Próximo:Definindo Macros, Anterior:Expansão, Acima:Macros [Conteúdo][Índice]

14.3 Compilação de Macros e Byte

Você pode perguntar por que nos damos ao trabalho de calcular uma expansão para uma macro e depois avaliar a expansão. Por que não fazer com que o macrocorpo produza os resultados desejados diretamente? A razão tem a ver com a compilação.

Quando uma chamada de macro aparece em um programa Lisp sendo compilado, o compilador Lisp chama a definição de macro da mesma forma que o interpretador faria e recebe uma expansão. Mas em vez de avaliar essa expansão, ele compila a expansão como se ela tivesse aparecido diretamente no programa. Como resultado, o código compilado produz o valor e os efeitos colaterais pretendidos para a macro, mas é executado em velocidade compilada total. Isso não funcionaria se o corpo da macro calculasse o valor e os efeitos colaterais em si — eles seriam calculados em tempo de compilação, o que não é útil.

Para que a compilação de chamadas de macro funcione, as macros já devem estar definidas em Lisp quando as chamadas para elas forem compiladas. O compilador tem um recurso especial para ajudá-lo a fazer isso: se um arquivo que está sendo compilado contém um `defmacro`formulário, a macro é definida temporariamente para o restante da compilação desse arquivo.

A compilação de bytes de um arquivo também executa todas as `require`chamadas de nível superior no arquivo, para que você possa garantir que as definições de macro necessárias estejam disponíveis durante a compilação, exigindo os arquivos que as definem (consulte Recursos nomeados). Para evitar carregar os arquivos de definição de macro quando alguém *executa* o programa compilado, escreva `eval-when-compile`em torno das `require`chamadas (consulte Eval durante a compilação).