Глава 4. Переменные и параметры. Введение.

Переменные -- это одна из основ любого языка программирования. Они учавствуют в арифметических операциях, в синтаксическом анализе строк и совершенно необходимы для абстрагирования каких либо величин с помощью символических имен. Физически переменные представляют собой ни что иное как участки памяти, в которые записана некоторая информация.

4.1. Подстановка переменных

Когда интерпретатор встречает в тексте сценария *имя* переменной, то он вместо него подставляет *значение* этой переменной. Поэтому ссылки на переменные называются *подстановкой переменных*.

$

Необходимо всегда помнить о различиях между *именем* переменной и ее *значением*. Если **variable1** -- это имя переменной, то **$variable1** -- это ссылка на ее *значение*. "Чистые" имена переменных, без префикса $, могут использоваться только при объявлении переменный, при присваивании переменной некоторого значения, при *удалении (сбросе)*, при экспорте и в особых случаях -- когда переменная представляет собой название сигнала (см. Пример 29-5). Присваивание может производится с помощью символа = (например: *var1=27*), инструкцией read и в заголовке цикла (*for var2 in 1 2 3*).

Заключение ссылки на переменную в двойные кавычки (" ") никак не сказывается на работе механизма подстановки. Этот случай называется "частичные кавычки", иногда можно встретить название "нестрогие кавычки". Одиночные кавычки (' ') заставляют интерпретатор воспринимать ссылку на переменную как простой набор символов, потому в одинарных кавычках операции подстановки не производятся. Этот случай называется "полные", или "строгие" кавычки. Дополнительную информацию вы найдете в Глава 5.

Примечательно, что написание **$variable** фактически является

упрощенной формой написания **${variable}**. Более строгая форма записи **${variable}** может с успехом использоваться в тех случаях, когда применение упрощенной формы записи порождает сообщения о синтаксических ошибках (см. Section 9.3, ниже).

Пример 4-1. Присваивание значений переменным и подстановка значений переменных

```
#!/bin/bash

# Присваивание значений переменным и подстановка значений переменных

a=375
hello=$a

#-------------------------------------------------------------------------
# Использование пробельных символов
# с обеих сторон символа "=" присваивания недопустимо.

#  Если записать "VARIABLE =value",
#+ то интерпретатор попытается выполнить команду "VARIABLE" с параметром
"=value".

#  Если записать "VARIABLE= value",
#+ то интерпретатор попытается установить переменную окружения
"VARIABLE" в ""
#+ и выполнить команду "value".
#-------------------------------------------------------------------------


echo hello    # Это не ссылка на переменную, выведет строку "hello".

echo $hello
echo ${hello} # Идентично предыдущей строке.

echo "$hello"
echo "${hello}"

echo

hello="A B  C   D"
echo $hello   # A B C D
echo "$hello" # A B  C   D
# Здесь вы сможете наблюдать различия в выводе echo $hello и echo
"$hello".
# Заключение ссылки на переменную в кавычки сохраняет пробельные
символы.

echo

echo '$hello'  # $hello
# Внутри одинарных кавычек не производится подстановка значений
переменных,
#+ т.е. "$" интерпретируется как простой символ.

# Обратите внимание на различия, существующие между этими типами
```

```
кавычек.


hello=    # Запись пустого значения в переменную.
echo "\$hello (пустое значение) = $hello"
#  Обратите внимание: запись пустого значения -- это не то же самое,
#+ что сброс переменной, хотя конечный результат -- тот же (см. ниже).

# --------------------------------------------------------------

#  Допускается присваивание нескольких переменных в одной строке,
#+ если они отделены пробельными символами.
#  Внимание! Это может снизить читабельность сценария и оказаться
непереносимым.

var1=variable1  var2=variable2  var3=variable3
echo
echo "var1=$var1   var2=$var2  var3=$var3"

# Могут возникнуть проблемы с устаревшими версиями "sh".

# --------------------------------------------------------------

echo; echo

numbers="один два три"
other_numbers="1 2 3"
# Если в значениях переменных встречаются пробелы,
# то использование кавычек обязательно.
echo "numbers = $numbers"
echo "other_numbers = $other_numbers"   # other_numbers = 1 2 3
echo

echo "uninitialized_variable = $uninitialized_variable"
# Неинициализированная переменная содержит "пустое" значение.
uninitialized_variable=   #  Объявление неинициализированной переменной
                          #+ (то же, что и присваивание пустого
значения, см. выше).
echo "uninitialized_variable = $uninitialized_variable"
                          # Переменная содержит "пустое" значение.

uninitialized_variable=23       # Присваивание.
unset uninitialized_variable    # Сброс.
echo "uninitialized_variable = $uninitialized_variable"
                                # Переменная содержит "пустое" значение.

echo

exit 0
```

Неинициализированная переменная хранит "пустое" значение - не ноль!. Использование неинициализированных переменных может приводить к ошибкам разного рода в процессе исполнения.

Не смотря на это в арифметических операциях допускается использовать неинициализированные переменные.

```
echo "$uninitialized"                                   #
(пустая строка)
let "uninitialized += 5"                                #
Прибавить 5.
echo "$uninitialized"                                   # 5

#  Заключение:
#  Неинициализированные переменные не имеют значения, однако
#+ в арифметических операциях за значение таких переменных
принимается число 0.
#  Это недокументированная (и возможно непереносимая)
возможность.
```

См. так же Пример 11-19.

Назад
Служебные символы

К началу
Наверх

Вперед
Присваивание значений переменным