Próximo:Exemplos de captura, Acima:Saídas não locais [Conteúdo][Índice]

11.7.1 Saídas Não Locais Explícitas: `catchethrow`

A maioria das construções de controle afeta apenas o fluxo de controle dentro da própria construção. A função `throwé` a exceção a essa regra de execução normal do programa: ela executa uma saída não local a pedido. (Existem outras exceções, mas elas são apenas para tratamento de erros.) `throwé` usado dentro de um `catch`, e volta para aquele `catch`. Por exemplo:

```
(defun foo-outer ()
  (pegue 'foo
    (foo-interior)))

(defun foo-inner ()
  …
  (se x
      (jogue 'pé t))
  …)
```

O `throw`formulário, se executado, transfere o controle diretamente de volta para o `catch`, que retorna imediatamente. O código após o `throw`não é executado. O segundo argumento de `throw`é usado como o valor de retorno do `catch`.

A função `throw`encontra a correspondência `catch`com base no primeiro argumento: ela procura um `catch`cujo primeiro argumento é `eq`aquele especificado no arquivo `throw`. Se houver mais de um aplicável `catch`, o mais interno terá precedência. Assim, no exemplo acima, o `throw`especifica `fooe` o `catch`in `foo-outer`especifica o mesmo símbolo, de modo que `catch`é o aplicável (assumindo que não há outra correspondência `catch`entre).

A execução `throw`encerra todas as construções Lisp até a correspondência `catch`, incluindo chamadas de função. Quando construções de vinculação, como `let`ou chamadas de função, são encerradas dessa maneira, as vinculações são desvinculadas, assim como quando essas construções são encerradas normalmente (consulte Variáveis locais). Da mesma forma, `throw`restaura o buffer e a posição salvos por `save-excursion`(consulte Excursões), e o status de restrição salvo por `save-restriction`. Ele também executa todas as limpezas estabelecidas com o `unwind-protect`formulário especial quando sai desse formulário (consulte Limpezas).

A `throw`necessidade não aparece lexicalmente dentro do `catch` que ela salta. Ele também pode ser chamado de outra função chamada dentro do `catch`. Desde que `throw`ocorra cronologicamente após a entrada no `catch`, e cronologicamente antes da saída dele, ele tem acesso a esse `catch`. É por isso que `throw`pode ser usado em comandos como aqueles `exit-recursive-edit` que retornam ao loop de comandos do editor (consulte Edição recursiva).

> **Nota do Common Lisp:** A maioria das outras versões do Lisp, incluindo o Common Lisp, tem várias maneiras de transferir o controle de forma não sequencial: `return`, `return-from`, e `go`, por exemplo. O Emacs Lisp tem apenas `throw`. O`cl-lib`biblioteca fornece versões de alguns deles. Consulte Bloqueios e Saídas em Extensões Common Lisp .

Forma especial: *corpo da etiqueta de* captura …

`catch`estabelece um ponto de retorno para a `throw`função. O ponto de retorno é diferenciado de outros pontos de retorno por *tag* , que pode ser qualquer objeto Lisp, exceto `nil`. *A tag de* argumento é avaliada normalmente antes que o ponto de retorno seja estabelecido.

Com o ponto de retorno em vigor, `catch`avalia as formas do *corpo* em ordem textual. Se os formulários forem executados normalmente (sem erro ou saída não local), o valor do último formulário do corpo será retornado do arquivo `catch`.

Se a `throw`for executado durante a execução de *body* , especificando a mesma *tag* de valor , o `catch`formulário sai imediatamente; o valor que ele retorna é o que foi especificado como o segundo argumento de `throw`.

Função: *valor da tag* de lançamento

O objetivo do `throw`é retornar de um ponto de retorno previamente estabelecido com `catch`. *A tag* de argumento é usada para escolher entre os vários pontos de retorno existentes; deve ser `eq`para o valor especificado no arquivo `catch`. Se vários pontos de retorno corresponderem à *tag* , o mais interno será usado.

O valor do argumento é usado como o valor a ser retornado `catch`.

Se nenhum ponto de retorno estiver em vigor com tag *tag* , então um `no-catch` erro é sinalizado com dados . (`tag value`)

Próximo:Exemplos de captura, Acima:Saídas não locais [Conteúdo][Índice]