Próximo:Processamento de Erros, Acima:Erros [Conteúdo][Índice]

11.7.3.1 Como sinalizar um erro

Sinalizar um erro significa iniciar o processamento do erro. O processamento de erros normalmente aborta todo ou parte do programa em execução e retorna a um ponto configurado para lidar com o erro (consulte Processamento de erros). Aqui descrevemos como sinalizar um erro.

A maioria dos erros são sinalizados automaticamente dentro de primitivas Lisp que você chama para outros propósitos, como se você tentar pegar o CAR de um inteiro ou avançar um caractere no final do buffer. Você também pode sinalizar erros explicitamente com as funções `error`e `signal`.

Sair, que acontece quando o usuário digita `C-g`, não é considerado um erro, mas é tratado quase como um erro. Consulte Desistir .

Cada erro especifica uma mensagem de erro, de uma forma ou de outra. A mensagem deve indicar o que está errado (“Arquivo não existe”), não como as coisas deveriam ser (“Arquivo deve existir”). A convenção no Emacs Lisp é que as mensagens de erro devem começar com letra maiúscula, mas não devem terminar com nenhum tipo de pontuação.

Função: error *format-string &rest args*

Essa função sinaliza um erro com uma mensagem de erro construída aplicando `format-message`(consulte Formatando Strings) a *format-string* e *args* .

Esses exemplos mostram usos típicos de `error`:

```
(erro "Isso é um erro - tente outra coisa")
     error→ Isso é um erro -- tente outra coisa


(erro "Nome inválido `%s'" "A%%B")
     erro→ Nome inválido 'A%%B'
```

`error`funciona chamando `signal`com dois argumentos: o símbolo `error`de erro e uma lista contendo a string retornada por `format-message`.

Normalmente, o acento grave e o apóstrofo no formato se traduzem em aspas curvas correspondentes, por exemplo, `"Faltando `%s'"` pode resultar em `"Faltando 'foo'"` . Consulte Estilo de citação de texto , para saber como influenciar ou inibir essa tradução.

Aviso: Se você quiser usar sua própria string como uma mensagem de erro literalmente, não escreva apenas . Se a *string string* contiver `'(error` *`string`*`) %'`, '`', ou `''` pode ser reformatado, com resultados indesejáveis. Em vez disso, use . `(error "%s"` *`string`*`)`

Função: *dados de símbolo de erro de* sinal

Esta função sinaliza um erro nomeado por *error-symbol* . *Os dados* do argumento são uma lista de objetos Lisp adicionais relevantes para as circunstâncias do erro.

O argumento *error-symbol* deve ser um símbolo de *erro* — um símbolo definido com `define-error`. É assim que o Emacs Lisp classifica diferentes tipos de erros. Consulte Símbolos de erro , para obter uma descrição dos símbolos de erro, condições de erro e nomes de condições.

Se o erro não for tratado, os dois argumentos serão usados na impressão da mensagem de erro. Normalmente, essa mensagem de erro é fornecida pela `error-message`propriedade *error-symbol* . Se *data* for non- `nil`, isso será seguido por dois pontos e uma lista separada por vírgulas dos elementos não avaliados de *data* . Para `error`, a mensagem de erro é o CAR dos *dados* (que deve ser uma string). As subcategorias de `file-error`são tratadas especialmente.

O número e o significado dos objetos nos *dados* dependem do *error-symbol* . Por exemplo, com um `wrong-type-argument`erro, deve haver dois objetos na lista: um predicado que descreve o tipo esperado e o objeto que não se ajusta a esse tipo.

Tanto *o símbolo de erro* quanto os *dados* estão disponíveis para qualquer manipulador de erro que lide com o erro: `condition-case`vincula uma variável local a uma lista do formulário (consulte Manipulando Erros). (`error-symbol . data`)

A função `signal`nunca retorna.

```
(sinal 'número errado de argumentos '(xy))
     erro→ Número incorreto de argumentos: x, y


(sinal 'no-tal-error '("Minha condição de erro desconhecida"))
     erro→ erro peculiar: "Minha condição de erro desconhecida"
```

Função: string de formato de erro do usuário *e argumentos restantes*

Essa função se comporta exatamente como `error`, exceto que usa o símbolo de erro `user-error`em vez de `error`. Como o nome sugere, isso se destina a relatar erros por parte do usuário, em vez de erros no próprio código. Por exemplo, se você tentar usar o comando `Info-history-back`(`l`) para voltar além do início do seu histórico de navegação de informações, o Emacs sinaliza um arquivo `user-error`. Tais erros não causam entrada no depurador, mesmo quando `debug-on-error`não é `nil`. Consulte Depuração de erros .

Nota do Common Lisp: O Emacs Lisp não tem nada como o conceito do Common Lisp de erros contínuos.

Próximo:Processamento de Erros, Acima:Erros [Conteúdo][Índice]