

K Syntax Highlighter

Tudor Nicoara, grupa 2E1

20.12.2017

1 Abstract

Implementation of a highlighter for the K Framework for gedit.

2 Definitions

2.1 Syntax highlighting

Syntax highlighting is a feature of text editors that are used for programming, scripting, or markup languages, such as HTML. The feature displays text, especially source code, in different colors and fonts according to the category of terms. This feature facilitates writing in a structured language such as a programming language or a markup language as both structures and syntax errors are visually distinct. Highlighting does not affect the meaning of the text itself; it is intended only for human readers.

2.2 Gedit

Gedit is the GNOME text editor. While aiming at simplicity and ease of use, gedit is a powerful general purpose text editor.

3 What I used

3.1 gedit for Windows

<https://wiki.gnome.org/Apps/Gedit>

3.2 .lang files

In order to observe the syntax and the editing technique of *.lang* files and highlighters in gedit, I took a look at some of the files in the language-specs folder.

> This PC > OS (C:) > Program Files (x86) > gedit > share > gtksourceview-2.0 > language-specs

For example: C language

```
24 <language id="c" _name="C" version="2.0" _section="Sources">
25 <metadata>
26 <property name="mimetypes">text/x-c;text/x-csrc:image/x-xpixmap</property>
27 <property name="globs">*.c</property>
28 <property name="line-comment-start">/*!</property>
29 <property name="block-comment-start">/*</property>
30 <property name="block-comment-end">*/</property>
31 </metadata>
32
33 <styles>
34 <style id="comment" _name="Comment" map-to="def:comment"/>
35 <style id="error" _name="Error" map-to="def:error"/>
36 <style id="string" _name="String" map-to="def:string"/>
37 <style id="preprocessor" _name="Preprocessor" map-to="def:preprocessor"/>
38 <style id="common-defines" _name="Common Defines" map-to="def:special-constant"/>
39 <style id="included-file" _name="Included File" map-to="def:string"/>
40 <style id="char" _name="Character" map-to="def:character"/>
41 <style id="keyword" _name="Keyword" map-to="def:keyword"/>
42 <style id="type" _name="Data Type" map-to="def:type"/>
43 <style id="storage-class" _name="Storage Class" map-to="def:type"/>
44 <style id="printf" _name="printf Conversion" map-to="def:special-char"/>
45 <style id="escaped-character" _name="Escaped Character" map-to="def:special-char"/>
46 <style id="floating-point" _name="Floating point number" map-to="def:floating-point"/>
47 <style id="decimal" _name="Decimal number" map-to="def:decimal"/>
48 <style id="octal" _name="Octal number" map-to="def:base-n-integer"/>
49 <style id="hexadecimal" _name="Hexadecimal number" map-to="def:base-n-integer"/>
50 <style id="boolean" _name="Boolean value" map-to="def:boolean"/>
51 </styles>
52
53 <definitions>
54 <!-- TODO: what about scanf ? -->
55 <!-- man 3 printf -->
56 <context id="printf" style-ref="printf" extend-parent="false">
57 <match extended="true">
58 \s*\%[\s\%
59 (?:[1-9][0-9]*\$)? # argument
60 [#0\-\ \+\|I]* # flags
61 (?:[1-9][0-9]*|\+)? # width
62 (?:\.\.\.?|[0-9]+\|\.*)? # precision
63 (?:[hh|ll|l|ll|q|j|z|t])? # length modifier
64 [diouxXeEfFgGaAcscSpnm] # conversion specifier
65 </match>
66 </context>
67
68 <define-regex id="escaped-character" extended="true">
69 \\( # leading backslash
70 [\\\"'\nr\b\t\fv\?]| # escaped character
71 [0-7]{1,3} | # one, two, or three octal digits
72 x[0-9A-Fa-f]+ # 'x' followed by hex digits
73 )
74 </define-regex>
75
76 <context id="c" class="no-spell-check">
77 <include>
78
```

3.3 Notepad++

In order to edit the *.lang* file.

4 Method

I took the C syntax highlither file and I modified it in order for it to work with the K Framework. I added some new styles for modules, for the printing part and for the operators. I removed some things from the previous C highlighter that I didn't need such as the preprocess part, define and include and I kept things like the comment, type, string and keyword.

```
<!-- Keywords -->
<context id="keywords" style-ref="keyword" class="keyword">
  <keyword>do</keyword>
  <keyword>else</keyword>
  <keyword>for</keyword>
  <keyword>if</keyword>
  <keyword>rule</keyword>
  <keyword>while</keyword>
  <keyword>syntax</keyword>
  <keyword>imports</keyword>
</context>

<context id="types" style-ref="type" class="type">
  <keyword>Bool</keyword>
  <keyword>Int</keyword>
  <keyword>Id</keyword>
  <keyword>String</keyword>
  <keyword>List</keyword>
</context>

<context id="boolean" style-ref="boolean" class="boolean">
  <keyword>>true</keyword>
  <keyword>>false</keyword>
</context>
```

5 How does it look like now

```
project.k
module PROIECT-SYNTAX
  syntax AExp ::= Id
    | Int
    | String
    | "read" "(" ")"
    | "++" Id [inc]
    | AExp "/" AExp [left, strict, division]
    | AExp "*" AExp [left, strict]
    | AExp "%" AExp [left, strict]
    > AExp "+" AExp [left, strict, plus]
    | "(" AExp ")" [bracket]

  syntax BExp ::= Bool
    | AExp ">" AExp [strict]
    | AExp "!=" AExp [strict]
    | AExp "==" AExp [strict]
    | "(" BExp ")" [bracket]

  syntax Block ::= "{" "}"
    | "{" Stmt "}"

  syntax Stmt ::= Block
    | Id "=" AExp ";" [strict(2)]
    | "int" Id ";"
    | "print" "(" AExp ")" ";" [strict]
    | "spawn" Stmt
    | AExp ";" [strict]
    | "if" BExp Block "else" Block [strict(1)]
    | "while" BExp Block
    | "mirror" "(" AExp ")" ";" [bracket]
    | "for" "(" Stmt BExp ";" Stmt ")" Block [bracket]
    > Stmt Stmt [right]

endmodule

module PROIECT
  imports PROIECT-SYNTAX
  syntax KResult ::= Int | Bool | String

  configuration <T>
    <threads>
      <thread multiplicity="*">
        <k> $PGM:Stmt </k>
        <env> .Map </env> // varname |-> address
        <stack> .List </stack>
      </thread>
    </threads>
```

6 Work in progress...

I still need to figure out how to highlight the whole "module..." line.
I need to learn how to use different colors than the ones already defined.
I need to do some refining, because some colors do not fit in some places together.

7 Conclusion

The work is nearly done, some things need some refining, but everything is coming to pieces.