Общие сведения:

Язык KolibriScript предназначен для выполнения различных командв командной среде Commander.

Любая последовательность конструкций на язык KolibriScript называется скриптом.

Скрипт может быть представлен как одной строкой ввода (при работе в терминале), так и отдельным файлом с расширением «.ks».

В языке KolibriScript допускается использование данных двух типов: целочисленной значение (соответствует типу int языка программирования C), строковое значение (соответствует типу char* языка программирования C).

Существует две версии переменных: глобальные и локальные. Глобальные сохраняются в отдельном файле и доступны из любого скрипта. Локальные переменные существуют только в рамках текущей сессии.

Под сессией понимается выполнение скриптов из файлов или выполнение последовательности команд в командной строке.

Результатов выполнения любого скрипта является строка или пустое выражение (void).

Язык KolibriScript поддерживает:

- переменные двух типов данных и двух типов видимости;

- метки – обозначение места, куда можно перейти по безусловному переходу;

- условия – поддерживаются цепочки условий, а также вложенные условия;

- безусловный переход к метке;

- перенаправления ввода и вывода;

- логические операции сравнения на равенство и неравенство, логическая операция «И», логическая операция «ИЛИ» с возможностью вложения;

- арифметические операции сложения и вычитания с возможностью вложения.

- выполнение команд с аргументами.

Грамматические и синтаксические правила:

Скрипт = {Конструкция, [Разделитель, Конструкция]}, «\0».

Тело условия = Конструкция, [Разделитель, Конструкция], Разделитель.

Разделитель = «\n» | «;».

Конструкция = Условие | Объявление переменной | Название переменной | Команда | Перенаправление вывода | Перенаправление ввода | Объявление метки | Переход к метке.

Условие = Условие «if», { Условие «elif» }, [Условие «else»].

Объявление переменной = Объявление глобальной переменной | Объявление локальной переменной.

Команда = Название команды, {Аргумент, [Аргумент]}.

Перенаправление вывода = Из файла в вывод | Из файла в переменную | Из переменной в файл | Из команды в файл | Из команды в переменную.

Перенаправление ввода = В файл | В переменную.

Из файла в вывод = Строка, Перенаправление вправо.

Из файла в переменную = Строка, Перенаправление вправо, Название.

Из переменной в файл = Название переменной, Перенаправление вправо, Строка.

Из команды в файл = Команда, Перенаправление вправо, Строка.

Из команды в переменную = Команда, Перенаправление вправо, Название переменной.

В файл = Перенаправление вправо, Строка.

В переменную = Перенаправление вправо, Название переменной.

Перенаправление вправо = «>» | «>>».

Объявление метки = «:», Название метки.

Переход к метке = «to», Название метки.

Условие «if» = «if», Логическое выражение, «then», Тело условия, «end».

Условие «elif» = «elif», Логическое выражение, «then», Тело условия, «end».

Условие «else» = «else», Тело условия, «end».

Название переменной = Название глобальной переменной | Название локальной переменной.

Название глобальной переменной = «$», Название.

Название локальной переменной = «@», Название.

Объявление глобальной переменной = Название глобальной переменной, «=», Значение переменной.

Объявление локальной переменной = Название локальной переменной, «=», Значение переменной.

Значение переменной = Арифметическое выражение | Сложение строк.

Аргумент = Флаг | Параметр.

Логическое выражение = Логический операнд, [Логическая операция, Логический операнд].

Логический операнд = «(», Логическое выражение, «)» | Название переменной | Число | Строка.

Логическая операция = «=» | «!» | «&» | «|».

Арифметическое выражение = Арифметический операнд, {Арифметическая операция, Арифметический операнд}.

Арифметический операнд = «(», Арифметическое выражение, «)» | Название переменной | Число.

Арифметическая операция = «+» | «-».

Сложение строк = Строковой операнд, {«+», Строковой операнд}.

Строковой операнд = Строка | Название переменной.

Строка = («’», {Символ, кроме апострофа}, «’») | (Символ, кроме зарезервированных, {Символ без пробела}).

Символ, кроме апострофа = <Любой символ, кроме апострофа>.

Символ без пробела = <Любой символ, кроме пробела>.

Символ, кроме зарезервированных = <Любой символ, кроме зарезервированных символов языка>.

