

General Information

PowerReplace interprets filter file line by line. A line must be as follow:

```
%comments-line
#define-line
first-stuff second-stuff
```

Comments-line has no effect. Define-line is used to set some option. The stuff-line means that the `first-stuff` will be replaced by the `second-stuff` everywhere in the text file. For the `first-stuff`, you can use the format:

```
"first-string"
$hex-string$
'pattern'
```

And for the `second-stuff`, you can use:

```
"second-string"
$hex-string$
```

In other words, a stuff-line must have one of the following format:

```
"first-string"
"second-string"
$hex-string$ "second-string"
'pattern' "second-string"
"first-string"
$hex-string$
$hex-string$ $hex-string$
'pattern' $hex-string$
```

In the following, first, we will give some examples. Then we will give a short description for each stuff. You can find at the end of this Chapter some bad examples with which you can understand better its use.

Note that a good way to learn is to study the sample filter files in the “Filter” folder.

Good Examples:

"´"	"é"	% OK. for HTML
"é"	"´"	% OK. for HTML
"à"	"A"	% OK. lower to upper
"è"	"e"	% OK. 8 bit to 7 bit
"è"	"\\e"	% OK. for TeX. \\ means the character \
"/*\r"	" "	% OK. delete C++ comments

First-String:

- In the simple case, a first-string is a normal string without asterisk-sign(*). You must use meta character for give special characters in the string like CR, NULL.

Example: `"abc"` and `"ab\ce"` are simple-strings. (`\` means the character `"`, see below.)

- In the more general case, a first-string may contain one asterisk-sign(*) in the middle of this string for any indeterminate substring.

For example, "abc*xyz" means all normal string beginning with "abc" and ending with "xyz". (See Chapter 7: Technical Note for a complicate example)

An other example: "/*\r" means all characters after "/*" of a line(C++ comments).

But neither "*ab" nor "a*b*c" is a first-string.

Second-String:

- In the simple case, a second-string is a normal string without asterisk-sign(*).

- In the more general case, a second-string may contain one tag \>. We'll discuss it later in the section "Insertion tag in the second-string."

Character set:

We can tell PowerReplace the special characters by using meta character as:

\\ representing \

\" representing "

* representing *

\t representing TAB

\n representing LF

\r representing CR

PowerReplace supports all characters set in filter, the control characters (0-31) in particular. We can specify characters in filter by using ASCII numbers in decimal, hexadecimal or octal base.

For example, All of "\d065" "\x41" "\o101" (or "\101") means the same character "A". Examples:

"\d65B" "B\101\x43" % OK. change "AB" to "BAC"

"\0" "" % OK. strip NULL

Define(#):

◇ The default meta character used by PowerReplace is "\". You can change it. For example, to set "/" as meta character, just insert the following define-line in your filter file:

#meta "/"

If you don't use any meta character, insert the following define-line:

#meta ""

For example, to replace è by \e, we can use the following line:

"è" "\e" % OK. for TeX. \e means the character \

or the following two lines:

#meta "/"

```
"è"      "\`e"      % OK. for TeX. now \ isn't meta character!
```

I use the default meta character "\`" in my documentation.

◇ The modification of the meta "\`" is not valid for the regular expression. It is only valid for the first-string and second-string (with ").

◇ See Chapter 7 - Technical Note for an other define #type.

Hexadecimal string:

The hex-string must be enclosed by dollar-sign(\$hex-string\$). You can use one of the following lines to convert "AB" to "BAC":

```
"AB"      "BAC"      % OK. change "AB" to "BAC"
$6566$    $666567$    % OK. change "AB" to "BAC"
$6566$    "BAC"      % OK. change "AB" to "BAC"
"AB"      $666567$    % OK. change "AB" to "BAC"
```

Regular expression (pattern):

This version supports regular expression (pattern) for searching string. The pattern must be enclosed by single quotation marks ('pattern'). We can only use pattern for first-stuff but not for the second-stuff. Here is a small description of pattern supported by PowerReplace:

An ordinary character (not mentioned below) matches that character.

```
^    matches beginning of line
$    matches end of line, including '\r' at the end of line.
\x   quotes character after it x, except
      \t   quotes   TAB
      \n   quotes   LF

      \r   quotes   CR

.    matches any character
*    a single character followed by * matches
      zero or more occurrences of the character.
      In particular, ".*" matches an arbitrary possibly empty string.
+    a single character followed by + matches
      one or more occurrences of the character.
[]   a set of characters in the set matches any single character in the set.
      [c1-c2] matches any character of ascii ranging from character c1 to character c2.
      [^set] matches any character not in set.
```

See also any Unix book for more information about pattern. Example:

```
'^ +'      % OK. strip spaces at beginning of line.
'^[ \t ]+' % OK. strip spaces and tab at beginning of line.
'\r'
"--\r"
% OK. replace blanc line by --.
'\r'
"--\r"
```

```
% OK. replace blanc line by --.
'^\r'
'''
```

```
% OK. remove blanc line
'\r+'
''
% OK. remove blanc line
```

Insertion tag in the second-string:

In the second-string, you can use a new tag `\>` for inserting a substring of the first-string-found (it is a variable string depending the input text, called also “source”). We use two parameters (x,y) to define this substring: It takes x letters at the beginning and y letters at the end of the source. By default, if (x,y) are missing, this substring is just the source.

Example1: My text is "hello". If the first-string is "e*o", then the source will be "ello". We study the following filter lines:

- (1) "e*o" "\> and goodbye"
- (2) "e*o" "\> and goodbye" (all, 0)
- (3) "e*o" "\> and goodbye" (0, all)
- (4) "e*o" "\> and goodbye" (all, all)
- (5) "e*o" "\> and goodbye" (0, 0)
- (6) "e*o" "\> and goodbye" (1, 0)
- (7) "e*o" "\> and goodbye" (0, 2)
- (8) "e*o" "\> and goodbye" (1, 1)

then, the output file will contain the following text for each case:

- (1,2,3) "hello and goodbye"
- (4) "helloello and goodbye"
- (5) "h and goodbye"
- (6) "he and goodbye"
- (7) "hlo and goodbye"
- (8) "heo and goodbye"

Example2: We want to change "à" to "a" if and only if it is the last letter of a word. The solution whitout the insertion tag is writing the following lines:

```
"à," "a,"
"à;" "a;"
"à:" "a:"
"à." "a."
"à!" "a!"
"à?" "a?"
"à " "a "
```

With the insert tag, you need only the following line:

```
'à[,:;!/? ]' "a\>" (0,1)
```

Or this line:

```
'à[^a-zA-Z]' "a\>" (0,1)
```

◇ You can put more than one insertion tag in the second string. For example,

```
"abc*xyz "1:\> ... 2:\> ... 3:\>" (0,1) (3,0)
```

The first insertion tag takes one character at the end of the source: (0,1), the second takes three character at the beginning of the source: (3,0), and the third takes all characters of the source.

Bad Examples:

```
'^ +'      'abc'      % BAD. pattern at the second position
$A9876$    ""         % BAD. bad hex-string, length odd
"a*b*c"    "\xB3"     % BAD. two * in the first-string, try "a*b\c" "\xB3"
'[aeiou]'  " "        % BAD. pattern syntax error
"a"a"      "AA"       % BAD. syntax error. try "a\"a" "AA"
"\da"      "a"        % BAD. decimal number error
"$23"      "98"       % BAD. syntax error. try "$23$ "98"
$A567BDG$  "98"      % BAD. bad hex-string
"toto"     "t*t"      % BAD. bad second-string, try "toto" "t\t"
```