Typeset Code Listings and Emulate Console Screenshots with LaTeX Beautifully

https://github.com/xziyue/latex-beautiful-listings-screenshot

Ziyue "Alan" Xiang

May 10, 2020

Contents

1 Quick Start Guide

1. Download `customlisting.sty` and place it in your project folder.

2. Load the package with `\usepackage{customlisting}`.

3. If you are using pdfLaTeX, make sure to include `\usepackage[T1]{fontenc}` in the preamble. Otherwise, symbols like ~ may not be displayed correctly.

This package provides the following environments:

- `tcbconsole`, `tcbconsole*`

- `tcbcode`, `tcbcode*`

- `tcbverbatim`, `tcbverbatim*`

This package also provides the following commands:

- `tcbinputcode`, `tcbinputcode*`

- `tcbinputverbatim`, `tcbinputverbatim*`

The starred environments/commands offer *unbreakable* listing boxes; while normal ones are *breakable*.

2 Typeset Source Code Listings

- Typeset source code inside TeX files

```
1  \begin{tcbcode}{cpp}
2  #include <iostream>
3  using namespace std;
4
5  int main(){
6      cout<<"Hello World\n";
7      return 0;
8  }
9  \end{tcbcode}
```

Code
```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout<<"Hello World\n";
6      return 0;
7  }
```

- Typeset source code from external source files

```
1  \tcbinputcode*{cpp}{../res/example.cpp}
```

Code
```cpp
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout<<"Hello World\n";
6      return 0;
7  }
```

- Inline source code

```
1  \cinline|printf("%s", "some text");|
2  \pyinline|map(lambda x:x, [1, 2])|
3  \rawinline|raw value|
```

```
printf("%s", "some text"); map(lambda x:x, [1, 2]) raw value
```

- Declare inline macros for other languages

```
1  \newmintinline[rubyinline]{ruby}{frame=none, fontsize=\fontsize{10}{10}}
2  \rubyinline|puts 'Hello, world!'|
```

```ruby
puts 'Hello, world!'
```

3 Typeset Generic Verbatims

- Typeset generic verbatims inside TEX files

```
1  \begin{tcbverbatim}
2   _____  ___
3  \__       ___/\_    _____/\   \/   /
4     |    |    |     __)_  \       /
5     |    |    |        \ /        \
6     |____|   /_____  //___/\  \
7                        \/         \_/
8  \end{tcbverbatim}
```

Verbatim
```
 _____  ___
\__       ___/\_    _____/\   \/   /
  |    |    |     __)_  \       /
  |    |    |        \ /        \
  |____|   /_____  //___/\  \
                     \/         \_/
```

- Typeset generic verbatims from external files

```
1  \tcbinputverbatim*{../res/wireshark.txt}
```

```
No.     Time         Source               Destination       Protocol Length Info
118 0.159070602  2604:6000:1419:404a::6 2607:f8b0:4009:805::2004 TCP      86     42136> 443 [ACK] Seq=936 Ack=52751 Win=2933 Len=0 TSval=2838670553 TSecr=302725484
119 0.177751097  192.168.0.1          192.168.0.100     DNS      140    Standard query response 0x829b A fonts.gstatic.com CNAME gstaticadssl.l.google.com A 172.217.1.35 OPT
120 0.178038905  192.168.0.1          192.168.0.100     DNS      152    Standard query response 0x0ec8 AAAA fonts.gstatic.com CNAME gstaticadssl.l.google.com AAAA 2607:f8b0
    :4009:802::2003 OPT
121 0.178444739  2604:6000:1419:404a::6 2607:f8b0:4009:802::2003 TCP      94     39596> 443 [SYN] Seq=0 Win=64800 Len=0 MSS=1440 SACK_PERM=1 TSval=4016727607 TSecr=0 WS=128
122 0.180362133  2607:f8b0:4009:805::2004 2604:6000:1419:404a::6 TCP      86     443> 42188 [ACK] Seq=1883 Ack=1337 Win=68096 Len=0 TSval=319629076 TSecr=2838670536
123 0.185824541  2607:f8b0:4009:806::2003 2604:6000:1419:404a::6 TCP      86     443> 53154 [ACK] Seq=1 Ack=591 Win=66816 Len=0 TSval=1161394810 TSecr=356743249
124 0.187455681  2607:f8b0:4009:806::2003 2604:6000:1419:404a::6 TLSv1.3  298    Server Hello, Change Cipher Spec, Application Data
125 0.187460441  2604:6000:1419:404a::6 2607:f8b0:4009:806::2003 TCP      86     53154> 443 [ACK] Seq=591 Ack=213 Win=64768 Len=0 TSval=356743285 TSecr=1161394811
126 0.187881868  2604:6000:1419:404a::6 2607:f8b0:4009:806::2003 TLSv1.3  150    Change Cipher Spec, Application Data
127 0.187975438  2604:6000:1419:404a::6 2607:f8b0:4009:806::2003 TLSv1.3  172    Application Data
128 0.188000758  2604:6000:1419:404a::6 2607:f8b0:4009:806::2003 TLSv1.3  323    Application Data
129 0.188267503  2604:6000:1419:404a::6 2607:f8b0:4009:806::2003 TLSv1.3  649    Application Data
130 0.194787446  2607:f8b0:4009:805::2004 2604:6000:1419:404a::6 TCP      86     443> 42136 [ACK] Seq=52751 Ack=936 Win=609 Len=0 TSval=302725520 TSecr=2838670548
131 0.211610817  2607:f8b0:4009:802::2003 2604:6000:1419:404a::6 TCP      94     443> 39596 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1360 SACK_PERM=1 TSval=3015134314 TSecr=4016727607
    WS=256
132 0.211641489  2604:6000:1419:404a::6 2607:f8b0:4009:802::2003 TCP      86     39596> 443 [ACK] Seq=1 Ack=1 Win=64896 Len=0 TSval=4016727640 TSecr=3015134314
```

