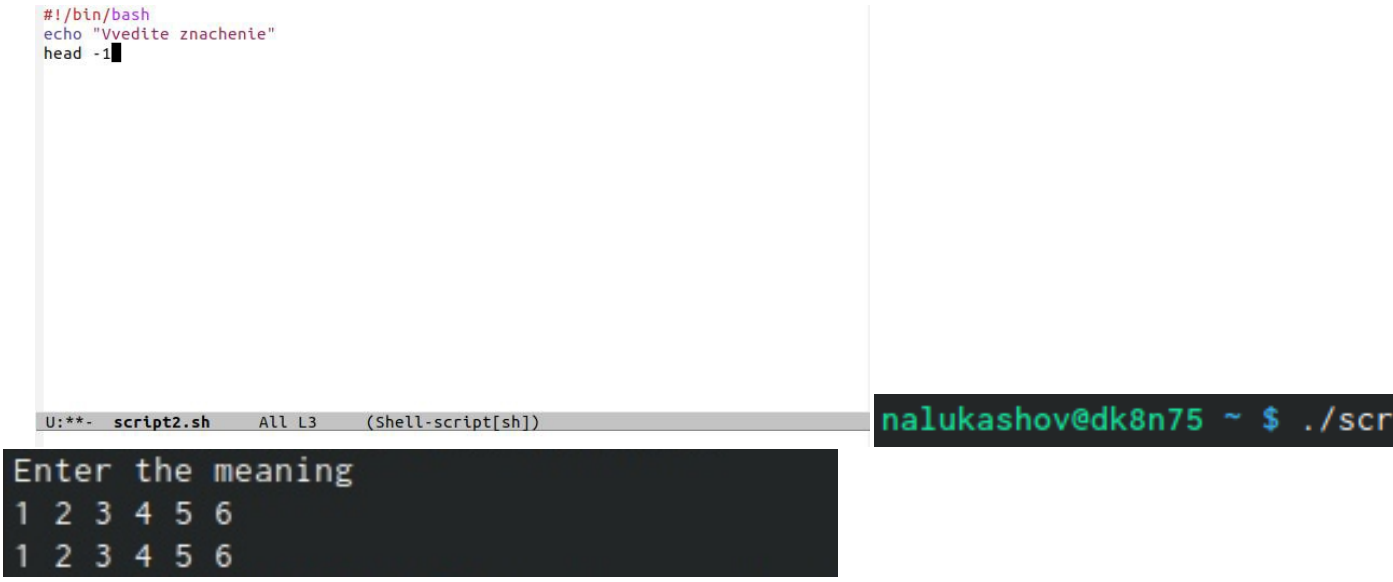
1. Написал скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации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







2. Написала пример командного файла, обрабатывающего любое произвольное



3. Написала 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 этого каталога.

```
#!/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
```

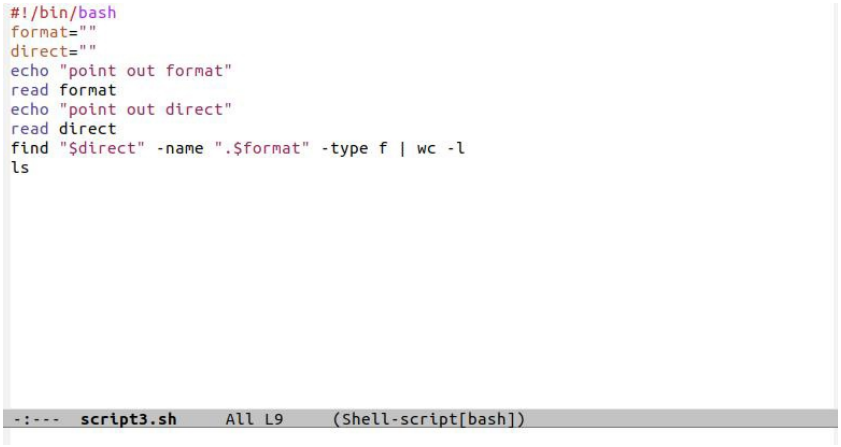
```
U:**-  file.sh        All L14    (Shell-script[sh])
```

```
ensavchenko@dk8n68 ~ $
file.sh: is a file andwr
file.sh~: is a file andw
GNUstep: is a directory
#lab07.sh#~: is a file a
lab07.sh: is a file andw
lab07.sh~: is a file and
LabaOS: is a directory
script2.sh: is a file ar
script2.sh~: is a file a
script.sh: is a file and
script.sh~: is a file ar
tmp: is a directory
work: is a directory
Видео: is a directory
Документы: is a director
Загрузки: is a directory
Изображения: is a direct
Музыка: is a directory
Общедоступные: is a dire
Шаблоны: is a directory
ensavchenko@dk8n68 ~ $
```

4. Написала 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.

```
nalukashov@dk8n75 ~ $ ./script3.sh
```

```
point out format
.sh
point out direct
/home
0
 file.sh   '#lab07.sh#~'   LabaOS       script3.sh   tm
 file.sh~   lab07.sh       script2.sh    script.sh    wo
 GNUstep    lab07.sh~      script2.sh~   script.sh~   Ви
```


#!/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
-:--- script3.sh   All L9   (Shell-script[bash])

4 Контрольные вопросы

1. 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;

– C-оболочка (или csh) — надстройка над оболочкой Борна, использующая Сподобный синтаксис команд с возможностью сохранения истории выполнения команд;

– оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;

– BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software

Foundation).

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/Linux-подобных 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.

Использование значения, присвоенного некоторой переменной, называется подстановкой. Для того чтобы имя переменной не сливалось с символами, которые могут следовать за ним в командной строке, при подстановке в общем случае используется следующая форма записи: ${имя переменной}

Например, использование команд b=/tmp/andyls -l myfile > $blssudoapt - getinstalltexlive - luatexls/tmp/andy - ls, ls - l >$bls приведёт к подстановке в командную строку значения переменной bls. Если переменной bls не было предварительно присвоено никакого значения, то её значением будет символ пробела.

Оболочка bash позволяет работать с массивами. Для создания массива используется команда set с флагом -A. За флагом следует имя переменной, а затем список

значений, разделённых пробелами. Например, set -A states Delaware Michigan “New Jersey”

Далее можно сделать добавление в массив, например, states[49]=Alaska.

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.

Команда read позволяет читать значения переменных со стандартного ввода:

echo “Please enter Month and Day of Birth ?”

read mon day trash

В переменные mon и day будут считаны соответствующие значения, введённые с клавиатуры, а переменная trash нужна для того, чтобы отобрать всю избыточно введённую информацию и игнорировать её.

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 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

8, 9. Такие символы, как ’ < > * ? | ” &, являются метасимволами и имеют для командного процессора специальный смысл. Снятие специального смысла с метасимвола называется экранированием метасимвола. 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,

5 Вывод

Изучил основы программирования в оболочке ОС UNIX/Linux, научился писать небольшие командные файлы.