第14学时　使 用 模 块

你可能已经发现，Perl是一种非常灵活的编程语言。它能够处理文件、文本、数学运算、算法和任何计算机语言中通常遇到的其他问题。该编程语言的很大一部分是专门用于编写特定目的的函数的。正则表达式是该语言的核心部分，对于Perl的使用方法来说，它们非常重要，不过许多编程语言没有正则表达式照样能够很好运行。　Perl对外部程序（反引号、管道和system函数）的使用是非常广泛的，不过许多语言根本不使用它们。

编程员都希望尽可能将任何有用的特性纳入该语言的核心中。具有这样的包容性，就会形成一种规模很大并且难以使用的语言。例如，有些语言的设计者认为，支持对　world wide web访问的特性应该纳入该语言的核心中。这是个非常好的思路，但是并不是每个人都需要这个特性。如果10年后web不再像现在这样重要，那么就必须下决心去掉这个特性，许多已经编好的软件就会变得支离破碎。

Perl采取了一种不同的路子。从 Perl 5开始，可以使用“ 模块 ”对语言进行扩展。模块是Perl例程的集合，它使你能扩展 Perl的功能范围。你会发现这些模块能将 web浏览、图形处理、Windows OLE、数据库和几乎任何想像到的特性添加给 Perl。不过请记住，Perl的运行并不一定需要这些模块，没有这些模块它照样能够很好地发挥作用。

使用模块，你就能够访问一个很大的工作代码库，以帮助你编写程序。本书的第三部分将专门介绍如何使用Perl模块来编写CGI程序。

在撰写本书时，Perl已经包含3500个以上的模块，有20多个模块已经可以销售给用户。这些模块大多数可以免费转用。可以将这些模块用在你自己的程序中，以实现你想要得到的任何功能。你想解决的许多难题都可以为你解决，你只需安装正确的模块，并且正确地使用这些模块。

在本学时中，你将要学习下面的内容：

- 学习如何在你的Perl程序中使用模块。
- 简单地了解某些内置模块的情况。
- 了解Perl提供的核心模块的列表。

14.1　模块的概述

若要在你的Perl程序中使用模块，可以使用 Perl的use命令。例如，若要将 Cwd模块纳入你的程序，只需将类似下面的命令插入你的代码：

```
use Cwd;
```

将use Cwd放在代码中的什么位置，这并不重要，不过为了清楚起见和便于维护，它应该放在靠近程序顶部的位置。

这个特定模块曾经用在第 10学时中。不过在第 10学时中，你不知道它是如何工作的。当

你运行带有use Cwd的程序时，就会出现下列情况：

1) Perl解释程序打开你的程序并读入所有代码，直到use Cwd语句被找到。

2) 当你的Perl解释程序安装时，它将得到关于它的安装目录的通知。该目录被搜索，以便找出称为Cwd的模块，该模块是包含Perl代码的一个文件。

3) Perl读取该模块，该模块运行时需要的所有函数和变量均被初始化。

4) Perl解释程序从上次终止的位置开始，继续读取和编译你的程序。

这就是该程序运行的情况。当Perl读取整个程序后，并且在它准备运行时，该模块具备的所有功能就可以供你使用。

你可能注意到use strict与use Cwd很相似。为了避免概念的混乱， use语句是个通用指令，它可以使 Perl解释程序执行某项操作。如果使用 use strict，它会改变解释程序的运行特性，使之对引用和裸单词变得比较严谨，不过并不存在称为strict的模块。如果使用use Cwd，它将一个模块纳入你的程序。你不必过分担心它们之间的差别，差别很小，不会对你产生很大的影响。

当你将use Cwd插入你的程序中时，一个新函数就可以供你使用，这就是函数 cwd。cwd函数能够返回你的当前工作目录的名字。

14.1.1 读取关于模块的文档

所有Perl模块都配有它们自己的文档资料。事实上，如果你能够使用某个模块，那么就可以访问它的文档，因为文档往往嵌入模块之中。

若要查看模块的文档，请使用带有模块名的 perldoc程序。例如，若要查看 Cwd的文档，只需在操作系统的命令提示符处键入下面的命令：

```
perldoc Cwd
```