4 Typeset Console Screenshots

Typesetting console screenshots is a bit trickier. By far, it can be done most conveniently on Ubuntu 18.04+. The key is to convert ANSI color codes used by the console into HTML. As it is shown in Figure 1, on Ubuntu 18.04+, this can be done simply by selecting the desired region, right click and select "Copy as HTML". On other platforms, this should be also doable by dumping the terminal output to a file and using a conversion tool such as `ansi2html`.

Figure 1: Converting terminal output to HTML on Ubuntu 18.04+.

Generally speaking, one needs to fulfill the following requirements:

1. Have a way of converting terminal output to HTML.

2. Be able to run the `html2latex` LaTeX Python script. Currently, the script is dependent on `wxPython`, `TexSoup`, colour and `PyLaTeX`. Please note that this software is very primitive and does not support many HTML features.

To typeset this screenshot in LaTeX, one needs to run `html2latex` and paste the HTML in the upper text box. By pressing the "Convert" button, the corresponding LaTeX code will appear in the lower text box, as it is shown in Figure 2. The result is shown as below.

```
1  \input{../res/console-dev.txt}
```

```
ashmem           loop17          psaux      tty30       ttyS2
autofs           loop18          ptmx       tty31       ttyS20
binder           loop19          ptp0       tty32       ttyS21
block            loop2           pts        tty33       ttyS22
bsg              loop20          random     tty34       ttyS23
btrfs-control    loop21          rfkill     tty35       ttyS24
bus              loop22
```

HTML2LaTeX   ● ● ●

```
style="background-color:#2E3436"><font color="#FCE94F"><b>tty33</b></font></span>    <span
style="background-color:#2E3436"><font color="#FCE94F"><b>ttyS22</b></font></span>
<font color="#729FCF"><b>bsg</b></font>         <span style="background-color:#2E3436"><font
color="#FCE94F"><b>loop20</b></font></span>        <span style="background-color:#2E3436"><font
color="#FCE94F"><b>random</b></font></span>  <span style="background-color:#2E3436"><font
color="#FCE94F"><b>tty34</b></font></span>        <span style="background-color:#2E3436"><font
color="#FCE94F"><b>ttyS23</b></font></span>
<span style="background-color:#2E3436"><font color="#FCE94F"><b>btrfs-control</b></font></span>  <span
style="background-color:#2E3436"><font color="#FCE94F"><b>loop21</b></font></span>        <span
style="background-color:#2E3436"><font color="#FCE94F"><b>rfkill</b></font></span>  <span
style="background-color:#2E3436"><font color="#FCE94F"><b>tty35</b></font></span>    <span
style="background-color:#2E3436"><font color="#FCE94F"><b>ttyS24</b></font></span>
<font color="#729FCF"><b>bus</b></font>         <span style="background-color:#2E3436"><font
color="#FCE94F"><b>loop22</b></font></span></pre>
```

```
{
\definecolor{xxxhtmlcolorIEJKJM}{HTML}{2E3436}
\definecolor{xxxhtmlcolorFCEPKF}{HTML}{FCE94F}
\definecolor{xxxhtmlcolorNIPFCF}{HTML}{729FCF}
\setlength{\fboxsep}{0pt}
\begin{tcbconsole}
%*\smash{\colorbox{xxxhtmlcolorIEJKJM}{{\color{xxxhtmlcolorFCEPKF}{\bfseries ashmem}}}}*)
%*\smash{\colorbox{xxxhtmlcolorIEJKJM}{{\color{xxxhtmlcolorFCEPKF}{\bfseries loop17}}}}*)
%*\smash{\colorbox{xxxhtmlcolorIEJKJM}{{\color{xxxhtmlcolorFCEPKF}{\bfseries psaux}}}}*)
%*\smash{\colorbox{xxxhtmlcolorIEJKJM}{{\color{xxxhtmlcolorFCEPKF}{\bfseries tty30}}}}*)
%*\smash{\colorbox{xxxhtmlcolorIEJKJM}{{\color{xxxhtmlcolorFCEPKF}{\bfseries ttyS2}}}}*)
%*\smash{\colorbox{xxxhtmlcolorIEJKJM}{{\color{xxxhtmlcolorFCEPKF}{\bfseries autofs}}}}*)
%*\smash{\colorbox{xxxhtmlcolorIEJKJM}{{\color{xxxhtmlcolorFCEPKF}{\bfseries loop18}}}}*)
%*\smash{\colorbox{xxxhtmlcolorIEJKJM}{{\color{xxxhtmlcolorFCEPKF}{\bfseries ptmx}}}}*)
```
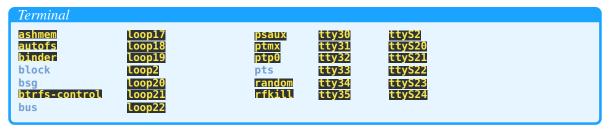
☐ XeLaTeX   ☐ Use Hex Char Code   Convert

