Perl 格式化输出 Perl 目录操作

Perl 文件操作

Perl 使用一种叫做文件句柄类型的变量来操作文件。

从文件读取或者写入数据需要使用文件句柄。

文件句柄(file handle)是一个I/O连接的名称。

Perl提供了三种文件句柄:STDIN,STDOUT,STDERR，分别代表标准输入、标准输出和标准出错输出。

Perl 中打开文件可以使用以下方式：

```
open FILEHANDLE, EXPR
open FILEHANDLE

sysopen FILEHANDLE, FILENAME, MODE, PERMS
sysopen FILEHANDLE, FILENAME, MODE
```

参数说明:

- FILEHANDLE：文件句柄，用于存放一个文件唯一标识符。
- EXPR：文件名及文件访问类型组成的表达式。
- MODE：文件访问类型。
- PERMS：访问权限位(permission bits)。

Open 函数

以下代码我们使用 open 函数以只读的方式(<)打开文件 file.txt：

```
open(DATA, "<file.txt");
```

<表示只读方式。

代码中的 DATA 为文件句柄用于读取文件，以下实例将打开文件并将文件内容输出：

实例

```
#!/usr/bin/perl
open(DATA, "<file.txt") or die "file.txt 文件无法打开, $!";
while(<DATA>){
print "$_";
}
```

以下代码以写入(>)的方式打开文件 file.txt:

```
open(DATA, ">file.txt") or die "file.txt 文件无法打开, $!";
```

>表示写入方式。

如果你需要以读写方式打开文件，可以在 > 或 < 字符前添加 + 号：

```
open(DATA, "+<file.txt"); or die "file.txt 文件无法打开, $!";
```

这种方式不会删除文件原来的内容，如果要删除，格式如下所示：

```
open DATA, "+>file.txt" or die "file.txt 文件无法打开, $!";
```

如果要向文件中追加数据，则在追加数据之前，只需要以追加方式打开文件即可：

```
open(DATA,">>file.txt") || die "file.txt 文件无法打开, $!";
```

>> 表示向现有文件的尾部追加数据，如果需要读取要追加的文件内容可以添加 + 号：

```
open(DATA,"+>>file.txt") || die "file.txt 文件无法打开, $!";
```

下表列出了不同的访问模式：

模式	描述
< 或 r	只读方式打开，将文件指针指向文件头。
> 或 w	写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之。
>> 或 a	写入方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建之。
+< 或 r+	读写方式打开，将文件指针指向文件头。
+> 或 w+	读写方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之。
+>> 或 a+	读写方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建之。

Sysopen函数

sysopen 函数类似于 open 函数，只是它们的参数形式不一样。

以下实例是以读写(+<filename)的方式打开文件：

```
sysopen(DATA, "file.txt", O_RDWR);
```

如果需要在更新文件前清空文件，则写法如下：

```
sysopen(DATA, "file.txt", O_RDWR|O_TRUNC );
```

你可以使用 O_CREAT 来创建一个新的文件， O_WRONLY 为只写模式， O_RDONLY 为只读模式。

The **PERMS** 参数为八进制属性值，表示文件创建后的权限，默认为 **0x666**。

下表列出了可能的模式值：

模式	描述
O_RDWR	读写方式打开，将文件指针指向文件头。
O_RDONLY	只读方式打开，将文件指针指向文件头。
O_WRONLY	写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之。
O_CREAT	创建文件
O_APPEND	追加文件
O_TRUNC	将文件大小截为零
O_EXCL	如果使用O_CREAT时文件存在,就返回错误信息,它可以测试文件是否存在
O_NONBLOCK	非阻塞I/O使我们的操作要么成功，要么立即返回错误，不被阻塞。

Close 函数

在文件使用完后，要关闭文件，以刷新与文件句柄相关联的输入输出缓冲区，关闭文件的语法如下：

```
close FILEHANDLE
close
```

FILEHANDLE 为指定的文件句柄，如果成功关闭则返回 true。

```
close(DATA) || die "无法关闭文件";
```

读写文件

向文件读写信息有以下几种不同的方式：

<FILEHANDL> 操作符

从打开的文件句柄读取信息的主要方法是 <FILEHANDLE> 操作符。在标量上下文中，它从文件句柄返回单一行。例如：

实例

```
#!/usr/bin/perl
print "菜鸟教程网址?\n";
$name = <STDIN>;
print "网址: $name\n";
```

以上程序执行后，会显示以下信息，我们输入网址后 print 语句就会输出：

当我们使用 <FILEHANDLE> 操作符时，它会返回文件句柄中每一行的列表，例如我们可以导入所有的行到数组中。

实现创建 import.txt 文件，内容如下：

```
$ cat import.txt
1
2
3
```

读取 import.txt 并将每一行放到 @lines 数组中：

实例

```
#!/usr/bin/perl
open(DATA,"<import.txt") or die "无法打开数据";
@lines = <DATA>;
print @lines; # 输出数组内容
close(DATA);
```

执行以上程序，输出结果为：

```
1
2
3
```

getc 函数

xgetc 函数从指定的 FILEHANDLE 返回单一的字符，如果没指定返回 STDIN：

```
getc FILEHANDLE
getc
```

如果发生错误，或在文件句柄在文件末尾，则返回 undef。

---

read 函数

read 函数用于从缓冲区的文件句柄读取信息。

这个函数用于从文件读取二进制数据。

```
read FILEHANDLE, SCALAR, LENGTH, OFFSET
read FILEHANDLE, SCALAR, LENGTH
```

