Documentation of source codes with the LaTeX package documentation*

Omar Salazar Morales
Laboratory for Automation, Microelectronics
and Computational Intelligence
Universidad Distrital Francisco José de Caldas
Engineering department
Bogotá, Colombia
osalazarm@correo.udistrital.edu.co
http://www.udistrital.edu.co

November 28, 2011

Abstract

This document shows the LaTeX implementation of the package documentation. This package is intented for all software's writers who want to document their source codes by using the comments of the programming language. The source files are processed with LaTeX in order to generate the documentation of them.

Contents

*This document is the version v0.1, 2011/11/28.

1 Introduction

On environments of software developtment is necessary to make the documentation of the source code according to its last modification. Sometimes, this is not easy, because of *documentation* and *source code* are written on different files.

In order to avoid this situation, where a software maker have to write two different files, LaTeX gives the possibility to handle an unique file where source code and documentation are together. Through the documentation package a software maker can write the source code for his application, and inside of its comments, he can document it. It's only neccessary to put LaTeX commands inside the comments of the source files. Source files are then proccesed by LaTeX to get a beautiful document (PDF, DVI, PS, ...)

The real advantage of this technique is that LaTeX will be present to handle all the documentation. If a software maker wants to put a complex formula where he explains a difficult algorithm, then he will be able to do it in the usual way with LaTeX commands.

2 Docstrip modules

This package has been developed with the typical LaTeX's documentation techniques. Docstrip has been used for the preparation of the source code of the package and its documentation.

The following docstrip modules were implemented to generate the different files of this project.

Option	Result
sty	It generates the package's file `*.sty`
drv	It generates the LaTeX documentation's master file `*.drv`

3 How to use

`\usepackage` This package is used in the usual way. You should use the `\usepackage` command in the preamble of your master's documentation file as follows

```
\usepackage[<options>]{documentation}
```

The `<options>` are presented in the next section.

4 Options

`java` `java` option is used when JAVA language is used. In this programming language the comment's character is `//` when one-line's comments are needed. If multi-line's comments are needed then `/*` and `*/` are neccessary. Then, all the comments inside your JAVA code have to start with `//`, or to be enveloped with `/*` and `*/`.

`c` `c` option is used when C language is used (or any of its variants like C++

or C#). This is the default option. This programming language uses the same comment's character as JAVA language, then, this option is the same as `java`.

`assembler` `assembler` option is used when assembler language is used, for example, when you are programming microcontrollers. In this programming language the comment's character is semicolon (`;`). All the comments (line-by-line) inside your assembler code have to start with `;`

5 How it works

1. You should create your source code as usual in C, C++, C#, JAVA or assembler languages (this step is done inside an IDE (Integrated Development Environment[1])). These programming languages use the following comment's characters

Programming language	Comment's character	Name of comment's character
C,C++,C#,JAVA	`//` or `/*` and `*/`	Double slash or slash with asterisk
Assembler	`;`	Semicolon

2. Now, you can add the documentation of your source code inside the comments. You can use LaTeX commands as usual. You should think that you are writing a LaTeX document. If you need to use the comment's character inside your source code, then you will be able to use a LaTeX command as is shown in the following table

Programming language	Character	LaTeX command
C,C++,C#,JAVA	`/`	`\/`
C,C++,C#,JAVA	`*`	`\*`
Assembler	`;`	`\;`

3. Any piece of source code have to be enveloped by `\begin{sourcecode}` and `\end{sourcecode}`. Just before of these two LaTeX commands, you have to put your comment's character *without spaces*. In C or JAVA, you have to use `//` before `\begin{sourcecode}` and `\end{sourcecode}`. For example, if you are writing a C code, then a piece of source code looks like

```
/*
   This is helloworld.c

   Comments with \LaTeX{} commands...

*/
//\begin{sourcecode}
#include<stdio.h> // Header file
//\end{sourcecode}
```

[1]Typical IDEs are *KDevelop* on Linux systems, *Eclipse* on Windows systems or *Microsoft Visual Studio* on Windows systems. Be careful when you're saving your source files on these IDEs. You should guarantee that they have the right coding as ASCII files

```
//
// More comments with \LaTeX{} commands...
//
//\begin{sourcecode}
void main (void)
{
   printf("Hello world!\n"); // "Hello world" message
}
//\end{sourcecode}
/*
   More comments with \LaTeX{} commands...
*/
```

In C or JAVA you can use `//` or `/*` and `*/` to add comments, but you have to use `//` before `\begin{sourcecode}` and `\end{sourcecode}`.

