Коды завершения

Переменная shell'а **?** содержит код завершения последней выполненной команды

0 команда выполнена без ошибки (true)

не 0 выполнение команды завершено связи с ошибкой (false)

Пример:

```
$ cd
$ echo $?
0
$ echo $?
0
$ cp
Usage : cp f1 f2
...
$ echo $?
1
$ cd aa bb 2> protocol
$ echo $?
1
$ echo $?
0
```

Условный оператор (команда test)

Синтаксис:

test *выражение* или **[** *выражение* **]**

Команда test оценивает истинность выражения и формирует код завершения.

Значение выражения	**Код завершения**
true	0
false	не 0 (обычно 1)

Команда test может оценивать истинность условия, в качестве аргументов которого могут быть:

- целые числа
- строки
- Файлы

Команда test ничего не пишет в стандартный вывод

Команда test: сравнение чисел

Синтаксис :

[*число отношение число*] Сравнивает числа в соответствии с отношением

Отношения :

-lt	меньше
-le	меньше или равно
-gt	больше
-ge	больше или равно
-eq	равно
-ne	не равно

Пример:

```
$ X=1
$[  $X  -lt  7  ]
$ echo $?
0
$ [  $X  -gt   7 ]
$ echo $?
1
```

Команда test: сравнение строк

Синтаксис :

[*строка1* = *строка2*]	Определяется эквивалентность строк
[*строка1* != *строка2*]	Определяется неэквивалентность строк

Пример:

```
$ X=abc
$ [ "$X" = "abc" ]
$ echo $?
0
$ [ "$X" != "abc" ]
$ echo $?
1
```

Операторы сравнения строк

строка1 = строка2	истина, если строки идентичны друг другу
строка1! = строка2	истина, если строки не идентичны друг другу
-z строка	истина, если строка нулевой длины
-n строка	истина, если строка ненулевой длины
строка	истина, если строка ненулевой длины

Пример:

$ X=“Yes we will”

$ [$X = yes] вызывает синтаксическую ошибку,
интерпретируется shell’ом как [Yes we will = yes]

$ [“$X” = yes] правильный синтаксис,
интерпретируется shell’ом как [“Yes we will” = yes], и
сравнение осуществляется корректным образом.

Особенности сравнения чисел и строк

Shell трактует все аргументы как числа в случае, если осуществляется сравнение чисел, и все аргументы как строки, если осуществляется сравнение строк.

Пример:

$ X=03	
$ Y=3	
$ [“$X” -eq “$Y”]	сравнивается число 03 с числом 3
$ echo $?	
0	истина, т.к. аргументы равны друг другу (сравниваются как числа)
$ [“$X” = “$Y”]	сравнивается строка «03» со строкой «3»
$ echo $?	
1	ложь, т.к. аргументы не равны друг другу (сравниваются как строки)

Команда test: тестирование файлов

Синтаксис :

test -*опция имя_файла* Оценивает характеристики *имя_файла* в соответствии с *опцией*

Опции:

-f *имя_файла*	истина, если файл существует и является обычным файлом. т.е. не каталогом и не файлом устройства
-s *имя_файла*	истина, если файл существует и его размер больше 0
-r *имя_файла*	истина, если файл существует и доступен для чтения
-w *имя_файла*	истина, если файл существует и доступен для записи
-x *имя_файла*	истина, если файл существует и доступен для выполнения
-d *каталог*	истина, если файл существует и является именно каталогом

Пример:

```
$ test  -f   file                или        [ -f  file ]
$ echo  $?
0
$ test  -d  file                 или        [ -d  file ]
$ echo  $?
1
```

Логические выражения

Синтаксис :

-o OR (ИЛИ)
-a AND (И)
! NOT (НЕ)
\ (\) ГРУППИРОВКА

Примеры:

$ [“$ANS” = y -o “$ANS” = Y]
$ [“$NUM” –gt 10 -a “$NUM” -lt 20]
$ test -s file -a -r file
$ test ! –d file
$ [\($# -eq 2 \) -a \(“$1” -eq “-m” \) -a \(-d “$2” -o -s “$2” \)]

Операторы ветвления

Простейшее ветвление:

Команда_А && *Команда_В*

Команда_В выполняется, если код завершения Команды_А равен 0.

Команда_А || *Команда_В*

Команда_В выполняется, если код завершения Команды_А не равен 0.

Операторы ветвления. Конструкция if

Синтаксис: (соответствует передаче управления на одну ветвь)

```
if
        список_A

then
        список_B
fi
```

Последовательность выполнения конструкции if:

1. Выполняются команды *списка_A* команд.

2. Если код завершения последней команды из *списка_A* равен 0 (ИСТИНА), то выполняются команды из *списка_B* команд и затем команды, следующие за fi.

3. Если код завершения последней команды из *списка_A* не равен 0 (ЛОЖЬ), то выполняются команды, следующие за fi.