然后就可以每次显示1页文档。下面是一页示例文档，它作了一定的压缩：

```
Cwd(3)          13/Oct/98 (perl 5.005, patch 02)          Cwd(3)

     NAME
          getcwd - get pathname of current working directory

     SYNOPSIS
               use Cwd;
               $dir = cwd;
               use Cwd;
               $dir = getcwd;

               use Cwd;
               $dir = fastgetcwd;

DESCRIPTION
          The getcwd() function re-implements the getcwd(3) (or
          getwd(3)) functions in Perl.

          The abs_path() function takes a single argument and returns
          the absolute pathname for that argument. It uses the same
```

```
algorithm as getcwd(). (actually getcwd() is abs_path("."))
:
:
```

在这个例子中，Cwd模块实际上允许你使用3个新函数，即cwd、getcwd和fastgetcwd。如果你想使用这些函数，请阅读关于Cwd模块的文档。

如果你很想知道模块是如何工作的，就应该去了解它。模块主要是用Perl编写的，存放在系统的文件树中。Cwd模块存放在Cwd.pm文件中。该文件的位置可以是不一样的，不过它通常存放在Perl的安装目录下的某个位置中。变量@INC包含Cwd.pm的可能存放位置的名字，若要输出该变量，请在命令提示符处键入perl -v。

由于许多模块是其他Perl程序员免费提供的，所以模块文档的质量差异很大。比较主流的模块，即销售的标准模块，本书中提到的模块，以及流行的模块，如TK和LMP等，都配有很好的文档。如果你不知道模块是如何工作的，请查阅第16学时的内容，以了解有关的资料，或者干脆询问模块的作者。

14.1.2 什么地方可能出错

如果你的Perl安装正确，并且根本没有受到破坏，那么不应该出现任何错误。但是，世界并不是完美无缺的，有时某些地方仍会出错。

如果你看到下面这个出错消息：

```
syntax error in file XXXX at line YYY, next two tokens "use Cwd"
```

那么你应该检查安装的Perl的版本。请在系统的命令行提示符处键入下面的命令：

```
perl -v
```

如果Perl报告的版本号小于5，比如4.036，那么你拥有的Perl版本就太旧了，必须对它进行升级。Perl 5的许多特性它都没有，而且所有老的软件中都存在着安全隐患。实际上第13学时中的代码举例都无法在Perl 4中运行，现在你应该注意到这个问题了，请立即进行版本的升级。

另一个潜在的错误消息如下所示：

```
Can't locate Cwd.pm in @INC (@INC contains: path...path...path...)
BEGIN failed--compilation aborted
```

这种错误通常意味着存在下列3个问题中的一个：

- 模块的名字拼写有误。

模块的名字是区分大小写字母的。Use Cwd不同于use cwd。有些模块名包含冒号（：），比如File：：Find，你必须正确键入冒号。

- 要使用的模块不是标准产品的组成部分，它没有安装在系统的正确位置上。

安装的每个Perl均配有大约150个模块，这属于“标准产品”。本学时的后面部分内容中列出了其中的一些模块。所有这些模块都应该能够正确运行。你或你的系统管理员必须另外安装不是“标准产品”中的模块。

本书的附录包含如何安装这些额外模块的说明。

- 安装的Perl不完整，或者受到了破坏，也可能安装不正确。这种情况是经常发生的。

Perl解释程序会按照出错消息输出的 @INC中的路径查看已安装的模块。如果这些模块移动了位置，被删除，或者无法使用，最简单的解决办法是重新安装 Perl。在处理这个问题之前，首先要搞清出错的模块是否属于标准模块。已经安装的任何附加模块都可能放到其他位置中，这是正常的。关于如何在非标准位置上安装和使用模块的详细说明，请参见本书的附录。

