

Syntax Highlighting in WinEdt

Colin Marquardt

November 21, 1999

Syntax Highlighting in WinEdt

Syntax Highlighting is one of the powerful features of WinEdt. Certain parts of your text can be highlighted, that means formatted visually in a different color or font. This allows to have your document structured in an easy to spot way. In addition, Syntax Highlighting can help you to avoid typos. The behavior of the Syntax Highlighting feature is dependent on the mode of your document. Of course, you can customize the highlighting completely. To show you how is the aim of this chapter.

1 Basics

Before experimenting with Syntax Highlighting settings, it is a good idea to save your current state of WinEdt's customization. All customizations are stored in the file `winedt.ini`¹. Use the menu entry **Options | Save Settings As...** to make a backup copy of your current `winedt.ini` under another name. If you want, you can then revert to your previous state by renaming this file to `winedt.ini`. You also have a "factory settings" `winedt.ini` in the subdirectory `\support`.

WinEdt provides a way to exchange settings from some dialogs (including the Syntax Highlighting) via a human-readable format. You can access this feature by clicking the secondary mouse button (usually the right button) in the dialog. Now, the context menu offers you the entries **Extract...**, **Append...** and **Load From...**



Syntax Highlighting in WinEdt is based on the idea of three different highlighting levels. These levels specify the syntax elements which are subject to highlighting.

¹If you haven't specified an alternative ini-file when calling WinEdt.

1.1 Filter Sets

Take, for example, a simple text where you give some numeric values. Some of these values are negative, and you want to spot them immediately. A good way to achieve this is to use a different color for the minus sign in front of these negative values, say blue. No problem, you tell WinEdt to color all occurrences of the minus sign blue.² To oppose the negative values, you decide to mark important positive values with an explicit plus sign in front of them. Again, you want this plus sign to be in an other color than black, this time red. This is nice and also easy, and would be done using a syntax element what WinEdt calls a *Filter Set*.

1.2 Switches

You look now at your partly colored text. It looks good, but you'd wish that also the values *itself* would be colored, not only minus and plus. From what you know now, you could tell WinEdt to color all '1's green, all '2's green, and so on. It would work, and you could even group the numbers together, using only one of WinEdt's Filter Sets. But no, you aren't satisfied with that, you want the negative numbers blue and the important positive numbers with a plus before it in red. You have now entered a new dimension of highlighting, the dimension of the *Switches*.

Switches set a number of successive, random characters to a certain color (or font), depending on a condition. In our case, the condition is the minus or plus sign. *If* a minus sign is followed by a number, *then* you want to have both in blue. You don't want numbers without minus or plus to be colored, nor do you want a blue hyphen or such. The condition itself is specified as a Filter Set, as we have already seen.

1.3 Reserved Words

There is another syntax element that you could want to highlight. Switches "switch" to another look depending on the condition, and the order of the characters to highlight doesn't matter; they only need to match some criteria (such as "being a number"³).

To impress and flatter your boss, you want now to have his name, Miller, in bold font. You don't want to have all uppercase 'M's, all 'i's, 'l's, 'e's and 'r's in bold, just this specific sequence that forms the word "Miller". This is a so-called *Reserved Word*, and of course you can base it on a certain condition too, e. g. a "Mr " before it.

Most problems that you have for highlighting can be solved with a combination of these three syntax elements.

2 Filter Sets

As we have seen, Filter Sets form the basic syntax element in WinEdt's Syntax Highlighting. They are used in the other two highlighting syntax elements, the Switches and the Reserved Words. In addition, Filter Sets are also used in the context of *Active Strings*.

²How this is done is explained later in this chapter.

³Often the criterion is "be any character".

2.1 The “Filter Sets” Dialog

If you look into the dialog for Filter Sets (open **Options | Highlighting** on the first tab page), you’ll see that there are a lot of predefined Filter Sets. This is the list on the left of this dialog. Browse through the entries and check how these look, but *do not* delete or modify any of these predefined sets.

You can, of course, add your own entries here. In fact, this is why this chapter is written. The order of the entries in the list box does not matter.⁴ Use the secondary mouse button to **I**nser a new entry. Type in a descriptive name for your Filter Set (it is not restricted to one word, use whatever you want). This name is not to be confused with the *definition* of the Filter Set!