Конструкция if

Примеры:

1.

```
if
        echo  Starting test…
        test  -s file
then
        echo file exists
fi
echo hello
```

2.

```
if
        grep  kingkong  /etc/passwd
then
        echo found kingkong
fi
```

Конструкция if

Примеры:

3. Использование для управления переходом в случае возникновения ошибок при выполнении программы

```
if
        [ $#  -ne  3  ]
then
        echo Incorrect syntax
        echo Usage: cmd arg1 arg2 arg3
        exit 99
fi
```

Конструкция if-else

Синтаксис: (соответствует передаче управления на одну из двух возможных ветвей)

```
if
        список_A
then
        список_B
else
        список_C
fi
```

Последовательность выполнения конструкции if-else:

1. Выполняются команды *списка_A* команд.

2. Если код завершения последней команды из *списка_A* равен 0 (ИСТИНА), то выполняются команды из *списка_B* команд и затем команды, следующие за fi.

3. Если код завершения последней команды из *списка_A* не равен 0 (ЛОЖЬ), то выполняются команды из *списка_C* команд и затем команды, следующие за fi.

Конструкция if-else

Пример:

```
if
        [ $X -lt 10 ]
then
        echo X is less then 10
else
        echo X is not less then 10
fi
```

Операторы ветвления. Конструкция case

Синтаксис: (соответствует передаче управления на одну из множества возможных ветвей)

```
case слово in
образец1) список_A
                  ;;
образец2) список_B
                  ;;
образецN) список_N
                  ;;
esac
```

Образцы задаются по формату имен файлов, сравнение основано на проверке двух строк на абсолютно точное совпадение. Могут использоваться следующие специальные символы:

*	сравнение любой строки символов, включая пустую
?	сравнение любого одиночного символа
[ ]	сравнение любого одного символа, помещенного между двумя символами обеспечивает сравнение с любым, попадающим в этот интервал
|	логический оператор OR («ИЛИ»)

Конструкция case

Примеры:

```
case $ANS in
  yes)       echo O.K.
       ;;
  no) echo no
       ;;
esac
```

```
case $OPT in
1) echo option 1 ;;
2) echo option 2 ;;
3) echo option 3 ;;
*) echo no option ;;
esac
```

Конструкция case: примеры задания образцов

В образцах конструкции case используются те же специальные символы, что и для генерации имен файлов

Пример:

```
$ cat menu_with-case
echo                  COMMAND MENU
echo   d to display time and date
echo   w to display logged-in users
echo   l to list contents of current directory
echo          Please enter your choice:
read choice
case $choice in
       [dD]*) date   ;;
       [wW]*) who    ;;
       l*|L*) ls     ;;
       *)     echo Invalid selection     ;;
esac
$
```

Циклы. Конструкция for

Для каждого элемента из *списка* выполнение цикла повторяется, после чего *переменной* присваивается значение следующего элемента *списка* и так до тех пор, пока *список* не будет исчерпан.

Синтаксис:

```
for переменная in список
do
        список_A
done
```

Выполнение цикла for происходит следующим образом:
1. Shell-переменной *переменная* присваивается в качестве значения первая строка из *списка.*
2. Выполняются команды *списка_A*.
3. Shell-переменной *переменная* присваивается в качестве значения следующая строка из *списка.*
4. Выполняются команды *списка_A*.
5. Цикл продолжается до тех пор, пока не будут исчерпаны все элементы из *списка.*

Конструкция for

Пример:

```
$ cat test_for
for X in 1 2 3 4 5
do
   echo “2*$X is \c”
   let X=X*2
   echo $X
done

$ test_for
2*1 is 2
2*2 is 4
2*3 is 6
2*4 is 8
2*5 is 10
```

Конструкция for

Пример:

Цикл выполняется по списку, который может быть создан посредством подстановки команд

```
for NAME in $(grep home /etc/passwd | cut -f1 -d: )
do
  mail $NAME < mesg.txt
  echo mailed mesg.txt to $NAME
done
```

Конструкция for

Пример:

```
for FILE in  $*
do
      if
            test -d $FILE
      then
            ls -F $FILE
      fi
done
```

Использование в списке цикла аргументов командной строки

Можно создать список цикла из аргументов командной строки следующим образом:

```
for i in $*
do
      cp $i $HOME/backups
done
```

или

```
for  i
do
      cp $i $HOME/backups
done
```

Циклы. Конструкция while

Повторение выполнения **когда** условие истинно.

Синтаксис:

```
while
        список_A
do
        список_B
done
```

Выполнение цикла while происходит следующим образом:
1. Выполняются команды *списка_A*.
2. Если код завершения последней команды из *списка_A* равен 0 (true), то выполняется *список_B* команд.
3. Возвращение к п.1.
4. Если код завершения последней команды из *списка_A* не равен 0 (false), то управление передается первой команде, следующей за ключевым словом done.