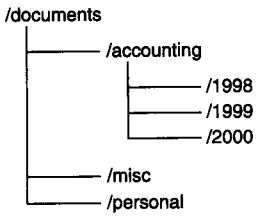
14.2 已安装模块简介

下面我们要简单介绍一下作为Perl核心产品的组成部分且已经安装在你的系统上的一些模块。

14.2.1 文件和目录简介

在第10学时中，我们介绍了如何打开目录并且读取这些目录中包含的文件名列表。接着，提出了如何读取子目录的问题，但是当时我们没有讲述这个问题。现在，我们就要说明如何遍历目录和子目录的方法。

你可以编写一个常用程序，以便在不知道文件所在的确切目录的情况下查找这个特定的文件。例如，你可能想在目录文档下的某个位置上查找名字为 important.doc的文件，如下所示：



这个插图显示了一个目录结构，它位于名字为 documents的父目录下。如果使用 opendir/readdir/closedir来查找documents下的某个位置上的一个文件，那是非常不容易的。首先，必须搜索documents来查找该文件。然后必须搜索 documents下的每个目录，即accounting、misc和personal，接着再搜索这些目录下的每个目录，如此等等。

这是过去30年来程序员一次又一次解决的一个老问题。如果你自己编写一个程序来解决这个问题，纯粹是浪费时间。为此，Perl的设计人员采取了一个简单的解决方案，加上了 File::Find模块。若要在你的程序中使用 File::Find模块，只需将下面这个命令输入到你的程序中的某个位置，最好是靠近程序的顶部：

```
use File::Find;
```

这时一个称为find的新函数就可以供你使用了。 find函数的句法如下所示：

```
find subref, dirlist
```

find函数的第二个参数是要搜索的一个目录列表。第一个参数对你来说是新的，它是个子例程引用。你创建的子例程引用很像是标量或数组的引用，它只不过是前面加上一个反斜杠的子例程名。必须使用子例程名前面的 &，才能取得子例程的引用。然后为 dirlist中找到的每个文件和目录调用指明的子例程。

程序清单14-1显示了为查找丢失的important.doc而使用的程序。

程序清单14-1 查找一个文件时所用的程序

```
1:   #!/usr/bin/perl -w
2:   use strict;
3:   use File::Find;
```

```
4:
5:    sub wanted {
6:          if ($_ eq "important.doc") {
7:                        print $File::Find::name;
8:          }
9:    }
10:   find \&wanted, '/documents';
```

第1~2行：这两行代码是Perl程序通常开始运行时的代码。 -w使警告特性激活， use strict用于捕获错误。

第3行：File：Find模块被插入你的程序。它使你可以运用 find函数。

第5行：为‘ /documents ’下的每个文件和目录调用该函数。如果你有 100个文件和12个目录，该例程将被调用112次。

第6行：当wanted()函数被调用时， $File ：： Find ：： name将包含到达被查看的当前文件的路径， $_只包含文件名。这行代码用于确定文件名是 importamt.doc。如果是，则输出全路径名。

第10行：用子例程引用\&wanted和一个目录调用find函数。并为‘ /documents ’下的每个文件和目录调用wanted()函数。

由find调用的函数将可以使用下列变量：

- $File ：： Find ：： name 当前路径名，目录和文件名。
- $File ：： Find ：： dir 当前目录名。
- $_ 当前文件名（不带目录）。有一点总是很重要，那就是你不能改变函数中的 $_的值。如果你改变了这个值，应该将它改回来。

程序清单14-2包含了另一个File ：： Find例子。这个例子用于删除C：和D：驱动器上带有扩展名.tmp的所有文件。这些文件在你的硬盘上不断增加并变得拥挤不堪。你可以很容易使用这个程序，从UNIX系统中删除文件，或者执行各种文件维护操作。

程序清单14-2 删除临时文件所用的程序

```
1:    #!/usr/bin/perl -w
2:    use strict;
3:    use File::Find;
4:
5:    sub wanted {
6:       # Check to see if the filename is not a directory
7:       if ( -f $File::Find::name ) {
8:               # Verify the filename eds in .tmp
9:               if ( $File::Find::name=~/\.tmp$/i) {
10:                      print "Removing $File::Find::name";
11:                       unlink $File::Find::name;
12:              }
13:      }
14:   }
15:   find(\&wanted, 'c:/', 'd:/');
```

