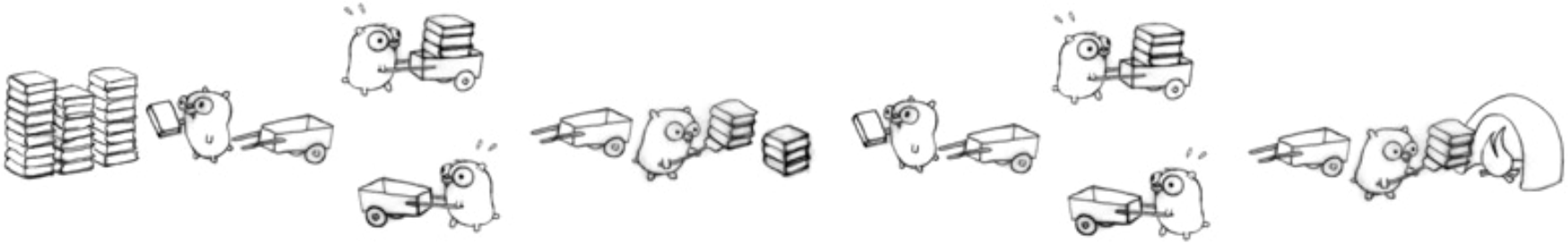
Programmieren mit Go

Sebastian ‘tokkee’ Harl
<sh@tokkee.org>

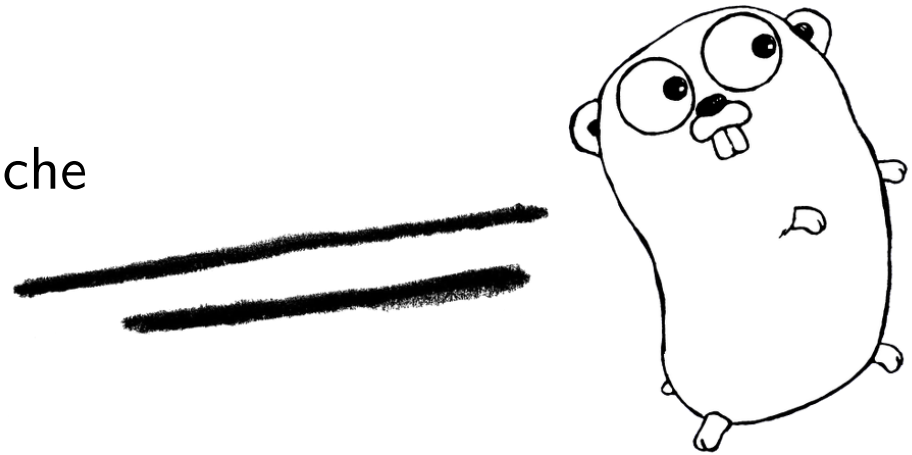


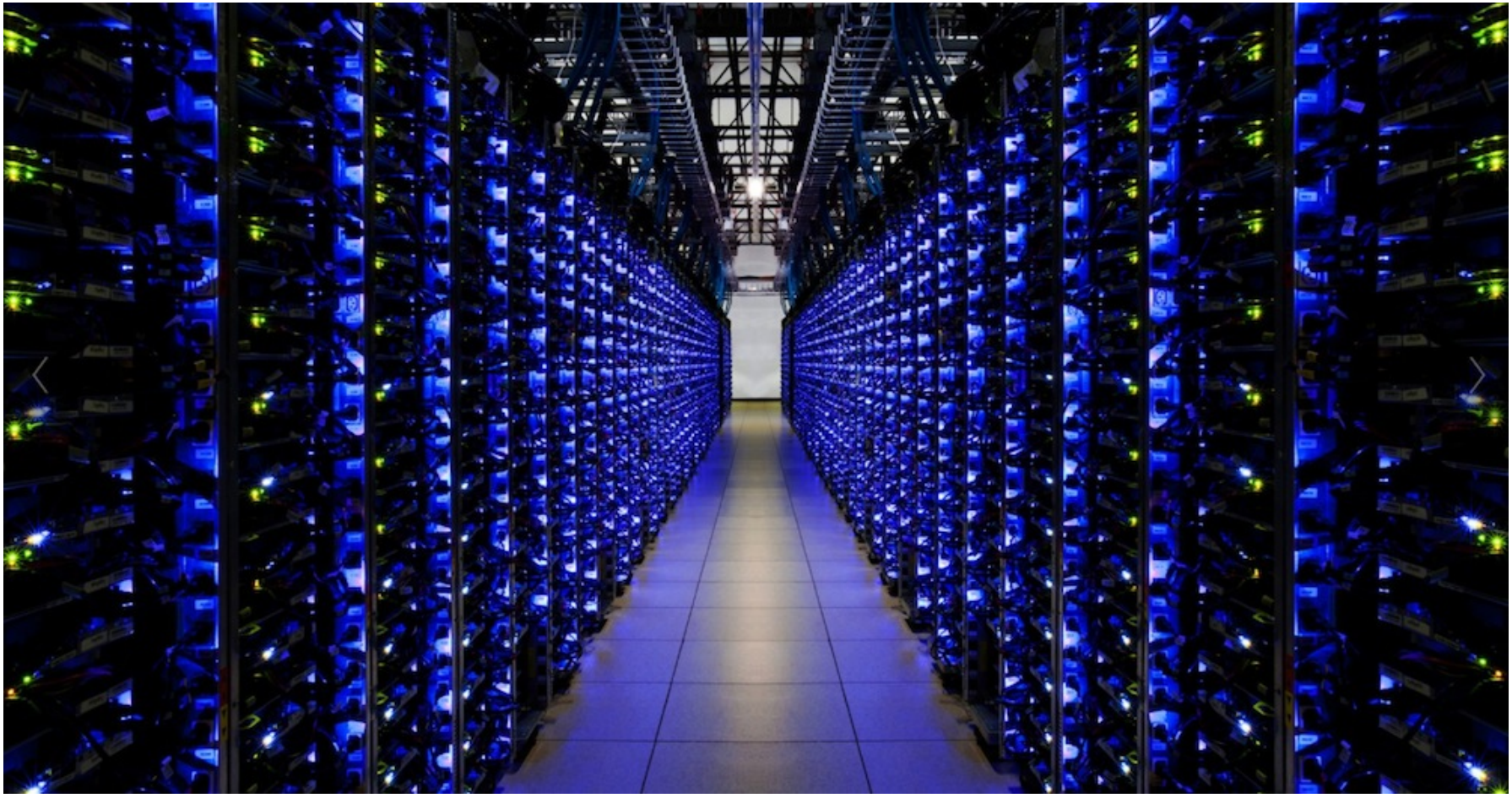
Grazer LINUXTAGE
29. + 30. April 2016 www.linuxtage.at

