

Próximo:[Noções básicas de flutuação](#), Acima:[Números](#) [[Conteúdo](#)][[Índice](#)]

3.1 Noções básicas de inteiro

O leitor Lisp lê um inteiro como uma sequência não vazia de dígitos decimais com sinal inicial opcional e ponto final opcional.

```
1; 0 inteiro 1.
1. ; 0 inteiro 1.
+1 ; Também o inteiro 1.
-1 ; 0 inteiro -1.
0; 0 inteiro 0.
-0 ; 0 inteiro 0.
```

A sintaxe para inteiros em bases diferentes de 10 consiste em '#' seguido por uma indicação de base seguida por um ou mais dígitos. As indicações de base são 'b' para binário, 'o' para octal, 'x' para hexadecimal, e '*raiz* r' para raiz *raiz*. Portanto, '#b *inteiro*' lê *inteiro* em binário, e '# *raiz* r *inteiro*' lê *inteiro* em radix *radix*. Os valores permitidos de *radix* vão de 2 a 36, e os dígitos permitidos são os primeiros caracteres *radix* retirados de '0'-'9', 'A'-'Z'. As letras maiúsculas são ignoradas e não há sinal inicial ou ponto final. Por exemplo:

```
#b101100 ⇒ 44
#o54 ⇒ 44
#x2c ⇒ 44
#24r1k ⇒ 44
```

Para entender como várias funções funcionam em números inteiros, especialmente os operadores bit a bit (consulte [Operações bit a bit](#)), geralmente é útil visualizar os números em sua forma binária.

Em binário, o inteiro decimal 5 se parece com isso:

```
...000101
```

(As reticências '...' representa um número conceitualmente infinito de bits que correspondem ao bit inicial; aqui, um número infinito de 0 bits. Exemplos posteriores também usam esta '...' notação.)

O inteiro -1 fica assim:

```
...111111
```

-1 é representado como todos. (Isso é chamado de notação *de complemento de dois*.)

Subtrair 4 de -1 retorna o inteiro negativo -5. Em binário, o inteiro decimal 4 é 100. Consequentemente, -5 fica assim:

```
...111011
```

Muitas das funções descritas neste capítulo aceitam marcadores para argumentos no lugar de números. (Consulte [Marcadores](#) .) Como os argumentos reais para essas funções podem ser números ou marcadores, geralmente damos a esses argumentos o nome *de número ou marcador* . Quando o valor do argumento é um marcador, seu valor de posição é usado e seu buffer é ignorado.

No Emacs Lisp, os caracteres de texto são representados por números inteiros. Qualquer número inteiro entre zero e o valor de (`max-char`), inclusive, é considerado válido como caractere. Consulte [Códigos de caracteres](#) .

Os inteiros no Emacs Lisp não estão limitados ao tamanho da palavra de máquina. Sob o capô, porém, existem dois tipos de inteiros: os menores, chamados *fixnums* , e os maiores, chamados *bignums* . Embora o código Emacs Lisp normalmente não deva depender se um inteiro é um fixnum ou um bignum, as versões mais antigas do Emacs suportam apenas fixnums, algumas funções no Emacs ainda aceitam apenas fixnums, e o código Emacs Lisp mais antigo pode ter problemas quando recebe bignums. Por exemplo, enquanto o código Emacs Lisp mais antigo pode comparar com segurança inteiros para igualdade numérica com `eq`, a presença de bignums significa que os predicados de igualdade gostam `eq` e `=` agora devem ser usados para comparar inteiros.

O intervalo de valores para bignums é limitado pela quantidade de memória principal, pelas características da máquina, como o tamanho da palavra usada para representar o expoente de um bignum, e pela `integer-width` variável. Esses limites são geralmente muito mais generosos do que os limites para números fixos. Um bignum nunca é numericamente igual a um fixnum; O Emacs sempre representa um inteiro no intervalo de fixnum como um fixnum, não um bignum.

O intervalo de valores para um número fixo depende da máquina. O intervalo mínimo é -536.870.912 a 536.870.911 (30 bits; ou seja, -2^{29} a $2^{29} - 1$), mas muitas máquinas oferecem um intervalo mais amplo.

Variável: `most-positive-fixnum`

O valor desta variável é o maior inteiro “pequeno” que o Emacs Lisp pode manipular. Os valores típicos são $2^{29} - 1$ em plataformas de 32 bits e $2^{61} - 1$ em plataformas de 64 bits.

Variável: `most-negative-fixnum`

O valor desta variável é o menor inteiro numericamente “pequeno” que o Emacs Lisp pode manipular. É negativo. Os valores típicos são -2^{29} em plataformas de 32 bits e -2^{61} em plataformas de 64 bits.

Variável: `largura inteira`

O valor desta variável é um inteiro não negativo que controla se o Emacs sinaliza um erro de intervalo quando um inteiro grande seria calculado. Inteiros com valores absolutos menores que 2^n , onde n é o valor desta variável, não sinalizam erro de intervalo. As tentativas de criar números inteiros maiores normalmente sinalizam um erro de intervalo, embora possa não haver sinal se um número inteiro maior puder ser criado de forma barata. Definir essa variável como um número grande pode ser caro se um cálculo criar números inteiros enormes.

Próximo: [Noções básicas de flutuação](#), Acima: [Números](#) [[Conteúdo](#)][[Índice](#)]