程序清单14-2中的大多数程序与程序清单14-1中的程序相似。

第7行：对传递过来的文件名进行测试，以确保它是个正规文件。请记住，这个子例程将同时为文件和目录而调用。

第9~11行：对文件名进行核实，以了解文件名的结尾是否包含 .tmp。如果包含.tmp，则用

unlink将文件删除。

14.2.2 拷贝文件

另一个常见操作是拷贝文件，可以在Perl中使用下列步骤执行这项操作：

1) 打开源文件，以便读取该文件。

2) 打开目标文件，以便写入。

3) 读取源文件并写入目标文件。

4) 关闭源文件和目标文件。

当然，在执行每个操作步骤后，必须确保没有发生任何错误，并且每个写入操作均取得了成功。告诉你一个比较容易的方法， Perl提供了File::copy模块，它能进行文件的拷贝操作。下面是该模块的一个例子：

```
use File::Copy;
copy("sourcefile", "destination") || warn "Could not copy files: $!";
```

上面这个代码段用于将sourcefile的内容拷贝到destination。如果拷贝成功，copy函数返回1，如果拷贝出现问题，则返回0，并且它会将变量$!设置为相应的错误代码。

File::copy模块也提供了一个move函数。move函数能够将文件从一个目录移到另一个目录。如果可以通过对文件改名来移动文件，那么该文件将被改名。当源文件和目标文件在同一个文件系统或磁盘上时，通常采取改名的方法。如果无法通过对文件改名来移动文件，那么文件首先拷贝到目标文件名，然后将原始文件删除。请看下面这个例子：

```
use File::Copy;
if (not move("important.doc", "d:/archives/documents/important.doc")) {
    warn "important.doc could not be moved: $!";
    unlink "d:/archives/documents/important.doc";
}
```

在上面的代码段中，文件importont.doc从当前目录移到目标目录d:/archives/documents中。如果move函数运行失败，那么就可能存在不完整的目标文件。如果 move运行失败，unlink函数能够删除部分拷贝的目标文件。

14.2.3 用于通信的Perl模块

Perl模块的功能并不限于对文件和目录进行操作。还可以使用 Net::Ping模块来确定你的系统是否能够在网络上正确地进行通信。

Net::Ping模块是根据UNIX实用程序ping而得名的，而实用程序ping又是根据潜水艇利用声波来测定位置的“乒”声音而得名。 ping实用程序将一个数据包发送到网络上的另一个系统。如果该系统正在运行，那么它就会发出应答，同时 ping命令报告数据包发送成功。下面显示的Net::Ping的工作方式与上面完全相同：

```
use Net::Ping;

if ( pingecho("www.yahoo.com", 15) ) {
     print "Yahoo is on the network.";
} else {
     print "Yahoo is unreachable.";
}
```

在上面这个代码段中，Net::Ping模块提供了一函数叫做pingecho。该函数拥有两个参数，

第一个参数是要查找的主机，在上例中是 www.yahoo.com。第二个参数用于指明pingecho应该等待多长时间才能收到对方的应答，这个时间以秒为单位计算。

由于Perl在Windows 95/98／NT上运行时所具备的性质，在撰写本书时（1999年夏季） Net：：Ping模块还不能运行。 Net：：Ping需要依赖alarm函数，而在Windows下该函数不能运行。 Activestate是开发在Windows下运行的Perl的主要公司，它已宣布准备实现用于 Windows的许多遗漏的功能，并将这些修改纳入Perl。

14.2.4 使用English模块

使用English模块，Perl的某些无名的特殊变量将采用比较长的名字，如下例所示：

```
use English;

while(<>) {
     print $ARG;
}
```

在上面的代码段中，while(< >)通常从STDIN中读取一个行输入，并将它赋予 $_。现在它仍然如此。但是，使用 use English，变量$_也叫做$ARG。下面显示了特殊变量及其对应的英文变量的部分列表。

