

l3customlisting

Typeset Code Listings and Emulate Console Screenshots with L^AT_EX Beautifully

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

Ziyue “Alan” Xiang

June 1, 2020

Contents

1	Quick Start Guide	1
2	Typeset Source Code Listings	1
3	Typeset Generic Verbatims	2
4	Typeset Console Screenshots	3
4.1	Unicode Support	5
5	Add Captions	6
6	Customize Listings	7
6.1	Changing appearance of existing listings	7
6.1.1	Common styles	7
6.1.2	Color and font	7
6.2	Changing style of existing inline commands	8
6.3	Declaring new listing environments/commands	8
6.4	More details	8

This is the L^AT_EX3 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 pdfL^AT_EX, 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 T_EX 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 \tcbininputcode*{cpp}{../res/example.cpp}
```

Code

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

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

3 Typeset Generic Verbatims

- Typeset generic verbatims inside T_EX files

```

1 \begin{tcbverbatim}
2
3 \begin{tikzpicture}
4 \draw[dashed] (0,0) -- (1,0) -- (1,1) -- (0,1) -- (0,0);
5 \draw[dashed] (1,0) -- (2,0) -- (2,1) -- (1,1) -- (1,0);
6 \draw[dashed] (2,0) -- (3,0) -- (3,1) -- (2,1) -- (2,0);
7 \draw[dashed] (3,0) -- (4,0) -- (4,1) -- (3,1) -- (3,0);
8 \draw[dashed] (4,0) -- (5,0) -- (5,1) -- (4,1) -- (4,0);
9 \draw[dashed] (5,0) -- (6,0) -- (6,1) -- (5,1) -- (5,0);
10 \draw[dashed] (6,0) -- (7,0) -- (7,1) -- (6,1) -- (6,0);
11 \draw[dashed] (7,0) -- (8,0) -- (8,1) -- (7,1) -- (7,0);
12 \draw[dashed] (8,0) -- (9,0) -- (9,1) -- (8,1) -- (8,0);
13 \draw[dashed] (9,0) -- (10,0) -- (10,1) -- (9,1) -- (9,0);
14 \draw[dashed] (10,0) -- (11,0) -- (11,1) -- (10,1) -- (10,0);
15 \draw[dashed] (11,0) -- (12,0) -- (12,1) -- (11,1) -- (11,0);
16 \draw[dashed] (12,0) -- (13,0) -- (13,1) -- (12,1) -- (12,0);
17 \draw[dashed] (13,0) -- (14,0) -- (14,1) -- (13,1) -- (13,0);
18 \draw[dashed] (14,0) -- (15,0) -- (15,1) -- (14,1) -- (14,0);
19 \draw[dashed] (15,0) -- (16,0) -- (16,1) -- (15,1) -- (15,0);
20 \draw[dashed] (16,0) -- (17,0) -- (17,1) -- (16,1) -- (16,0);
21 \draw[dashed] (17,0) -- (18,0) -- (18,1) -- (17,1) -- (17,0);
22 \draw[dashed] (18,0) -- (19,0) -- (19,1) -- (18,1) -- (18,0);
23 \draw[dashed] (19,0) -- (20,0) -- (20,1) -- (19,1) -- (19,0);
24 \draw[dashed] (20,0) -- (21,0) -- (21,1) -- (20,1) -- (20,0);
25 \draw[dashed] (21,0) -- (22,0) -- (22,1) -- (21,1) -- (21,0);
26 \draw[dashed] (22,0) -- (23,0) -- (23,1) -- (22,1) -- (22,0);