Immediately below the list of Filter Sets there is a field which is called “Definition”. You specify the criteria for the Filter Set selected above here. Please see section 2.2 for information about these Set Expressions.

In the middle column in this dialog there are two fields which are “Filters”, “Before:” and “After:”. If your Filter Set is dependent on another Filter Set, you specify this here. To extend our introductory example, you could want to have a Filter Set which is “+–”. This is either a plus “Before” a minus or a minus “After” a plus. To use a Filter Set in this list box it has to be defined first. Although you could write directly in this box (say, a “+”), this alone is not functioning without a matching Filter Set entry on its own.

The checkboxes for the filters can specify further restrictions:

- if the option “Before:” is checked then the beginning of line is *not* an admissible position for a character to belong to the filter,
- if the option “After:” is checked then the end of line is *not* an admissible position for a character to belong to the filter.

Below the “Filters” field is “Options”, where you can enable a Filter Set for highlighting of certain modes. You *don’t* have to enable highlighting for a Filter Set in order to use it elsewhere! You only enable a Filter Set if you want to highlight something directly with it.

A Filter Set can be given a priority from “0” (lowest) to “9” (highest), which determines highlighting inside Switches and Reserved Words.



Filter Sets (as well as Switches and Reserved Words) are highlighted with respect to the “Priority” value. Priority can be specified to determine highlighting of the currently selected filter set inside Switches and Reserved Words. Note that this *does not* mean that when two or more Filter Sets compete, the one with the higher priority is highlighted. In such a case, always the last specified Filter Set applies, regardless of priority. However, a Filter Set with a high priority rules over a Switch with a lower priority.

The “Sample” field shows you how your Filter Set will look if you have given it certain “Font” or “Color” attributes in the right column of this dialog. Of course, this does only apply in your document if it matches the enabled mode definition.

◇

Note that certain fields in this dialog (or, to be correct, in almost every WinEdt dialog) give useful context-sensitive menus, that means they respond to the secondary mouse button.

⁴As long as you do not specify entries with the same name. In this case, the last of these entries is used.

2.2 Set Expressions

You use Set Expressions to specify the “Definition” field in the Filter Set dialog. There are four predefined sets: `Numeric`, `Alpha`, `Upper` and `Lower`. These sets are language sensitive, that means their definition is dependent on the Language Settings in Windows’ Control Panel.

To define sets, you use the following syntax:

character: Either "`?`", where `?` stands for any character, or
`# {code}`, with $0 < \text{code} \leq 255$.

interval: `character . . character`

set: Either a predefined set or a [*sequence of characters and intervals*].
`[]` is also allowed and denotes the empty set.

operators: `~` (not),
`+` (union),
`-` (difference),
`*` (intersection)

2.2.1 Examples:

Character:

`"a"` the letter *a*,
`"b"` the letter *b*,
`#32` the ASCII code 32.

Interval:

`a . . z` the lowercase letters *a, b, c . . . z*.

Set:

<code>Numeric</code>	the numbers 0 to 9 (a predefined set),
<code>["ab"]</code>	the letters <i>a</i> and <i>b</i> ,
<code>["a", "b"]</code>	also the letters <i>a</i> and <i>b</i> ,
<code>["+-*/^<>=(){}&#[]"]</code>	Math stuff (from the default WinEdt entry),
<code>[]</code>	the empty set.

Operators:

<code>~[]</code>	not empty, that means All,
<code>Alpha+Numeric</code>	union of the predefined sets <i>Alpha</i> and <i>Numeric</i> , thus specifying all letters and all numbers,
<code>Alpha-Lower</code>	all letters minus the lowercase ones, giving the uppercase ones (which happen to have their own predefined set, <i>Upper</i>),
<code>SetA*SetB</code>	gives you the elements that are in <i>SetA</i> as well as in <i>SetB</i> . (I cannot make up a straightforward example, sorry . . .)

These examples show the basics of defining Set Expressions. We will come to some more complicated definitions in later, concrete examples. Making up not-too-easy examples for Filter

Sets only is hard. Since a good and sensible Switch usually requires a good and sensible Filter Set too, we will give its both definitions in the section about Switches.

