

Próximo:[Usando Ligação Lexical](#), Anterior:[Dicas de vinculação dinâmica](#), Acima:[Escopo Variável](#) [ [Conteúdo](#) ][[Índice](#)]

### 12.10.3 Ligação Lexical

A vinculação léxica foi introduzida no Emacs, como um recurso opcional, na versão 24.1. Esperamos que sua importância aumente com o tempo. A vinculação léxica abre muito mais oportunidades para otimização, portanto, os programas que a utilizam provavelmente serão executados mais rapidamente em versões futuras do Emacs. A vinculação léxica também é mais compatível com a simultaneidade, que foi adicionada ao Emacs na versão 26.1.

Uma variável vinculada lexicalmente tem *escopo léxico*, o que significa que qualquer referência à variável deve ser localizada textualmente dentro da construção de vinculação. Aqui está um exemplo (consulte [Using Lexical Binding](#), para saber como realmente habilitar a vinculação léxica):

```
(let ((x 1)); x é lexicalmente vinculado.
  (+ x 3))
  ⇒ 4

(defun getx()
  x); x é usado gratuitamente nesta função.

(let ((x 1)); x é lexicalmente vinculado.
  (getx))
error→ 0 valor do símbolo como variável é nulo: x
```

Aqui, a variável `x` não tem valor global. Quando está lexicalmente vinculado a um `let` formulário, pode ser usado nos limites textuais desse `let` formulário. Mas não pode ser usado de dentro de uma `getx` função chamada a partir do `let` formulário, pois a definição da função de `getx` ocorre fora do `let` próprio formulário.

Aqui está como a ligação léxica funciona. Cada construção de vinculação define um *ambiente léxico*, especificando as variáveis que estão vinculadas à construção e seus valores locais. Quando o avaliador Lisp deseja o valor atual de uma variável, ele procura primeiro no ambiente léxico; se a variável não for especificada lá, ela procura na célula de valor do símbolo, onde o valor dinâmico está armazenado.

(Internamente, o ambiente léxico é uma lista de pares símbolo-valor, com o elemento final na lista sendo o símbolo `t` em vez de uma célula cons. Essa lista pode ser passada como o segundo argumento para a `eval` função, a fim de especificar um ambiente léxico no qual avaliar um formulário. Consulte [Eval](#). A maioria dos programas Emacs Lisp, no entanto, não deve interagir diretamente com ambientes léxicos dessa maneira; apenas programas especializados como depuradores.)

As ligações léxicas têm extensão indefinida. Mesmo depois que uma construção de vinculação termina de ser executada, seu ambiente léxico pode ser “mantido” em objetos Lisp chamados *closures*. Um encerramento é criado quando você define uma função nomeada ou anônima com ligação léxica habilitada. Consulte [Fechamentos](#), para obter detalhes.

Quando um encerramento é chamado como uma função, qualquer referência de variável léxica dentro de sua definição usa o ambiente léxico retido. Aqui está um exemplo:

```
(defvar meu-ticker nil); Usaremos isso vinculado dinamicamente
                        ; variável para armazenar um fechamento.
```

```
(let ((x 0)) ; x é lexicalmente vinculado.
  (setq meu-ticker (lambda()
                     (conjunto x (1+ x))))))
⇒ (fechamento ((x . 0) t) ()
    (conjunto x (1+ x)))
```

```
(funcall meu-ticker)
⇒ 1
```

```
(funcall meu-ticker)
⇒ 2
```

```
(funcall meu-ticker)
⇒ 3
```

x; Observe que x não tem valor global.  
error→ 0 valor do símbolo como variável é nulo: x

A letvinculação define um ambiente léxico no qual a variável x é localmente vinculada a 0. Dentro dessa construção de vinculação, definimos uma expressão lambda que incrementa x em um e retorna o valor incrementado. Essa expressão lambda é automaticamente transformada em um encerramento, no qual o ambiente léxico vive mesmo após a saída da letconstrução de vinculação. Cada vez que avaliamos o fechamento, ele incrementa x, usando a ligação de x nesse ambiente léxico.

Observe que, diferentemente das variáveis dinâmicas que estão vinculadas ao próprio objeto símbolo, a relação entre variáveis lexicais e símbolos está presente apenas no interpretador (ou compilador). Portanto, funções que recebem um argumento de símbolo (como `symbol-value`, `boundp`, e `set`) só podem recuperar ou modificar a ligação dinâmica de uma variável (ou seja, o conteúdo da célula de valor de seu símbolo).

Próximo:[Usando Ligação Lexical](#), Anterior:[Dicas de vinculação dinâmica](#), Acima:[Escopo Variável](#) [ [Conteúdo](#) ][[Índice](#)]