Próximo:Formulários de função, Anterior:Listas de classificação, Acima:Formulários [Conteúdo][Índice

]

10.2.4 Símbolo Função Indireção

Se o primeiro elemento da lista for um símbolo, a avaliação examina a célula de função do símbolo e usa seu conteúdo em vez do símbolo original. Se o conteúdo for outro símbolo, esse processo, chamado de *função de símbolo indireção* , é repetido até obter um não-símbolo. Consulte Nomes de Função , para obter mais informações sobre a indireção da função de símbolo.

Uma possível consequência desse processo é um loop infinito, caso a célula de função de um símbolo se refira ao mesmo símbolo. Caso contrário, eventualmente obtemos um não-símbolo, que deveria ser uma função ou outro objeto adequado.

Mais precisamente, devemos agora ter uma função Lisp (uma expressão lambda), uma função de código de byte, uma função primitiva, uma macro Lisp, um formulário especial ou um objeto de carregamento automático. Cada um desses tipos é um caso descrito em uma das seções a seguir. Se o objeto não for um desses tipos, o Emacs sinaliza um `invalid-function`erro.

O exemplo a seguir ilustra o processo de indireção do símbolo. Usamos `fset`para definir a célula de função de um símbolo e `symbol-function`obter o conteúdo da célula de função (consulte Células de função). Especificamente, armazenamos o símbolo `car`na célula de função de `first`, e o símbolo `first`na célula de função de `erste`.

```
;; Construa esta ligação de célula de função:
;; ------------- ----- ------- -------
;; | #<carro secundário> | <-- | carro | <-- | primeiro | <-- | outro |
;; ------------- ----- ------- -------
(símbolo-função 'carro)
     ⇒ #<carro secundário>
(fset 'primeiro' carro)
     ⇒ carro
(fset 'erste' primeiro)
     ⇒ primeiro
(erste '(1 2 3)); Chame a função referenciada por erste.
     ⇒ 1
```

Por outro lado, o exemplo a seguir chama uma função sem nenhuma indireção de função de símbolo, porque o primeiro elemento é uma função Lisp anônima, não um símbolo.

```
((lambda (arg) (anterior arg))
 '(1 2 3))
     ⇒ 1
```

A execução da função em si avalia seu corpo; isso envolve a indireção da função de símbolo ao chamar `erste`.

Este formulário é raramente usado e agora está obsoleto. Em vez disso, você deve escrevê-lo como:

```
(funcall (lambda (arg) (erste arg))
         '(1 2 3))
```

ou apenas

```
(deixe ((arg '(1 2 3))) (erste arg))
```

A função `indirect-function`interna fornece uma maneira fácil de executar a indireção da função de símbolo explicitamente.

Função: função de função indireta *e noerror opcional*

Esta função retorna o significado de *função* como uma função. Se a *função* for um símbolo, então ela encontra a definição da *função* da função e recomeça com esse valor. Se *a função* não for um símbolo, ela retornará a própria *função* .

Esta função retorna `nil`se o símbolo final for desvinculado. Ele sinaliza um `cyclic-function-indirection`erro se houver um loop na cadeia de símbolos.

O argumento opcional *noerror* é obsoleto, mantido para compatibilidade com versões anteriores e não tem efeito.

Aqui está como você pode definir `indirect-function`em Lisp:

```
(função indireta defun (função)
  (se (função de símbolo)
      (função indireta (função de função de símbolo))
    função))
```

Próximo:Formulários de função, Anterior:Listas de classificação, Acima:Formulários [Conteúdo][Índice

]