27 \draw[dashed] (23,0) -- (24,0) -- (24,1) -- (23,1) -- (23,0);
28 \draw[dashed] (24,0) -- (25,0) -- (25,1) -- (24,1) -- (24,0);
29 \draw[dashed] (25,0) -- (26,0) -- (26,1) -- (25,1) -- (25,0);
30 \draw[dashed] (26,0) -- (27,0) -- (27,1) -- (26,1) -- (26,0);
31 \draw[dashed] (27,0) -- (28,0) -- (28,1) -- (27,1) -- (27,0);
32 \draw[dashed] (28,0) -- (29,0) -- (29,1) -- (28,1) -- (28,0);
33 \draw[dashed] (29,0) -- (30,0) -- (30,1) -- (29,1) -- (29,0);
34 \draw[dashed] (30,0) -- (31,0) -- (31,1) -- (30,1) -- (30,0);
35 \draw[dashed] (31,0) -- (32,0) -- (32,1) -- (31,1) -- (31,0);
36 \draw[dashed] (32,0) -- (33,0) -- (33,1) -- (32,1) -- (32,0);
37 \draw[dashed] (33,0) -- (34,0) -- (34,1) -- (33,1) -- (33,0);
38 \draw[dashed] (34,0) -- (35,0) -- (35,1) -- (34,1) -- (34,0);
39 \draw[dashed] (35,0) -- (36,0) -- (36,1) -- (35,1) -- (35,0);
40 \draw[dashed] (36,0) -- (37,0) -- (37,1) -- (36,1) -- (36,0);
41 \draw[dashed] (37,0) -- (38,0) -- (38,1) -- (37,1) -- (37,0);
42 \draw[dashed] (38,0) -- (39,0) -- (39,1) -- (38,1) -- (38,0);
43 \draw[dashed] (39,0) -- (40,0) -- (40,1) -- (39,1) -- (39,0);
44 \draw[dashed] (40,0) -- (41,0) -- (41,1) -- (40,1) -- (40,0);
45 \draw[dashed] (41,0) -- (42,0) -- (42,1) -- (41,1) -- (41,0);
46 \draw[dashed] (42,0) -- (43,0) -- (43,1) -- (42,1) -- (42,0);
47 \draw[dashed] (43,0) -- (44,0) -- (44,1) -- (43,1) -- (43,0);
48 \draw[dashed] (44,0) -- (45,0) -- (45,1) -- (44,1) -- (44,0);
49 \draw[dashed] (45,0) -- (46,0) -- (46,1) -- (45,1) -- (45,0);
50 \draw[dashed] (46,0) -- (47,0) -- (47,1) -- (46,1) -- (46,0);
51 \draw[dashed] (47,0) -- (48,0) -- (48,1) -- (47,1) -- (47,0);
52 \draw[dashed] (48,0) -- (49,0) -- (49,1) -- (48,1) -- (48,0);
53 \draw[dashed] (49,0) -- (50,0) -- (50,1) -- (49,1) -- (49,0);
54 \draw[dashed] (50,0) -- (51,0) -- (51,1) -- (50,1) -- (50,0);
55 \draw[dashed] (51,0) -- (52,0) -- (52,1) -- (51,1) -- (51,0);
56 \draw[dashed] (52,0) -- (53,0) -- (53,1) -- (52,1) -- (52,0);
57 \draw[dashed] (53,0) -- (54,0) -- (54,1) -- (53,1) -- (53,0);
58 \draw[dashed] (54,0) -- (55,0) -- (55,1) -- (54,1) -- (54,0);
59 \draw[dashed] (55,0) -- (56,0) -- (56,1) -- (55,1) -- (55,0);
60 \draw[dashed] (56,0) -- (57,0) -- (57,1) -- (56,1) -- (56,0);
61 \draw[dashed] (57,0) -- (58,0) -- (58,1) -- (57,1) -- (57,0);
62 \draw[dashed] (58,0) -- (59,0) -- (59,1) -- (58,1) -- (58,0);
63 \draw[dashed] (59,0) -- (60,0) -- (60,1) -- (59,1) -- (59,0);
64 \draw[dashed] (60,0) -- (61,0) -- (61,1) -- (60,1) -- (60,0);
