Rocket software

# Rocket Uniface Library 10.4

## Struct Variables

A Struct variable is a variable, parameter, or non-database field of type `struct` that holds an ordered collection of references to zero or more Structs or Struct members.

Struct variables are similar to handles in that both `struct` and `handle` are reference data types that cannot hold a value. Instead they hold references to data in memory. This is in contrast to scalar variables, which hold a value.

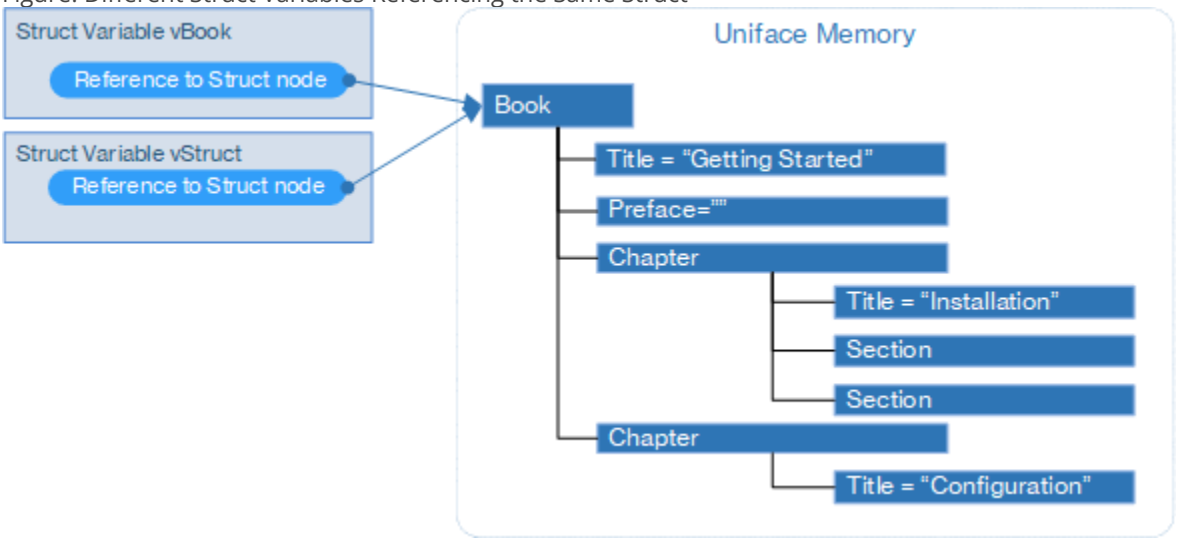
Unlike a `handle`, which can only hold one reference, a `struct` variable holds a collection of references.

The nature of the `struct` data type has a number of consequences when accessing a Struct:

- Several `struct` variables can refer to the same Struct. For example:

  ```
  variables
   struct vStruct, vBook, vChapter, vTitle
  endvariables
  ; Convert an XML document to a Struct
  xmlToStruct vBook, "file://book.xml"
  vStruct = vBook
  ```

