株式会社 ITS MORE
2020年4月設立

2020年5月19日 投稿者: YSATO@DELEGATE.ORG

goによるpngの生成（終了）

さて、pngの容れ物はわかったので、中身を詰めてみよう。イメージデータを詰めるには、、、どうも zlib 形式での圧縮（deflate）は必須らしい。なんでだろう。

同一マシンの中で揮発性で小さめの画像を飛ばすだけなので、圧縮・伸長のオーバヘッドを避けたいのに。マシン内なら楽勝で10Gbps出るのに、Zlibかませたら遅くなる。そもそもCRCもいらない。あれもかなり重い。100Mbpsくらいに落ちてしまう感じだ。たかだか4GHzのCPUで処理してるんだし。そう言えば、zlib の圧縮のパラメータで、圧縮無しってあったっけか…

まあ、せっかく go がPNG用のエンコーダを用意してくれてるんだから、まずはそれを使ってみよう。再び　Packege png のページを見る。たぶんこの Encodeという関数だろう。引数はどうなっているのか　…　「image.Image」。なんだろう、これ。Rectangle …。…。…。えーっ、つまりこれでこれでごにょごにょイメージを作ってやって、Package png に渡してやると、PNG 形式にしてくれるって事？　そうなんですか。

なーんだ

なーんだぁ！

まあ、そういうものが無い方がおかしいと思った。でもまあいい、10年の時の流れを実体験しながら1日で飛び越えられたから。PNGのフォーマットも勉強できたし、何かの役に立つかも知れない。

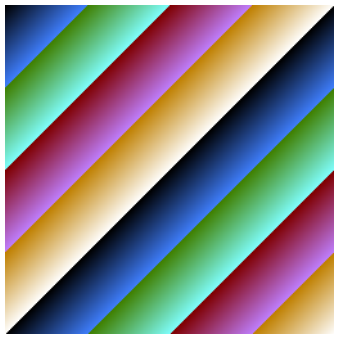
png Packageによる png の生成

それで、それってどう使うのって、これが Encode の Example ですか。

```
package main

import (
        "image"
        "image/color"
        "image/png"
        "log"
        "os"
)

func main() {
        const width, height = 256, 256

        // Create a colored image of the given width and height.
        img := image.NewNRGBA(image.Rect(0, 0, width, height))

        for y := 0; y < height; y++ {
                for x := 0; x < width; x++ {
                        img.Set(x, y, color.NRGBA{
                                R: uint8((x + y) & 255),
                                G: uint8((x + y) << 1 & 255),
                                B: uint8((x + y) << 2 & 255),
                                A: 255,
                        })
                }
        }

        f, err := os.Create("image.png")
        if err != nil {
                log.Fatal(err)
        }

        if err := png.Encode(f, img); err != nil {
                f.Close()
                log.Fatal(err)
        }
```

```
        if err := f.Close(); err != nil {
                log.Fatal(err)
        }
}
```

New して Set したら Encode して終了ですか。そうですかー。どれどれ。go run go。出来た。



ステキ。いやはや極楽極楽。いい世の中になったものたろうだなー :-)。もう道具は全て揃ってるって感じ。

自分で作ったのも、記念にアーカイブしておこう。これはこれで、楽しかった。やはり、有名でがっちりした仕様書を読みながら実装するのはワクワクする。しかし、単にバイト列を直線的にバッファに書いていくのをGo言語で効率的にやる方法がまだわからない。まあ、string じゃなくて byte というのでやるんだろう…

/*

<base href=https://www.w3.org/TR/PNG/>

```
        2020-0518-03 pnggen/0.0.3 (PNG generator) by ITS more :-)
        Reference: PNG Ed.2 Spec.
```

```
*/

package main
import (
        "os"
        "fmt"
        "hash/crc32"
)
func myid() string {
        return "pnggen/0.0.3 (PNG Generator)"
```

```
}
func hton32b(hint int)(string){
        buf := make([]byte, 4)
        buf[0] = byte(hint >> 24)
        buf[1] = byte(hint >> 16)
        buf[2] = byte(hint >>  8)
        buf[3] = byte(hint)
        return string(buf)
}
func putNetInt32(file *os.File, hint int) int {
        file.Write([]byte(hton32b(hint)))
        return 4
}
// 5.3 Chunk layout | Chunk := Length+Type+Data+CRC
func putChunk(file *os.File, ctype string, cdata string) int {
        crc := crc32.ChecksumIEEE([]byte(ctype+cdata))  // heavy if large
        olen := putNetInt32(file,len(cdata))    // length of data
        file.Write([]byte(ctype))               // chunk type
        file.Write([]byte(cdata))               // chunk data
        olen += putNetInt32(file,int(crc))      // CRC of type plus data
        return olen + len(ctype) + len(cdata)
}
func pnggen() {
        outfile := "x3.png"
        genw := 64;
        genh := 64;
        fmt.Fprintf(os.Stderr,"-- %s --\n",myid())
        file, _ := os.Create(outfile)
        defer file.Close()

        //---- Signature -- | 5.2 PNG signagure |
        PNGsignature := "\x89PNG\r\n\x1A\n"
        file.Write([]byte(PNGsignature))
        olen := len(PNGsignature)

        //---- IHDR -- | 11.2.2.2 IHDR Image header |
        ihdr := hton32b(genw) // Width
        ihdr += hton32b(genh) // Hight
        ihdr += string(1) // Bit depth, 1 for monocrome
        ihdr += string(0) // Color type, 0 for Grayscalse
        ihdr += string(0) // Compression method, 0 for deflate/inflate
        ihdr += string(0) // Filter method, 0 for adaptive
        ihdr += string(0) // Interlace method, 0 for no interlace
        olen += putChunk(file,"IHDR",ihdr)

        //---- PLTE -- | 11.2.2.3 PLTE Pallete |
        olen += putChunk(file,"PLTE","");
```

```
        //---- IDAT -- 11.2.2.4 IDAT Image data
        olen += putChunk(file,"IDAT","")

        //---- IEND -- 11.2.2.5 IEND Image trailer
        olen += putChunk(file,"IEND","")

        fmt.Fprintf(os.Stderr,"-- created %s (%d x %d) %d bytes\n",
                outfile,genw,genh,olen)
}
func main() {
        pnggen()
}
//
```