Figure 2: Using `html2latex` to convert HTML to LaTeX.

Other classic command-line tools, such as `emacs`, are supported as well.

```
1  \input{../res/console-emacs.txt}
```

```
File Edit Options Buffers Tools Help
Emacs tutorial.  See end for copying conditions.

Emacs commands generally involve the CONTROL key (sometimes labeled
CTRL or CTL) or the META key (sometimes labeled EDIT or ALT).  Rather than
write that in full each time, we'll use the following abbreviations:

 C-<chr>  means hold the CONTROL key while typing the character <chr>
          Thus, C-f would be: hold the CONTROL key and type f.
 M-<chr>  means hold the META or EDIT or ALT key down while typing <chr>.
          If there is no META, EDIT or ALT key, instead press and release the
          ESC key and then type <chr>.  We write <ESC> for the ESC key.

Important note: to end the Emacs session, type C-x C-c.  (Two characters.)
To quit a partially entered command, type C-g.
The characters ">>" at the left margin indicate directions for you to
```

4

```
  try using a command.  For instance:

>> Now type C-v (View next screen) to move to the next screen.
   (go ahead, do it by holding down the CONTROL key while typing v).
         From now on, you should do this again whenever you finish
         reading the screen.
-UU-:----F1  TUTORIAL     Top L1     (Fundamental) --------------------------
```

4.1 Unicode Support

Very frequently, the terminal output contains Unicode characters. For TeX distribution that supports Unicode input natively (e.g. XƎLATEX, LuaLATEX), this should not be a problem. Just remember to tick the "XeLaTeX" check box in `html2latex`.

As for the most commonly used pdfLATEX, special treatment is needed. The solution is to use the `\unichar` command provided by loading `\usepackage[utf8x]{inputenc}`. Therefore, if you are using pdfLATEX and there is Unicode character inside the terminal output, you should do the following:

1. Make sure to include `\usepackage[utf8x]{inputenc}` in your preamble.

2. Put `customlisting-unicode.sty` into your project folder and load it with

   `\usepackage{customlisting-unicode}`.

3. In `html2latex`, make sure "XeLaTeX" is unchecked.

A pdfLATEX example is shown as below.

```
1  \input{../res/console-unicode.txt}
```

Terminal
```
(base) user@machine:~/latex_typeset_listings/res$ cat unicode-test.txt
Basic Latin
! " # $ % & ' ( ) * +
Latin-1 Supplement
  ¡ ¢ £ ¤ ¥ ¦ § ¨ © ª
Latin Extended-A
Ā ā Ă ă Ć ć Ĉ ĉ Ċ
```

However, keep in mind that **this Unicode support is extremely limited**: many characters are simply unavailable in pdfLATEX. For better Unicode support, one should use XƎLATEX or LuaLATEX.

5 Add Captions

To support captions, one needs to load the `caption` package in the preamble and add some related definitions.

```
1  \usepackage{caption}
2
3  \newenvironment{mylisting}{\medskip\captionsetup{type=listing, labelsep=space}}{\medskip}
4  \DeclareCaptionType{lstcap}[Listing][List of Code Listings]
```

This allows one to add caption to code listings with the following code. The "List of Code Listings" can be generated with `\listoflstcaps`.

```
\begin{mylisting}
\begin{tcbcode*}{julia}
function quadratic2(a::Float64, b::Float64, c::Float64)
    # unlike other languages 2a is equivalent to 2*a
    # a^2 is used instead of a**2 or pow(a,2)
    sqr_term = sqrt(b^2-4a*c)
    r1 = quadratic(a, sqr_term, b)
    r2 = quadratic(a, -sqr_term, b)
    # multiple values can be returned from a function using tuples
    # if the return keyword is omitted, the last term is returned
    r1, r2
end
\end{tcbcode*}
\end{mylisting}
\listoflstcaps
```

Code

```
function quadratic2(a::Float64, b::Float64, c::Float64)
    # unlike other languages 2a is equivalent to 2*a
    # a^2 is used instead of a**2 or pow(a,2)
    sqr_term = sqrt(b^2-4a*c)
    r1 = quadratic(a, sqr_term, b)
    r2 = quadratic(a, -sqr_term, b)
    # multiple values can be returned from a function using tuples
    # if the return keyword is omitted, the last term is returned
    r1, r2
end
```

Listing 1: Some random Julia function.

List of Code Listings