或-nan（后面可能跟着一对圆括号，圆括号里面有一系列的字符）。A、E、F和G与a、e、f和g是等价的，区别仅在于使用大写字母（INF、INFINITY、NAN）。

- **支持宽字符**。C99的另一个特性是使用fprintf来输出宽字符。%lc转换说明用于输出一个宽字符，%ls用于输出一个由宽字符组成的字符串。
- **之前未定义的转换说明现在允许了**。在C89中，使用%le、%lE、%lf、%lg以及%lG的效果是未定义的。这些转换说明在C99中都是合法的（l长度修饰符被忽略）。

22.3.4 …printf 转换说明示例

现在来看一些示例。在前面章节中我们已经看过大量日常转换说明的例子了，所以下面将集中说明一些更高级的应用示例。与前面章节一样，这里将用 • 表示空格字符。

我们首先来看看标志作用于%d转换的效果（对其他转换的效果也是类似的）。表22-7的第一行显示了不带任何标志的%8d的效果。接下来的四行分别显示了带有标志-、+、空格以及0的效果（标志#从不用于%d）。剩下的几行显示了标志组合所产生的效果。

表22-7　标志作用于%d转换的效果

转换说明	对123应用转换说明的结果	对-123应用转换说明的结果
%8d	•••••123	••••-123
%-8d	123•••••	-123••••
%+8d	••••+123	••••-123
% 8d	•••••123	••••-123
%08d	00000123	-0000123
%-+8d	+123••••	-123••••
%- 8d	•123••••	-123••••
%+08d	+0000123	-0000123
% 08d	•0000123	-0000123

556

表22-8说明了标志#作用于o、x、X、g和G转换的效果。

表22-8　标志#的效果

转换说明	对123应用转换说明的结果	对123.0应用转换说明的结果
%8o	•••••173	
%#8o	••••0173	
%8x	••••••7b	
%#8x	••••0x7b	
%8X	••••••7B	
%#8X	••••0X7B	
%8g		•••••123
%#8g		•123.000
%8G		•••••123
%#8G		•123.000

在前面的章节中，表示数值时已经使用过最小字段宽度和精度了，所以这里不再给出更多的示例了，只在表22-9中给出最小字段宽度和精度作用于%s转换的效果。

表22-9 最小字段宽度和精度作用于转换%s的效果

转换说明	对"bogus"应用转换说明的结果	对"buzzword"应用转换说明的结果
%6s	•bogus	buzzword
%-6s	bogus•	buzzword
%.4s	bogu	buzz
%6.4s	••bogu	••buzz
%-6.4s	bogu••	buzz••

表22-10说明了%g转换如何以%e和%f的格式显示数。表中的所有数都用转换说明%.4g进行了书写。前两个数的指数至少为4，因此它们是按照%e的格式显示的。接下来的8个数是按照%f的格式显示的。最后两个数的指数小于-4，所以也用%e的格式进行显示。

表22-10 %g转换的示例

数	对数应用转换%.4g的结果
123456.	1.235e+05
12345.6	1.235e+04
1234.56	1235
123.456	123.5
12.3456	12.35
1.23456	1.235
0.123456	0.1235
0.0123456	0.01235
0.00123456	0.001235
0.000123456	0.0001235
0.0000123456	1.235e-05
0.00000123456	1.235e-06

557

过去，我们假设最小字段宽度和精度都是嵌在格式串中的常量。用字符*取代最小字段宽度或精度通常可以把它们作为格式串之后的实际参数加以指定。例如，下列printf函数的调用都产生相同的输出：

```
printf("%6.4d", i);
printf("%*.4d", 6, i);
printf("%6.*d", 4, i);
printf("%*.*d", 6, 4, i);
```

注意，为字符*填充的值刚好出现在待显示的值之前。顺便说一句，字符*的主要优势就是它允许使用宏来指定字段宽度或精度：

```
printf("%*d", WIDTH, i);
```

我们甚至可以在程序执行期间计算字段宽度或精度：

```
printf("%*d", page_width / num_cols, i);
```

最不常见的转换说明是%p和%n。%p转换允许显示指针的值：

```
printf("%p", (void *) ptr); /* displays value of ptr */
```

虽然在调试时%p偶尔有用，但它不是大多数程序员日常使用的特性。C标准没有指定用%p显示指针的形式，但很可能会以八进制或十六进制数的形式显示。

转换%n用来找出到目前为止由…printf函数调用所显示的字符数量。例如，在调用

```
printf("%d%n\n", 123, &len);
```

之后len的值将为3，因为在执行转换%n的时候printf函数已经显示3个字符（123）了。注意，在len前面必须要有&（因为%n要求指针），这样就不会显示len自身的值。