The Mabel Literate Programming Tool

A Simple Tangler

M-CS-ME

Introduction

I wanted to take notes with executable code snippets inside, and I didn't want it to be glued to my text editor. This little project is supposed to (eventually) do that.

I chose markdown because it's simple, light weight, and has many implementations. A markdown engine or `pandoc` will be better than any weaver that I could write.

The source in `src/mabel.go` is generated with `mabel` through `mabel mabel.md > src/mabel.go`.

Imports and packages

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

```
package main

import (
    "os"
    "bufio"
    "fmt"
)
```

Dealing with errors

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 f.Scan() {
        ln := f.Text()
        if len(ln) >= 3 {
            if ln[:3] == "```" {
                open = !open
                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`)