65 \draw[dashed] (61,0) -- (62,0) -- (62,1) -- (61,1) -- (61,0);
66 \draw[dashed] (62,0) -- (63,0) -- (63,1) -- (62,1) -- (62,0);
67 \draw[dashed] (63,0) -- (64,0) -- (64,1) -- (63,1) -- (63,0);
68 \draw[dashed] (64,0) -- (65,0) -- (65,1) -- (64,1) -- (64,0);
69 \draw[dashed] (65,0) -- (66,0) -- (66,1) -- (65,1) -- (65,0);
70 \draw[dashed] (66,0) -- (67,0) -- (67,1) -- (66,1) -- (66,0);
71 \draw[dashed] (67,0) -- (68,0) -- (68,1) -- (67,1) -- (67,0);
72 \draw[dashed] (68,0) -- (69,0) -- (69,1) -- (68,1) -- (68,0);
73 \draw[dashed] (69,0) -- (70,0) -- (70,1) -- (69,1) -- (69,0);
74 \draw[dashed] (70,0) -- (71,0) -- (71,1) -- (70,1) -- (70,0);
75 \draw[dashed] (71,0) -- (72,0) -- (72,1) -- (71,1) -- (71,0);
76 \draw[dashed] (72,0) -- (73,0) -- (73,1) -- (72,1) -- (72,0);
77 \draw[dashed] (73,0) -- (74,0) -- (74,1) -- (73,1) -- (73,0);
78 \draw[dashed] (74,0) -- (75,0) -- (75,1) -- (74,1) -- (74,0);
79 \draw[dashed] (75,0) -- (76,0) -- (76,1) -- (75,1) -- (75,0);
80 \draw[dashed] (76,0) -- (77,0) -- (77,1) -- (76,1) -- (76,0);
81 \draw[dashed] (77,0) -- (78,0) -- (78,1) -- (77,1) -- (77,0);
82 \draw[dashed] (78,0) -- (79,0) -- (79,1) -- (78,1) -- (78,0);
83 \draw[dashed] (79,0) -- (80,0) -- (80,1) -- (79,1) -- (79,0);
84 \draw[dashed] (80,0) -- (81,0) -- (81,1) -- (80,1) -- (80,0);
85 \draw[dashed] (81,0) -- (82,0) -- (82,1) -- (81,1) -- (81,0);
86 \draw[dashed] (82,0) -- (83,0) -- (83,1) -- (82,1) -- (82,0);
87 \draw[dashed] (83,0) -- (84,0) -- (84,1) -- (83,1) -- (83,0);
88 \draw[dashed] (84,0) -- (85,0) -- (85,1) -- (84,1) -- (84,0);
89 \draw[dashed] (85,0) -- (86,0) -- (86,1) -- (85,1) -- (85,0);
90 \draw[dashed] (86,0) -- (87,0) -- (87,1) -- (86,1) -- (86,0);
91 \draw[dashed] (87,0) -- (88,0) -- (88,1) -- (87,1) -- (87,0);
92 \draw[dashed] (88,0) -- (89,0) -- (89,1) -- (88,1) -- (88,0);
93 \draw[dashed] (89,0) -- (90,0) -- (90,1) -- (89,1) -- (89,0);
94 \draw[dashed] (90,0) -- (91,0) -- (91,1) -- (90,1) -- (90,0);
95 \draw[dashed] (91,0) -- (92,0) -- (92,1) -- (91,1) -- (91,0);
96 \draw[dashed] (92,0) -- (93,0) -- (93,1) -- (92,1) -- (92,0);
97 \draw[dashed] (93,0) -- (94,0) -- (94,1) -- (93,1) -- (93,0);
98 \draw[dashed] (94,0) -- (95,0) -- (95,1) -- (94,1) -- (94,0);
99 \draw[dashed] (95,0) -- (96,0) -- (96,1) -- (95,1) -- (95,0);
100 \draw[dashed] (96,0) -- (97,0) -- (97,1) -- (96,1) -- (96,0);
101 \draw[dashed] (97,0) -- (98,0) -- (98,1) -- (97,1) -- (97,0);
102 \draw[dashed] (
```

Verbatim

1. **Tracing**

- Typeset generic verbatims from external files

