PERL FORMATS - WRITING REPORTS

http://www.tutorialspoint.com/perl/perl_format.htm

As stated earlier that Perl stands for Practical Extraction and Reporting Language, and we'll now discuss using Perl to write reports.

Perl uses a writing template called a 'format' to output reports. To use the format feature of Perl, you must −

- Define a Format
- Pass the data that will be displayed on the format
- Invoke the Format

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
.
```

`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.

`fieldline` can contain any text or fieldholders. 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.

Invoke the Format to write Data

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

```
write EMPLOYEE; #send to the output
```

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 $~

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

When we now do a write, the data would be sent to STDOUT. Remember: if you didn't have STDOUT set as your default file handle, you could revert back to the original file handle by assigning the return value of select to a scalar value, and using select along with this scalar variable after the special variable is assigned the format name, to be associated with STDOUT.

The above example will generate a report in the following format

```
Kirsten              12
Mohammad             35
Suhi                 15
Namrat               10
```

Defining 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 which will have same name but appended with _TOP keyword as follows

```
format EMPLOYEE_TOP =
------------------------
Name                  Age
------------------------
.
```

Now your report will look like

```
------------------------
Name                  Age
------------------------
Kirsten              12
Mohammad             35
Suhi                 15
Namrat               10
```

Defining a Pagination & Number of Lines on a Page

What about if your report is taking more than one page ? You have a solution for that. Use $% vairable along with header as follows

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

Now your output will look like

```
------------------------
Name                  Age   Page 1
------------------------
```

```
Kirsten               12
Mohammad              35
Suhi                  15
Namrat                10
```

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

Defining a Report Footer

One final thing is left which is footer. Very similar to header, you can define a footer and it will be written after each page. Here you will use _BOTTOM keyword instead of _TOP.

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

This will give you following result

```
-------------------------
Name                  Age   Page 1
-------------------------
Kirsten               12
Mohammad              35
Suhi                  15
Namrat                10
End of Page 1
```

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