Конструкция while

Пример:

```
$ cat test_while
X=1
while [ “$X” -le 10 ]
do
      echo hello X is $X; let X=X+1
done

$ test_while
hello X is 1
hello X is 2
      .
      .
      .
hello X is 10
```

Конструкция while

Пример:

Повторение выполнения цикла, когда ans соответствует yes

ans=yes

```
while
        [ “$ans” = yes ]
do
        echo Enter a name
        read name
        echo $name >> file.names
        echo “Continue?”
        enter yes or no
        read ans
done
```

Конструкция while

Пример:

Повторение выполнения цикла, когда имеются аргументы в командной строке

```
while [ "$#" != 0 ]
do
        if test -d $1
        then
                echo contents of $1:
                ls -F $1
        fi
        shift
        echo There $# items
        echo left on the cmd line
done
```

Циклы. Конструкция until

Повторение выполнения тела цикла **до тех пор, пока** условие истинно.

Синтаксис:

```
until
        список_A
do
        список_B
done
```

Выполнение цикла until происходит следующим образом:
1. Выполняются команды *списка_A*.
2. Если код завершения последней команды из *списка_A* не равен 0 (false), то выполняется *список_B* команд.
3. Возвращение к п.1.
4. Если код завершения последней команды из *списка_A* равен 0 (true), то управление передается первой команде, следующей за ключевым словом done.

Конструкция until

Пример:

```
$ cat test_until
X=1
until [ “$X” -gt 10 ]
do
      echo hello X is $X; let X=X+1
done

$ test_until
hello X is 1
hello X is 2
      .
      .
      .
hello X is 10
```

Конструкция until

Пример:

Повторение выполнения цикла, когда ans не станет no

```
ans=yes
until
        [ “$ans” = no ]
do
        echo Enter a name
        read name
        echo $name >> file.names
        echo “Continue?”
        enter yes or no
        read ans
done
```

Конструкция until

Пример:

Повторение выполнения тела цикла до тех пор, пока в командной строке не окажется аргументов

```
until [ “$#” == “0” ]
do
        if test -d $1
        then
                echo contents of $1:
                ls -F $1
        fi
        shift
        echo There $# items
        echo left on the cmd line
done
```

Вычисление значений арифметических выражений – команда let

Синтаксис:

let *выражение* или **((*выражение*))**

При построении арифметических выражений можно использовать операторы (в порядке убывания приоритета выполнения):

Оператор	Описание
-	Одноместный минус (операция изменения знака арифметического выражения
!	Логическое отрицание
* / %	Умножение, деление, остаток от деления
+ -	Сложение, вычитание
<= >= < >	Сравнение
== !=	Равно, не равно
=	Присвоение
( )	Скобки используются для изменения порядка вычисления

Вычисление значений арифметических выражений – команда let

Пример:

```
$ x=10
$ y=2
$ let x=x+2
$ echo $x
12
$ let "x=x/(y+1)"
$ echo $x
4
$ (( x=x+1 ))
$ echo $x
5
```

```
$ x=12
$ let "x < 10"
$ echo $?
1
$ (( x > 10 ))
$ echo $?
0
$ if (( x > 10 ))
> then echo x greater
> else echo x not greater
> fi
x greater
```

Вычисление значений арифметических выражений – команда expr

Синтаксис:

expr ***выражение*** - выводит значение арифметического выражения в stdout

Примеры

```
$ x=10
$ y=5
$ expr  $x+$y
15
$ z=$(expr  $x+$y)
$ echo $x+$y=$z
10+5=15
$
```

shell-функции

Синтаксис:

function *имя_функции {shell_текст}*

или

имя_функции (){*shell_текст}*

Пример:

```
$ function install
> {
>       echo Install file:$1
>       chmod +x $1
>       mv $1 $HOME/bin
>       echo Install complete
> }
$ install myfile
Install file: myfile
Install complete
```

```
install()
{
        echo Install file:$1
        chmod +x $1
        mv $1 $HOME/bin
        echo Install complete
}
```

Команды `exit` и `return`

`exit` [**аргумент**]

•Завершает выполнение shell-программы и формирует код возврата.

•Если команда задана без аргументов, то код возврата устанавливается равным коду возврата последней команды, выполненной перед exit.

`return` [**аргумент**]

•Останавливает выполнение функции и возвращает в вызвавшую процедуру аргумент, содержащий код завершения.

•Если аргумент не указан, то код зовершения соответствует последней команде, выполненной в функции перед выходом из нее.

•Когда прекращение выполнения функции обусловлено какими-то внешними по отношению к ней причинами, то это равносильно exit.

Команды exit и return

Примеры:

```
$ cat exit_test
echo exiting program now
exit 99

$ exit_test
exiting program now

$echo $?
99
```

==========

```
$ cat rtn_test
function rtn
{
echo Returning from function
return 99
}

$ rtn_test
Returning from function

$echo $?
99
```

Конструкции break, continue, exit

break [n]	Прекращает текущее выполнение цикла и передает управление первой (или с номером n) команде после done.
continue [n]	Останавливает выполнение текущей итерации цикла и передает управление на начало очередной (или с номером n) итерации вложенного цикла.
exit [n]	Останавливает выполнение shell-программы и формирует код завершения n.

Команды break и continue: примеры

```
while
      true
do
      echo “Enter file to remove:\c”
      read FILE
      if test ! -f $FILE
      then
            echo $FILE is not a regular file
            continue
      fi
      echo removing $FILE
      rm $FILE
      break
done
```