```
1 \tcbininputverbatim*{../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 CHAME.gstaticadsel1.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 CHAME.gstaticadsel1.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`.

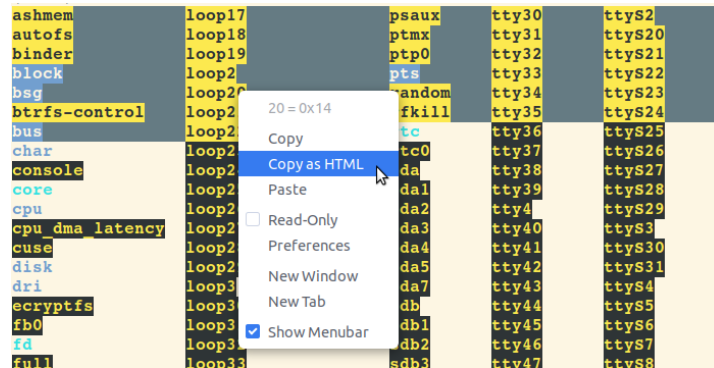


Figure 1: Converting terminal output to HTML on Ubuntu 18.04+.

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}
```

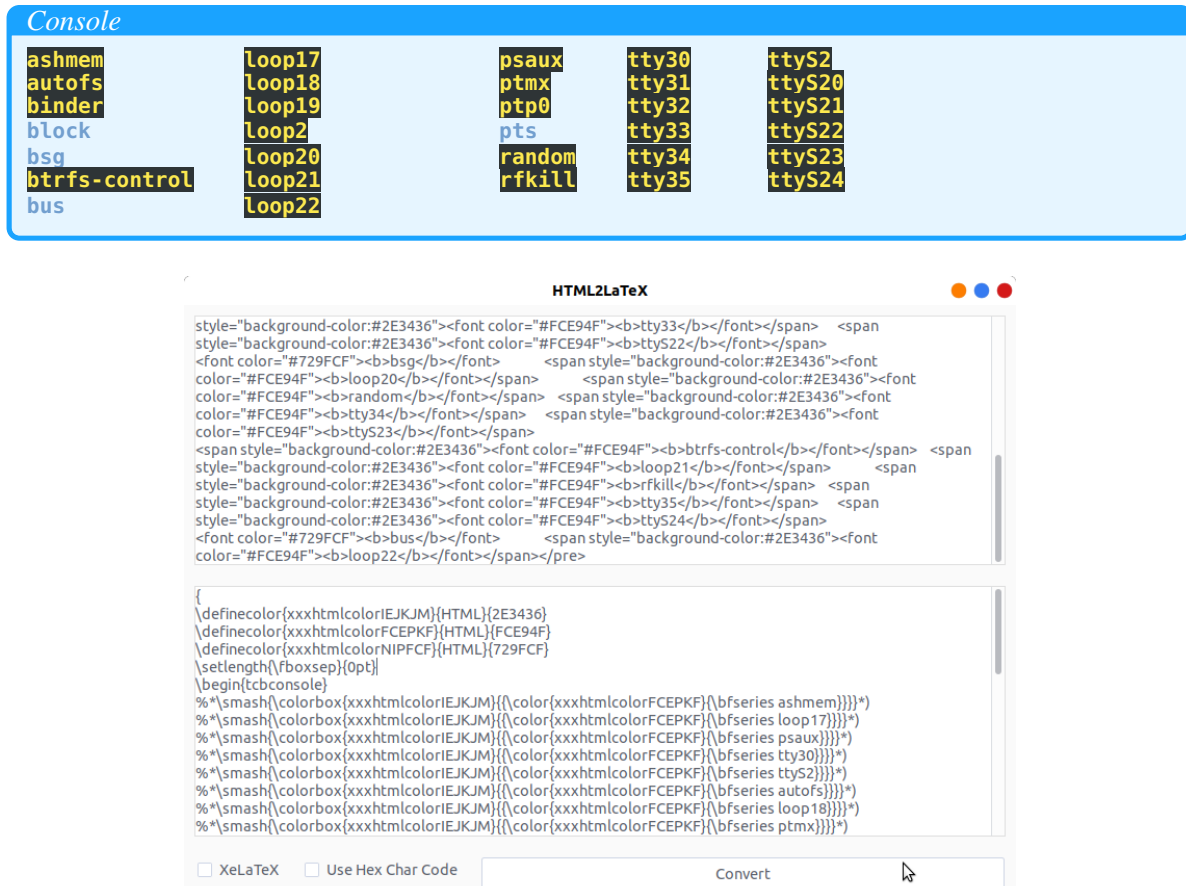
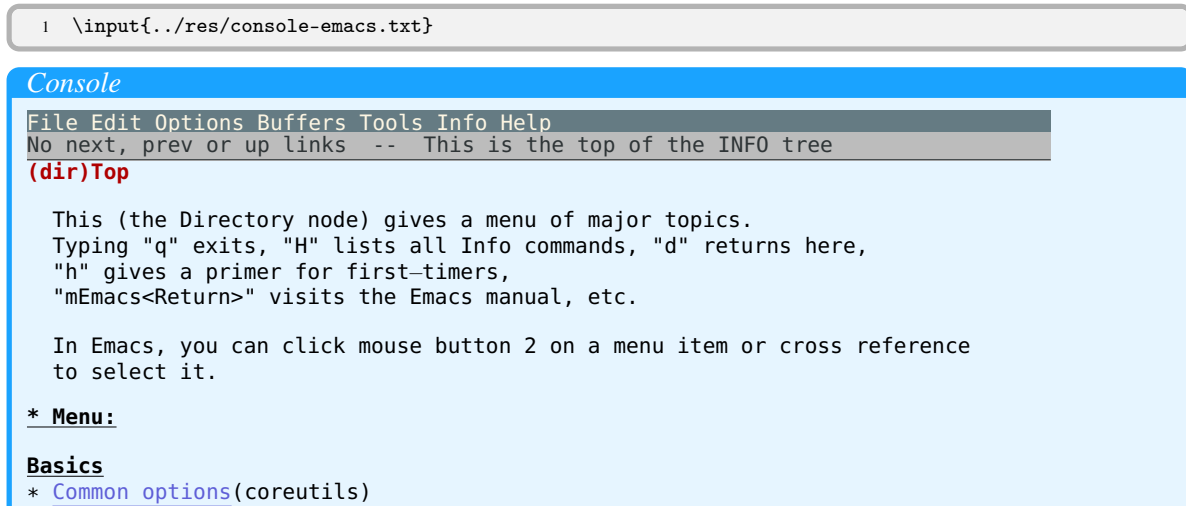


Figure 2: Using `html2latex` to convert HTML to \LaTeX .

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



```
* 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 F3Search F4Filter F5Free F6SortBy F7Nice F8Nice F9Kill F10Quit

4.1 Unicode Support

Very frequently, the terminal output contains Unicode characters. For \TeX distribution 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
  i ç £ ¤ ¥ ¦ § ¨ © ª
Latin Extended-A
Ā ā Ă ă Ć ć Ĉ ĉ Ċ ċ
```

However, keep in mind that **this Unicode support is extremely limited**: many characters are simply unavailable in pdf \LaTeX . Many packages are not compatible with `\usepackage[utf8x]{inputenc}`. One most notable example is biblatex. Therefore, for better Unicode support, one should use Xe \LaTeX or Lua \LaTeX .

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`.