参数说明:

- FILEHANDLE：文件句柄，用于存放一个文件唯一标识符。
- SCALAR：存贮结果，如果没有指定OFFSET，数据将放在SCALAR的开头。否则数据放在SCALAR中的OFFSET字节之后。
- LENGTH：读取的内容长度。
- OFFSET：偏移量。

如果读取成功返回读取的字节数，如果在文件结尾返回 0，如果发生错误返回 undef。

print 函数

对于所有从文件句柄中读取信息的函数，在后端主要的写入函数为 print：

```
print FILEHANDLE LIST
print LIST
print
```

利用文件句柄和 print 函数可以把程序运行的结果发给输出设备(STDOUT：标准输出)，例如：

```
print "Hello World!\n";
```

文件拷贝

以下实例我们将打开一个已存在的文件 file1.txt ，并读取它的每一行写入到文件 file2.txt 中：

实例

```
#!/usr/bin/perl
# 只读方式打开文件
open(DATA1, "<file1.txt");
# 打开新文件并写入
open(DATA2, ">file2.txt");
# 拷贝数据
while(<DATA1>)
{
print DATA2 $_;
}
close( DATA1 );
close( DATA2 );
```

文件重命名

以下实例，我们将已存在的文件 file1.txt 重命名为 file2.txt，指定的目录是在 /usr/runoob/test/ 下：

```
#!/usr/bin/perl

rename ("/usr/runoob/test/file1.txt", "/usr/runoob/test/file2.txt" );
```

函数 **renames** 只接受两个参数，只对已存在的文件进行重命名。

删除文件

以下实例我们演示了如何使用 **unlink** 函数来删除文件：

实例

```
#!/usr/bin/perl
unlink ("/usr/runoob/test/file1.txt");
```

指定文件位置

你可以使用 **tell** 函数来获取文件的位置，并通过使用 **seek** 函数来指定文件内的的位置：

tell 函数

tell 函数用于获取文件位置：

```
tell FILEHANDLE
tell
```

如果指定 FILEHANDLE 该函数返回文件指针的位置，以字节计。如果没有指定则返回默认选取的文件句柄。

seek 函数

seek()函数是通过文件句柄来移动文件读写指针的方式来读取或写入文件的，以字节为单位进行读取和写入：

```
seek FILEHANDLE, POSITION, WHENCE
```

参数说明:

- FILEHANDLE：文件句柄，用于存放一个文件唯一标识符。
- POSITION：表示文件句柄(读写位置指针)要移动的字节数。
- WHENCE：表示文件句柄(读写位置指针)开始移动时的起始位置，可以取的值为0、1、2；分别表示文件开头、当前位置和文件尾。

以下实例为从文件开头读取 256 个字节：

```
seek DATA, 256, 0;
```

文件信息

Perl 的文件操作也可以先测试文件是否存在，是否可读写等。

我们可以先创建 file1.txt 文件，内如如下：

```
$ cat file1.txt
www.runoob.com
```

实例

```
#/usr/bin/perl
my $file = "/usr/test/runoob/file1.txt";
my (@description, $size);
if (-e $file)
{
push @description, '是一个二进制文件' if (-B _);
push @description, '是一个socket(套接字)' if (-S _);
push @description, '是一个文本文件' if (-T _);
push @description, '是一个特殊块文件' if (-b _);
push @description, '是一个特殊字符文件' if (-c _);
push @description, '是一个目录' if (-d _);
push @description, '文件存在' if (-x _);
push @description, (($size = -s _)) ? "$size 字节" : '空';
print "$file 信息: ", join(', ',@description),"\n";
}
```

执行以上程序，输出结果为：

```
file1.txt 信息: 是一个文本文件, 15 字节
```

文件测试操作符如下表所示：

操作符	描述
-A	文件上一次被访问的时间(单位：天)
-B	是否为二进制文件
-C	文件的(inode)索引节点修改时间(单位：天)
-M	文件上一次被修改的时间(单位：天)
-O	文件被真实的UID所有
-R	文件或目录可以被真实的UID/GID读取
-S	为socket(套接字)
-T	是否为文本文件
-W	文件或目录可以被真实的UID/GID写入
-X	文件或目录可以被真实的UID/GID执行

-b	为block-special (特殊块)文件(如挂载磁盘)
-c	为character-special (特殊字符)文件(如I/O 设备)
-d	为目录
-e	文件或目录名存在
-f	为普通文件
-g	文件或目录具有setgid属性
-k	文件或目录设置了sticky位
-l	为符号链接
-o	文件被有效UID所有
-p	文件是命名管道(FIFO)
-r	文件可以被有效的UID/GID读取
-s	文件或目录存在且不为0(返回字节数)
-t	文件句柄为TTY(系统函数isatty()的返回结果；不能对文件名使用这个测试)
-u	文件或目录具有setuid属性
-w	文件可以被有效的UID/GID写入
-x	文件可以被有效的UID/GID执行
-z	文件存在，大小为0(目录恒为false)，即是否为空文件，

← Perl 格式化输出 Perl 目录操作 →