codebox: programming code box

Nan Geng
nangeng@nwafu.edu.cn

2022/01/28   v1.0.4[*][†]

Abstract

codebox is a tcolorbox-based package developed with LaTeX3, which provides environments `codebox` and `codeview`, and macros `\codefile` and `\cvfile` for typsetting programming source code box.

The environments create codebox with it's body and macros is used to read in the source code file and output is in the codebox.

The starred environments and macros are also provided to get codebox with comments at the bottom of box.

All codebox style can be setted by `\codeset` macro or environment's and macro's key-value `[`⟨*options*⟩`]`.

Contents

1 introduction

codebox is a LaTeX3 package for typesetting programming source code box.

Both `codebox` and `codeview` environment are provided with enironment body. At the same time, both `\codefile` and `\cvfile` macros are created for reading source code file.

The starred environments(`codebox*` and `codeview*`) and macros(`\codefile*` and `\cvfile*`) are also provided to get codebox with comments at the bottom of box.

---

[*] https://github.com/registor/codebox
[†] https://gitee.com/nwafu_nan/codebox

2 interface

2.1 codebox and codebox* environments

codebox
codebox*

New: 2021-12-25
Updated: 2021-12-25

```
\begin{codebox}[⟨options⟩]{⟨codebox title⟩}
.....
\end{codebox}
\begin{codebox*}[⟨options⟩]{⟨codebox title⟩}
.....
\end{codebox*}
```

Typesetting codebox with environment body. You can set the title of the codebox with {⟨*codebox title*⟩}.

The appearance of the codebox is set by key-value in [⟨*options*⟩].

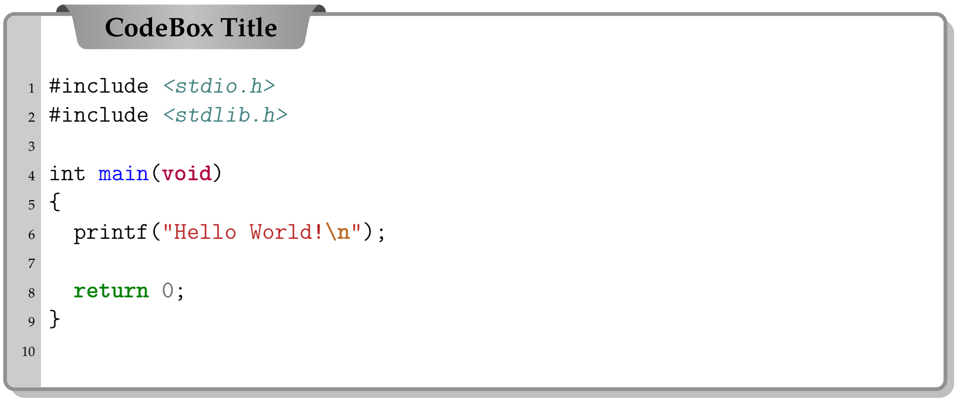
The starred environment `codebox*` is used to add comments at the bottom of the codebox, note that this needs to be done with ⟨`comments`⟩ = ⟨`texts`⟩ in [⟨*options*⟩].

Of course the key-value [⟨*options*⟩] can alse be set via the comma-separated key-value list of the `\codeset` macro.

```
\centering
\begin{codebox}{CodeBox Title}
  #include <stdio.h>
  #include <stdlib.h>

  int main(void)
  {
      printf("Hello World!\n");

      return 0;
  }
\end{codebox}
```



CodeBox Title

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  printf("Hello World!\n");

  return 0;
}

```

2.2 `\codefile` and `\codefile*` macros

\codefile
\codefile*

New: 2021-12-25
Updated: 2021-12-25

```
\codefile  [⟨options⟩] {⟨codebox title⟩} {⟨code file⟩}
\codefile* [⟨options⟩] {⟨codebox title⟩} {⟨code file⟩}
```

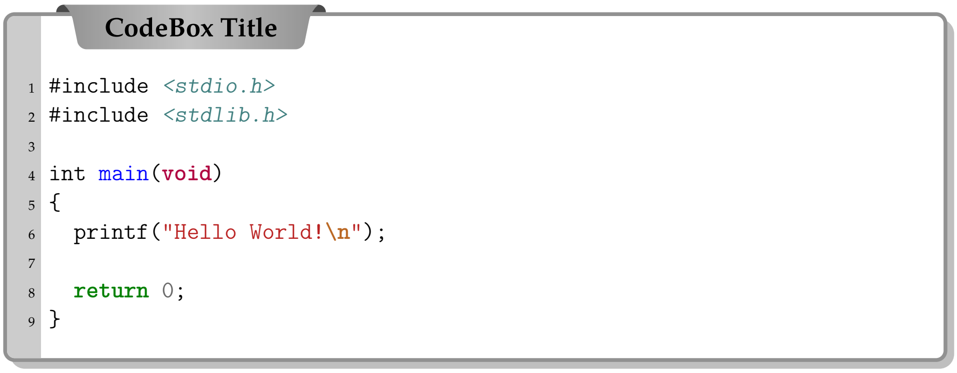
Typesetting codebox from a source code file. You can set the title of the codebox with {⟨*codebox title*⟩}.

The appearance of the codebox is set by key-value in [⟨*options*⟩].

The starred environment `\codefile*` is used to add comments at the bottom of the codebox, note that this needs to be done with ⟨`comments`⟩ = ⟨`texts`⟩ in [⟨*options*⟩].

Of course the key-value [⟨*options*⟩] can alse be set via the comma-separated key-value list of the `\codeset` macro.

