

Re: Using Japanese and English strings, encodings

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2006-04/msg00638.html>

- *From:* Pascal Bourguignon <pjb@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 14 Apr 2006 19:56:25 +0200
-

"drrobot" <drrobot@xxxxxxxxx> writes:

[...]
I'm a bit surprised that Lisp macros have to evaluate always to a list or a single value; I was sure there was some sort of way of getting them to evaluate to multiple or zero values like the ordinary functions they (superficially) resemble.

Usually, you can return from a macro a side-effect-free form, like NIL, and it will be ignored at the "call" site.

```
(defconstant +debug+ nil)
(defmacro when-debug (&body body)
  (when +debug+ `(progn ,body)))

(macroexpand '(when-debug (print :debugging)))
--> NIL ;
T
```

Therefore, when you use it:

```
(progn
  (print 'working)
  (when-debug (print :debugging))
  (print 'hard!))
```

it expands to:

```
#+clisp (ext:expand-form '(progn
  (print 'working)
  (when-debug (print :debugging))
  (print 'hard!)))
-->
(PROGN (PRINT 'WORKING) NIL (PRINT 'HARD!)) ; T
^^^
```

but since evaluating NIL returns NIL with no side effect and it's

Re: Using Japanese and English strings, encodings

immediately ignored, the compiler doesn't generate code for it.

```
(disassemble (compile nil (lambda ()
  (print 'working)
  (when-debug (print :debugging))
  (print 'hard!))))
```

```
Disassembly of function NIL
(CONST 0) = WORKING
(CONST 1) = HARD!
0 required arguments
0 optional arguments
No rest parameter
No keyword parameters
7 byte-code instructions:
0 (CONST&PUSH 0) ; WORKING
1 (PUSH-UNBOUND 1)
3 (CALLS1 132) ; PRINT
5 (CONST&PUSH 1) ; HARD!
6 (PUSH-UNBOUND 1)
8 (CALLS1 132) ; PRINT
10 (SKIP&RET 1)
NIL
```

See? Nothing between the PRINT and HARD!

Now, since you `_imposed_` a usage pattern, the solution Common Lisp provide is the `_reader_` macros `#+` and `#-`; but of course, this means that you can only choose between the language when you `_read_` the `_source_` of your program.

If you want to be able to select the language at run-time, then you shouldn't impose this usage pattern, or if you insist, you must bear the cost: you'd have to write your own parser and loader, since you're not reading `s-expr` anymore.

[You could write a reader macro that would parse un-parenthesized tokens and build the form to select the language at run-time:

```
(print #L (eng "Hello") (jap "S'kao, z † Š"))
```

You could try, but the reader macro function would need to look ahead to know when to stop reading `#L` forms, and since we can unread only one character on standard streams, we'd be stuck with such forms:

```
(print #L (eng "Hello") (jap "S'kao, z † Š") output-stream) ]
```

As a bit of background may clarify why I wanted to do this. I'm an american grad student in Japan right now, and I often find myself

Re: Using Japanese and English strings, encodings

writing documents twice, once in english, and once in japanese, over and over again, and getting annoyed at the constant back-and-forth revisions required to both documents.

I found it a useful time-saver to write all my research reports as Lisp source file, and use some scripts I wrote a while ago to transform the lisp source file into something else (either LaTeX document or HTML/XML) using some very, very simple macros and some perl scripts from before I came to Japan.

Here's the start of the source of my tri-lingual CV:

```
(defparameter pascal-bourguignon
  (quote
    (:resume
     (:person
      (:name "Pascal BOURGUIGNON")
      (:nationality (:text (:en "(French)") (:fr "(Français)") (:es "(Frances)"))))
      (:birth-date 2005 3 15)
      (:birth-place "Hayange, France")
      (:address "Apartado de correo 69 "
                "30380 La Manga del Mar Menor "
                "España")
      (:mail "pjb@xxxxxxxxxxxxxxxxxxxx")
      (:web "www.informatimago.com")
      (:phone "+34 968 140 492"))
      (:summary
       (:text (:en "Common Lisp Application Development "
                  "& Web application development (UncommonWeb).")
              (:fr "Développement d'applications en Common Lisp "
                  "& applications Web (UncommonWeb).")
              (:es "Desarrollo de aplicaciones Common Lisp "
                  "y aplicaciones Web (UncommonWeb)."))
       (:text (:en "OpenStep (MacOSX or GNUstep) & WebObject (GNUstepWeb) "
                  "Application Development")
              (:fr "Développement d'applications OpenStep (MacOSX ou GNUstep) "
                  "& WebObject (GNUstepWeb)")
              (:es "Desarrollo de aplicaciones OpenStep (MacOSX o GNUstep) "
                  "y WebObject (GNUstepWeb)."))
       (:text (:en "UNIX System & Application Development.")
              (:fr "Développement d'applications & développement système UNIX.")
              (:es "Desarrollo de aplicaciones y sistema UNIX.))
       (:text (:en "Internet UNIX Server Administration.")
              (:fr "Administration de serveurs Internet UNIX.")
              (:es "Administración de servidores Internet UNIX.))))
      (:skills
       (:category
        (:text (:en "Compilers; "
                  "Operating Systems; "
```

