Anterior:Avaliação, Acima:Avaliação [Conteúdo][Índice]

10.6 Avaliação Adiada e Preguiçosa

Às vezes é útil atrasar a avaliação de uma expressão, por exemplo, se você quiser evitar realizar um cálculo demorado se o resultado não for necessário no futuro do programa. ObaqueA biblioteca fornece as seguintes funções e macros para dar suporte a essa *avaliação adiada* :

Macro: *formas* de atraso de conversão …

Retorne um *thunk* para avaliar os *formulários* . Um thunk é um encerramento (consulte Closures) que herda o ambiente léxico da `thunk-delay`chamada. Usar esta macro requer `lexical-binding`.

Função: thunk-force *thunk*

Força a *conversão* para realizar a avaliação dos formulários especificados no `thunk-delay`arquivo que criou a conversão. O resultado da avaliação do último formulário é retornado. A *conversão* também “lembra” que foi forçada: Qualquer chamada adicional de `thunk-force`com a mesma *conversão* apenas retornará o mesmo resultado sem avaliar os formulários novamente.

Macro: thunk-let *(ligações…) formas…*

Essa macro é análoga, `let`mas cria ligações de variáveis “preguiçosas”. Qualquer ligação tem a forma . Ao contrário de , a avaliação de qualquer *forma de valor* é adiada até que a ligação do *símbolo* correspondente seja usada pela primeira vez ao avaliar as *formas* . Qualquer *forma de valor* é avaliada no máximo uma vez. Usar esta macro requer . `(symbol value-form)`letlexical-binding

Exemplo:

```
(defun f (número)
  (thunk-let ((número derivado
               (progn (mensagem "Calculando 1 mais 2 vezes %d") número)
                      (1+ (* 2 número)))))
    (se (> número 10)
        número derivado
      número)))


(f 5)
⇒ 5


(f 12)
-| Calculando 1 mais 2 vezes 12
⇒ 25
```

Por causa da natureza especial das variáveis de ligação lenta, é um erro defini-las (por exemplo, com `setq`).

Macro: thunk-let* *(ligações…) formas…*

Isso é semelhante, `thunk-let`mas qualquer expressão em *associações* pode se referir a associações anteriores neste `thunk-let*`formulário. Usar esta macro requer `lexical-binding`.

```
(thunk-let* ((x (prog2 (mensagem "Calculando x...")
                    (+ 1 1)
                  (mensagem "Concluído o cálculo x")))
             (y (prog2 (mensagem "Calculando y...")
                    (+ x 1)
                  (mensagem "Concluído o cálculo y")))
             (z (prog2 (mensagem "Calculando z...")
                    (+ e 1)
                  (mensagem "Cálculo z concluído")))
             (a (prog2 (mensagem "Calculando a...")
                    (+z 1)
                  (mensagem "Concluído o cálculo a"))))
  (*zx))

-| Calculando z...
-| Calculando v...
-| Calculando x...
-| Terminado o cálculo x
-| Terminado o cálculo y
-| Cálculo z concluído
⇒ 8
```

`thunk-let`e `thunk-let*`use thunks implicitamente: sua expansão cria símbolos auxiliares e os vincula a thunks envolvendo as expressões de ligação. Todas as referências às variáveis originais nos *formulários* do corpo são então substituídas por uma expressão que chama `thunk-force`com a variável auxiliar correspondente como argumento. Portanto, qualquer código usando `thunk-let`ou `thunk-let*`pode ser reescrito para usar thunks, mas em muitos casos usar essas macros resulta em um código melhor do que usar thunks explicitamente.

Anterior:Avaliação, Acima:Avaliação [Conteúdo][Índice]