l3customlisting
Typeset Code Listings and Emulate Console Screenshots with LaTeX Beautifully

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

Ziyue "Alan" Xiang

June 1, 2020

Contents

This is the LaTeX3 version of the old `customlisting` package, which allows users to control the styles and declare new listing environments more easily. The old package can be found in `archive` folder.

1 Quick Start Guide

1. Download `l3customlisting.sty` and place it in your project folder.
2. Load the package with `\usepackage{l3customlisting}`.
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

```
\begin{tcbcode}{cpp}
#include <iostream>
using namespace std;

int main(){
    cout<<"Hello World\n";
    return 0;
}
\end{tcbcode}
```

Code

```cpp
#include <iostream>
using namespace std;

int main(){
    cout<<"Hello World\n";
    return 0;
}
```

- Typeset source code from external source files

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

Code

```cpp
#include <iostream>
using namespace std;

int main(){
    cout<<"Hello World\n";
    return 0;
}
```

- Inline source code

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

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

- Declare inline macros for other languages

```
\ctmlstnewinline{rubyinline}{ruby} %{macro name}{language}
\rubyinline|puts 'Hello, world!'|
```

`puts 'Hello, world!'`

3 Typeset Generic Verbatims

- Typeset generic verbatims inside TEX files

```
\begin{tcbverbatim}
 _____________________________  ___
\__    ___/\_   _____/\   \/  /
  |    |    |    __)_  \     /
  |    |    |        \ /     \
  |____|   /_______  //___/\  \
                   \/       \_/
\end{tcbverbatim}
```

Verbatim

```
 _____________________________  ___
\__    ___/\_   _____/\   \/  /
  |    |    |    __)_  \     /
  |    |    |        \ /     \
  |____|   /_______  //___/\  \
                   \/       \_/
```

- Typeset generic verbatims from external files

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

Verbatim

```
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`.

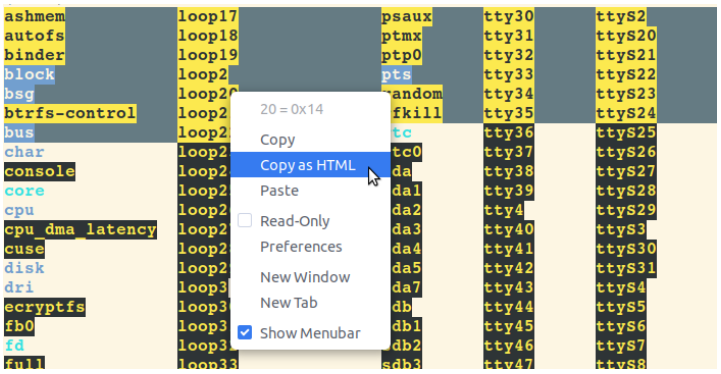

Figure 1: Converting terminal output to HTML on Ubuntu 18.04+.

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` and `colour`. Please note that this software is very primitive and does not support many HTML features. If any problem occurs, you can try the old version in the `archive` folder.

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}
```

Console

```
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
```

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

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

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

Console

```
File Edit Options Buffers Tools Info Help
No next, prev or up links  --  This is the top of the INFO tree
(dir)Top

  This (the Directory node) gives a menu of major topics.
  Typing "q" exits, "H" lists all Info commands, "d" returns here,
  "h" gives a primer for first-timers,
  "mEmacs<Return>" visits the Emacs manual, etc.

  In Emacs, you can click mouse button 2 on a menu item or cross reference
  to select it.

* Menu:

Basics
* Common options(coreutils)
```

```
* Coreutils             Core GNU (file, text, shell) utilities.
* Date input formats(coreutils)
* Ed                    The GNU line editor
* File permissions(coreutils)
                        Access modes.
* Finding files         Operating on files matching certain criteria.
-UUU:%%--F1  *info*(dir) Top  Top L2     (Info Narrow) ----------------------
Info file emacs does not exist
```

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

Console

```
1  [      0.0%]   4  [      0.0%]   7  [      0.0%]   10 [|     0.7%]
  2  [||    2.1%]   5  [      0.0%]   8  [|     0.7%]   11 [||    5.4%]
  3  [      0.0%]   6  [      0.0%]   9  [||    4.1%]   12 [      0.0%]
  Mem[||||||||||         4.66G/31.3G]   Tasks:  192, 718thr; 1running
  Swp[                      0K/8.00G]   Load average:  0.67 0.64 0.68
                                        Uptime:  03:54:45

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
20803 alan       20   0 13.8G 1628M 85092 S  0.7  5.1  1:03.72 /snap/pycharm-pro
20812 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:09.76 /snap/pycharm-pro
20813 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:06.06 /snap/pycharm-pro
20840 alan       20   0 13.8G 1628M 85092 S  0.7  5.1  0:04.65 /snap/pycharm-pro
20824 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:01.94 /snap/pycharm-pro
20843 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:00.96 /snap/pycharm-pro
 5364 alan       20   0 4547M  454M 98400 S  8.7  1.4 13:26.42 /usr/bin/gnome-sh
20823 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:00.97 /snap/pycharm-pro
20807 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:01.54 /snap/pycharm-pro
20845 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:00.56 /snap/pycharm-pro
20828 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:01.29 /snap/pycharm-pro
 5225 root       20   0  578M  100M 80364 S 12.0  0.3 13:34.04 /usr/lib/xorg/Xor
20844 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:02.04 /snap/pycharm-pro
20846 alan       20   0 13.8G 1628M 85092 S  0.0  5.1  0:00.55 /snap/pycharm-pro
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

4.1 Unicode Support

Very frequently, the terminal output contains Unicode characters. For TEXdistribution that supports Unicode input natively (e.g. XELATEX, 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. In `html2latex`, make sure "XeLaTeX" is unchecked.

A pdfLATEX example is shown as below.

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

Console

```
(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. Many packages are not compatible with `\usepackage[utf8x]{inputenc}`. One most notable example is `biblatex`. Therefore, for better Unicode support, one should use XeLaTeX or LuaLaTeX.

