Próximo:Fechamentos, Anterior:Funções genéricas, Acima:Funções [Conteúdo][Índice]

13.9 Acessando o Conteúdo da Célula de Função

A *definição* de função de um símbolo é o objeto armazenado na célula de função do símbolo. As funções descritas aqui acessam, testam e configuram a célula de função de símbolos.

Veja também a função `indirect-function`. Consulte Definição de função indireta .

Função: símbolo-função *símbolo*

Isso retorna o objeto na célula de função do *símbolo* . Ele não verifica se o objeto retornado é uma função legítima.

Se a célula de função for nula, o valor de retorno será `nil`. Para distinguir entre uma célula de função que é nula e uma definida como `nil`, use `fboundp`(veja abaixo).

```
(barra de defunção (n) (+ n 2))
(símbolo-função 'barra)
     ⇒ (lambda (n) (+ n 2))
(fset 'baz' bar)
     ⇒ barra
(função-símbolo 'baz)
     ⇒ barra
```

Se você nunca deu a um símbolo nenhuma definição de função, dizemos que a célula de função desse símbolo é *void* . Em outras palavras, a célula de função não possui nenhum objeto Lisp. Se você tentar chamar o símbolo como uma função, o Emacs sinaliza um `void-function`erro.

Observe que void não é o mesmo que `nil`ou o símbolo `void`. Os símbolos `nil`e `void`são objetos Lisp e podem ser armazenados em uma célula de função assim como qualquer outro objeto pode ser (e podem ser funções válidas se você os definir com `defun`). Uma célula de função void não contém nenhum objeto.

Você pode testar a vacuidade da definição de função de um símbolo com `fboundp`. Depois de ter dado a um símbolo uma definição de função, você pode anulá-lo mais uma vez usando `fmakunbound`.

Função: *símbolo* fboundp

Esta função retorna `t`se o símbolo tiver um objeto em sua célula de função, `nil`caso contrário. Ele não verifica se o objeto é uma função legítima.

Função: *símbolo* fmakunbound

Esta função anula a célula de função do *símbolo* , de modo que uma tentativa subsequente de acessar esta célula causará um `void-function`erro. Ele retorna o *símbolo* . (Veja também `makunbound`, em Variáveis Vazias .)

```
(defun foo (x) x)
(foo 1)
     ⇒1
(fmakunbound 'foo)
     ⇒ fo
```

```
(foo 1)
error→ A definição da função do símbolo é void: foo
```

Função: definição do *símbolo* fset

Esta função armazena a *definição* na célula de função do *símbolo* . O resultado é a *definição* . Normalmente , a *definição* deve ser uma função ou o nome de uma função, mas isso não é verificado. O *símbolo* do argumento é um argumento avaliado comum.

O uso primário desta função é como uma sub-rotina por construções que definem ou alteram funções, como `defun`ou `advice-add`(veja Funções de Orientação). Você também pode usá-lo para dar a um símbolo uma definição de função que não é uma função, por exemplo, uma macro de teclado (consulte Macros de teclado):

```
;; Defina uma macro de teclado nomeada.
(fset 'kill-two-lines "\^u2\^k")
     ⇒ "\^u2\^k"
```

Se você deseja usar `fset`para criar um nome alternativo para uma função, considere usar `defalias`em vez disso. Consulte Definição de defalias .

Próximo:Fechamentos, Anterior:Funções genéricas, Acima:Funções [Conteúdo][Índice]