Subject: Pointer Behavior Objects Vs Plain routines?
Posted by savoie on Wed, 11 Sep 2002 14:54:29 GMT
View Forum Message <> Reply to Message

Hey all,

O.k.  I'm looking at some pointer weridness.  Well, I'm calling it weirdness because I obviously don't understand something that is happening.  There are two examples below.

The first is just two routines.  test: creates a pointer, calls changePtr with a null pointer as an argument; and changePtr: which just assigns a string the the passedPtr.  This examples shows that if you pass a pointer to a procedure, assign something to that pointer, you can retrieve it after exit.

The rest of the routines are a simple object with a couple of methods, showing exactly the opposite effect.  When the object's CHANGEPTR method is called, self.myptr doesn't seem to be able to be changed on return.

So if someone here can please enlighten me, I would be very appreciative.
What's different between the object calls and the plain procedure calls?

Thanks

Matt Savoie
National Snow and Ice Data Center, Boulder, CO

p.s.  Just because I haven't had to ask questions here in a while, doesn't mean I haven't been getting them answered.  Google groups for this has saved my behind on many more than one occasion.  So additional thanks for everyone helping answer questions on this group.

```
;;  ------------------------------------------------------------ ----------
;; These are distilled as much as I could think of to produce the behavior
;; that I don't understand.  PRO CHANGEPTR, passedPtr

  passedPtr =  ptr_new('This is weird?')
END

PRO TEST
```

```
  ptr =  ptr_new()
  print,  ptr_valid(ptr)
  changePtr,  ptr
  print,  ptr_valid(ptr)
  print,  *ptr
END

 ;;-------------------------------------------------------- ------------

PRO WEIRD::DOIT,  ptrInside
  ptrInside =  ptr_new('Why can not I change this?')
END


PRO WEIRD::CHANGEPTR
  self -> doIt,  self.myptr
END

PRO WEIRD::SHOWME
  print,  ptr_valid(self.myptr)
  IF (ptr_valid(self.myptr)) THEN   print,  *self.myptr
END

FUNCTION INIT
  return,  1
END

PRO WEIRD__DEFINE
  objectClass = {weird, $
                myptr:ptr_new() $
               }
END


PRO TESTWEIRD

  w =  obj_new('weird')
  w -> showMe
  w -> ChangePtr
  w -> ShowMe

END
```