Próximo:Variáveis nulas, Anterior:Variáveis Constantes, Acima:Variáveis [Conteúdo][Índice]

12.3 Variáveis Locais

As variáveis globais têm valores que duram até serem substituídos explicitamente por novos valores. Às vezes é útil dar a uma variável um *valor local* — um valor que tem efeito apenas dentro de uma certa parte de um programa Lisp. Quando uma variável tem um valor local, dizemos que ela está *localmente vinculada* a esse valor e que é uma *variável local* .

Por exemplo, quando uma função é chamada, suas variáveis de argumento recebem valores locais, que são os argumentos reais fornecidos à chamada da função; essas ligações locais entram em vigor no corpo da função. Para dar outro exemplo, o `let`formulário especial estabelece explicitamente ligações locais para variáveis específicas, que têm efeito apenas dentro do corpo do `let`formulário.

Também falamos da *ligação global* , que é onde (conceitualmente) o valor global é mantido.

Estabelecer uma ligação local salva o valor anterior da variável (ou a falta de um). Dizemos que o valor anterior está *sombreado* . Tanto os valores globais quanto os locais podem ser obscurecidos. Se uma associação local estiver em vigor, usar `setq`na variável local armazena o valor especificado na associação local. Quando essa ligação local não está mais em vigor, o valor anteriormente sombreado (ou a falta de um) volta.

Uma variável pode ter mais de uma vinculação local por vez (por exemplo, se houver `let`formulários aninhados que vinculam a variável). A *ligação atual* é a ligação local que está realmente em vigor. Ele determina o valor retornado avaliando o símbolo da variável e é a ligação atuada por `setq`.

Para a maioria dos propósitos, você pode pensar na ligação atual como a ligação local mais interna ou a ligação global se não houver nenhuma ligação local. Para ser mais preciso, uma regra chamada regra de *escopo* determina onde em um programa uma ligação local entra em vigor. A regra de escopo padrão no Emacs Lisp é chamada *de escopo dinâmico* , que simplesmente afirma que a vinculação atual em qualquer ponto da execução de um programa é a vinculação criada mais recentemente para essa variável que ainda existe. Para obter detalhes sobre o escopo dinâmico e uma regra de escopo alternativa chamada *escopo léxico* , consulte Escopo variável .

Os formulários especiais `let`e `let*`existem para criar ligações locais:

Forma Especial: formas let *(encadernações…)…*

> Esse formulário especial configura associações locais para um determinado conjunto de variáveis, conforme especificado por *bindings* e, em seguida, avalia todos os *formulários* em ordem textual. Seu valor de retorno é o valor do último formulário em *forms* . As ligações locais configuradas por `let` entrarão em vigor apenas dentro do corpo de *formulários* .
>
> Cada uma das *ligações* é (i) um símbolo, caso em que esse símbolo está localmente ligado a `nil`; ou (ii) uma lista da forma , em cujo caso o *símbolo* está localmente ligado ao resultado da avaliação *da forma de valor* . Se *a forma de valor* for omitida, será usado. `(symbol value-form)`nil
>
> Todos os s *de forma de valor* nas *associações* são avaliados na ordem em que aparecem e *antes* de vincular qualquer um dos símbolos a eles. Aqui está um exemplo disso: zestá vinculado ao valor antigo de y, que é 2, não ao novo valor de y, que é 1.

```
(conjunto e 2)
     ⇒ 2


(seja ((y 1)
       (zy))
  (lista yz))
     ⇒ (1 2)
```

Por outro lado, a ordem das *ligações* não é especificada: no exemplo a seguir, 1 ou 2 podem ser impressos.

```
(deixe ((x 1)
       (x 2))
  (imprima x))
```

Portanto, evite vincular uma variável mais de uma vez em um único `let`formulário.

Formulário especial: formulários let* *(encadernações…)*…

Essa forma especial é como `let`, mas vincula cada variável logo após calcular seu valor local, antes de calcular o valor local para a próxima variável. Portanto, uma expressão em *associações* pode se referir aos símbolos anteriores vinculados neste `let*`formulário. Compare o exemplo a seguir com o exemplo acima para `let`.

```
(conjunto e 2)
     ⇒ 2


(let* ((y 1)
       (zy)); Use o valor recém-estabelecido de y.
  (lista yz))
     ⇒ (1 1)
```

Formulário especial: formulários letrec *(encadernações…)*…

Essa forma especial é como `let*`, mas todas as variáveis são vinculadas antes que qualquer um dos valores locais seja calculado. Os valores são então atribuídos às variáveis vinculadas localmente. Isso só é útil quando a associação léxica está em vigor e você deseja criar encerramentos que se referem a associações que, de outra forma, ainda não estariam em vigor ao usar `let*`.

Por exemplo, aqui está um encerramento que se remove de um gancho após ser executado uma vez:

```
(letrec ((hookfun (lambda ())
                  (mensagem "Executar uma vez")
                  (remover-gancho 'pós-gancho de comando-hookfun))))
  (add-hook 'pós-gancho de comando-hookfun))
```

Aqui está uma lista completa dos outros recursos que criam ligações locais:

- Chamadas de função (consulte Funções).
- Chamadas de macro (consulte Macros).
- `condition-case`(veja Erros).

As variáveis também podem ter ligações de buffer-local (veja Buffer-Local Variables); algumas variáveis têm ligações terminal-local (veja Múltiplos Terminais). Esses tipos de ligações funcionam como ligações locais comuns, mas são localizadas dependendo de onde você está no Emacs.

Opção do usuário: max-specpdl-size

Essa variável define o limite do número total de associações e `unwind-protect`limpezas de variáveis locais (consulte Limpando de Saídas Não Locais) que são permitidas antes que o Emacs sinalize um erro (com data `"Variable binding depth exceeds max-specpdl-size"`).

Este limite, com o erro associado quando ele é excedido, é uma maneira que Lisp evita a recursão infinita em uma função mal definida. `max-lisp-eval-depth`fornece outro limite na profundidade de aninhamento. Veja Eval .

O valor padrão é 1600. A entrada no depurador Lisp aumenta o valor, se houver pouco espaço sobrando, para garantir que o próprio depurador tenha espaço para execução.

Próximo:Variáveis nulas, Anterior:Variáveis Constantes, Acima:Variáveis [Conteúdo][Índice]