特 殊 变 量	英 文 名
$_	$ARG
@_	@ARG
$!	$OS_ERROR
$^O	$OSNAME
$0	$PROGRAM_NAME

若要了解特殊变量及其对应的英文变量的完整列表，请查看 English模块的在线文档。

14.2.5 diagnostics模块

Perl模块diagnostics能够帮助你查找程序中的错误。当你键入本书中的代码例子时， Perl解释程序肯定会发出你不太理解的出错消息。例如，请看下面这个短程序：

```
#!/usr/bin/perl -w

use strict;
print "For help, send mail to help@support.org\n";
```

它使Perl发出下面的警告消息：

```
In string, @support now must be written as \@support at line 4
Global symbol "@support" requires explicit package name at line 4
```

diagnostics模块会使Perl具体说明它的错误和警告。可以修改这个示例程序，使它像下面这样包含诊断模块：

```
#!/usr/bin/perl -w
use strict;
```

```
use diagnostics;

print "For help, send mail to help@support.com\n";
```

修改后的程序能够输出一个文字更详细的诊断消息：

```
In string, @support now must be written as \@support at line 4
Global symbol "@support" requires explicit package name at ./diag.pl line 5 (#1)

    (F) You've said "use strict vars", which indicates that all variables
    must either be lexically scoped (using "my"), or explicitly qualified to
    say which package the global variable is in (using "::").
```

如果认真观察一下这两个消息，就会发现它们之间有着明显的关系。第一条消息的意思很清楚，Perl要求你的电子邮件地址应该写成 help\@support.com。第二条消息经过解释，变得更加清楚了一些。由于 use strict是有效的，@support变量应该已经用 my作了声明。但是@support不是个变量，它是电子邮件地址的一部分，不过它被 Perl转换错了。

该消息前面的字母用于指明你遇到的是何种类型的错误。（W）表示是个警告，（D）表示你使用了一个不该使用的语句，（S）是个严重警告，（F)表示这是个致命的错误。除了（ F）之外的所有消息类型，你的Perl程序都会继续运行。

Perl共有60页用于描述它的出错消息。如果你在理解 Perl的简要出错消息时遇到了问题，use diagnostics有时能够帮助你理解出错消息的含义。

通过浏览perldiag在线手册页，就可以看到出错消息和诊断消息的完整列表。

14.3 标准模块的完整列表

关于Perl中包含的模块的完整列表，本书将不作详细的说明。下面是标准 Perl产品中的模块列表及其简单的说明。如果想知道模块的作用以及它如何运行，请使用 perldoc，以查看该模块的文档资料。

模 块 名	说 明
AutoLoader	允许Perl只在需要时对函数进行编译
AutoSplit	对模块进行分割，以便自动加载
Benchmark	允许对Perl函重复定时，以便加速基准测试
CGI	允许非常容易地访问用于Web编程的Common Gateway Interface (公用网关接口，第17~24学时介绍)
CPAN	用于访问Perl模块的存档文件，以便安装新模块
Carp	生成出错消息
DirHandle	提供与目录句柄之间的对象接口
Env	将操作系统的环境映射到变量中
Exporter	允许你编写自己的模块
ExtUtils∷*	允许你编写自己的模块或者安装模块
File∷*	提供更多的文件操作模块，如File∷Copy
File∷Spec∷*	允许对文件名进行跨操作系统的操作
FileCache	打开的文件数量可以超过操作系统通常允许的数量
FindBin	找出当前正在运行的程序的名字
Getopt∷*	允许你处理程序中的命令行选项

(续)