  Figure: Different Struct Variables Referencing the Same Struct

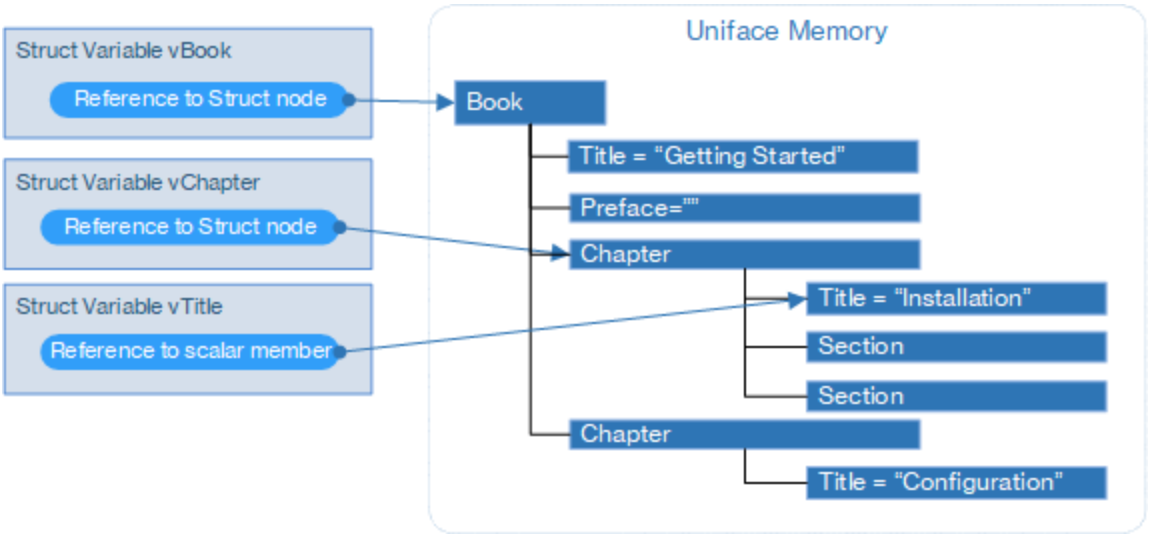
  
  

- A `struct` variable can refer to a member of a Struct. Access operators enable you specify the member.

  ```
  vChapter = vBook->chapter{1} ; Struct index operator
  vTitle = vChapter->Title ; Struct de-reference operator
  ```

  `vChapter` now refers to the first member named chapter in the Struct referenced by `vBook`, and `vTitle` refers to the title of that chapter. The chapter member is a nested Struct, whereas the title member is a scalar member.

  Figure: Struct Variables Referencing Members of a Struct




All these variables point to the same Struct in memory.

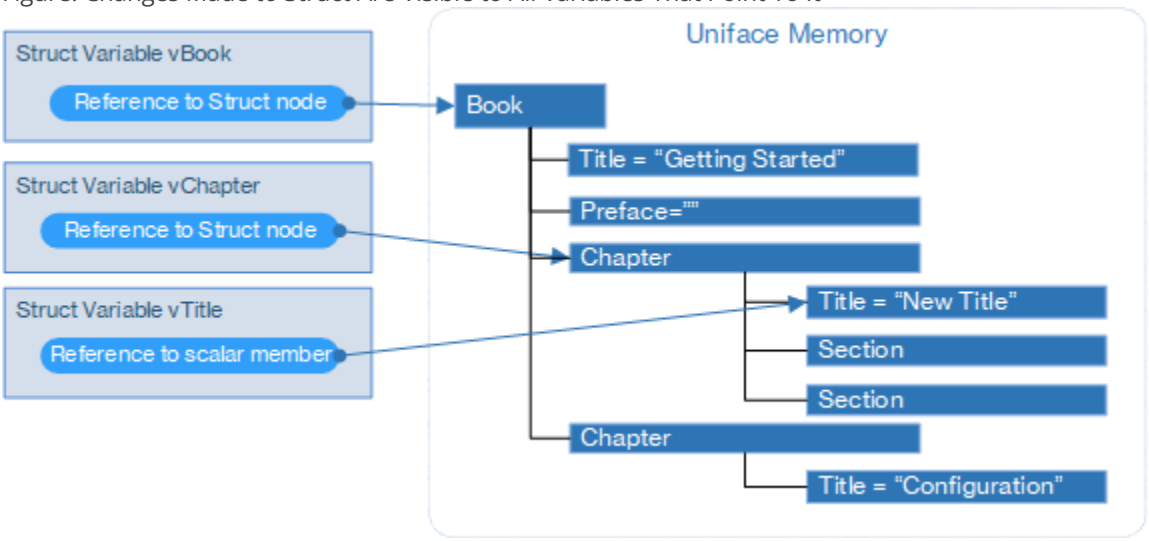
- A change made through one `struct` variable, is reflected in other variables that refer to the same Struct or Struct member. For example, changing the title of the first chapter (`vChapter`):

```
vChapter->Title = "New Title"
```

means that the following expressions all evaluate to `"New Title"`.

```
$1 = vTitle
$2 = vChapter->Title
$3 = vBook->chapter{1}->Title
; $1 == $2 == $3 = "New Title"
```