If you are not sure about the correct syntax of a Filter Set's definition: don't hesitate to experiment. You cannot do much wrong as long as you *do not modify WinEdt's default entries*. Enable the Filter Set you are working on and give it an easy to spot color. If you think it is right, you can again switch it off and use it in e. g. the Reserved Words tab page (remember, you don't have to highlight a Filter Set to use it elsewhere).

3 Switches

A Switch tells WinEdt to highlight certain parts of your text. These parts are *started* by a clearly defined condition, and are *ended* on another condition. Unless in Filter Sets, you don't define the part of the text to highlight, you define which condition "switches" highlighting on instead.

3.1 Global Switches

Global Switches are one of the things that WinEdt 1.414 (the 32-bit version) introduced. This allows highlighting to extend over more than one line. Because of the complicated algorithm WinEdt has to use, enabling Global Switches can be slow, especially for longer documents. By default, Global Switches are only enabled for T_EX documents. You can change these settings for other modes via the menu Options | Preferences | Defaults. Here is a "Disable Global Switches" checkbox. Global Switches for the current document can be enabled or disabled in Project | Document Settings | Options (accessible also from the Mode Field in the status line). In the examples it is assumed that you have turned on Global Switches, but you will not always need it. If highlighting over more than one line doesn't work, check this setting.

3.2 The "Switches" Dialog

To define or modify a Switch, you use the second tab page in Options | Highlighting. As in the tab page for Filter Sets, there are already some entries, which give an idea how such an entry looks. If you use WinEdt to write T_EX or L^AT_EX documents, you will recognize some of these Switches.

As in the other dialogs for Syntax Highlighting (and almost every dialog in WinEdt), a mouse click with the secondary button pops up a context menu with commands for manipulating entries, such as Inserting a new one.

In the dialog for defining Switches, only the middle column differs from that of the "Filter Sets" dialog. Since you should know a little about Filter Sets when playing with Switches, I'm not going to explain the rest of the dialog here. The remarks in section 2.1 (The "Filter Sets" Dialog) apply here as well.

In the middle column in this dialog there are two fields which are "Filters". These say "Begin:" and "End:" here. As you can imagine, this tells WinEdt what Filter Set begins or ends a Switch here. To select such a Filter Set from the list, it has to be defined before.

To understand what the checkboxes in front of the "Begin:" and "End:" fields do, we first must explain how the "Definition" field comes into play.

3.2.1 The “Begin:” Filter and “Definition”

To take up our example in the introduction of this chapter, assume we want to specify a Switch for marking all negative numbers.

There are basically two different possibilities to make such a combination of a Filter Set and a Switch work.

In a first case, you only need to give the “minus” Filter Set in the “Begin:” field, provided you have set up this Filter Set. The field “Definition” can stay empty. Since highlighting starts according to the “Begin:” field, the minus and the following number will be highlighted.

The second possibility is to specify a “-” (without the quote signs) in the “Definition” field, and `Numeric` in the “Begin:” field (with the checkbox *not* checked). Since `Numeric` is a predefined set, you don’t even have to create a new Filter Set. As already said, highlighting starts according to the “Begin:” field. In this case, we have `Numeric` there, so the numbers in our document will be highlighted. But, and this is the crucial point, the minus will *not* be highlighted! You have to decide whether this is acceptable. You could, of course, set up and enable a Filter Set with “minus” in addition. While this is practicable, it is usually better to think about other ways.



Now, there seems to be a third way of setting up this Switch. This would be to make “-” the “Begin:” specifier, and to set `Numeric` as the “Definition”. Good idea, but . . . With Filter Sets, you were allowed to have (predefined) sets in the “Definition” field. This, however, is not possible with Switches. WinEdt treats text in the “Definition” field in the Switches dialog verbatim! Specifying `Numeric` here would expect the *word* “Numeric” after the “minus” sign.

Let’s sum up what this “Begin:” checkbox does:

“Begin:” unchecked: `Definition`, then `Filter Set` – The Switch starts if the characters specified in the “Definition” field (taken verbatim) precede the syntax elements as specified by the Filter Set in the “Begin:” field. The characters in the “Definition” field are not highlighted since Syntax Highlighting starts only with the Filter Set.