模 块 名	说 明
I18N∷Collate	允许按特定语言排序
IPC∷*	用于进程间的通信，比如使用双通或三通管道进行通信
Math∷*	允许你使用带有任意精度浮点数、整数和复数的扩展数学运算库
Net∷*	允许你获得关于网络主机的信息。例如， Net∷hostent可将IP地址（如204.71.200.68）转换成主机名（如www.Yahoo.com）
Pod∷*	用于访问Perl的Plain Old Documentation格式化例程
Symbol	允许你对Perl自己的符号表进行查看和操作
Sys∷Hostname	用于获取你的系统的IP主机名
Sys∷Syslog	允许将信息写入UNIX系统的出错记录
Term∷*	为光标位置和清屏等提供终端控制的函数接口
Text∷Abbrev	创建缩写表
Text∷ParseWords	允许对文本进行分析，以便搜索单词
Text∷Soundex	使用Soundex方法，根据标点对单词进行分类
Tie∷*	将Perl的变量与函数连接起来，使你可以实现自己的数组和哈希结构
Time∷*	允许对时间进行分析和处理。例如，你可以将“ Sat Jul 24 16:21:38 EDT 1999 ”这种格式的时间转换成1970年1月1日以来的秒数
constant	允许定义常量值
integer	使Perl有时能够用整数而不是浮点数进行数学运算
Locale	允许进行基于语言的字符串比较（各国语言字符的字符串比较）

下一步进行的操作

如果你想免费了解能使用哪些种类的模块，请使用 Web浏览器，以便访问网址 http：//www.cpan.org。模块按类别进行大致的排列。

有些模块需要C编译器和起码的开发环境来进行安装。在Windows计算机上可能没有这些模块。Activestate的Perl包含一个名叫PPM的实用程序，它可以用来浏览和安装预安装的模块。

本书的附录包含一个按步骤操作的说明，用于在UNIX和Windows计算机上安装模块。这些操作说明将告诉你如何使用CPAN模块（用于UNIX）和Activestate的用于安装新模块的PPM实用程序。

14.4 课时小结

在本学时中，我们介绍了如何使用模块来扩展Perl语言的功能，以便执行许多其他的任务。这种将新功能添加给Perl的通用方法将在本书的其他学时内容中广泛使用。另外，本学时介绍了一些常用的模块，并且给出了标准Perl产品包含的模块的完整列表。

14.5 课外作业

14.5.1 专家答疑

问题：在File∷Find模块中，变量名中的双冒号（∷）表示什么？它是否与$File∷Find∷dir中的相同？

解答：Perl模块能够为变量名建立一些备用区域，称为名空间，这样，模块的全局变量名

与你自己的全局变量名就不会混淆在一起了。因此在Cwd模块中的全局变量将称为$Cwd::x。你的大多数全局变量实际上拥有$main::x这个全名，而不是简名$x。不过就目前来说，这并不重要。

问题：我有一台安装了Windows 95/98/NT的计算机，我想使用的模块无法通过Activestate的PPM实用程序进行安装。我应该如何安装该模块？

解答：很遗憾，CPAN的大多数模块要求你拥有完整的UNIX型开发环境，可以对模块进行编译和安装，这种环境在Windows计算机上很难安装。如果你能够非常方便地使用C编译器，则可以下载一个开发环境，创建你自己的模块，但是这样做并不容易。

问题：我有一个包含require而不是use的老式Perl程序。require的功能是什么？

解答：require语句与use相类似。由于Perl 4没有use关键字，它使用的是require。require语句可使解释程序查找一个库文件，并将它纳入你的程序，这个功能类似use的功能。但是它们之间的主要差别是：每当require语句在运行时（运行期），便执行require功能；而use命令则是在你的程序第一次加载（编译时）时执行其功能。

14.5.2 思考题

1) 如果你想在程序中两次使用cwd函数，应该使用use Cwd; 几次？
   a. 一次。
   b. cwd的每个实例使用一次，因此是两次。
   c. 一次也不用，因为cwd是个内置函数。
2) 什么模块能够为$_变量提供一个别名？
   a. LongVars
   b. English
   c. $_没有别名

14.5.3 解答

1) 答案是a。当你用use将模块插入你的程序后，它的所有函数均可供程序的其余部分使用。
2) 答案是b。use English使得$_可以使用别名$ARG。

14.5.4 实习

- 请翻到本书的附录，设法按照那里的说明，通过CPAN安装模块Bundle::LWP。对于第24学时中的代码例子来说，需要使用该组模块中的一个模块。