# LaTeX Listings - PVL Syntax Highlighting

Ömer Şakar

25 Jan. 2020

A package defining a `listings` language to (syntax) highlight PVL.

## Contents



## 1 Requirements

The following LaTeXpackages are required to be installed on your system:

- `listings`
- `xcolor`

## 2 How to use

The listings language is called PVL and can be used by setting the listings option `language`.

```
\begin{lstlisting}[language=PVL]
    // PVL CODE
\end{lstlisting}
```

Or,

```
\lstinputlisting[language=PVL]{path/to/pvlfile.pvl}
```

Other listings options can be set after specifying the language. If you have encoding issues (for example by changing the default font), try adding \usepackage[T1]{fontenc} after specifying the document class.

# 3 Details about the package

## 3.1 Groups of keywords

Keywords are grouped (loosely) based on the PVL Syntax page:

- **Class 2**: int, boolean, void, resource, frac, zfrac, process, seq, set, bag, option
- **Class 3**: return, if, else, for, while, par, vec, atomic, barrier, fork, join, wait, notify, lock, unlock
- **Class 4**: class, pure, new, this
- **Class 5**: Perm, PointsTo, Value, History, HPerm, Hist, action, AbstractState, create, destroy, split, merge, modifies, accessible, Future, choose
- **Class 6**: assert, assume, requires, ensures, context, loop_invariant, invariant, given, yields, with, then, unfolding, in, refute, inhale, exhale, fold, unfold
- A group for keywords starting with a backslash
- A group for infix operators

## 3.2 Adding more keywords

How keywords are added depends on which group of keywords it belongs to.

- If the keyword starts with a backslash, then it is added by using the listings option `moretexcs`. For example, if we want to highlight `\myFunc`, then we add the following option:

  ```
  \lstset{
      language=PVL,
      moretexcs={myFunc},
  }
  ```

- If the keyword is an infix operator, then it is added by using the listings option `literate`. For example, if we want to highlight an imaginary operator !@, then we add the following option:

  ```
  \lstset{
      language=PVL,
      literate={\ !@\ }{{\color{red}\  !@ }}{3}
  }
  ```

  For the exact syntax of the `literate` option, please see the documentation for the `listings` package.

- If the keyword does not fall into one of the previous categories and you want to add it to an existing class, then it is added by using the listings option `morekeywords`. For example, if we want to add a type `float` to class 2, then we add the following option:

```
\lstset{
    language=PVL,
    morekeywords=[2]{
        float
    }
}
```

- If the keyword should be in a new class, then we use a combination of the listings options `morekeywords` for adding the keyword and `keywordstyle` to define the style/color. The first six classes are in use, so any class above class 6 should be free. For example, if we want to add a keyword `context_everywhere` to a new class 8 and it should be colored red, then we add the following options:

```
\lstset{
    language=PVL,
    morekeywords=[8]{
        context\_everywhere
    },
    keywordstyle=[8]\color{red}
}
```

### 3.3 Colors

The different groups of keywords have the following coloring scheme:

- Comments -> `\color{pvlgrey}`
- Class 2 -> `\color{pvlblue}`,
- Class 3 -> `\color{pvlblue}`,
- Class 4 -> `\color{pvlblue}`,
- Class 5 -> `\color{\color{purple}`,
- Class 6 -> `\color{pvlgreen}`,
- Keywords starting with backslash -> `\color{orange}`,
- The infix operators &&, ||, ==> -> `\color{pvlinfixcolor}`,
- The rest of the infix operators -> `\color{darkpvlinfixcolor}`,

These colors are defined as follows:

```
    \definecolor{pvlgrey}{rgb}{0.46,0.45,0.48}
    \definecolor{pvlgreen}{HTML}{65CC2D}
    \definecolor{pvlblue}{HTML}{4D5BFF}
    \definecolor{pvlinfixcolor}{HTML}{999C94}
    \definecolor{darkpvlinfixcolor}{HTML}{5A5C57}
```

To change the colors, either redefine the color with the names above or overwrite the style option for the correct group of keywords. For example, if we want to have all keywords black and bold again, then we can do the following:

```
\lstset{
    language=PVL,
    commentstyle=\bfseries\color{black}, % For comments
    keywordstyle=[2]\bfseries\color{black}, % For class 2 keywords
    keywordstyle=[3]\bfseries\color{black}, % For class 3 keywords
    keywordstyle=[4]\bfseries\color{black}, % For class 4 keywords
    keywordstyle=[5]\bfseries\color{black}, % For class 5 keywords
    keywordstyle=[6]\bfseries\color{black}, % For class 6 keywords
    texcsstyle=*\bfseries\color{black}, % For keywords starting with backslash
}
```

# 4 Example PVL code

Since this package only defines syntax highlighting for PVL and no styling/formatting for the listings, it is probably a good idea to add some styling to your PVL code. This could be either by using \lstset{language=PVL, <options>} or define a new language using \lstdefinelanguage{NewLanguageName}{language=PVL, <options>}.

Below we show a piece of PVL code twice. Figure 1 is plain PVL highlighting for the example (i.e. no other options set) and Figure 2 is a styled version using a listings language `AnotherPVL` defined as in Figure 3.

```
class Incrementer {
        requires n >= 0;
        ensures |\result| == |input|;
        ensures (\forall int k; 0 <= k && k < |\result|; \result[k] ==
            input[k]+n);
        seq<int> incrementAllByN(seq<int> input, int n) {
                seq<int> res = seq<int> {};
                int i = 0;

                loop_invariant 0 <= i && i <= |input|;
                loop_invariant i == |res|;
                loop_invariant (\forall int k; 0 <= k && k < i; res[k] ==
                    input[k]+n);
                while (i < |input|) {
                        res = res + seq<int> {input[i]+n};
                        i = i + 1;
                }
                return res;
        }
}
```

Figure 1: Plain PVL Syntax highlighting

```
1  class Incrementer {
2      requires n >= 0;
3      ensures |\result| == |input|;
4      ensures (\forall int k; 0 <= k && k < |\result|; \result[k] == input[k]+n);
5      seq<int> incrementAllByN(seq<int> input, int n) {
6          seq<int> res = seq<int> {};
7          int i = 0;
8
9          loop_invariant 0 <= i && i <= |input|;
10         loop_invariant i == |res|;
11         loop_invariant (\forall int k; 0 <= k && k < i; res[k] == input[k]+n);
12         while (i < |input|) {
13             res = res + seq<int> {input[i]+n};
14             i = i + 1;
15         }
16         return res;
17     }
18 }
```

Figure 2: Styled PVL Syntax highlighting (with the language in Figure 3)

```
\lstdefinelanguage{AnotherPVL} {
    language=PVL,
    numbers=left,
    numberstyle=\small,
    numbersep=8pt,
    frame=single,
    framexleftmargin=15pt,
    tabsize=2,
    basicstyle=\footnotesize\ttfamily,
    breaklines=true,
    postbreak=\mbox{\textcolor{red}{$\hookrightarrow$}\space},
    showspaces=false,
    showtabs=false,
    showstringspaces=true,
    breakatwhitespace=true,
}
```

Figure 3: The definition of the listings language `AnotherPVL`