```
1 \begin{mylisting}
2 \begin{tcrcode*}{julia}
3 function quadratic2(a::Float64, b::Float64, c::Float64)
4     # unlike other languages 2a is equivalent to 2*a
5     # a^2 is used instead of a**2 or pow(a,2)
6     sqr_term = sqrt(b^2-4a*c)
7     r1 = quadratic(a, sqr_term, b)
8     r2 = quadratic(a, -sqr_term, b)
9     # multiple values can be returned from a function using tuples
10    # if the return keyword is omitted, the last term is returned
11    r1, r2
12 end
13 \end{tcrcode*}
14 \end{mylisting}
15 \listoflstcaps
```

Code

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

Listing 1: Some random Julia function.

List of Code Listings

1	Some random Julia function.	7
---	-------------------------------------	---

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

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

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

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

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:

```
1 \definecolor{tcbverbatimcb}{HTML}{efefef}  
2 \renewcommand{\tcbverbatimfont}{\linespread{0.9}\fontsize{8}{8}\fontfamily{lmr}}  
3 \begin{tcbverbatim}  
4 modified verbatim  
5 \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`.


```

1 \cinline|int main();|
2 \renewcommand{\ctmlstinlineoptions}{frame=none, fontsize=\fontsize{15}{15}}
3 \ctmlstrenewinline{cinline}{c}
4 \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:

```

1 \definecolor{tcbtextcf}{HTML}{000000}
2 \definecolor{tcbtextcb}{HTML}{efefef}
3 \newcommand{\tcbtextfont}{\fontfamily{lmr}\fontsize{10}{10}}
4 \ExplSyntaxOn
5 \__ctmlst_new_listings:nNnnn {text} {\__ctmlst_listingbased_style:VVcn} {0} {breakable} {}
6 \ExplSyntaxOff
7
8 \begin{tcbtext}
9 Sample text.
10 \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.

```

1 % command to generate style list for listing-based tcblisting
2 % #1: name of the listing (no star involved, e.g. console, code, verbatim, etc.)
3 % #2: title of the listing
4 % #3: csname of font style
5 % #4: additional parameters
6 \cs_set:Npn \__ctmlst_listingbased_style:nnNn #1#2#3#4 {
7   ctmlstlistingstyle,
8   colback=#1cb,
9   colframe=#1cf,
10  title=\exp_not:n{#2},
11  listing\space options={style=ctmlststyle,
12    backgroundcolor=\exp_not:N\color{#1cb},

```

```

13     basicstyle=\exp_not:n{#3\selectfont}},
14     #4
15 }
16
17 % command to generate style list for minted-based tcbinputlisting
18 % #1: name of the listing (no star involved, e.g. console, code, verbatim, etc.)
19 % #2: title of the listing
20 % #3: csname of font style
21 % #4: additional parameters
22 \cs_set:Npn \__ctmlst_mintedbased_style:nnNn #1#2#3#4 {
23     ctmlstmintedstyle,
24     listing\space engine=minted,
25     colback=#1cb,
26     colframe=#1cf,
27     title=\exp_not:n{#2},
28     minted\space options={
29         \ctmlstmintedoptions,
30         fontsize=\exp_not:n{#3\selectfont}
31     },
32     #4
33 }

```

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

```

1 \__ctmlst_new_listings:nNnnn {console} {\__ctmlst_listingbased_style:VVcn} {0} {breakable} {}
2 \__ctmlst_new_inputcmd:nNnnn {console} {\__ctmlst_listingbased_style:VVcn} {1} {breakable,
  ↳ listing\space file=##1} {listing\space file=##1}
3
4 \__ctmlst_new_listings:nNnnn {code} {\__ctmlst_mintedbased_style:VVcn} {1} {breakable,
  ↳ minted\space language=##1} {minted\space language=##1}
5 \__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}

```