“Begin:” checked: `Filter Set`, then `Definition` – The Switch starts if the Filter Set in the “Begin:” field matches and is followed by the “Definition”, which is taken verbatim. Here, “Definition” is highlighted, since it comes after the Filter Set which begins the Switch.

In the “Options” section of this dialog there is a checkbox named “BOLN” (Beginning Of Line). If this box is checked, then the whole Switch only applies if it starts at the beginning of a line.

3.2.2 Ending a Switch

After setting up a condition which begins a Switch, you now need another condition which ends this Switch. This is achieved by the “End:” field in the “Filters” section. Also, a Switch has a “Scope” in which it is highlighted. Text outside the scope of a Switch is not highlighted; this is the second condition which can end a Switch.

As you see, the listbox for “End:” gives the list of all defined Filter Sets. The Switch is ended when the “End:” Filter Set applies. The checkbox before the “End:” field tells whether the occurrence of this Filter Set is part of the Switch, that means whether it is highlighted or not.

Now, what is the scope of a Switch? The “Scope” field in the “Options” section can have a number ranging from “0” to “4”. The meaning of these numbers is:

- 0 The Switch ends with the first non-alpha character. This is meant for \TeX control sequences.
- 1 The Switch ends at the end of the line. This is useful for one-line comments in programming languages.
- 2 The Switch ends at the hard return (not used).
- 3 The Switch ends at the first empty line. This can be used for \TeX Math sequences ($\$. . . \$$).
- 4 The Switch ends with the specified “End:” Filter Set. Take care, as this is potentially slow in large documents.

Also see the comments about Global Switches in section 3.1.

3.2.3 Priority

Switches can be nested. The default settings of nested switches are combined if the switches have equal priorities and the inner switch has the option “Default Font” or “Default Color” enabled. A Switch with a lower priority is ignored inside a Switch with greater priority.

3.3 Examples

The following examples are given in the syntax which WinEdt uses to Extract... the entries in the Syntax Highlighting dialogs. This syntax should be self-explanatory.

3.3.1 The Negative Number scheme

Let us see how the example with the negative number can be set up. We already noted that there are (at least) two ways to achieve this. We begin with the first possibility, which only needs a Filter Set to begin the Switch.

The Filter Set with which our Switch begins is simply a “minus” sign, followed by a number. We know already that there is a predefined set called `Numeric` which provides exactly this. Thus, the Filter Set with the name “`-Numeric`” is:

```
Name:-Numeric
Definition:["-"]
Before:0
After:1Numeric
Enabled:0
Priority:0
Color:1040
```

You can see, the “Definition” of the Filter Set is a minus, in a syntax that follows the rules from

section 2.2, “Set Expressions”. The “Before:” field is not checked (the ‘0’) and empty. The “After:” field contains the (predefined) set `Numeric` and is checked, so that a minus ending a line does not get highlighted (without this, in addition to `Numeric`, the end of the line would be valid). The Filter Set itself is not enabled, since we only use it indirectly in the Switch. Priority is zero, the color is the standard color, but this does not come to effect since the Filter Set is not enabled.

Okay. We have all what we need to begin the Switch. But how do we switch it off? The highlighting should stay in effect for all digits in the negative number, but not for text or other non-numeric characters. Therefore, a good “End:” condition is “not numeric”. But before we come to set this Filter Set up, let’s think a little: a number, if it is not an integer, can also have a decimal point. If we would simply say: “switch off on all but numbers”, the decimal point would switch highlighting off too. Since we do not want this, we must also allow the decimal point. We can count the decimal point as numeric in this case, but of course it is not included in the predefined set `Numeric`. Thus the Filter Set with the name `Not Numeric` is:

```
Name:Not Numeric
  Definition:~Numeric-["."]
  Before:0
  After:0
  Enabled:0
  Priority:0
  Color:1040
```

We remember: the tilde (~) is the negation symbol in the Set Expression syntax. The “Definition” reads as follows: we want “not numeric”, which leaves all other characters in the ASCII table *except* the numbers from ‘0’ to ‘9’. From this remaining set of characters we subtract also the minus sign.⁵ Thus, the Filter Set specifies all characters *except* the numbers from ‘0’ to ‘9’ *and* the ‘minus’. We can now use this set to end our Switch.

