ПРИЛОЖЕНИЯ UNIX СИСТЕМ

Лекция №3

Ст. преподаватель
кафедры ПДСиМ
Квиткова Ирина Геннадьевна

Вычисления в Shell

Командная оболочка позволяет выполнять арифметические операции:

a +(-) b – сложение (вычитание);
a * b – умножение (**a * b**) – для **expr**;
a / b – деление;
a % b – остаток от деления;
a ** b – возведение в степень (**a ^ b** – для **bc**);
sqrt (a) – квадратный корень.

1) **echo $((a+b))**; **c=$(($x * $y))**.

2) **echo "sqrt(16)" | bc**; **c=$(echo "2+2" | bc)**;
echo 'scale=4; sqrt(9^3)' | bc *# scale – точность.*

3) **expr 15 + 25**; **c=`expr $x \* $y + 34 / 2`**.

4) **let "c = a % b"**.

Системы счисления

Возможен перевод из одной системы счисления в другую.

Перевод из двоичной в десятичную систему:

echo ''ibase=2; obase=A; 1100''|bc

12

 # ibase – указывает исходную систему счисления;

obase – указывает систему, в которую переводят.

но

echo ''obase=10; ibase=2; 1100''|bc

- из binary в hex

echo 'obase=16; ibase=2; 1100'|bc.

echo "obase=2; $NUM"| bc | xargs printf "%08d\n"

– перевод десятичного числа $NUM в двоичное, представленное 8-ю цифрами.

Структурные операторы Shell

Условный оператор "if"

Оператор "if" имеет структуру

```
if  [ условие ]
  then  список
     (elif  условие
      then  список)
  else  список
fi
```

Условия записываются в форме ключей

-gt (>), -lt (<), -le (<=), -ge (>=), -eq (=), -ne (не равно). (для сравнения числовых значений)

Пример:

```
if  [ $1 –ne $2 ]
          then  cat a1
          else  echo a1
       fi
```

Проверка условий (для файлов):

[-d FILE]	FILE директорией.
[-e FILE]	FILE существует.
[-f FILE]	FILE является обычным файлом.
[-h (-L) FILE]	FILE является символической ссылкой.
[-p FILE]	FILE является именованным каналом (FIFO).
[-r FILE]	FILE доступен для чтения.
[-s FILE]	размер FILE больше нуля.
[-w FILE]	FILE доступен для записи.
[-x FILE]	FILE является исполняемым.
[-O FILE]	владелец FILE - действующий пользователь.

Проверка условий (для строк):

[-z "STRING"]	длина строки "STRING" равна нулю.
[-n "STRING"]	длина строки "STRING" ненулевая.
[STRING1 = STRING2]	строки равны.
[STRING1 != STRING2]	строки не равны.

Пример 1:

```
echo "Tell me a word"
read a
if [ "$a" = "yes" ] # "$a"=${a}
then
        head -1 $1
else
        echo "You didn’t guess"
fi
```

Пример 2:

```
if [ -d $1 ]
then
   du -sh $1
else
  echo "$1 is file with $(wc –l $1| cut –f1 –d” ”) str."
fi
```

Пример 3:

```
a=`find . -empty -type f`
if [ $? -eq 0 ]
then
     echo "$a\n" # \n  - перевод строки
fi
```

Оператор выбора "case".

Оператор выбора "case" имеет структуру:

```
case   переменная  in
    шаблон)  список команд;;
    шаблон)  список команд;;
        …
esac
```

Пример:

```
case $1 in
        1) sort a1;;
        2) sort –r a1;;
        *) cat a1;;
        esac
```

При запуске **sh script 1** выполняется сортировка a1, при - **sh script 2 -** сортировка a1 в обратном порядке, при - **sh script 3 –** вывод содержимого a1.

Оператор цикла с перечислением "for" имеет структуру:

```
for переменная in список значений
      do
         список команд
      done
```

Пример 1: ***for i in ****
```
            do
              cat $i
            done
```
-вывести все файлы текущего каталога

Пример 2: ***for f in $(ls /var); do***
```
              echo $f
            done
```

Оператор цикла с истинным условием.
Оператор цикла **"while"** имеет структуру:

```
while [ условие ]
    do
        список команд
    done
```

- Список команд в теле цикла (между "do" и "done") повторяется до тех пор, пока сохраняется истинность условия.
- При первом входе в цикл условие должно выполняться.
- Начальное истинное значение задаётся перед циклом.

Оператор цикла с ложным условием.
Оператор цикла **"until"** имеет структуру:

```
until [ условие ]
      do
         список команд
      done
```

- Список команд в теле цикла (между "do" и "done") повторяется до тех пор, пока сохраняется ложность условия.
- При первом входе в цикл условие не должно выполняться.
- Начальное значение условия задаётся перед циклом.

Пример 1: **x=0**

```
while [ $x –lt 5 ]
 do
     cat a1
     x=`expr $x + 1`
 done
```

Выводится содержимое файла **a1** на экран **5** раз.

Пример 2: **x="ok"**

```
until [ "$x" = "not ok" ]
 do
    cat a1
    echo "show again? "
    read x
 done
```

Пока **x** не равен **ok**, на экране – содержимое **a1**. Значение **x** задаётся с клавиатуры.

Функция в shell.

Функция позволяет подготовить список команд для последующего выполнения.

Описание функции имеет вид:

```
имя()
 {
 список команд
 }
 имя
```

- Функция используется для выполнения действия рекурсивно.

Пример 1:

```
fn()
{
  for i in *
       do
         if [ -d $i ]
           then
             cd $i; fn; cd ..
           else
             grep 3 $i && echo $i
         fi
       done
}
fn
```

Рекурсивный поиск (в каталоге и подкаталогах) файлов, содержащих строки с цифрой 3.

Пример 2 (позиционная переменная в функции):

```
factorial()
{
  i=1; n=1
  while [ $i -le $1 ]
  do
     n=`expr $n \* $i`
     i=`expr $i + 1`
  done
     echo $n
}
factoriall $1
```

- вычисление факториала.

При запуске скрипта необходимо указать значение для $1.