Próximo:Propriedades do símbolo, Anterior:Definições, Acima:Símbolos [Conteúdo][Índice]

9.3 Criação e Internação de Símbolos

Para entender como os símbolos são criados no GNU Emacs Lisp, você deve saber como o Lisp os lê. O Lisp deve garantir que encontra o mesmo símbolo toda vez que lê o mesmo conjunto de caracteres. Não fazer isso causaria uma confusão completa.

Quando o leitor Lisp encontra um símbolo, ele lê todos os caracteres do nome. Em seguida, ele faz o hash desses caracteres para encontrar um índice em uma tabela chamada *obarray* . Hashing é um método eficiente de procurar algo. Por exemplo, em vez de pesquisar uma lista telefônica de capa a capa ao procurar Jan Jones, você começa com os J's e continua a partir daí. Essa é uma versão simples de hash. Cada elemento do obarray é um *balde* que contém todos os símbolos com um determinado código de hash; para procurar um determinado nome, basta procurar em todos os símbolos no bucket o código hash desse nome. (A mesma ideia é usada para tabelas gerais de hash do Emacs, mas elas são um tipo de dados diferente; veja Tabelas de Hash .)

Se for encontrado um símbolo com o nome desejado, o leitor usa esse símbolo. Se o obarray não contiver um símbolo com esse nome, o leitor cria um novo símbolo e o adiciona ao obarray. Encontrar ou adicionar um símbolo com um certo nome é chamado de *internamento* , e o símbolo é então chamado de *símbolo interno* .

Interning garante que cada obarray tenha apenas um símbolo com qualquer nome específico. Outros símbolos de mesmo nome podem existir, mas não no mesmo obarray. Assim, o leitor obtém os mesmos símbolos para os mesmos nomes, desde que continue lendo com a mesma obarray.

O internamento geralmente acontece automaticamente no leitor, mas às vezes outros programas precisam fazer isso. Por exemplo, depois que o `M-x`comando obtém o nome do comando como uma string usando o minibuffer, ele então interna a string, para obter o símbolo interno com esse nome.

Nenhum obarray contém todos os símbolos; na verdade, alguns símbolos não estão em nenhuma ordem. Eles são chamados *de símbolos não internados* . Um símbolo não internado tem as mesmas quatro células que outros símbolos; no entanto, a única maneira de obter acesso a ele é encontrando-o em algum outro objeto ou como o valor de uma variável.

Criar um símbolo não internado é útil na geração de código Lisp, porque um símbolo não internado usado como uma variável no código que você gera não pode entrar em conflito com nenhuma variável usada em outros programas Lisp.

No Emacs Lisp, um obarray é na verdade um vetor. Cada elemento do vetor é um balde; seu valor é um símbolo interno cujo nome tem hashes para esse bucket ou 0 se o bucket estiver vazio. Cada símbolo interno possui um link interno (invisível ao usuário) para o próximo símbolo no balde. Como esses links são invisíveis, não há como encontrar todos os símbolos em um obarray, exceto usando `mapatoms`(abaixo). A ordem dos símbolos em um bucket não é significativa.

Em um obarray vazio, cada elemento é 0, então você pode criar um obarray com . **Esta é a única maneira válida de criar um obarray.** Números primos como comprimentos tendem a resultar em um bom hash; comprimentos um a menos que uma potência de dois também são bons. `(make-vector` *`length`* `0)`

Não tente colocar símbolos em um obarray você mesmo. Isso não funciona - apenas `intern`pode inserir um símbolo em um obarray corretamente.

> **Nota do Common Lisp:** Ao contrário do Common Lisp, o Emacs Lisp não fornece um único símbolo em vários obarrays.

A maioria das funções abaixo recebe um nome e às vezes um obarray como argumentos. Um `wrong-type-argument`erro é sinalizado se o nome não for uma string ou se o obarray não for um vetor.

Função: símbolo nome- ***símbolo***

Esta função retorna a string que é o nome do *símbolo* . Por exemplo:

```
(nome-símbolo 'foo)
     ⇒ "foi"
```

Aviso: Alterar a string substituindo caracteres altera o nome do símbolo, mas não atualiza o obarray, então não faça isso!

