YAML support for the Go language

Introduction

The yaml package enables Go programs to comfortably encode and decode YAML values. It was developed within Canonical as part of the juju project, and is based on a pure Go port of the well-known libyaml C library to parse and generate YAML data quickly and reliably.

Compatibility

The yaml package supports most of YAML 1.1 and 1.2, including support for anchors, tags, map merging, etc. Multi-document unmarshalling is not yet implemented, and base-60 floats from YAML 1.1 are purposefully not supported since they're a poor design and are gone in YAML 1.2.

Installation and usage

The import path for the package is *gopkg.in/yaml.v2*.

To install it, run:

```
go get gopkg.in/yaml.v2
```

API documentation

If opened in a browser, the import path itself leads to the API documentation:

- https://gopkg.in/yaml.v2

API stability

The package API for yaml v2 will remain stable as described in gopkg.in.

License

The yaml package is licensed under the Apache License 2.0. Please see the LICENSE file for details.

Example

```go
package main

import (
        "fmt"
        "log"

        "gopkg.in/yaml.v2"
```

```
)

var data = `
a: Easy!
b:
  c: 2
  d: [3, 4]
`

// Note: struct fields must be public in order for unmarshal to
// correctly populate the data.
type T struct {
        A string
        B struct {
                RenamedC int   `yaml:"c"`
                D        []int `yaml:",flow"`
        }
}

func main() {
        t := T{}

        err := yaml.Unmarshal([]byte(data), &t)
        if err != nil {
                log.Fatalf("error: %v", err)
        }
        fmt.Printf("--- t:\n%v\n\n", t)

        d, err := yaml.Marshal(&t)
        if err != nil {
                log.Fatalf("error: %v", err)
        }
        fmt.Printf("--- t dump:\n%s\n\n", string(d))

        m := make(map[interface{}]interface{})

        err = yaml.Unmarshal([]byte(data), &m)
        if err != nil {
                log.Fatalf("error: %v", err)
        }
        fmt.Printf("--- m:\n%v\n\n", m)

        d, err = yaml.Marshal(&m)
        if err != nil {
                log.Fatalf("error: %v", err)
        }
        fmt.Printf("--- m dump:\n%s\n\n", string(d))
}
```

This example will generate the following output:

```
--- t:
{Easy! {2 [3 4]}}

--- t dump:
a: Easy!
b:
  c: 2
  d: [3, 4]


--- m:
map[a:Easy! b:map[c:2 d:[3 4]]]

--- m dump:
a: Easy!
b:
  c: 2
  d:
  - 3
  - 4
```