READER

Programování v Common Lispu

Funkce `read`

- čte ze streamu
- výsledek zcela závisí na aktuální packagi
- varianta `read-from-string`

```
> (read-from-string "ahoj")
ahoj

> (read-from-string "(")
chyba "unexpected eof"
```

- syntax funkce `read`:

```
(read &optional stream eof-error-p eof-value recursive-p)
```

Funkce `read`

- čtení čísla: vrací číslo
- čtení symbolu: dívá se do aktuální package a nepřipustí duplicity
- dále makroznaky:

```
( ) ' ; " ` , #
```

- poslední je tzv. "dispatch macro character":

```
#' #B #O #X #nR #C #\ #A #( #* #S #P #|
#mezera #) #<
#+ #- #: #n= #n# #.
```

dispatch makroznaky

#+, #- podmíněné čtení podle proměnné `*features*` (implementačně závislé):

```
> *features*
(:LISPWORKS-PERSONAL-EDITION :COMMON-LISPWORKS
:LW-EDITOR :CAPI :COCOA :UNIX-WITHOUT-MOTIF :COMPILER
:ANSI-CL :COMMON-LISP :IEEE-FLOATING-POINT :LISPWORKS :CLOS
:UNICODE :UNIX :LISPWORKS-32BIT :MACOSX :LITTLE-ENDIAN ...)

> (progn #+unix(+ 1 1) #-unix(+ 2 2))
2 na unixu, 4 jinde
```

#: nový symbol bez package:

```
> (eql '#:a '#:a)
nil
```

dispatch makroznaky

#*n*=, #*n*# označení již přečteného a odkaz:

```
> '#1=(nil . #1#)
nekonečný seznam (nil nil nil ...)

> (loop for elem = (read stream nil #1=#:eof)
        while (not (eql elem #1#))
        collect elem)
seznam výrazů přečtených z stream
```

dispatch makroznaky

#. vyhodnotit během čtení:

```
(defvar *last-build-time* #.(get-universal-time))
```

kompilace objektu do souboru:

```
(defun store-obj (obj file)
  (let ((*store* obj))
    (declare (special *store*))
    (compile-file  *helper* :output-file file)
    obj))

(defun load-obj (file)
  (let (*load*)
    (declare (special *load*))
    (load file)
    *load*))

;; obsah souboru *helper*: (setf *load* '#.*store*)
```

lze potlačit proměnnou `*read-eval*`

Reader lze programovat

```
(defun single-quote-reader (stream char)
  (declare (ignore char))
  (list (quote quote) (read stream t nil t)))

(set-macro-character #\' #'single-quote-reader)
```

```
(defun semicolon-reader (stream char)
  (declare (ignore char))
  (read-line stream nil nil t)
  (values))

(set-macro-character #\; #'semicolon-reader)
```

Reader: poznámky

- package je v proměnné `*package*` (raději zatím nepoužívat)
- readtable je v proměnné `*readtable*`
- modifikovat vlastní kopii (`copy-readtable`)
- kdy programovat reader?