Überblick

Was ist Go?

- `https://golang.org/`
- Open Source Programmiersprache
- imperativ, Interfaces, Pakete
- statisch typisiert, kompiliert
- Nebenläufigkeit
- Garbage Collection





Hallo Welt!

```
1 package main
2
3 import "fmt"
4
5 func main() {
6   fmt.Println("Hallo Welt")
7 }
```

Programm übersetzen / ausführen

```
% go build -o hallo hallo.go
% ./hallo
Hallo Welt

% go run hallo.go
Hallo Welt
```

Go Playground

The Go Playground Run Format Imports Share

```
1  package main
2
3  import "fmt"
4
5  func main() {
6          fmt.Println("Hello, 世界")
7  }
8
9
10
```

```
Hello, 世界

Program exited.
```

Pakete / Bibliotheken

```
1  package zeichenkette
2
3  func Rückwärts(s string) string {
4    r := []rune(s)
5    for i := 0; i < len(r)/2; i++ {
6      j := len(r) - i - 1
7      r[j], r[i] = r[i], r[j]
8    }
9    return string(r)
10 }
```

Pakete verwenden

- `$GOPATH/src` ist der Suchpfad für extra Pakete
- `import "<paketname>"`

```
1  package main
2
3  import (
4    "fmt"
5
6    zk "tokkee.org/zeichenkette"
7  )
8
9  func main() {
10   fmt.Println(zk.Rückwärts("Hallo Welt"))
11 }
```

Arbeitsumgebung

```
$GOPATH
  |\
  | `- bin/
  |     \
  |      `- hallo
   \
    `- src/
        \
         `- hallo/
             \
              `- hallo.go
```

Arbeitsumgebung (2)

- go build -o hallo_welt hallo
- go install hallo
- go doc hallo

Fehlerbehandlung

```
1  func machEs() (*Object, error) {
2    o := &Object{}
3    if err := o.init(); err != nil {
4      return nil, err
5    }
6    return o, nil
7  }
8
9  func main() {
10   o, err := machEs()
11   if err != nil {
12     log.Exitf("Fehler: %v\n", err)
13   }
14   // ...
```

Komplexe Datenstrukturen

Array / Slice

```
1 var array [3]string
2 array[0] = "ein"
3 array[1] = "String"
4 fmt.Println(array)
```

```
1 var slice []string
2 slice = append(slice, "ein", "String")
3 slice = []string{"ein", "anderer", "String"}
4 fmt.Println(slice)
```

Komplexe Datenstrukturen (2)

Map

```
1 var m map[string]int
2 m = make(map[string]int) // Array anlegen
3 m = map[string]int{      // Array initialisieren
4     "a": 1,
5     "b": 2,
6 }
7 m["c"] = 3
8 fmt.Println(m["a"])
```

Komplexe Datenstrukturen (3)

