NAME

Highlight - a universal sourcecode to formatted text converter

SYNOPSIS

highlight [*OPTIONS*]... [*FILES*]...

DESCRIPTION

Highlight converts sourcecode to HTML, XHTML, RTF, ODT, LaTeX, TeX, BBCode, Pango markup, SVG, XTERM or ANSI escape sequences. There are several colour themes available. **Highlight** recognizes keywords, numbers, strings, comments, symbols and preprocessor directives. It supports about 180 programming languages, which are defined in Lua scripts.

It's easily possible to enhance highlight's database of programming languages and colour themes. See the README file for details.

GENERAL OPTIONS

-B, **--batch-recursive**=<*wildcard*>
convert all files matching the wildcard (uses recursive search)

-D, **--data-dir**=<*path*>
set path to highlight data directory

--config-file=<*file*>
set path to a lang or theme file

-h, **--help**[=*topic*]
print this help or a topic description <topic> = [syntax, theme, plugin, config]

-i, **--input**=<*file*>
name of input file

-o, **--output**=<*file*>
name of output file

-d, **--outdir**=<*output directory*>
name of output directory

-P, **--progress**
print progress bar in batch mode

-S, **--syntax**=<*type*|*path*>
set type of source code, necessary if input file suffix is missing. The syntax may also be defined as path of the language file.

--syntax-by-name=<*name*>
specify type of source code by given name. Will not read a file of this name, useful for stdin and to determine the syntax of the file before piping its content to highlight. This option overrides --syntax.

--syntax-supported
test if the given syntax can be loaded and print the result (assumes -S or --syntax-by-name)

-v, **--verbose**
print debug info to stderr; repeat to show more information

-q, **--quiet**
suppress progress info in batch mode

--force[=*syntax*]
generate output if input syntax is unknown. The fallback syntax may be set here, Plain Text is default.

--list-scripts=<*type*>
list installed scripts <type> = [langs, themes, plugins]

--list-cat=<*categories*>
filter the scripts by the given categories (example: --list-cat='source;script')

--max-size=<*size*>
set maximum input file size (examples: 512M, 1G; default: 256M)

--plug-in=<*script*>
execute Lua plug-in script; repeat option to apply multiple plug-ins

--plug-in-param
set plug-in input parameter. This might be an input file name (ie. 'tags').

--print-config
print path configuration

--print-style
print stylesheet only (see --style-outfile)

--skip=<*list*>
ignore listed unknown file types (example: --skip='bak;c~;h~')

--stdout
output to stdout (batch mode, --print-style)

--validate-input
test if input is a valid text file

--service-mode
run in service mode, not stopping until signaled

--version
print version and copyright info

OUTPUT FORMATTING OPTIONS

-O, **--out-format**=<*format*>
output file in given format <format>=[html, xhtml, latex, tex, rtf, odt, ansi, xterm256, truecolor, bbcode, pango, svg]

-c, **--style-outfile**=<*file*>
name of style definition file

-T, **--doc-title**
document title

-e, **--style-infile**=<*file*>
name of file to be included in style-outfile

-f, **--fragment**
omit header and footer of the output document (see --keep-injections)

-F, **--reformat**=<*style*>

reformat output in given style. <style>=[allman, gnu, google, horstmann, java, kr, linux, lisp, mozilla, otbs, pico, vtk, ratliff, stroustrup, webkit, whitesmith]

-I, **--include-style**
include style definition in output

-J, **--line-length**=<*num*>
line length before wrapping (see -V, -W)

-j, **--line-number-length**=<*num*>
line number length incl. left padding. Default length: 5

-k, **--font**=<*font*>
set font (specific to output format)

-K, **--font-size**=<*num?*>
set font size (specific to output format)

-l, **--line-numbers**
print line numbers in output file

-m, **--line-number-start**=<*cnt*>
start line numbering with cnt (assumes -l)

--line-range=<*start-end*>
output only lines from number <start> to <end>

-s, **--style**=<*style name|path*>
set highlighting style (theme). Add 'base16/' prefix to use a Base16 theme. The theme may also be defined as path of the theme file.

-t **--replace-tabs**=<*num*>
replace tabs by num spaces

-u, **--encoding**=<*enc*>
set output encoding which matches input file encoding; omit encoding information if set to "NONE"

-V, **--wrap-simple**
wrap lines after 80 (default) characters without indenting function parameters and statements.

-W, **--wrap**
wrap lines after 80 (default) characters (use with caution).