5 Add Captions

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

```
\usepackage{caption}

\newenvironment{mylisting}{\medskip\captionsetup{type=listing, labelsep=space}}{\medskip}
\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

6 Customize Listings

6.1 Changing appearance of existing listings

6.1.1 Common styles

Common styles can be updated with `\tcbset`.

- The common style of `listing`-based environments are defined by

```
\tcbset{ctmlstlistingstyle/.style=
    {listing only, enhanced jigsaw, boxsep=0pt, top=0pt, bottom=0pt, left=2mm, right=2mm,
    ↪ boxrule=2pt}
}
```

- The common style of `minted`-based environments are defined by

```
\tcbset{ctmlstmintedstyle/.style=
    {listing only, enhanced jigsaw, boxsep=0pt, top=3pt, bottom=3pt, left=8mm, right=2mm,
    ↪ boxrule=2pt}
}
```

6.1.2 Color and font

Denote name of environment name by `<name>`:

- The frame color is given by `tcb<name>cf` (modify with `\definecolor`)
- The background color is given by `tcb<name>cb`
- The font style is given by `\tcb<name>font` (modify with `\renewcommand`)

For example, to change the background color and font of `tcbverbatim`, we can simply run:

```
\definecolor{tcbverbatimcb}{HTML}{efefef}
\renewcommand{\tcbverbatimfont}{\linespread{0.9}\fontsize{8}{8}\fontfamily{lmr}}
\begin{tcbverbatim}
modified verbatim
\end{tcbverbatim}
```

Verbatim

```
modified verbatim
```

6.2 Changing style of existing inline commands

Inline styles are defined in `\ctmlstinlineoptions`. After updating them, one should refresh macro definitions with `\ctmlstrenewinline`.

```
\cinline|int main();|
\renewcommand{\ctmlstinlineoptions}{frame=none, fontsize=\fontsize{15}{15}}
\ctmlstrenewinline{cinline}{c}
\cinline|int main();|
```

`int main();` int main();

6.3 Declaring new listing environments/commands

To declare a new `listing`-based environment (denote the name by `<name>`), one needs declare the following variables:

- Frame color `tcb<name>cf`
- Background color `tcb<name>cb`
- Font style `\tcb<name>font`

For example, if one wants to define a new environment called `tcbtext`, the following commands should be called:

```
\definecolor{tcbtextcf}{HTML}{000000}
\definecolor{tcbtextcb}{HTML}{efefef}
\newcommand{\tcbtextfont}{\fomtfamily{lmr}\fontsize{10}{10}}
\ExplSyntaxOn
\__ctmlst_new_listings:nNnnn {text} {\__ctmlst_listingbased_style:VVcn} {0} {breakable} {}
\ExplSyntaxOff

\begin{tcbtext}
Sample text.
\end{tcbtext}
```

The output is shown as below.

Text

```
Sample text.
```

For more information on `\__ctmlst_new_listings:nNnnn`, please refer to the comments.

6.4 More details

For more control over new environments/commands, one needs to create a *style generator*. There are two existing style generators, one for `listing`-based environments and another for `minted`-based environments. Their definitions are shown below.

```
% command to generate style list for listing-based tcblisting
% #1: name of the listing (no star involved, e.g. console, code, verbatim, etc.)
% #2: title of the listing
% #3: csname of font style
% #4: additional parameters
\cs_set:Npn \__ctmlst_listingbased_style:nnNn #1#2#3#4 {
    ctmlstlistingstyle,
    colback=#1cb,
    colframe=#1cf,
    title=\exp_not:n{#2},
    listing\space options={style=ctmlststyle,
        backgroundcolor=\exp_not:N\color{#1cb},
```

```
        basicstyle=\exp_not:n{#3\selectfont}},
    #4
}

% command to generate style list for minted-based tcbinputlisting
% #1: name of the listing (no star involved, e.g. console, code, verbatim, etc.)
% #2: title of the listing
% #3: csname of font style
% #4: additional parameters
\cs_set:Npn \__ctmlst_mintedbased_style:nnNn #1#2#3#4 {
    ctmlstmintedstyle,
    listing\space engine=minted,
    colback=#1cb,
    colframe=#1cf,
    title=\exp_not:n{#2},
    minted\space options={
        \ctmlstmintedoptions,
        fontsize=\exp_not:n{#3\selectfont}
    },
    #4
}
```

The macros to declare new environments/commands takes a style generator and additional parameters as inputs.

```
\__ctmlst_new_listings:nNnnn {console} {\__ctmlst_listingbased_style:VVcn} {0} {breakable} {}
\__ctmlst_new_inputcmd:nNnnn {console} {\__ctmlst_listingbased_style:VVcn} {1} {breakable,
↪  listing\space file=##1} {listing\space file=##1}

\__ctmlst_new_listings:nNnnn {code} {\__ctmlst_mintedbased_style:VVcn} {1} {breakable,
↪  minted\space language=##1} {minted\space language=##1}
\__ctmlst_new_inputcmd:nNnnn {code} {\__ctmlst_mintedbased_style:VVcn} {2} {breakable,
↪  minted\space language=##1, listing\space file=##2} {minted\space language=##1, listing\space
↪  file=##2}
```