Лабораторная работа № 11

Программирование в командном процессоре ОС UNIX. Командные файлы

Маметкадыров Ынтымак

Цель работы

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

Выполнение лабораторной работы

1. Создали директорию backup в нашем домашнем каталоге (рис. 1).
2. Создали файл backup.sh (рис. 1).

```
[itmametkadihrov@itmametkadihrov ~]$ mkdir backup
[itmametkadihrov@itmametkadihrov ~]$ touch backup.sh
```

Рис. 1. Создание файл backup.sh

3. Написали скрипт (рис. 2), который при запуске будет делать резервную копию самого себя в директорию backup в нашем домашнем каталоге (рис. 3).

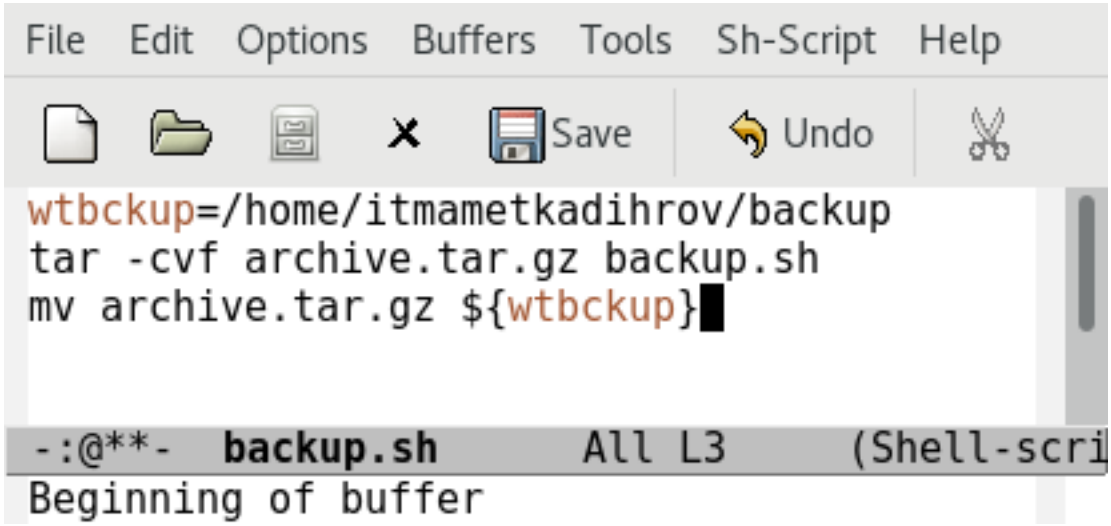

Рис. 2. Содержимое файл backup.sh

```
[itmametkadihrov@itmametkadihrov ~]$ ./backup.sh
backup.sh
[itmametkadihrov@itmametkadihrov ~]$ cd backup
[itmametkadihrov@itmametkadihrov backup]$ ls
archive.tar.gz
[itmametkadihrov@itmametkadihrov backup]$
```

Рис. 3. Созданный архив

4. Создали командный файл print.sh (рис. 4), который последовательно распечатывает значения всех переданных аргументов, в том числе превышающих десять (рис. 5).

