Próximo:Conversão de Caso, Anterior:Formatando Strings, Acima:Strings e Personagens [Conteúdo][Índice]

## 4.8 Strings de Formato Personalizado

Às vezes, é útil permitir que usuários e programas Lisp controlem como determinado texto é gerado por meio de strings de controle de formato personalizado. Por exemplo, uma string de formato pode controlar como exibir o nome próprio, o sobrenome e o endereço de e-mail de alguém. Usando a função `format`descrita na seção anterior, a string de formato pode ser algo como `"%s %s <%s>"`. No entanto, essa abordagem rapidamente se torna impraticável, pois pode não ser claro qual caractere de especificação corresponde a qual parte da informação.

Uma string de formato mais conveniente para esses casos seria algo como `"%f %l <%e>"`, onde cada caractere de especificação carrega mais informações semânticas e pode ser facilmente reorganizado em relação a outros caracteres de especificação, tornando essas strings de formato mais facilmente personalizáveis pelo usuário.

A função `format-spec`descrita nesta seção executa uma função semelhante a `format`, exceto que opera em strings de controle de formato que usam caracteres de especificação arbitrários.

**Função: modelo de especificação de formato *spec-alist &opcional only-present***

Esta função retorna uma string produzida a partir do *modelo* de string de formato de acordo com as conversões especificadas em *spec-alist* , que é uma lista (consulte Listas de Associação ) do formato . Cada especificação no *modelo* será substituída por *substituição* ao formatar a string resultante. `(`*`letter . replacement`*`)%`*`letter`*

Os caracteres em *template* , além das especificações de formato, são copiados diretamente na saída, incluindo suas propriedades de texto, se houver. Quaisquer propriedades de texto das especificações de formato são copiadas para suas substituições.

O uso de um alist para especificar conversões dá origem a algumas propriedades úteis:

- Se *spec-alist* contiver mais chaves de *letras* exclusivas do que caracteres de especificação exclusivos em *template* , as chaves não utilizadas serão simplesmente ignoradas.
- Se *spec-alist* contiver mais de uma associação com a mesma *letra* , será usada a mais próxima do início da lista.
- Se o *template* contiver o mesmo caractere de especificação mais de uma vez, a mesma *substituição* encontrada em *spec-alist* será usada como base para todas as substituições desse caractere.
- A ordem das especificações no *template* não precisa corresponder à ordem das associações em *spec-alist* .

O argumento opcional *only-present* indica como lidar com caracteres de especificação no *template* que não são encontrados em *spec-alist* . Se for `nil`ou omitido, a função sinaliza um erro. Caso contrário, essas especificações de formato e quaisquer ocorrências de '`%%`' no *modelo* são deixados literalmente na saída, incluindo suas propriedades de texto, se houver.

A sintaxe das especificações de formato aceitas por `format-spec`é semelhante, mas não idêntica, àquela aceita por `format`. Em ambos os casos, uma especificação de formato é uma sequência de caracteres

começando com '%' e terminando com uma letra alfabética como 's'.

Ao contrário `format`de , que atribui significados específicos a um conjunto fixo de caracteres de especificação, `format-spec`aceita caracteres de especificação arbitrários e os trata todos igualmente. Por exemplo:

```
(setq my-site-info
      (lista (cons ?s nome do sistema)
            (cons ?t (tipo de sistema de nome de símbolo))
            (cons ?c configuração do sistema)
            (cons ?v emacs-versão)
            (cons ?e nome de invocação)
            (cons ?p (número para string (emacs-pid))))
            (contras ?um endereço de email do usuário)
            (contras ?n nome completo do usuário))))

(formato-especificação "%e %v (%c)" my-site-info)
     ⇒ "emacs 27.1 (x86_64-pc-linux-gnu)"

(especificação de formato "%n <%a>" my-site-info)
     ⇒ "Desenvolvedores Emacs <emacs-devel@gnu.org>"
```

Uma especificação de formato pode incluir qualquer número dos seguintes caracteres sinalizadores imediatamente após o '%' para modificar aspectos da substituição.

**'0'**

Este sinalizador faz com que qualquer preenchimento especificado pela largura consista em '0' caracteres em vez de espaços.

**'-'**

Esse sinalizador faz com que qualquer preenchimento especificado pela largura seja inserido à direita e não à esquerda.

**'<'**

Esse sinalizador faz com que a substituição seja truncada à esquerda para a largura especificada, se especificada.

**'>'**

Este sinalizador faz com que a substituição seja truncada à direita para a largura especificada, se especificada.

**'^'**

Este sinalizador converte o texto substituído em maiúsculas (consulte Conversão de maiúsculas e minúsculas ).

**'_'**

Este sinalizador converte o texto substituído em letras minúsculas (consulte Conversão de maiúsculas e minúsculas ).

O resultado do uso de sinalizadores contraditórios (por exemplo, maiúsculas e minúsculas) é indefinido.

Como é o caso de `format`, uma especificação de formato pode incluir uma largura, que é um número decimal que aparece após qualquer sinalizador. Se uma substituição contiver menos caracteres do que a largura especificada, ela será preenchida à esquerda:

```
(formato-especificação "% 8a é preenchido à esquerda com espaços"
             '((?a . "alfa")))
     ⇒ " alfa é preenchido à esquerda com espaços"
```

Aqui está um exemplo mais complicado que combina vários recursos mencionados acima:

```
(setq my-battery-info
      (lista (cons ?p "73"); Porcentagem
            (contras ?L "Bateria"); Status
            (cons ?t "2:23"); Tempo restante
            (cons ?c "24330"); Capacidade
            (cons ?r "10.6"))); Taxa de descarga

(especificação de formato "%>^-3L : %3p%% (%05t à esquerda)" my-battery-info)
     ⇒ "BAT: 73% (02:23 à esquerda)"

(especificação de formato "%>^-3L : %3p%% (%05t à esquerda)"
             (contras (contras ?L "AC")
                   minha-bateria-info))
     ⇒ "AC: 73% (02:23 à esquerda)"
```

Como os exemplos nesta seção ilustram, `format-spec`é frequentemente usado para formatar seletivamente uma variedade de diferentes informações. Isso é útil em programas que fornecem strings de formato personalizáveis pelo usuário, pois o usuário pode optar por formatar com uma sintaxe regular e em qualquer ordem desejada apenas um subconjunto das informações que o programa disponibiliza.

Próximo:Conversão de Caso, Anterior:Formatando Strings, Acima:Strings e Personagens [Conteúdo][Índice]