Re: Using Japanese and English strings, encodings

```
"Object-Oriented Development.")
(:fr "Compilateurs; "
"Systemes d'exploitation; "
"Développement orienté-objet.")
(:es "Desarrollo de compiladores; "
"Desarrollo Orientado a Objetos; "
"Sistemas Operativos.))
(:category (:text (:en "Operating systems: ")
(:fr "Systemes d'exploitation : ")
(:es "Sistemas operativos : "))
(:list
"UNIX (Linux, BSD, MacOSX, Solaris) "
"MacOS "
(:text (:en "Programming (system & applications);")
(:fr "Programmation (système & applications); ")
(:es "Programación (Sistema & aplicaciones); "))
(:text (:en "Unix system administration.")
(:fr "Administration de systemes Unix.")
(:es "Administración de sistemas Unix.))
(:category (:text (:en "Internet:") (:fr "Internet :") (:es "Internet :"))
(:flow
"SMTP"
"POP3"
"IMAP"
"HTTP/HTML/CGI"
"FTP"
"DNS"
"NFS" "..."))
(:category (:text (:en "Programming Languages: ")
(:fr "Langages de programmation : ")
(:es "Lenguajes de programación : "))
(:flow
"Common-Lisp"
"emacs-lisp"
"Smalltalk"
"Java"
"C/C++"
(:text (:en "& other OO, procedural & scripting languages.")
(:fr "& autres langages OO ou procéduraux.")
(:es "& otros lenguajes OO, procedural y de script.))
(:category (:text (:en "Development environment: ")
(:fr "Environnement de développement : ")
(:es "Entornos de desarrollo : "))
(:list
(:text (:en "UNIX: emacs, MacOSX/GNUstep development tools, "
"Standard UNIX & GNU development tools.")
(:fr "UNIX: emacs, outils de développement MacOSX "
"(Xcode) & GNUstep (Gorm); "
"Outils de développement standard UNIX & GNU.")
(:es "UNIX: emacs, Herramientas de desarrollo MacOSX "
"(Xcode) y GNUstep (Gorm); "
```

Re: Using Japanese and English strings, encodings

```
"Herramientas de desarrollo UNIX & GNU."))
(:text (:en "MacOS: CodeWarrior, MPW.")))
(:category (:text (:en "CASE Tools: ")
(:fr "Ateliers de génie logiciel : ")
(:es "Heramientas CASE : "))
(:list
(:text (:en "Objecteering UML (from Softeam SA); ")
(:fr "Objecteering UML (de Softeam SA); ")
(:es "Objecteering UML (de Softeam SA); "))
(:text (:en "ObjectTeam (from Cayenne Inc.); ")
(:fr "ObjectTeam (de Cayenne Inc.); ")
(:es "ObjectTeam (de Cayenne Inc.); "))
(:text (:en "OMTool (from Martin Marietta, Inc.). ")
(:fr "OMTool (de Martin Marietta, Inc.). ")
(:es "OMTool (de Martin Marietta, Inc.). ")))
(:category (:text (:en "Methodologies: ")
(:fr "Méthodologies : ")
(:es "Metodologias : "))
(:flow "UML" "OMT" "Booch"))
(:category (:text (:en "Databases: ")
(:fr "Bases de données : ")
(:es "Bases de datos : "))
(:flow "PostgreSQL" "MySQL" "Oracle" "Sybase")))
...)))
```

You can see the rest and the code I use to generate the HTML versions of my CV at <http://www.informatimago.com/cv.lisp>

It's rather Q&D, but it should give you some idea. You can also fetch the markup code Peter used to write "Practical Common Lisp", which could be more practical for less structured documents, merging the multi-lingual stuff with it.

--
__Pascal Bourguignon__ <http://www.informatimago.com/>

WARNING: This product attracts every other piece of matter in the universe, including the products of other manufacturers, with a force proportional to the product of the masses and inversely proportional to the distance between them.

.