Generating direct Latex using multi-line raw string (or HERE DOCUMENT)

Nasser M. Abbasi

June 16, 2016 Compiled on January 30, 2024 at 5:47am

Contents

These are small examples showing the use of `https://en.wikipedia.org/wiki/Here_document` which is also called multi-line raw strings or literal string.

This is done in Perl, Ruby, Python and C++11. These raw strings can contain any text in them, including embedded strings and escape characters. In addition, one can use `sprintf` like operations on them by replacing some tokens that have `%s` in them, with variables after building the strings. The main use of this is to be able to generate Latex directly from another language, such as Mathematica or Perl or Python, etc... since there are cases where it is easier to do this than coding the logic direclty in Latex itself. Some usage examples are below

1 Ruby

```
#!/usr/local/bin/ruby -w

str = <<'EOT'
\documentclass[12pt,titlepage]{article}
\begin{document}
\title{%{title}}
\date{%{date}}

The following in known equation $\sin^2(x)+\cos^2(x)=1$ and this
is 100%{p} correct. This was written on %{date} and this is a
string "I am a string" and this is \\n which is left as is.

\end{document}
```

```
EOT

puts str % {title:"title",date:"1/1/2015",p:"%"}
```

Now running the above script `t.rb` gives

```
\documentclass[12pt,titlepage]{article}
\begin{document}
\title{title}
\date{1/1/2015}

The following in known equation $\sin^2(x)+\cos^2(x)=1$ and this
is 100% correct. This was written on 1/1/2015 and this is a
string "I am a string" and this is \\n which is left as is.

\end{document}
```

Notice that `%` in the raw string (the percentage sign) had to also have a place holder to avoid conflict with the `%` used for string replacement. In the Perl example below, this is not needed.

2 Python

```
s =r"""
\documentclass[12pt,titlepage]{article}
\begin{document}
\title{%(title)s}
\date{%(date)s}

The following in known equation $\sin^2(x)+\cos^2(x)=1$ and this
is 100%(p)s correct. This was written on %(date)s and this is a
string "I am a string" and this is \n which is left as is.

\end{document}
"""
print s % {"title": "main report","date": "1/1/2015", "p":"%"}
```

Which is very similar to Ruby. Except in Ruby there was no need to add `s` after the place holder. Also, the `%` in the raw string had to be special handled the same as with Ruby.

3 Perl

In Perl however, the %, does not need special handling

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

use Text::Sprintf::Named qw(named_sprintf);

my $s =<<'EOT';
\documentclass[12pt,titlepage]{article}
\begin{document}
\title{%(title)s}
\date{%(date)s}

The following in known equation $\sin^2(x)+\cos^2(x)=1$ and this
is 100% correct. This was written on %(date)s and this is a
string "I am a string" and this is \\n which is left as is.

\end{document}
EOT

print named_sprintf($s, title=> 'My report', date=>'1/1/2016');
```

Notice how the 100% was left as is. No place holder was needed for it in Perl.

4 C++11

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <string>
using namespace std;

int main()
{
        char buffer [500];
        string s =

R"(
\documentclass[12pt,titlepage]{article}
\begin{document}
\title{%s}
\date{%s}

The following in known equation $\sin^2(x)+\cos^2(x)=1$ and this
is 100%s correct. This was written on %s and this is a
string "I am a string" and this is \\n which is left as is.

\end{document}
)";

 snprintf(buffer,500,s.c_str(),"main report","1/1/2015","%","1/1/2015");
 cout<<buffer<<endl;
 return 0;
}
```

and now

```
>g++ -Wall -std=c++0x ./t1.cpp
>./a.out
```

```
\documentclass[12pt,titlepage]{article}
\begin{document}
\title{main report}
\date{1/1/2015}
```

```
The following in known equation $\sin^2(x)+\cos^2(x)=1$ and this
is 100% correct. This was written on 1/1/2015 and this is a
string "I am a string" and this is \\n which is left as is.
\end{document}
```

However, the `C++` solution is not as flexible as the others, since it does not support named place holders which makes it little harder to replace tokens in the raw string if there are many of them.

References

1. C++ string literals
2. Ruby accepted answer here
3. Perl Named.pm