Since this is possibility One, let us call this Switch `Negative Number 1`.

```
Name:Negative Number 1
  Definition:
  Begin:1-Numeric
  End:0Not Numeric
  Enabled:1*
  Priority:2
  Color:3082
  Case Sensitive:0
  BOLN:0
  Scope:1
```

As said before, this switch does not need a “Definition”, since all is done with Filter Sets.

The “Begin:” field is checked (the ‘1’). This means, we remember, that WinEdt expects the sequence Filter Set, then Definition. As our “Definition” is empty, it would not matter whether this field is checked or not; the Filter Set alone does the work. But in contrast to possibility Two, let us have this field checked.

⁵Of course, you can specify this Filter Set in other ways, as long as it is logically correct and follows the syntax for Set Expressions.

The “End:” field is not checked, since we do not want the Filter Set after our `Negative Number` to be highlighted. This Filter Set is our just created `Not Numeric` set.

The Switch itself is enabled (since we want the highlighting), the ‘*’ means “all modes”. Of course, you can restrict highlighting of this Switch to specific modes, just enter the names here (multiple entries divided by semicolons).

Priority is ‘2’, that means the Switch can be overridden by Filter Sets, Switches and Reserved Words with higher priority values. The color is set to 3082, which happens to be bright green. The Switch is not case sensitive (why should it?) and it does not need to start at the beginning of the line (but it can). The scope of the Switch is ‘1’, that means it is switched off at the end of the line if it is not already switched off before.

Now we have set up our first highlighting scheme!

4 Reserved Words

Reserved Words are the third syntax element in WinEdt’s Syntax Highlighting algorithm. They are sequences of characters given explicitly, and like Switches, their activation can be controlled by certain conditions, the Filter Sets.

Reserved Words are mostly used to get highlighting of keywords of programming languages. Given WinEdt’s `TEX` background (a programming language to typeset texts), it is not surprising that most of the entries in the default setup of WinEdt’s Reserved Words dialog are aimed at `TEX`.

Highlighting of Reserved Words, however, is not restricted to programming languages. To give a relatively useful example not connected to programming, we wanted to get the word “Miller” highlighted in an ordinary text. If this example doesn’t convince you, think about a foreign business partner, whose complicated name you have to write over and over (I have to resist the temptation to invent a funny name here). Politeness commands you to have the name spelled correctly, and highlighting helps you with this. *Only* the correct spelling will be highlighted. (Of course, WinEdt can help you also with Active Strings, spell checking or word completion.)

4.1 The “Reserved Words” Dialog

The “Reserved Words” Dialog is not very complicated. In the left column, there are two fields, called “Reserved Words” and “Words”. The upper one lists groups of Reserved Words that build a unit, e. g. all keywords for the programming language “C”. In the list-box below you give the words itself, e. g. “while” and “volatile” for the language “C”. To get back to our example, “Miller” could be one word (given one per line in the “Words” field) in the group “Complicated Names” (given in the “Reserved Words” field).

As in all the other dialogs for Syntax Highlighting, a mouse click with the secondary (usually the right) button pops up a context menu with commands for manipulating entries, such as `Inserting` a new one in the field “Reserved Words”.

The field “Filters” lets you specify whether a Filter Set needs to come “Before:” or “After:” the Reserved Word. The checkboxes here define whether the beginning of line (the “Be-

fore:” checkbox) or the end of line (the “After:” checkbox) is *not* an admissible position for the Reserved Word. That means, if you check the “Before:” field, a Reserved Word will not be highlighted if it starts immediately at the beginning of the line.

Reserved Words can be enabled for certain modes. The priority can be specified to determine highlighting of the currently selected group of Reserved Words inside Switches. The “Case Sensitive” checkbox tells whether the words given should be treated case sensitive or case insensitive.

About speed, the WinEdt online help has to say this (in T_EX, every command starts with a backslash): “It makes a difference to specify a T_EX command by its first letter and use ‘\’ as a Filter Set or to define all T_EX Control Sequences under ‘\CS’. The second method is less efficient!” (read: “slower”).