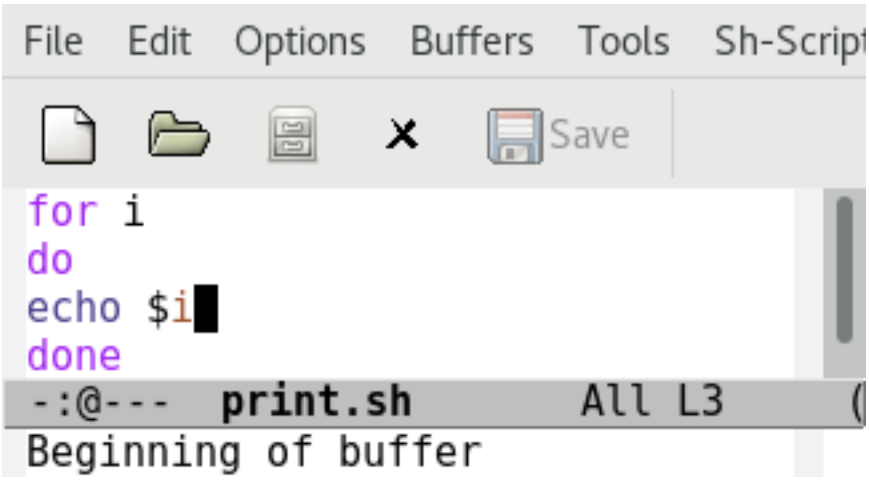

Рис. 4. Командный файл print.sh

```
[itmametkadihrov@itmametkadihrov ~]$ ./print.sh 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

Рис. 5. Работа командного файла print.sh

5. Создали командный файл ls.sh - аналог команды ls (рис. 6). Этот файл выдает информацию о нужном каталоге и выводит информацию о возможностях доступа к файлам этого каталога без использования команды ls и dir (рис. 7).

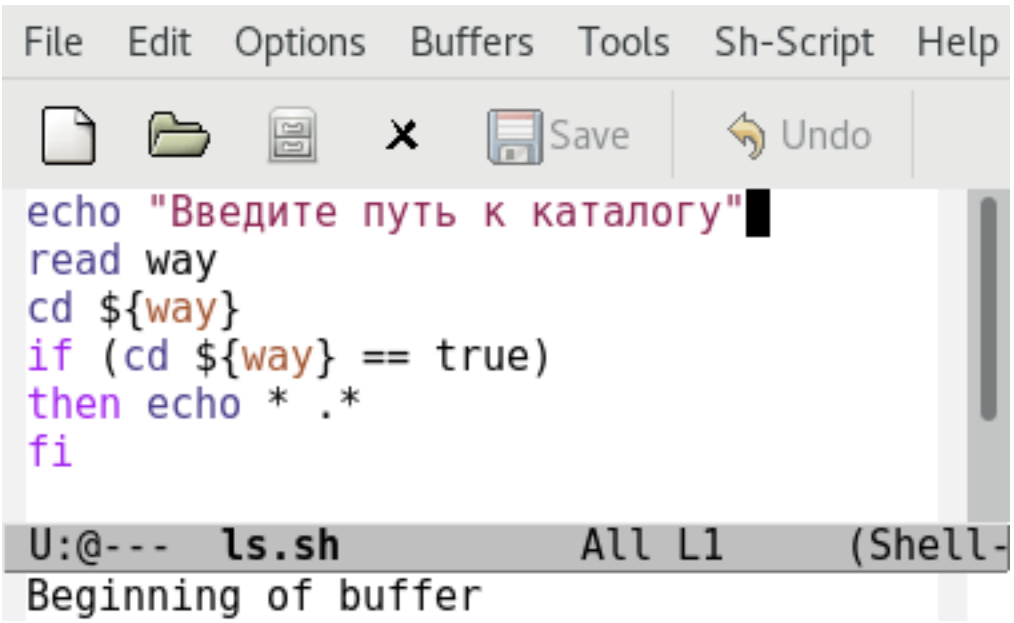

Рис. 6. Командный файл ls.sh

```
[itmametkadihrov@itmametkadihrov ~]$ ./ls.sh
Введите путь к каталогу
/home/itmametkadihrov
1.sh~ academic-laboratory-report-template academic-presentation-markdown-template a
.txt backup #backup.sh# backup.sh b.txt #count.sh# count.sh count.sh~ hello.sh hell
o.sh~ iftex.sty lab08 lab10.sh~ lab.sh lab.sh~ ls.sh ls.sh~ pandoc-2.5 pandoc-cross
ref #print.sh# print.sh print.sh~ SCR snap work Видео Документы Загрузки Изображени
я Музыка Общедоступные Рабочий стол Шаблоны . .. .atom .#backup.sh .bash_history .b
ash_logout .bash_profile .bashrc .cabal .cache .config .#count.sh .dbus .emacs.d .e
sd_auth .ghc .git .gitconfig .gnupg .gstreamer-0.10 .gtkrc-2.0-kde4 .ICEauthority .
java .kde .local .lyx .mozilla .pki .ssh .texlive2012 .text.md.swp .thumbnails .vbo
xclient-clipboard.pid .vboxclient-display.pid .vboxclient-draganddrop.pid .vboxclie
nt-seamless.pid .viminfo .xfce4-session.verbose-log .xfce4-session.verbose-log.last
 .xsession-errors .xsession-errors.old
[itmametkadihrov@itmametkadihrov ~]$
```

Рис. 7. Запуск командного файла ls.sh

6. Написали командный файл (рис. 8), 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 (рис. 9).

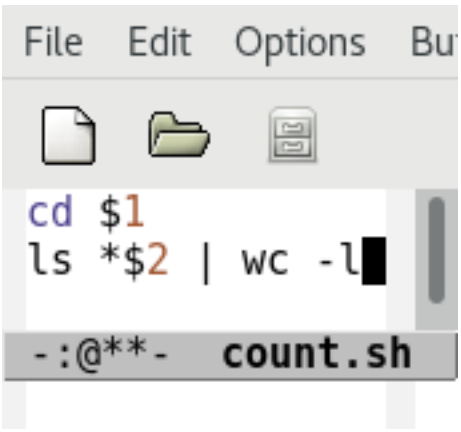

Рис. 8. Командный файл count.sh