Função: ***nome*** **do símbolo make**

Essa função retorna um símbolo recém-alocado e não interno cujo nome é *name* (que deve ser uma string). Seu valor e definição de função são void e sua lista de propriedades é `nil`. No exemplo abaixo, o valor de não `sym`é porque é um símbolo não interno distinto cujo nome também é '`eqfoofoo`'.

```
(setq sym (fazer-símbolo "foo"))
     ⇒ fo
(eq sym 'foo)
     ⇒ nada
```

Função: gensym ***& prefixo opcional***

Esta função retorna um símbolo usando `make-symbol`, cujo nome é feito anexando `gensym-counter`ao *prefixo* . O prefixo é padronizado para `"g"`.

Função: nome interno ***e obarray opcional***

Esta função retorna o símbolo interno cujo nome é *name* . Se não houver tal símbolo no obarray *obarray* , `intern` cria um novo, adiciona-o ao obarray e o retorna. Se *obarray* for omitido, o valor da variável global `obarray`será usado.

```
(setq sym (interno "foo"))
     ⇒ fo
(eq sym 'foo)
     ⇒ t

(setq sym1 (interno "foo" outro-obarray))
     ⇒ fo
(eq sym1 'foo)
     ⇒ nada
```

Nota do Common Lisp: No Common Lisp, você pode inserir um símbolo existente em um obarray. No Emacs Lisp, você não pode fazer isso, porque o argumento `intern`deve ser uma string, não um

símbolo.

Função: nome interno-soft *e obarray opcional*

Esta função retorna o símbolo em *obarray* cujo nome é *name* , ou `nil`se *o obarray* não possui nenhum símbolo com esse nome. Portanto, você pode usar `intern-soft`para testar se um símbolo com um determinado nome já está internado. Se *obarray* for omitido, o valor da variável global `obarray`será usado.

O nome do argumento também pode ser um símbolo; nesse caso, a função retorna *name* se *name* estiver internado no obarray especificado e, caso contrário, `nil`.

```
(intern-soft "frazzle"); Não existe tal símbolo.
     ⇒ nada
(fazer-símbolo "frazzle"); Crie um não internado.
     ⇒ fraquejar
(intern-soft "frazzle"); Esse não pode ser encontrado.
     ⇒ nada
(setq sym (interno "frazzle")); Crie um interno.
     ⇒ fraquejar
(intern-soft "frazzle"); Esse pode ser encontrado!
     ⇒ fraquejar
(eq sym 'frazzle); E é o mesmo.
     ⇒ t
```

Variável: obarray

Essa variável é o obarray padrão para uso por `interne read`.

Função: função mapatoms *e obarray opcional*

Esta função chama a *função* uma vez com cada símbolo no obarray *obarray* . Então ele retorna `nil`. Se *obarray* for omitido, o padrão será o valor de `obarray`, o obarray padrão para símbolos comuns.

```
(conjunto de contagem 0)
     ⇒ 0
(defun count-syms (s)
  (contagem setq (1+ contagem)))
     ⇒ contagem-sim
(mapatoms 'count-syms)
     ⇒ nada
contar
     ⇒ 1871
```

Veja `documentation`em Acessando a Documentação , para outro exemplo usando `mapatoms`.

Função: *símbolo unintern* obarray

Esta função exclui o *símbolo* do obarray *obarray* . Se `symbol`não estiver realmente no obarray, `unintern`não faz nada. Se *obarray* for `nil`, o obarray atual será usado.

Se você fornecer uma string em vez de um símbolo como *symbol* , ela representa um nome de símbolo. Em seguida, `unintern`exclui o símbolo (se houver) no obarray que tem esse nome. Se não houver tal símbolo, `unintern`não faz nada.

Se `unintern`excluir um símbolo, ele retornará `t`. Caso contrário, ele retorna `nil`.

Próximo:Propriedades do símbolo, Anterior:Definições, Acima:Símbolos [Conteúdo][Índice]