-z, **--zeroes**
fill leading space of line numbers with zeroes

--isolate
output each syntax token in separate tags (verbose output)

--keep-injections
output plug-in header and footer injections in spite of -f

--kw-case=<*upper*|*lower*|*capitalize*>
output all keywords in given case if language is not case sensitive

--no-trailing-nl[=*mode*]
omit trailing newline. If mode is "empty-file", omit only for empty input

--no-version-info
omit version info comment

--wrap-no-numbers
omit line numbers of wrapped lines (assumes -l)

(X)HTML OPTIONS

-a, **--anchors**
attach anchors to line numbers (HTML only)

-y, **--anchor-prefix**=<*str*>
set anchor name prefix

-N, **--anchor-filename**
use input file name as anchor name

-C, **--print-index**
print index file with links to all output files

-n, **--ordered-list**
print lines as ordered list items

--class-name=<*str*>
set CSS class name prefix; omit class name if set to "NONE"

--inline-css
output CSS within each tag (verbose output)

--enclose-pre
enclose fragmented output with pre tag (assumes -f)

LATEX OPTIONS

-b, **--babel**
disable Babel package shorthands

-r, **--replace-quotes**
replace double quotes by \dq

--beamer
adapt output for the Beamer package

--pretty-symbols
improve appearance of brackets and other symbols

RTF OPTIONS

--page-color
include page color attributes

-x, **--page-size**=<*size*>
set page size, <size>=[a3, a4, a5, b4, b5, b6, letter]

--char-styles
include character stylesheets

SVG OPTIONS

--height=<*h*>
set image height (units allowed)

--width=<*w*>

set image size (see --height)

TERMINAL ESCAPE OUTPUT OPTIONS (XTERM256 OR TRUECOLOR)

--canvas[=*width*]
set background colour padding (default: 80)

LANGUAGE SERVER OPTIONS

--ls-profile=<*server*>
load LSP configuration from lsp.conf

--ls-delay=<*ms*>
set server initialization delay in milliseconds

--ls-exec=<*bin*>
set server executable name

--ls-option=<*option*>
set server CLI option (can be repeated)

--ls-hover
execute hover requests (HTML output only)

--ls-semantic
query server for semantic token types (requires LSP 3.16)

--ls-syntax=<*lang*>
set syntax which is understood by the server

--ls-syntax-error
retrieve syntax error information (assumes --ls-hover or --ls-semantic)

--ls-workspace=<*dir*>
set workspace directory to initialize the server

--ls-legacy
do not require a server capabilities response

ENV VARIABLES

Highlight recognizes these variables:

HIGHLIGHT_DATADIR
sets the path to highlight's configuration scripts

HIGHLIGHT_OPTIONS
may contain command line options, but no input file paths.

HINTS

If no in- or output files are specified, stdin and stdout will be used for in- or output. Reading from stdin can also be triggered by the '-' option.

Default output format: xterm256 or truecolor if appropriate, HTML otherwise.

Style definitions are stored in highlight.css (HTML, XHTML, SVG) or highlight.sty (LaTeX, TeX) if neither -c nor -I is given. For CSS, definitions are stored in the output document header with -I, if -f is also given there will be no style definitions.

Reformatting code (-F) will only work with C, C++, C# and Java input files.

LSP features require absolute input paths and disable reformatting (-F).

BUGS

Wrapping lines with -V or -W will cause faulty highlighting of long single line comments and directives. Using line-range might interfere with multi line syntax elements. Use with caution.

FILES

The configuration files are stored in */usr/share/highlight/*. Language definitions, themes and plugins are located in subdirectories.

Documentation files are stored in */usr/share/doc/highlight/*, configuration files in */etc/highlight/*.

See README how to install own scripts in the home directory.

EXAMPLES

Single file conversion:

highlight -o hello.html -i hello.c

highlight -o hello.html hello.c

highlight -o hello.html -S c < hello.c

highlight -S c < hello.c > hello.html

Note that a file highlight.css is created in the current directory.

Batch file processing:

highlight --out-format=xhtml -B '*.cpp' -d /home/you/html_code/

converts all *.cpp files in the current directory and its subdirectories to xhtml files, and stores the output in /home/you/html_code.

highlight --out-format=latex * -d /home/you/latex_code/

converts all files to LaTeX, stored in /home/you/latex_code/.

Use --quiet to improve performance of batch file processing (recommended for usage in shell scripts).

Use highlight --out-format=xterm256 <yourfile> | less -R to display a source file in a terminal.

Run highlight --list-scripts=langs to see all supported syntax types.

AUTHORS

Andre Simon <as@andre-simon.de>

SEE ALSO

README files and http://www.andre-simon.de/.