Anterior:Avaliação durante a expansão, Acima:Problemas com macros [Conteúdo][Índice]

14.5.5 Quantas vezes a macro é expandida?

Ocasionalmente, os problemas resultam do fato de que uma chamada de macro é expandida toda vez que é avaliada em uma função interpretada, mas é expandida apenas uma vez (durante a compilação) para uma função compilada. Se a definição de macro tiver efeitos colaterais, eles funcionarão de maneira diferente dependendo de quantas vezes a macro for expandida.

Portanto, você deve evitar efeitos colaterais no cálculo da expansão da macro, a menos que realmente saiba o que está fazendo.

Um tipo especial de efeito colateral não pode ser evitado: construir objetos Lisp. Quase todas as expansões de macro incluem listas construídas; esse é o ponto principal da maioria das macros. Isso geralmente é seguro; há apenas um caso em que você deve ter cuidado: quando o objeto que você constrói faz parte de uma constante citada na expansão da macro.

Se a macro for expandida apenas uma vez, na compilação, o objeto será construído apenas uma vez, durante a compilação. Mas na execução interpretada, a macro é expandida toda vez que a chamada da macro é executada, e isso significa que um novo objeto é construído a cada vez.

Na maioria dos códigos Lisp limpos, essa diferença não importa. Só pode importar se você executar efeitos colaterais nos objetos construídos pela definição de macro. Assim, para evitar problemas, **evite efeitos colaterais em objetos construídos por definições de macro** . Aqui está um exemplo de como esses efeitos colaterais podem causar problemas:

```
(defmacro objeto vazio ()
  (list 'citação (cons nil nil))))



(defun inicializa (condição)
  (let ((objeto (objeto-vazio))))
    (se condição
        (condição do objeto setcar))
    objeto))
```

Se `initialize`for interpretado, uma nova lista `(nil)`será construída cada vez que `initialize`for chamado. Assim, nenhum efeito colateral sobrevive entre as chamadas. Se `initialize`for compilado, a macro `empty-object`é expandida durante a compilação, produzindo uma única constante `(nil)`que é reutilizada e alterada cada vez que `initialize`é chamada.

Uma maneira de evitar casos patológicos como esse é pensar nisso `empty-object`como um tipo engraçado de constante, não como uma construção de alocação de memória. Você não usaria `setcar`em uma constante como `'(nil)`, então, naturalmente, você também não a usaria `(empty-object)` .

Anterior:Avaliação durante a expansão, Acima:Problemas com macros [Conteúdo][Índice]