
Subject: Reference counting

Posted by [Robbie](#) on Wed, 12 Sep 2007 06:32:43 GMT

[View Forum Message](#) <> [Reply to Message](#)

I am using reference counting to automatically free pointers which are no longer in use. I've come across a problem when I use save and restore. I find that I really need to recalculate the number of references after I restore an object. Is there any way to programatically determine the number of references to a pointer?

Thanks

Robbie

Example code attached:

```
pro refcount::GetProperty, VALUE=value
```

```
if (arg_present(value) and (n_elements(*self.value) gt 0)) then value  
= *self.value
```

```
end
```

```
pro refcount::SetProperty, VALUE=value
```

```
if (n_elements(value) gt 0) then *self.value = value
```

```
end
```

```
function refcount::Copy
```

```
obj =
```

```
obj_new('refcount',P_VALUE=self.pValue,P_REFERENCE=self.pReference)
```

```
*self.pReference++
```

```
return, obj
```

```
end
```

```
pro refcount::Cleanup
```

```
*self.pReference--
```

```
if (*self.pReference le 0) then begin
```

```
    ptr_free, self.pReference
```

```
    ptr_free, self.pValue
```

```
endif
```

```
end
```

```
function refcount::init, P_VALUE=pValue, P_REFERENCE=pReference,  
VALUE=value
```

```
if (n_elements(pValue) gt 0) then self.pValue = pValue $
```

```
else self.pValue = ptr_new(/ALLOCATE_HEAP)
```

```
if (n_elements(pReference) gt 0) then self.pReference = pReference $
```

```
else self.pReference = ptr_new(1)
```

```
if (n_elements(value) gt 0) then *self.pValue = value
```

```
return, 1b
```

```
end
```

```
pro refcount__define, struct
```

```
struct = {refcount, $
```

```
    pValue: ptr_new(), $
```

```
    pReference: ptr_new() $
```

```
}
```

```
end
```

```
pro refcount_example1  
a = obj_new('refcount',VALUE=indgen(100))  
b = a -> copy()  
obj_destroy, [a,b]  
help, /heap  
end
```

```
pro refcount_example2_restore  
restore  
obj_destroy, [a]  
help, a  
end
```

```
pro refcount_example2  
a = obj_new('refcount',VALUE=indgen(100))  
b = a -> copy()  
save, a  
obj_destroy, [a,b]  
help, a, b  
refcount_example2_restore  
help, /heap  
end
```