```
\centering
\codefile{CodeBox Title}{test.c}
```



CodeBox Title

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  printf("Hello World!\n");

  return 0;
}
```

2.3 codeview and codeview* environments

`codeview`
`codeview*`

New: 2021-12-26
Updated: 2021-12-26

```
\begin{codeview}[⟨options⟩]{⟨codeview title⟩}
.....
\end{codeview}
\begin{codeview*}[⟨options⟩]{⟨codeview title⟩}
.....
\end{codeview*}
```

Typesetting code viewer with environment body. You can set the title of the code viewer with `{`⟨*codeview title*⟩`}`.

The appearance of the code viewer is set by key-value in `[`⟨*options*⟩`]`.

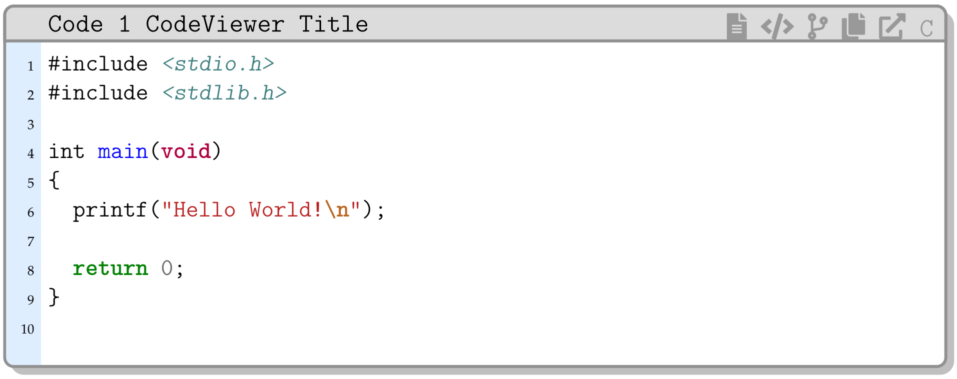
The starred environment `codeview*` is used to add comments at the bottom of the codebox, note that this needs to be done with ⟨*comments*⟩ = ⟨*texts*⟩ in `[`⟨*options*⟩`]`.

Of course the key-value `[`⟨*options*⟩`]` can alse be set via the comma-separated key-value list of the `\codeset` macro.

```
\centering
\begin{codeview}{CodeViewer Title}
  #include <stdio.h>
  #include <stdlib.h>

  int main(void)
  {
      printf("Hello World!\n");

      return 0;
  }
\end{codeview}
```



Code 1 CodeViewer Title

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  printf("Hello World!\n");

  return 0;
}

```

2.4 `\cvfile` and `\cvfile*` macros

`\cvfile`
`\cvfile*`

New: 2021-12-26
Updated: 2021-12-26

`\cvfile  [⟨options⟩] {⟨codeview title⟩} {⟨code file⟩}`
`\cvfile* [⟨options⟩] {⟨codeview title⟩} {⟨code file⟩}`

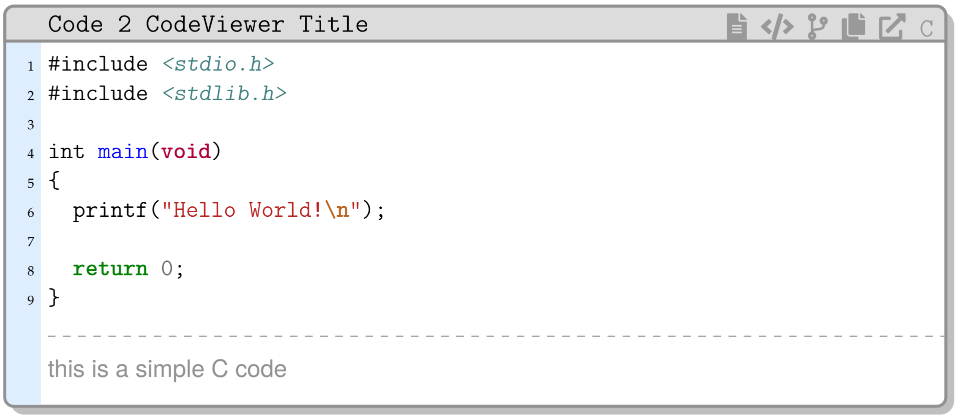
Typesetting code viewer from a source code file. You can set the title of the code viewer with {⟨*codeview title*⟩}.

The appearance of the code viewer is set by key-value in [⟨*options*⟩].

The starred environment `\vcfile*` is used to add comments at the bottom of the codebox, note that this needs to be done with ⟨`comments`⟩ = ⟨`texts`⟩ in [⟨*options*⟩].

Of course the key-value [⟨*options*⟩] can alse be set via the comma-separated key-value list of the `\codeset` macro.

