

Re: portable finalizer

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2007-07/msg00233.html>

- *From:* Tamas Papp <tkpapp@xxxxxxxxxx>
 - *Date:* Wed, 04 Jul 2007 10:40:58 +0200
-

Marco Gidde <marco.gidde@xxxxxxxxxx> writes:

Tamas Papp <tkpapp@xxxxxxxxxx> writes:

I am wrapping a void * cffi pointer and some extra information in a CLOS object. When the object is garbage-collected, I would like to have the memory location freed. I have found out how to do finalization in SBCL, but I want to make my code portable.

I have seen finalize implementations with #+ directives floating around, could somebody post a reasonably comprehensive version? Or is there a library to do this?

there is a small library out there called trivial-garbage with finalization code for several CL implementations.

Thanks Marco,

I also found that there is a cffi:finalize that does the thing I want.

Since this thread has drifted towards resources, I have a conceptual question. Suppose I have a resource (eg a Cairo context in cl-cairo2), is a pointer to a C structure.

```
(defclass foo () ((pointer :initform nil)))
```

The user has to be able to close the resource explicitly (eg in cl-cairo2, this indicates that he has finished drawing and the file can be closed), so there is a destroy method:

```
(defmethod destroy ((object foo))  
  (with-slots (pointer) object  
    (when pointer  
      (call_c_api_to_destroy pointer)  
      (setf pointer nil))))
```

Re: portable finalizer

But suppose that I also want to register a finalizer, and have the object destroyed when it is garbage collected. The documentation of `ffi:finalize` says

For portability reasons, function should not attempt to look at object by closing over it because, in some lisps, object will already have been garbage-collected and is therefore not accessible when function is invoked.

So how is the finalizer supposed to look at pointer to determine if the resource has already been freed explicitly with `destroy`?

Thanks,

Tamas

.