Лабораторная работа № 3

Использование операторов цикла в сценариях командного интерпретатора BASH.

Цель работы: Изучение сценариев командного интерпретатора bash, включающих операторы цикла и использующих переменные командного интерпретатора, а также переменные сценариев в shell-сценариях и условных операторах..

Продолжительность работы - 4 ч.

Теоретические сведения

Оператор цикла с перечислением for

Оператор цикла **for** имеет структуру:

for имя параметра цикла [in список значений]
do
список команд
done

где **for** - служебное слово, определяющее тип цикла, **do** и **done** - служебные слова, выделяющие тело цикла. О завершении цикла говорит слово **done**. Фрагмент **in список** значений может отсутствовать.

Пример 1. Требуется отсортировать три текстовых файла. Пусть сценарий **sortfile** представлен следующим командным файлом:

```
for i in f1 f2 f3
do
        sort $i
done
```

В этом примере переменная **i** играет роль параметра цикла. Это имя можно рассматривать как shell-переменную, которой последовательно присваиваются перечисленные значения (**i=f1**, **i=f2**, **i=f3**). Команда **sort $i** выполняется в цикле.

В циклах часто используется форма **for i in ***, означающая перечисление всех файлов текущего каталога.

Пусть вместо команды **sort** используется последовательность команд: **$ cat $1 | sort | tee /dev/lp > ${1}sorted**,
тогда последовательно сортируются указанные файлы, результаты сортировки выводятся на печать (на устройство /**dev/lp**) и направляются в файлы **f1sorted**, **f2sorted** и **f3sorted**.

Сценарий можно сделать более универсальным, если не перечислять файлы в команде, а передавать параметрами, причем их число может быть произвольным. Тогда программа будет следующей:

```
for i
do
    sort $i
done
```

Здесь отсутствие после переменной **i** служебного слова **in** с перечислением имен означает то, что список сортируемых данных поступает через параметры команды. Результат предыдущего примера можно получить, набрав команду:

```
$ sortfile f1 f2 f3
```

Пример 2. Вывести имена всех поддиректориев директория с именем **$dir**, имя которого можно задать в качестве параметра сценария.

```
for i in $dir/*
do
    if [ -d $i ]
    then
            echo $i
    fi
done
```

Пример 3. Следует отметить различие в специальных переменных **$*** и **$@**, представляющих перечень параметров. Первый представляет параметры, как строку, а второй, как совокупность слов. Пусть командный файл **compare** имеет вид:

```
for i in "$*"
do
     echo $i
done
echo
```

```
for i in "$@"
do
    echo $i
done
```

При таком запуске программы: **$ compare aa bb cc** на терминал будет выведено:

```
aa bb cc
aa
bb
cc
```

Пример 4. (Делать не надо) Массовая рассылка по адресам, находящимся в файле всех пользователей **spisok.txt**, за исключением тех, кто есть в файле **stop.txt.** В примере рассылка направлена в файл **mess.txt,** при работе этот адрес следует заменить на адрес на сервере:

```
for user in `cat users.txt`
do
   if grep $user stop.txt < /dev/null 2 < &1
   then
            echo skip $user
   else
            mail -s "Рассылка" $user > mess.txt
   fi
done
```

Смысл не изменится, если первую строку сценария записать как **for i in $***, поскольку значение **$*** - есть список значений параметров.

Пример 5. Организации пятикратного выполнения команды. Переменная **i** принимает здесь пять значений: 1, 2, 3, 4, 5, но внутри цикла эта переменная отсутствует и поэтому ее значение никакой роли не играет и ничего не меняет. Переменная **i** могла принимать другие значения, в результате также было бы пять раз повторено одно и то же вычисление содержимого цикла без изменений.

```
# program volume5
for i in 1 2 3 4 5
do
    cat file-22 > /dev/lp
done
```

Оператор цикла с истинным условием (while)

Структура **while**, также обеспечивающая выполнение циклов, предпочтительнее тогда, когда неизвестен заранее точный список значений параметров или этот список должен быть получен в результате вычислений в цикле.

