Perl Grabbag

Some useful bits'n'pieces that every Perl programmer should know

Strictness

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

# bestrict - demonstrating the effect of strictness.

use strict;

$message = "This is the message.\n";

print $message;
```

Results from bstrict ...

```
Global symbol "$message" requires explicit package name at bestrict line 7.
Global symbol "$message" requires explicit package name at bestrict line 9.
Execution of bestrict aborted due to compilation errors.
```

Using my To Fix bestrict

```
my $message = "This is the message.\n";
```

Maxim 8.1

Unless you have a really good reason not to, always switch on strictness at the top of your program

Perl One-Liners

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

```
$ perl -e 'use ExampleModule'
$ perl -e 'print "Hello from a Perl one-liner.\n";'
$ perl -e 'printf "%0.2f\n", 30000 * .12;'
```

```
$ perldoc -f printf
$ perldoc -f sprintf
```

Perl One-Liners: Equivalents

```
$ perl -ne 'print if /ctgaatagcc/;' embl.data
```

```
while ( <> )
{
    print if /ctgaatagcc/;
}
```

```
$ grep 'ctgaatagcc' embl.data
```

Perl One-Liners: More Options

```
$ perl -npe 'last if /\d{4}$/;' embl.data
```

```
while ( <> )
{
    last if /\d{4}$/;
}
continue {
    print $_;
}
```

```
$ grep -v '[0123456789][0123456789][0123456789][0123456789]$' embl.data
```

Running Other Programs From perl

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

# pinvoke - demonstrating the invocation of other programs
# from Perl.

use strict;

my $result = system( "ls -l p*" );

print "The result of the system call was as follows:\n$result\n";

$result = `ls -l p*`;

print "The result of the backticks call was as follows:\n$result\n";

$result = qx/ls -l p*/;

print "The result of the qx// call was as follows:\n$result\n";
```

Results from pinvoke ...

```
-rw-rw-r-- 1 barryp barryp 403 Aug 16 16:48 pinvoke
-rw-rw-r-- 1 barryp barryp 145 Aug 7 12:36 prepare_embl
-rw-rw-r-- 1 barryp barryp 422 Jul 22 15:10 private_scope
The result of the system call was as follows:
0
The result of the backticks call was as follows:
-rw-rw-r-- 1 barryp barryp 403 Aug 16 16:48 pinvoke
-rw-rw-r-- 1 barryp barryp 145 Aug 7 12:36 prepare_embl
-rw-rw-r-- 1 barryp barryp 422 Jul 22 15:10 private_scope

The result of the qx// call was as follows:
-rw-rw-r-- 1 barryp barryp 403 Aug 16 16:48 pinvoke
-rw-rw-r-- 1 barryp barryp 145 Aug 7 12:36 prepare_embl
-rw-rw-r-- 1 barryp barryp 422 Jul 22 15:10 private_scope
```

Recovering From Errors

```
my $first_filename = "itdoesnotexist.txt";

open FIRSTFILE, "$first_filename"
    or die "Could not open $first_filename. Aborting.\n";




eval {
    my $first_filename = "itdoesnotexist.txt";

    open FIRSTFILE, "$first_filename"
        or die "Could not open $first_filename. Aborting.\n";
};
if ( $@ )
{
    print "Calling eval produced this message: $@";
}
```

Maxim 8.2

Use eval to protect potentially erroneous code

Sorting

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

# sortexamples - how Perl's in-built sort subroutine works.

use strict;

my @sequences = qw( gctacataat attgttttta aattatattc cgatgcttgg );

print "Before sorting:\n\t-> @sequences\n";

my @sorted = sort @sequences;
my @reversed = sort { $b cmp $a } @sequences;
my @also_reversed = reverse sort @sequences;

print "Sorted order (default):\n\t-> @sorted\n";
print "Reversed order (using sort { \$b cmp \$a }):\n\t-> @reversed\n";
print "Reversed order (using reverse sort):\n\t-> @also_reversed\n";
```

Results from sortexamples ...

```
Before sorting:
        -> gctacataat attgttttta aattatattc cgatgcttgg
Sorted order (default):
        -> aattatattc attgttttta cgatgcttgg gctacataat
Reversed order (using sort { $b cmp $a }):
        -> gctacataat cgatgcttgg attgttttta aattatattc
Reversed order (using reverse sort):
        -> gctacataat cgatgcttgg attgttttta aattatattc
```

Another Sorting Example

```
my @chromosomes = qw( 17 5 13 21 1 2 22 15 );

print "Before sorting:\n\t-> @chromosomes\n";


@sorted = sort { $a <=> $b } @chromosomes;
@reversed = sort { $b <=> $a } @chromosomes;

print "Sorted order (using sort { \$a <=> \$b }):\n\t-> @sorted\n";
print "Reversed order (using sort { \$b <=> \$a }):\n\t-> @reversed\n";
```

And its results ...

```
Before sorting:
        -> 17 5 13 21 1 2 22 15
Sorted order (using sort { $a <=> $b }):
        -> 1 2 5 13 15 17 21 22
Reversed order (using sort { $b <=> $a }):
        -> 22 21 17 15 13 5 2 1
```

The sortfile Program

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

# sortfile - sort the lines in any file.

use strict;

my @the_file;

while ( <> )
{
    chomp;
    push @the_file, $_;
}

my @sorted_file = sort @the_file;

foreach my $line ( @sorted_file )
{
    print "$line\n";
}
```

Results from sortfile ...

```
Zap! Zoom! Bang! Bam!
Batman, look out!
Robin, behind you!
Aaaaah, it's the Riddler!


$ perl sortfile sort.data

Aaaaah, it's the Riddler!
Batman, look out!
Robin, behind you!
Zap! Zoom! Bang! Bam!


$ sort sort.data
```

Learning More About Sorting

```
$ perldoc -f sort
```

```
$ man sort
```

Maxim 8.3

Take the time to become familiar with the utilities included in the operating system

HERE Documents

```
Shotgun Sequencing

This is a relatively simple method of reading
a genome sequence. It is ''simple'' because
it does away with the need to locate
individual DNA fragments on a map before
they are sequenced.

The Shotgun Sequencing method relies on
powerful computers to assemble the finished
sequence.
```

Without HERE Documents

```
print "Shotgun Sequencing\n\n";
print "This is a relatively simple method of reading\n";
print "a genome sequence. It is ''simple'' because\n";
print "it does away with the need to locate\n";
print "individual DNA fragments on a map before\n";
print "they are sequenced.\n\n";
print "The Shotgun Sequencing method relies on\n";
print "powerful computers to assemble the finished\n";
print "sequence.\n";
```

With HERE Documents

```
my $shotgun_message = <<ENDSHOTMSG;
Shotgun Sequencing

This is a relatively simple method of reading
a genome sequence. It is ''simple'' because
it does away with the need to locate
individual DNA fragments on a map before
they are sequenced.

The Shotgun Sequencing method relies on
powerful computers to assemble the finished
sequence.
ENDSHOTMSG

print $shotgun_message;
```

Even Better HERE Documents

```
print <<ENDSHOTMSG;
Shotgun Sequencing

This is a relatively simple method of reading
a genome sequence. It is ''simple'' because
it does away with the need to locate
individual DNA fragments on a map before
they are sequenced.

The Shotgun Sequencing method relies on
powerful computers to assemble the finished
sequence.
ENDSHOTMSG
```

Where To From Here