PERL - FORMATS

Perl uses a writing template called a 'format' to output reports. To use the format feature of Perl, you have to define a format first and then you can use that format to write formatted data.

Define a Format

Following is the syntax to define a Perl format −

```
format FormatName =
fieldline
value_one, value_two, value_three
fieldline
value_one, value_two
.
```

Here **FormatName** represents the name of the format. The **fieldline** is the specific way, the data should be formatted. The values lines represent the values that will be entered into the field line. You end the format with a single period.

Next **fieldline** can contain any text or fieldholders. The fieldholders hold space for data that will be placed there at a later date. A fieldholder has the format −

```
@<<<<
```

This fieldholder is left-justified, with a field space of 5. You must count the @ sign and the < signs to know the number of spaces in the field. Other field holders include −

```
@>>>> right-justified
@|||| centered
@####.## numeric field holder
@* multiline field holder
```

An example format would be −

```
format EMPLOYEE =
===================================
@<<<<<<<<<<<<<<<<<<<<<< @<<
$name $age
@#####.##
$salary
===================================
.
```

In this example, $name would be written as left justify within 22 character spaces and after that age will be written in two spaces.

Using the Format

In order to invoke this format declaration, we would use the write keyword −

```
write EMPLOYEE;
```

The problem is that the format name is usually the name of an open file handle, and the write statement will send the output to this file handle. As we want the data sent to the STDOUT, we must associate EMPLOYEE with the STDOUT filehandle. First, however, we must make sure that that STDOUT is our selected file handle, using the select function.

```
select(STDOUT);
```

We would then associate EMPLOYEE with STDOUT by setting the new format name with STDOUT, using the special variable *or*FORMAT_NAME as follows −

```
$~ = "EMPLOYEE";
```

When we now do a write, the data would be sent to STDOUT. Remember: if you are going to write your report in any other file handle instead of STDOUT then you can use select function to select that file handle and rest of the logic will remain the same.

Let's take the following example. Here we have hard coded values just for showing the usage. In actual usage you will read values from a file or database to generate actual reports and you may need to write final report again into a file.

```
#!/usr/bin/perl

format EMPLOYEE =
====================================
@<<<<<<<<<<<<<<<<<<<<<<< @<<
$name $age
@#####.##
$salary
====================================
.

select(STDOUT);
$~ = EMPLOYEE;

@n = ("Ali", "Raza", "Jaffer");
@a  = (20,30, 40);
@s = (2000.00, 2500.00, 4000.000);

$i = 0;
foreach (@n){
   $name = $_;
   $age = $a[$i];
   $salary = $s[$i++];
   write;
}
```

When executed, this will produce the following result −

```
====================================
Ali                      20
  2000.00
====================================
====================================
Raza                     30
  2500.00
====================================
====================================
Jaffer                   40
  4000.00
====================================
```

Define a Report Header

Everything looks fine. But you would be interested in adding a header to your report. This header will be printed on top of each page. It is very simple to do this. Apart from defining a template you would have to define a header and assign it to ^{o}rFORMAT_TOP_NAME variable −

```
#!/usr/bin/perl

format EMPLOYEE =
====================================
@<<<<<<<<<<<<<<<<<<<<<<< @<<
$name $age
```

```
@#####.##
$salary
===================================
.

format EMPLOYEE_TOP =
===================================
Name                     Age
===================================
.

select(STDOUT);
$~ = EMPLOYEE;
$^ = EMPLOYEE_TOP;

@n = ("Ali", "Raza", "Jaffer");
@a  = (20,30, 40);
@s = (2000.00, 2500.00, 4000.000);

$i = 0;
foreach (@n){
   $name = $_;
   $age = $a[$i];
   $salary = $s[$i++];
   write;
}
```

Now your report will look like –

```
===================================
Name                     Age
===================================
===================================
Ali                      20
  2000.00
===================================
===================================
Raza                     30
  2500.00
===================================
===================================
Jaffer                   40
  4000.00
===================================
```

Define a Pagination

What about if your report is taking more than one page? You have a solution for that, simply use **$%** or $FORMAT_PAGE_NUMBER vairable along with header as follows –

```
format EMPLOYEE_TOP =
===================================
Name                     Age Page @<
                                  $%
===================================
.
```

Now your output will look like as follows –

```
===================================
Name                     Age Page 1
===================================
===================================
Ali                      20
  2000.00
===================================
===================================
Raza                        30
  2500.00
====================================
====================================
Jaffer                      40
  4000.00
====================================
```

Number of Lines on a Page

You can set the number of lines per page using special variable **$=** $or\$FORMAT_LINES_PER_PAGE$, By default $= will be 60.

Define a Report Footer

While ^orFORMAT_TOP_NAME contains the name of the current header format, there is no corresponding mechanism to automatically do the same thing for a footer. If you have a fixed-size footer, you can get footers by checking variable $-or$FORMAT_LINES_LEFT before each write and print the footer yourself if necessary using another format defined as follows −

```
format EMPLOYEE_BOTTOM =
End of Page @<
            $%
.
```

For a complete set of variables related to formating, please refer to Perl Special Variables section.

Loading [MathJax]/jax/output/HTML-CSS/jax.js