```
\centering
\cvfile*[comments=this is a simple C code]{CodeViewer Title}{test.c}
```



Code 2 CodeViewer Title

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  printf("Hello World!\n");

  return 0;
}
```

this is a simple C code

3 Options

The codebox package provides a number of options to set the style of the codebox. The following options can be set with `\codeset` macro. Also, these options can be set with the all environment's or command's [⟨*options*⟩].

3.1 code engine

`minted`

New: 2021-12-26
Updated: 2021-12-26

`minted = ⟨`**`true`**`|`*`false`*`⟩` Init = `true`

`minted` is used to set code highlight engine, if it is **true** then the minted package is used, if it is **false** then the listings package is used. The default is **true**.

3.2 language

`lang`

New: 2021-12-26
Updated: 2021-12-26

`lang = {⟨source code language⟩}` Init = `C`

`lang` is used to set source code language. The default is **C**.

3.3 title prefix

`pretitle`

New: 2021-12-26
Updated: 2021-12-26

`pretitle = {⟨title prefix⟩}` Init = `Code`

`pretitle` is used to set prefix of code counter. The default is **Code**。

3.4 code highlight style

`codestyle`
New: 2021-12-26
Updated: 2021-12-26

`codestyle = {⟨highlight style⟩}` Init = `codeblocks`

`codestyle` is used to set code highlight style, valid only for the minted engine. The default is **codeblocks**.

3.5 code fontsize

`codesize`
New: 2021-12-26
Updated: 2021-12-26

`codesize = {⟨fontsize macro⟩}` Init = `\small`

`codesize` is used to set code fontsize, valid only for minted engine. The default is`\small`.

3.6 comment contents

`comments`
New: 2021-12-26
Updated: 2021-12-26

`comments = {⟨texts⟩}` Init = `nothing`

`comments` is used to set comment contents. The default is **nothing**.

3.7 comment format

`commentf`
New: 2021-12-26
Updated: 2021-12-26

`commentf = {⟨format macros⟩}` Init = `\small\sffamily`

`commentf` is used to set comment format at codebox bottom. The default is `\small\sffamily`.

3.8 code baseline stretch

`codestretch`
New: 2021-12-26
Updated: 2021-12-26

`codestretch = {⟨float number⟩}` Init = `1.0`

`codestretch` is used to set code baseline stretch, valid only for minted engine. The default is**1.0**.

3.9 seperation between line number and code

`linenumsep`
New: 2021-12-26
Updated: 2022-1-28

`linenumsep = {⟨float number⟩}` Init = `1.80`

`linenumsep` is used to set the seperation between line number and code, valid only for minted engine. Note the unit is mm. The default is**3.0**.

3.10 label

`label`
New: 2022-1-4
Updated: 2022-1-4

`label = {⟨label name⟩}` Init = `nothing`

`label` is used to set `\ref` 's label name, it is for `codeview/codeview*` and `\cvfile/\cvfile*`. The default is **nothing**.

4 The counter

`cvconuter`

New: 2021-12-28
Updated: 2021-12-28

The codebox package provides a `cvcounter` counter that can be used to count code boxes with environment `codeview/codeview*` and the command `\cvfile/\cvfile*`.

By default, if `\thechapter` exists, its parent counter is set to **chapter** otherwise it will be counted uniformly by full text.

You can use `\renewcommand{\thecvcounter}{\thechapter.\arabic{cvcounter}}` or something like this macro to change the numbered output.

5 Examples

The codebox package can be used in situations where the highlight programming source code needs to be typeset to avoid the use of screenshots. Code box can be with or without underline comments.

5.1 Java code

The language can be set with `\codeset` macro.

```
\centering
\codeset{lang=java}
\codefile{Java CodeBox}{hellojava.java}
```

Java CodeBox

```java
public class HelloWorld {
  public static void main(String[] args){
      System.out.println("Hello World!");
  }
}
```

5.2 Python code

The language can be set with `options`, of course you can label and ref it such as code 3.

```
\centering
\cvfile[lang=python,label=code-test]{Python CodeBox}{hellopy.py}
```

Code 3 Python CodeBox PYTHON

```python
import tensorflow as tf
import numpy as np
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

# Create 100 phony x, y data points in Numpy, y = x * 0.1 + 0.3
x_data = np.random.random(100).astype("float32")
y_data = x_data * 0.1 + 0.3

# Try to find values for W and b that compute y_data = W * x_data + b
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

# Minimize the mean squared errors.
loss = tf.reduce_mean(tf.square(y -y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

# Before starting, initialize the variables. We will 'run' this first
init = tf.global_variables_initializer()

# Launch the graph.
sess = tf.Session()
sess.run(init)

# Fit the line.
for step in range(201):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(W), sess.run(b))

```

5.3 listings engine

listings engine can be set with ⟨*minted*⟩ = ⟨*false*⟩.

```
\centering
\cvfile[minted=false,lang=c]{C CodeBox}{test.c}
```

Code 4 C CodeBox

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    printf("Hello World!\n");

    return 0;
}
```