Оператор цикла **while** имеет структуру:

```
while условие
do
    список команд
done
```

где **while** - служебное слово определяющее тип цикла с истинным условием. Список команд в теле цикла (между **do** и **done**) повторяется до тех пор, пока сохраняется истинность условия (т.е. код завершения последней команды в теле цикла равен "0") или цикл не будет прерван изнутри специальными командами (**break**, **continue** или **exit**). При первом входе в цикл условие должно выполняться.

Пример 1. Вывод 50 копий документа

```
# print50: Структура "while"
# Расчет позволяет напечатать 50
# экземпляров файла "file22"
n=0
while [ $n -lt 50 ]     # пока n < 50
do
   n=`expr $n + 1`
   cat file22 > /dev/lp
done
```

Следует обратить внимание на то, что переменной **n** вначале присваивается значение **0**, а не пустая строка, поскольку команда **expr** работает с **shell**-переменными как с целыми числами, а не как со строками:

```
n=`expr $n + 1`
```

при каждом выполнении значение **n** увеличивается на **1**.

Оператор цикла с ложным условием (until)

Оператор цикла **until** имеет структуру:

until условие
do
список команд
done

где **until** - служебное слово определяющее тип цикла с ложным условием. Список команд в теле цикла (между **do** и **done**) повторяется до тех пор, пока сохраняется ложность условия или цикл не будет прерван изнутри специальными командами (**break**, **continue** или **exit**). При первом входе в цикл условие не должно выполняться. Отличие от оператора **while** состоит в том, что условие цикла проверяется на ложность (на ненулевой код завершения последней команды тела цикла) после каждого (в том числе и первого) выполнения команд тела цикла.

Пример 1.

```
until false
do
    read x
    if [ $x = 5 ]
    then echo достаточно ; break
    else echo продолжить
    fi
done
```

Здесь программа с бесконечным циклом ждет ввода слов (повторяя на экране фразу **продолжить**), пока не будет введено **5**. После этого выдается **достаточно** и команда **break** прекращает выполнение цикла.

Пример 2. В операторе условия можно использовать вычисления. До тех пор пока не наступит 12 часов дня, программа не активизируется. Каждые 30 секунд выполняется командная строка условия. Команда **date** выдает текущую дату и время и передает ее через программный канал команде **grep**. Она получает эту информацию и пытается совместить заданный шаблон "**12:00**:" с временем, выдаваемым командой **date**:

```
until date | grep 12:00:
do
    sleep 30
done
```

При несовпадении **grep** выдает код возврата **1**, что соответствует значению **ложь**, и цикл **выполняет ожидание** в течение 30 секунд, после чего повторяется проверка выполнения условия. В полдень (возможно с несколькими секундами) произойдет сравнение, условие станет истинным, команда **grep** выдаст на экран соответсвующую строку, работа цикла закончится.

Лабораторное задание и порядок выполнения работы

Изучить теоретический материал, выполнить рекомендуемые задания с использованием своих файлов и шаблонов. Выполнить примеры по архивированию и сжатию, составлению сценариев, используя свои переменные, данные и наборы команд. В команде test условия можно изменять. Законспектировать материал по новым командам, входящим в выполненные примеры и задания. Оформить отчет и защитить работу.

Контрольное задание

Составить циклические сценарии:

1. Расчет n! с вводом значения переменной n;
2. организации пятикратного выполнения команды записи информации со стандартного устройства ввода в файл. Переменная **i** принимает пять значений: 1, 2, 3, 4, 5.
3. Сценарий вывода количества печатаемых экземпляров (оно равно количеству параметров). Здесь после **for i in** для переменной **i** следует установить не перечень имен, а перечень произвольных параметров, которые можно задать в командной строке.

Требования к отчету

Отчет должен содержать:

1. Описания нескольких разнотипных сценариев;
2. краткие сведения о работе;
3. материал по новым командам, входящим в выполненные примеры и задания;
4. описание последовательности выполнения основных команд по поиску в файлах по образцу и команд создания архивов с Вашими именами файлов и каталогов выполнить подробно.