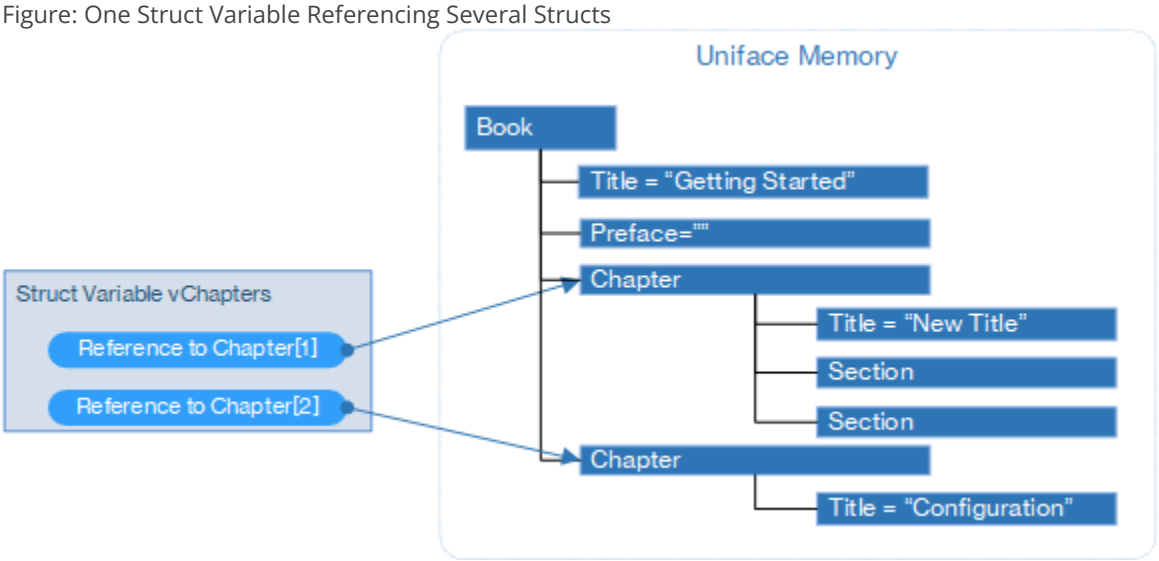
Figure: Changes Made to Struct Are Visible to All Variables That Point To It




- A single Struct variable can refer to multiple Structs, which makes it possible to manipulate a group of Structs, instead of handling them individually. For example, each `chapter` member of the `vBook` is itself a nested Struct. The following code results in `vStruct` containing a collection of Structs named `chapter`:

```
vChapters = vBook->chapter
```

Figure: One Struct Variable Referencing Several Structs




- A `struct` variable that contains a collection of only one reference to a Struct node can be handled as a single Struct rather than a collection. This means, for example, that you do not have to specify an index when accessing it. For example, there is only one Preface member in a book Struct, so you can use:

  ```
  vBook->preface
  ```

  instead of using the Struct index operator to specify a particular reference in the collection:

  ```
  vStruct{1}
  ```

- A `struct` variable can refer to zero Structs. In this case, the collection size (`$collsize`) is 0.
- A `struct` variable or expression whose value is NULL is interpreted as being a collection of references to zero Structs. For more information, see Null Values.
- The term `struct` variables also encompasses `struct` parameters and non-database fields. For more information, see Passing Struct Parameters.
- ProcScript variables and parameters of type `any` can also be used to refer to Structs, because Uniface implicitly converts the data type.

  ```
  variables
   any vAny
   struct vStruct
  endvariables
  vStruct = $newstruct ; Create a Struct
  vStruct->title = "Installation Guide" ; Add a member
  vAny = vStruct ; Assign vStruct to vAny
  ```

  vAny now points to same Struct as `vStruct`.