Assignment Tutorial

Assigning values

Setting the value of a variable is an *assignment*. You can read about assignments in section 2.4.3 of the Reference Manual.

```
> x = 1
> y = "hello"
> print(x,y)
1       hello
```

Multiple assignment

In Lua we can can perform *multiple assignments*, e.g.,

```
> x, y = 2, "there"
> print(x,y)
2       there
```

The list of values on the right is assigned to the list of variables on the left of the `=`. We can assign as many values as we like and they don't all have to be of the same type. e.g.,

```
> a,b,c,d,e,f = 1,"two",3,3.14159,"foo",{ this="a table" }
> print(a,b,c,d,e,f)
1       two     3       3.14159 foo     table: 0035BED8
```

Evaluation order

Any expressions on the left and the values on the right are evaluated before the assignments are made.

```
> i = 7
> i, x = i+1, i
> print(i, x)
8       7
```

Notice that `x` has the value of `i` before it was modified by the list of values on the right hand side.

Swapping values

Because values are assigned as though all assignments are simultaneous, you can use multiple assignment to swap variable values around.

```
> a,b = 1,2  -- set initial values
> print(a,b)
1       2
> a,b = b,a  -- swap values around
> print(a,b)
2       1
> a,b = b,a  -- and back again
> print(a,b)
1       2
```

Assignment order

The order in which multiple assignments are performed is not defined. This means you should not assume that the assignments are made left to right; if the same variable or table reference occurs twice in the assignment list, you may be surprised by the results.

```
> a, a = 1, 2
> print(a)
1
```

Note, in the above example Lua does assignments right-to-left, i.e., `a=2` and then `a=1`. However, we should not depend on this being consistent in future versions of Lua. If the order of assignment is important, you should use separate assignment statements.

In particular, watch out for statements like the following. If `i==j`, these two statements do different things:

```
> table[i], table[j] = table[j], table[k]
> table[j], table[i] = table[k], table[j]
```

It should be written as two separate statements. This always swaps the two values, though:

```
> table[i], table[j] = table[j], table[i]
```

Mismatched list sizes

If a value list is longer than the variable list the extra values are ignored.

```
> a,b,c = 1,2,3,4,5,6
> print(a,b,c)
1       2       3
```

If a value list is shorter than the variable list Lua assigns the value `nil` to the variables without a value.

```
> a,b,c,d = 1,2
> print(a,b,c,d)
1       2       nil     nil
```

FindPage · RecentChanges · preferences
edit · history
Last edited May 9, 2003 2:41 pm PDT (diff)