file-types Manual

Max Rottenkolber

Sunday, 26 July 2015

Table of Contents

1 Using file-types

file-types is a simplistic approach to identify types of files based on their names (e.g. no magic numbers like UNIX's `file(1)`). It exposes a way to find the common *MIME type* of a file but also comes with a novel *file tag* system.

`file-mime` may be used to get the MIME type for a file e.g.:

```
(file-mime #P"foo.sh") ⇒ ("application" "x-sh")
```

`file-tags` on the other hand exposes access to the file tagging system. When invoked with a single filename argument it will return a list of keywords, each being an increasingly specific description of the file's type. If invoked with a keyword as the second parameter, `file-tags` will act as a predicate to test if the file has the tag designated by the supplied keyword.

```
(file-tags #P"test.lisp") ⇒ (:TEXT :COMMON-LISP)
(file-tags #P"Makefile") ⇒ (:TEXT :MAKE-FILE)

(and (file-tags #P"test.lisp" :text)
     (file-tags #P"Makefile" :text))
  ⇒ true
```

Example of using `file-tags`.

2 Extending the type and name databases

The file-types database is created at compile time. It is sourced from two Lisp files—`types.lisp` and `names.lisp`. The former contains a special parameter variable `*file-type-list*` which stores a list of `pathname-type` to file tag and MIME type assignments. In specific, filenames whose `pathname-type` are `equalp` to one of the `pathname-type` strings in an assignment will inherit the tags and MIME type of that assignment.

```
(defparameter *file-type-list*
  '(;; Text files
    (("txt")                    ; List of PATHNAME-TYPEs.
     :tags (:text)              ; Tags.
     :mime ("text" "plain"))))  ; MIME type as returned by FILE-MIME.
```

Example of a exemplary `types.lisp`.

Now the file tag definition format has one special rule: Multiple assignments to a single type are valid and the specified tags will be appended in the database (MIME types will be superseded). Consider an exended version of the example above.

```
(defparameter *file-type-list* '(

   ;; Text file class
   (("txt" "lisp" "asd" "html" "htm")
    :tags (:text)
    :mime ("text" "plain"))

   ;; HTML file class
   (("html" "htm")
    :tags (:hyper-text-markup-language)
    :mime ("text" "html"))

   ;; Lisp file class
   (("lisp" "asd")
    :tags (:common-lisp))

   ;; ASDF file class
   (("asd")
    :tags (:asdf-system-definition))

))
```

Using type classes.

We specify a set of `pathname-types` to designate plain text files. Further down we specialize on some of those types. For instance, `html` and `htm` get assigned their tag `:hyper-text-markup-language` and their correct MIME type. Then we define `lisp` and `asd` to be `:common-lisp` files but let them retain the plain text MIME type. Then we fan out further among the lisp files and append the `asdf-system-definition` tag to the `asd` type.

```
"txt"  ("text" "plain") (:TEXT)
"lisp" ("text" "plain") (:TEXT :COMMON-LISP)
"asd"  ("text" "plain") (:TEXT :COMMON-LISP :ASDF-SYSTEM-DEFINITION)
"html" ("text" "html")  (:TEXT :HYPER-TEXT-MARKUP-LANGUAGE)
"htm"  ("text" "html")  (:TEXT :HYPER-TEXT-MARKUP-LANGUAGE)
```

The resulting database.

The second file is `name.lisp`. It contains the special parameter variable `*file-name-list*` which stores simple `euqalp` mappings from

pathname-names to pathname-types. Some types of files share a conventional name but have no type suffix—for instance consider Makefiles. In this case file-types will try to match the pathname-name against the names in names.lisp and if successful continue with the type recommended.

```
(defparameter *file-name-list*
  '(;; Conventions
    ("README" "txt")
    ("Makefile" "mk")
    ;; Init files
    (".emacs" "el")
    (".clisprc" "lisp")
    (".sbclrc" "lisp")))
```

Exemplary contents of names.lisp.