```
[itmametkadihrov@itmametkadihrov ~]$ ./count.sh /home/itmametkadihrov .txt
2
[itmametkadihrov@itmametkadihrov ~]$
```

Рис. 9. Работа командного файла count.sh

Вывод

Изучили основы программирования в оболочке ОС UNIX/Linux. Научились писать небольшие командные файлы.

Ответы на 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 на 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.

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.

   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.

4. 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.

!	!exp	Если exp равно 0, то возвращает 1; иначе 0
!=	exp1 !=exp2	Если exp1 не равно exp2, то возвращает 1; иначе 0
%	exp1%exp2	Возвращает остаток от деления exp1 на exp2
%=	var=%exp	Присваивает остаток от деления var на exp переменной var
&	exp1&exp2	Возвращает побитовое AND выражений exp1 и exp2
&&	exp1&&exp2	Если и exp1 и exp2 не равны нулю, то возвращает 1; иначе 0
&=	var &= exp	Присваивает переменной var побитовое AND var и exp
*	exp1 * exp2	Умножает exp1 на exp2
*=	var *= exp	Умножает exp на значение переменной var и присваивает результат переменной var
+	exp1 + exp2	Складывает exp1 и exp2
+=	var += exp	Складывает exp со значением переменной var и результат присваивает переменной var
-	-exp	Операция отрицания exp (унарный минус)
-	exp1 - exp2	Вычитает exp2 из exp1
-=	var -= exp	Вычитает exp из значения переменной var и присваивает результат переменной var
/	exp / exp2	Делит exp1 на exp2
/=	var /= exp	Делит значение переменной var на exp и присваивает результат переменной var
<	exp1 < exp2	Если exp1 меньше, чем exp2, то возвращает 1, иначе возвращает 0
<<	exp1 << exp2	Сдвигает exp1 влево на exp2 бит
<<=	var <<= exp	Побитовый сдвиг влево значения переменной var на exp
<=	exp1 <= exp2	Если exp1 меньше или равно exp2, то возвращает 1; иначе возвращает 0
=	var = exp	Присваивает значение exp переменной var
==	exp1==exp2	Если exp1 равно exp2, то возвращает 1; иначе возвращает 0
>	exp1 > exp2	1, если exp1 больше, чем exp2; иначе 0
>=	exp1 >= exp2	1, если exp1 больше или равно exp2; иначе 0
>>	exp >> exp2	Сдвигает exp1 вправо на exp2 бит
>>=	var >>=exp	Побитовый сдвиг вправо значения переменной var на exp
^	exp1 ^ exp2	Исключающее OR выражений exp1 и exp2
^=	var ^= exp	Присваивает переменной var побитовое XOR var и exp
|	exp1 | exp2	Побитовое OR выражений exp1 и exp2
|=	var |= exp	Присваивает переменной var результат операции XOR var и exp
||	exp1 || exp2	1, если или exp1 или exp2 являются ненулевыми значениями; иначе 0
~	~exp	Побитовое дополнение до exp

5.

Рис. 6. Арифметические операторы оболочки bash

6. Для облегчения программирования можно записывать условия оболочки bash в двойные скобки — (( )).

7. PS1, PS2, HOME, IFS, MAIL, TERM, LOGNAME.

8. Такие символы, как ’ < > * ? | " &, являются метасимволами и имеют для командного процессора специальный смысл.

9. Снятие специального смысла с метасимвола называется экранированием метасимвола. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа , который, в свою очередь, является метасимволом.

10. Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде:

    bash командный_файл [аргументы]

11. Для этого существует ключевое слово function, после которого следует имя функции и список команд, заключённых в фигурные скобки.

12. test -f file — истина, если файл file существует;

    test -d file — истина, если файл file является каталогом.

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

    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.

14. Чтобы передать параметры в командный файл неоходимо после названия командного файла через пробел написать передаваемые параметры.

– `$*` — отображается вся командная строка или параметры оболочки;
– `$?` — 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`.

15.

Рис. 7. Арифметические операторы оболочки bash