
Multi Progress Bar



mpb is a Go lib for rendering progress bars in terminal applications.

Features

- **Multiple Bars:** Multiple progress bars are supported
- **Dynamic Total:** Set total while bar is running
- **Dynamic Add/Remove:** Dynamically add or remove bars
- **Cancellation:** Cancel whole rendering process
- **Predefined Decorators:** Elapsed time, ewma based ETA, Percentage, Bytes counter
- **Decorator's width sync:** Synchronized decorator's width among multiple bars

Usage

Rendering single bar

```
1 package main
2
3 import (
4     "math/rand"
5     "time"
6
7     "github.com/vbauerster/mpb/v8"
8     "github.com/vbauerster/mpb/v8/decor"
9 )
10
11 func main() {
12     // initialize progress container, with custom width
13     p := mpb.New(mpb.WithWidth(64))
14
15     total := 100
16     name := "Single Bar:"
17     // create a single bar, which will inherit container's width
18     bar := p.New(int64(total),
19         // BarFillerBuilder with custom style
20         mpb.BarStyle().Lbound("||").Filler("█").Tip("▮").Padding("░").
21             Rbound("||"),
22         mpb.PrependDecorators(
23             // display our name with one space on the right
24             decor.Name(name, decor.WC{C: decor.DindentRight | decor.
25                 DextraSpace}),
26             // replace ETA decorator with "done" message, OnComplete
27             event
```

```

25         decor.OnComplete(decor.AverageETA(decor.ET_STYLE_GO), "done
26         "),
27         mpb.AppendDecorators(decor.Percentage()),
28     )
29     // simulating some work
30     max := 100 * time.Millisecond
31     for i := 0; i < total; i++ {
32         time.Sleep(time.Duration(rand.Intn(10)+1) * max / 10)
33         bar.Increment()
34     }
35     // wait for our bar to complete and flush
36     p.Wait()
37 }
```

Rendering multiple bars

```

1     var wg sync.WaitGroup
2     // passed wg will be accounted at p.Wait() call
3     p := mpb.New(mpb.WithWaitGroup(&wg))
4     total, numBars := 100, 3
5     wg.Add(numBars)
6
7     for i := 0; i < numBars; i++ {
8         name := fmt.Sprintf("Bar#%d:", i)
9         bar := p.AddBar(int64(total),
10            mpb.PrependDecorators(
11                // simple name decorator
12                decor.Name(name),
13                // decor.DSyncWidth bit enables column width
14                // synchronization
15                decor.Percentage(decor.WCSyncSpace),
16            ),
17            mpb.AppendDecorators(
18                // replace ETA decorator with "done" message,
19                // OnComplete event
20                decor.OnComplete(
21                    // ETA decorator with ewma age of 30
22                    decor.EwmaETA(decor.ET_STYLE_GO, 30, decor.
23                        WCSyncWidth), "done",
24                ),
25            ),
26        )
27        // simulating some work
28        go func() {
29            defer wg.Done()
30            rng := rand.New(rand.NewSource(time.Now().UnixNano()))
31            max := 100 * time.Millisecond
32            for i := 0; i < total; i++ {
33                // start variable is solely for EWMA calculation
34                // EWMA's unit of measure is an iteration's duration

```

```
32         start := time.Now()
33         time.Sleep(time.Duration(rng.Intn(10)+1) * max / 10)
34         // we need to call EwmaIncrement to fulfill ewma
           decorator's contract
35         bar.EwmaIncrement(time.Since(start))
36     }
37     }()
38 }
39 // wait for passed wg and for all bars to complete and flush
40 p.Wait()
```

Dynamic total

Complex example

Bytes counters