File-Local Variables

CDR 9, Version 1.0, DOI: 10.5281/zenodo.3414042

Didier Verna <didier@didierverna.net>

Copying

1 Motivation

The Common Lisp standard defines two special variables, `*package*` and `*readtable*`, that are treated in a special way: the functions `load` and `compile-file` establish a new dynamic binding for each of them, so that any modification to their value at load or compile time becomes local to the file being processed. It is this particular treatment of these variables that allows for `in-package` or `in-readtable` (from the `named-readtables` library) to essentially have a "file-local" effect.

The motivation for the present document is the claim that this behavior could be useful for other, user-defined variables, although there is currently no way to do so in standard Common Lisp.

2 Example

XFormat is a library that extends the capabilities of the standard `format` function by letting the user modify the set of available format directives and their behavior. At the heart of XFormat lies the notion of "format table". A format table stores the association between directive characters and their meaning. XFormat defines a special variable named `*format-table*` which holds the value of the current format table. Conceptually, `*format-table*` is very close to `*package*` or `*readtable*` in the sense that it centralizes the definition of a specific part of the behavior of a Common Lisp program.

XFormat provides constructs such as `with-format-table`, to temporarily and locally bind `*format-table*` to a specific value. XFormat also provides a macro called `in-format-table`, the intent of which is obviously to do something similar to `in-package` or `in-readtable`. However, it is currently impossible to define it properly in all situations because the variable `*format-table*` won't be restored to its former value after loading or compiling the file.

3 Proposal

An extension to the Common Lisp standard is not strictly required to achieve the desired goal. As a matter of fact, we have written a library called `asdf-flv` which does this for ASDF systems (see Appendix A [ASDF-FLV], page 3).

However, we still think it is worthwhile to have it implemented directly by Common Lisp vendors. In doing so, one would not depend on a specific system management utility to get the functionality, and moreover, such an extension is very cheap to implement (see below).

The API to the requested functionality can be as simple as providing a function called `make-variable-file-local` in whatever extension package a Common Lisp implementation happens to use (the suggested name is inspired from (X)Emacs's `make-variable-buffer-local` function).

`make-variable-file-local` *SYMBOL* [Function]

Make special variable *SYMBOL* behave in a file-local manner.

File-local variables behave like `*package*` and `*readtable*`: `load` and `compile-file` bind them to the values they held before loading or compiling the file.

The function above would `pushnew` the variable to an internal list of special variables. Assuming this list is called `*file-local-variables*`, the functions `load` and `compile-file` would in turn wrap their functionality within a call to `progv`, as follows:

```
(progv *file-local-variables* (mapcar #'symbol-value *file-local-variables*)
  #| do the loading or compiling |#)
```

Appendix A ASDF-FLV

For reference, below is the implementation of the `asdf-flv` library. A vendor-based implementation of our proposal would do something very similar.

```
(defvar *file-local-variables* ()
  "List of file-local special variables.")

(defun make-variable-file-local (symbol)
  "Make special variable named by SYMBOL have a file-local value."
  (pushnew symbol *file-local-variables*))

(defmethod asdf:perform :around
    ((operation asdf:load-op) (file asdf:cl-source-file))
  "Establish new dynamic bindings for file-local variables."
  (progv *file-local-variables*
      (mapcar #'symbol-value *file-local-variables*)
    (call-next-method)))

(defmethod asdf:perform :around
    ((operation asdf:compile-op) (file asdf:cl-source-file))
  "Establish new dynamic bindings for file-local variables."
  (progv *file-local-variables*
      (mapcar #'symbol-value *file-local-variables*)
    (call-next-method)))
```