4.2 A small example

Let us look at one of the default entries in WinEdt. It highlights some words that are used in the introductory text file to WinEdt. This group of Reserved Words is called “Special”:

```
Name:Special
  Before:0~Alpha
  After:0~Alpha
  Enabled:1
  Priority:4
  Color:1027
  Case Sensitive:1
TeX
LaTeX
WinEdt
BibTeX
DVIWIN
MiKTeX
YAP
emTeX
```

Both “Before:” and “After:” fields are not checked, so the Reserved Words in this group can start at the beginning of the line and can end at the end of the line. The definition `~Alpha` (“not Alpha”) ensures that the words are *not* highlighted if they are part of another word. This group is enabled, the empty field here tells that highlighting is in effect in all modes. Priority is “4”, so all highlighting with a lower priority is overridden. These words are highlighted with a certain color.

`Case Sensitive:1` tells WinEdt to take care to highlight only words which match the entries in the group in case. Thus, “Latex” is not highlighted, only “LaTeX” is.

This group of Reserved Words contains eight words, namely ‘TeX’, ‘LaTeX’, ‘WinEdt’, ‘BibTeX’, ‘DVIWIN’, ‘MiKTeX’, ‘YAP’ and ‘emTeX’.

Most other entries in the Reserved Words dialog are no more complicated than this one. If you understand what the Filter Set in the “Before:” and “After:” field does, you will have no problem.

4.2.1 “I get incorrect highlighting!”

Sort your keywords in decreasing order (WinEdt has a “Sort Lines” function). If you do not do this, you get wrong highlighting with words that begin with the same letter(s). Sorting the words in decreasing order makes sure that longer words become before shorter ones when having the first letters in common, which is essential in most cases.



Note: this example is a T_EX example. You do not need to know about this language, just note that all T_EX commands start with a backslash.

Say you have the Reserved Words `\par` and `\part`. If you have not taken care to sort the words in decreasing order, WinEdt would highlight only the letters ‘p’, ‘a’ and ‘r’ of `part`. When applying its algorithm, WinEdt goes through the list of Reserved Words and sees that the pattern `\par` matches the word “`\part`” perfectly (see it as a stencil), and highlights it. This looks like “`\par`”. WinEdt does not go further down the list (which can be rather long, and usually is) for speed reasons, so it does not see that the stencil `\part` would match even better.

If, however, you have sorted the words in decreasing order, then `\part` is found first, and correctly applied. WinEdt would not need to go further down the list of Reserved Words, since there will be only `\par`, which does not fit the stencil because of the missing “t”.

There is another way to get correct highlighting in this case, and this has a pleasing side-effect. To be honest, this second way was ‘invented’ in the first place to get the side-effect. So let’s see what the side-effect is. In T_EX or L^AT_EX you can define functions yourself. You could now define a function called `\parent`. Since this is neither standard T_EX or L^AT_EX you will not have it in the Reserved Words list. Without taking care, you will have the “`\par`” part of your function highlighted, resulting in “`\parent`”. This is really annoying.

The solution is to specify an “After:” Filter Set. In our T_EX example, a `~Alpha` (*not* Alpha, you remember) Filter Set matches best, since normally T_EX commands may only have alphabetic characters. The `~Alpha` Filter Set here says: “After a Reserved Word may come everything *except* an alphabetic character.” This prevents highlighting of words that do not fit exactly in writing (while it leniently overlooks all non-alphabetic characters after the word). Our `\parent` function does not fit exactly in writing, and thus is not highlighted. Great.

Let’s get back to the “`\part`” example from the beginning: here too, `\par` does not fit “`\part`” exactly anymore. It does not matter whether the list of Reserved Words is sorted in increasing or decreasing order: WinEdt can not apply its highlighting. It has to find an exact match, and only `\part` matches “`\part`”.

Contents

1	Basics	1
1.1	Filter Sets	2
1.2	Switches	2
1.3	Reserved Words	2
2	Filter Sets	2
2.1	The “Filter Sets” Dialog	3
2.2	Set Expressions	4
2.2.1	Examples:	4
3	Switches	5
3.1	Global Switches	5
3.2	The “Switches” Dialog	5
3.2.1	The “Begin:” Filter and “Definition”	6
3.2.2	Ending a Switch	6
3.2.3	Priority	7
3.3	Examples	7
3.3.1	The Negative Number scheme	7
4	Reserved Words	9
4.1	The “Reserved Words” Dialog	9
4.2	A small example	10
4.2.1	“I get incorrect highlighting!”	11