The lit Tangler

A Simple Tangler that's Lit

Kian Dianati

Introduction

Sometime in late 2020 I became interested int the Scheme programming language and so decided to go through the phenoumenal *Structure and Interpretation of Computer Programms*. For recording my answers to exercises I was in need of a literate programming tool.

And so I searched for possible tools. None of which were quite up to the task. They were either stuck to a particular environment, not-portable, or language dependant. Not to mention that none of them were particularly simple in design or implementation and that they all used custom weavers with custom formats. Pure suck. So I decided to write my own.

Markdown is by far the most popular markup language around right now. It is widely used by developers in all settings and fields. And since it is a standard not an implementation there are a variety of markdown engines around for all platforms. And as it has the ability to insert code, it is the perfect choice for a literate programming tool that strives for simplicity and portability.

Imports and packages

I let the package name be `main` for now although I will change it later on. For this file we will need the `os`, `bufio`, and `fmt` packages from the stdlib.

```
package main

import (
```

```
    "os"
    "bufio"
    "fmt"
)
```

Dealing with erros

Since we'll be doing a lot of file i/o, lets make a `check` function that will panic if it recieves an error.

```
func check(e error) {
    if e != nil {
        panic(e)
    }
}
```

Tangling

This is still an early implmentation and so doesn't have all the features I want. For now it looks through the code, and if it's in a code block (determined by the `open` boolean) it will print the line into stdout which then could be piped into a file.

```
func tangle(in string) {
    file, err := os.Open(in)
    check(err)
    f := bufio.NewScanner(file)
    var open bool = false
    for i := 0; f.Scan(); i++ {
        ln := f.Text()
        if len(ln) >= 3 {
            if ln[:3] == "```" {
                open = !open
                fmt.Println()
                continue
            }
        }
        if open {
            fmt.Println(ln)
```

```
        }
    }
}
```

Main and dealing with input

For now just tangle all argument given to it, or if no arguments are given, take one from stdin.

```
func main() {
    if len(os.Args) > 1 {
        for _, i := range os.Args[1:] {
            tangle(i)
        }
    } else {
        var file string
        fmt.Scan(&file)
        tangle(file)
    }
}
```

What's next?

- ☐ Add concurency for speed
- ☐ Ability to execute/print to stdout a selected group of code blocks (like `org-babel`)