4. You can add a master's documentation file (`*.tex`) in your IDE. In the preamble of this file you have to use `\usepackage[<options>]{documentation}`. Now, you can read all your source files of your project with the LaTeX command `\inputsourcecode{<source file>}` where `<source file>` is the path of your source file with its extension[2].

   For the previous example, this file looks like

   ```
   %
   % This is dochelloword.tex
   %
   \documentclass{article}
   \usepackage[c]{documentation} % Needed
   \begin{document}
   \inputsourcecode{helloworld.c} % input your source code
   \end{document}
   ```

   You can use any LaTeX class (for example, article, book, report, ...) and any number of `\inputsourcecode` commands in order to read any number of source files.

5. Run LaTeX as usual to get the documentation of your source code. In the previous example, run `dochelloword.tex` through LaTeX. `\inputsourcecode` will read your source files and it will extract the documentation from the comments.

[2]Extension is needed because of some IDEs create different files with the same name and different extensions

6 Implementation

6.1 Package's identification

\NeedsTeXFormat
\ProvidesPackage

Package documentation was created to use it with LaTeX 2_ε.

```
%<*sty>
\NeedsTeXFormat{LaTeX2e}%
\ProvidesPackage{documentation}%
[2011/11/28 v0.1 Make the documentation for your source code]%
```

6.2 Preliminaries

\ifDOC@javalang
\ifDOC@Clang
\ifDOC@assemblerlang

The boolean variables `\ifDOC@...lang` are used to determine which programming language is especified by the user according to the following table.

Variable	Programming language
`\ifDOC@javalang`	JAVA
`\ifDOC@Clang`	C (or C++ and C#)
`\ifDOC@assemblerlang`	Assembler

These variables begin with a `false` value.

```
\newif\ifDOC@javalang     \DOC@javalangfalse
\newif\ifDOC@Clang        \DOC@Clangfalse
\newif\ifDOC@assemblerlang\DOC@assemblerlangfalse
```

6.3 Options

java

`java` option calls all the necessary code which is needed to make the documentation for JAVA language. This programming language uses the comment's characters `//` for one-line's comments and `/*...*/` for multi-line's comments.

This option gives `true` value to `\ifDOC@javalang` and `false` to others. Inside this option, `\DOC@changeccofcommentchar` and `\DOC@definecsofcommentchar` are defined in order to change the `\catcode` of `/` and `*` as desire, and to define the macros `\/` and `\*` to print `/` and `*` inside the text of the source files.

```
\DeclareOption{java}{%
  \DOC@javalangtrue \DOC@Clangfalse
  \DOC@assemblerlangfalse
  \gdef\DOC@changeccofcommentchar#1{\catcode`/=#1
                                    \catcode`*=#1}%
  \gdef\DOC@definecsofcommentchar{\chardef\/=`/
                                  \chardef\*=`*}}%
```

c

This option is almost the same as `java`.

```
\DeclareOption{c}{%
  \DOC@javalangfalse \DOC@Clangtrue
  \DOC@assemblerlangfalse
  \gdef\DOC@changeccofcommentchar#1{\catcode`/=#1
```

```
                                        \catcode`*=#1}%
  \gdef\DOC@definecsofcommentchar{\chardef\/=`/
                                  \chardef\*=`*}}%
```

assembler

`assembler` option calls all the necessary code which is needed to make the documentation for assembler language. This programming language uses the comment's characters `;` for all kind of comments.

