LitCode

Another Tool for Literate Programming

Praneet Kapoor

July 5, 2023

Contents

Chapter 1

Introduction

Chapter 2

Using LitCode

Chapter 3

The Source Code

3.1 Custom Style file: litcode.sty

3.1.1 The Type of Styling I Want

Before delving into the code making up `litcode` command line tool, I want to first talk about the styling which I wanted for my code-blocks or *chunks* .
Firstly, I wanted to write the chunks in an evironment whose name wasn't greater than five characters. Why? When working on a project for an excessive period of time, a person will naturally have some fatigue and might misspell the environment name. Also, I am not a professional typer: I still look at the keys from time to time while typing and I tend to use my right hand more than my left hand for typing. I know it for a fact that if I were to type

```
\begin{lstlisting}{caption = 'hello world', label = 'hello world'}
```

a 100 times, I am going to make several typing mistakes 80% of the time. Yes, I am that bad! It would be better for me to type

```
\begin{code}{hello world}
```

than the *monstrosity* I previously wrote. This means that I have make sure that the environments I was going to build had as few arguments as possible, and their caption is the same as their label.
Another important feature which must be there in the environment is inline chunk referencing. A program written in literate style comprises of several chunks which may refer to each other. For example the chunk below refers to a chunk named 'Print hello world 100 times':

```
#include <stdio.h>

int main()
{
    @<\coderef{Print hello world 100 times}@>
    return 0;
}
```

. . . which has been described here:

⟨Print hello world 100 times⟩≡

```
for (int i = 0; i < 100; ++i) {
    fprintf(stdout, "Hello, world\n");
    @<yaka@>
```

This meant that I had to find a way to break the *"verbatimness"* of the listing environments and develop a new type of referencing style which matches those defined by Knuth and Ramsey. The end result was `litcode` package.

I am going to use this package for the majority of my projects and I suggest others to try it out too. This package defines several variables and environment in `litcode.sty` file. I will be honest: I am not that good in LaTeX. Most of the styling file — and by most I mean about 70% — has been written by ChatGPT. It is a fascinating tool but it does make mistakes many many times. Once the core portion of styling file had been implemented, I did some major and minor fixes by doing hit and trial till I was satisfied by the result.

The `litcode` package defines the following flavours of 'listing' environments:

1. `code` : An environment for your regular coding tasks. You can code in any language you want in this environment except for LaTeX. This is because if I were to use the escape characters I had described for `code` and `vcode` environments inside a chunk of LaTeXcode using the same character combination, I would have broken lot's of stuff. A mouthful sentence — just remember that if you are writing a chunk in anything but LaTeX, use this environment. `@<` and `@>` are to used to escape any piece of LaTeXcode in this environment, usually `\coderef`.

2. `latex` : Now this is the environment in which you can write bits of LaTeX code. `/*<` and `>*/` are the escape characters in this environment.

3. `vcode` : An environment for listing some text without any captions or labels. No escape characters are defined for this environment.

To refer to a chunk either from inside the chunk or from anywhere else in the document, use `\coderef{}` macro as has been shown in the example in fig. 3.1.

```
We begin by printing \texttt{Hello world} on the screen. \coderef{Hello world in C++}
    is going to print \texttt{Hello world} on the terminal.

\begin{code}[Hello world in C++]

@<\coderef{Setup}@>

int main()
{
    cout << "Hello, world" << endl;
    return 0;
}

\end{code}

\begin{code}[Setup]

#include <iostream>

using namespace std;

\end{code}
```

Figure 3.1: Using `code` environment along with `coderef`

Chapter 4

A Few Examples

Listings