Próximo:Chamando funções, Anterior:Nomes de funções, Acima:Funções [Conteúdo][Índice]

13.4 Definindo Funções

Geralmente damos um nome a uma função quando ela é criada pela primeira vez. Isso é chamado *de definição de uma função* e geralmente fazemos isso com a `defun`macro. Esta seção também descreve outras maneiras de definir uma função.

Macro: defun *name args [doc] [declare] [interactive] body…*

`defun`é a maneira usual de definir novas funções Lisp. *Ele define o nome* do símbolo como uma função com argumentos da lista de argumentos *(* consulte Lista de argumentos) e formas de corpo fornecidas por *body* . Nem o *nome* nem *os argumentos* devem ser citados.

doc , se presente, deve ser uma string especificando a string de documentação da função (consulte Documentação da Função). *declare* , se presente, deve ser um `declare`formulário especificando metadados de função (consulte Declare Form). *interativo* , se presente, deve ser um `interactive`formulário especificando como a função deve ser chamada interativamente (consulte Chamada interativa).

O valor de retorno de `defun`é indefinido.

aqui estão alguns exemplos:

```
(defun foo () 5)
(foo)
     ⇒ 5


(barra defun (a &opcional b &rest c)
    (lista ab))
(barra 1 2 3 4 5)
     ⇒ (1 2 (3 4 5))
(barra 1)
     ⇒ (1 zero zero)
(Barra)
erro→ Número incorreto de argumentos.


(defun capitalizar para trás ()
  "Materialize a última letra da palavra no ponto."
  (interativo)
  (palavra invertida 1)
  (palavra avançada 1)
  (backward-char 1)
  (palavra 1 maiúscula))
```

Tenha cuidado para não redefinir funções existentes acidentalmente. `defun`redefine até mesmo funções primitivas, como `car` sem qualquer hesitação ou notificação. O Emacs não impede que você faça isso, porque a redefinição de uma função às vezes é feita deliberadamente, e não há como distinguir a redefinição deliberada da redefinição não intencional.

Função: definição de *nome* defalias & documento opcional

Esta função define o *nome* do símbolo como uma função, com definição de *definição* (que pode ser qualquer função Lisp válida). Seu valor de retorno é *indefinido* .

Se *doc* for non- `nil`, ele se tornará a documentação da função *name* . Caso contrário, qualquer documentação fornecida por *definição* é usada.

Internamente, `defalias`normalmente usa `fset`para definir a definição. Se *name* tiver uma `defalias-fset-function`propriedade, no entanto, o valor associado será usado como uma função para chamar no lugar de `fset`.

O local apropriado para usar `defalias`é onde um nome de função específico está sendo definido—especialmente onde esse nome aparece explicitamente no arquivo de origem que está sendo carregado. Isso porque `defalias`registra qual arquivo definiu a função, assim como `defun` (veja Descarregando).

Por outro lado, em programas que manipulam definições de funções para outros fins, é melhor usar `fset`, que não mantém esses registros. Consulte Células de Função .

Você não pode criar uma nova função primitiva com `defun`ou `defalias`, mas pode usá-los para alterar a definição da função de qualquer símbolo, mesmo um como `car`ou `x-popup-menu`cuja definição normal seja primitiva. No entanto, isso é arriscado: por exemplo, é quase impossível redefinir `car`sem quebrar completamente o Lisp. Redefinir uma função obscura, como `x-popup-menu`é menos perigoso, mas ainda pode não funcionar como você espera. Se houver chamadas para a primitiva do código C, elas chamam a definição C da primitiva diretamente, portanto, alterar a definição do símbolo não terá efeito sobre elas.

Veja também `defsubst`, que define uma função como `defun` e diz ao compilador Lisp para realizar a expansão inline nela. Consulte Funções embutidas .

Para indefinir um nome de função, use `fmakunbound`. Consulte Células de Função .

Próximo:Chamando funções, Anterior:Nomes de funções, Acima:Funções [Conteúdo][Índice]