This option gives `true` value to `\ifDOC@assemblerlang` and `false` to others. Inside this option, `\DOC@changeccofcommentchar` and `\DOC@definecsofcommentchar` are defined in order to change the `\catcode` of `;` as desire, and to define the macro `\;` to print `;` inside the text of the source files.

```
\DeclareOption{assembler}{%
  \DOC@javalangfalse \DOC@Clangfalse
  \DOC@assemblerlangtrue
  \gdef\DOC@changeccofcommentchar#1{\catcode`;=#1}%
  \gdef\DOC@definecsofcommentchar{\chardef\;=`;}}%
```

All the other options which are specified by the user, but which are not defined, give an error message as *unknown options*.

```
\DeclareOption*{%
  \PackageError{documentation}%
  {Unknown option `\CurrentOption'}%
  {See the documentation for more details}}%
```

Now, it's necessary to process all the options which were especified by the user. `c` option is used as the default.

```
\ExecuteOptions{c}\ProcessOptions\relax
```

6.4 The source code

sourcecode

`\begin{sourcecode}` and `\end{sourcecode}` is the way as an user gives the source code for his application. All source code has to be enveloped with this environment in order to write it `verbatim`. The real difference with respect to `verbatim` is that `sourcecode` recognizes the comment's character of the programming language. This environment permits to write the comment's character inside without any special LaTeX command (like `\/`, `\*` or `\;`). The only restriction is that you have to use the comment's character of your programming language just before `\begin{sourcecode}` and `\end{sourcecode}` *without spaces* between them. `sourcecode` uses the internal macros `\@verbatim`, `\frenchspacing`, `\@vobeyspaces`, `\if@newlist`, `\leavevmode` and `\endtrivlist`. See `ltmiscen.dtx` for more details.

```
\def\sourcecode{\DOC@changeccofcommentchar{12}%
                \@verbatim \frenchspacing\@vobeyspaces \DOC@sourcecode}%
\def\endsourcecode{\if@newlist \leavevmode\fi\endtrivlist}%
```

\DOC@sourcecode

`\DOC@sourcecode` recognizes the beginning and the end of the real source code by using the comment's character of your programming language. Then, this macro

depends on the language. This macro begins changing some `\catcode`s of some characters because it's needed to say to LaTeX where is the end of the source code. This is done inside a group because it's needed to keep local all changes.

```
\begingroup
  \catcode`|=0   \catcode`[= 1
  \catcode`]=2   \catcode`\{=12
  \catcode`\}=12 \catcode`\\=12
  |catcode`/=12  |catcode`;=12
```

Now, `\DOC@sourcecode` is defined according to the language. It's needed to say to LaTeX that source code will finish with the comment's character of the programming language which is followed by `\end{sourcecode}` *without spaces*. At the end, the group is closed.

```
  |ifDOC@javalang
    |gdef|DOC@sourcecode#1//\end{sourcecode}[#1|end[sourcecode]]%
  |fi
  |ifDOC@Clang
    |gdef|DOC@sourcecode#1//\end{sourcecode}[#1|end[sourcecode]]%
  |fi
  |ifDOC@assemblerlang
    |gdef|DOC@sourcecode#1;\end{sourcecode}[#1|end[sourcecode]]%
  |fi
|endgroup
```

`\inputsourcecode` This is the way as an user `\input`s his source code. This command uses the classical `\input` LaTeX command. It begins with the definition of `\DOC@path` as the path of the source file. `\DOC@path` is necessary because of UNIX systems use `/` as a delimiter on its directory tree (for example `/usr/share/local/`), and JAVA and C use the same character as comment's character. Everything is done inside a group.

```
\def\inputsourcecode#1{%
\begingroup
  \def\DOC@path{#1}%
```

Now, it's time to define the right command secuence if an user wants to print the comment's character inside the documentation, also, it's changed the `\catcode` of the comment's character which is treated as an space (catcode 10).

```
  \DOC@definecsofcommentchar
  \DOC@changeccofcommentchar{10}%
```

At the end, the source file is red. Notice that `\inputsourcecode` keeps all changes local. Then, your source file doesn't affect any other part of your LaTeX document.

```
  \expandafter\input\DOC@path
\endgroup}%
```
⟨/sty⟩

Change History

v0.1

General: Initial version 1