Название команды = Название.

Название метки = Название.

Название = Буква, {Цифра | Буква}.

Флаг = «-», Буква, {Буква | Цифра}.

Параметр = Строка | Название переменной.

Число = [«+» | «-»] (Цифра | (Цифра без нуля, {Цифра})).

Цифра = «0» | Цифра без нуля.

Цифра без нуля = «1»-«9».

Буква = «a»-«z» | «A»-«Z» | «_».

Обозначения:

<Любой символ, кроме апострофа> - любой символ ASCII, кроме апострофа.

<Любой символ, кроме пробела> - любой символ ASCII, кроме пробела.

Семантические правила:

Работа с переменными:

- переменные одного и того же уровня видимости можно переопределять, задавая другое значение;

- нельзя назвать локальную/глобальную переменную именем другой глобальной/локальной переменной;

- переменные обязательно должны иметь какое-либо значение;

- нельзя присвоить переменной значение этой же переменной, пока она не будет инициализирована.

Работа с логическими выражениями:

- результатом выполнения логического выражения является число 0 или 1;

Работа с условиями:

- если в качестве условия выступает один операнд:

1) если операнд – арифметическое выражение, то условие истинно, если значение арифметического значения отлично от 0, иначе - ложь;

2) если операнд – строка, то условие истинно, если строка непустая, иначе - ложь.

Работа с арифметическими выражениями:

- операндами арифметического выражения могут быть только числа и переменные, содержащие числа.

Работа со сложением строк:

- операндами операции сложения строк могут быть только строки и переменные, содержащие строки.

Работа со строками:

- строки представляют собой последовательности экранированных символов, то есть все символы, находящиеся внутри строки, не несут каких-либо управляющих инструкций.

Работа с метками:

- нельзя создать две метки с одинаковыми именами;

- метки и переходы по ним не работают в режиме интерактивного ввода.

Работа с ошибками:

Скрипт может завершить работу с ошибкой. Обычно ошибки выводятся прямо в терминал, но пользователь также может повлиять на работу с ошибками:

- локальная переменная @IS_ERROR предназначена для хранения успешности выполнения последней конструкции: 1 – если имеется ошибка, 0 – если нет ошибок;

- локальная переменная @ERROR предназначена для хранения возникшей в ходе выполнения последней конструкции ошибки: 1 – если имеется ошибка, 0 – если нет ошибок;

- локальная переменная @DISPLAY_ERRORS предназначена для задания механизма вывода ошибок: 1 – если требуется отображать ошибки в терминал, 0 – если ошибки должны сохраняться только в переменных для ошибок.

Примеры использования:

Пример 1. Hello, world!

echo ‘Hello, world!’
Вывод: Hello, world!

Пример 2. Числа Фибоначчи

```
@count = 5
@number1 = 0
@number2 = 1
@counter = 0
:do
if  @counter  !  @count  then
    @counter = @counter + 1
    if @counter = 1 then
        echo @number1
    end
    elif  @counter  =  2  then
        echo @number2
    end
   else
       @temp = @number1 + @number2
       @number1 = @number2
       @number2 = @temp
    end
to do
end
echo End
```

```
Вывод:
0
1
1
2
3
End
```

Пример 3. Вывод всех директорий, названия которых содержат цифры

```
@root = /dirs/
@dirs_count = 0
@index = 0
dc @root > @dirs_count
dl @root > dirs.txt
:do
if @index ! @dirs_count then
   getL dirs.txt @index > @dir_name
   contain_number @dir_name > @is_number
   if @is_number then
      echo @dir_name
   end
   @index = @index + 1
   to do
end
```

```
Вывод:
dir1
dir2
dir3
```

Примечания:

В примере 3 использовались команды командной оболочки:

- dc <путь к папке> - возвращает число папок по указанному пути (вывод: число);

- dl <путь к папке> - выдает список всех директорий по указанному пути (вывод: строка, содержащая названия всех папок, разделенных символом «\n»);

- getL <путь к файлу> <индекс строки> - читает строку файла по указанному символу (вывод: строка файла, за исключением символа «\n»);

- contain_number <строка> - проверяет строку на наличие цифр (вывод: 0 – если нет цифр, 1 – если есть цифры).