Struct

```
1  type Ding struct {
2      privat      float32
3      Öffentlich  string
4  }
5  d := Ding{
6      Öffentlich: "Daten für Alle!",
7      privat:     47.11,
8  }
9  d.privat = 3.1415926535
10 fmt.Println(d)
```

Komplexe Datenstrukturen (4)

Interface / Methoden

```
1 type Macher interface {
2   Mach(int, int) string
3 }
4 type MeinMacher struct {}
5 func (m MeinMacher) Mach(a, b int) string {
6   return fmt.Sprintf("%d x %d = %d", a, b, a*b)
7 }
```

Übung: Quadratwurzel

`https://go-tour-de.appspot.com/flowcontrol/8`
`https://de.wikipedia.org/wiki/Newton-Verfahren`

Berechne die Quadratwurzel (Wurzel(x **float64**)) mit Hilfe des Newton-Verfahren:

$$z_{n+1} = z_n - \frac{z_n^2 - x}{2 * z_n}$$

Erstelle dazu ein eigenes Paket und ein Programm zum testen. Verwende entweder eine Schleife mit fester Anzahl an Durchläufen oder bis sich das Ergebnis nicht (kaum) mehr ändert.

Die Standard-Bibliothek

`https://golang.org/pkg/`

- Crypto
- Datenbanken
- Go Parser
- Netzwerk, HTTP, SMTP, etc.
- Datenstrukturen

Weitere Bibliotheken

`https://godoc.org/`

- Vielzahl an Open Source Bibliotheken
- go get github.com/user/pkg
- `https://gopkg.in`
  - Versionierung von Bibliotheken auf Github
  - `v3` zeigt auf Branch/Tag `v3`, `v3.N` oder `v3.N.M`
  - `gopkg.in/pkg.v3` → `github.com/go-pkg/pkg`
  - `gopkg.in/user/pkg.v3` → `github.com/user/pkg`

Ein Webserver

https://golang.org/pkg/net/http/

```
1  func main() {
2    http.HandleFunc("/hallo", sageHallo)
3    log.Fatal(http.ListenAndServe(":9999", nil))
4  }
5
6  func sageHallo(w http.ResponseWriter,
7    r *http.Request) {
8
9    fmt.Fprintf(w, "Hallo %s", r.RemoteAddr)
10 }
```

Nebenläufigkeit

Alle Anfragen werden nebenläufig behandelt.

- Goroutinen
  - `go f(args)`
  - leichtgewichtige Threads
  - hunderte oder tausende werden effizient verwaltet
- Channels
  - Kommunikation zwischen Goroutinen
- `select`
  - warten auf I/O und Channels

Goroutinen

```
1  errCh := make(chan error, 2)
2
3  go func() {
4      errCh <- machEs()
5  }()
6  go func() {
7      errCh <- undDas()
8  }()
9
10 go machEtwasImHintergrund()
```

Goroutinen (2)

https://golang.org/pkg/time/

```
1  timeout := time.After(50*time.Millisecond)
2  for i := 0; i < cap(errCh); i++ {
3    select {
4    case err := <- errCh:
5      if err != nil {
6        return err
7      }
8    case <-timeout:
9      return fmt.Errorf("timeout(%d)", i)
10   }
11 }
```

Goroutinen (3)

Synchronisierung

`https://golang.org/pkg/sync/`

```
1  daten := []Object{...}
2  var wg sync.WaitGroup
3  for _, d := range daten {
4    wg.Add(1)
5    go func(o Object) {
6      defer wg.Done()
7      verarbeite(o)
8    }(d) // <- !!!
9  }
10 wg.Wait()
```

Unit Tests

https://golang.org/pkg/testing/

```
1  package zeugs
2
3  // mult multipliziert die beiden
4  // übergebenen Argumente und gibt
5  // das Ergebnis zurück.
6  func mult(a, b int) int {
7    return a * b
8  }
```

Unit Tests (2)

```
1  package zeugs
2  func TestMult(t *testing.T) {
3  for _, test := range []struct{
4    a, b, want int
5  }{
6    {3, 4, 12},
7    {7, 8, 56},
8  } {
9    if r := mult(test.a, test.b); r != test.want {
10     t.Errorf("mult(%d, %d) = %d; want %d",
11       test.a, test.b, r, test.want)
12   }
13 }
```

Übung: Äquivalente Binärbäume

`https://go-tour-de.appspot.com/concurrency/7`
`https://godoc.org/golang.org/x/tour/tree`
`https://de.wikipedia.org/wiki/Binärbaum`

- Implementiere die Same(t1, t2 ∗tree.Tree) Funktion aus der Online-Übung.
- Zusätzlich: Erstelle einen Unit-Test für die Funktion.

Werkzeuge

Go ist dazu gedacht, in Werkzeugen verwendet zu werden (`go/ast`, etc.)

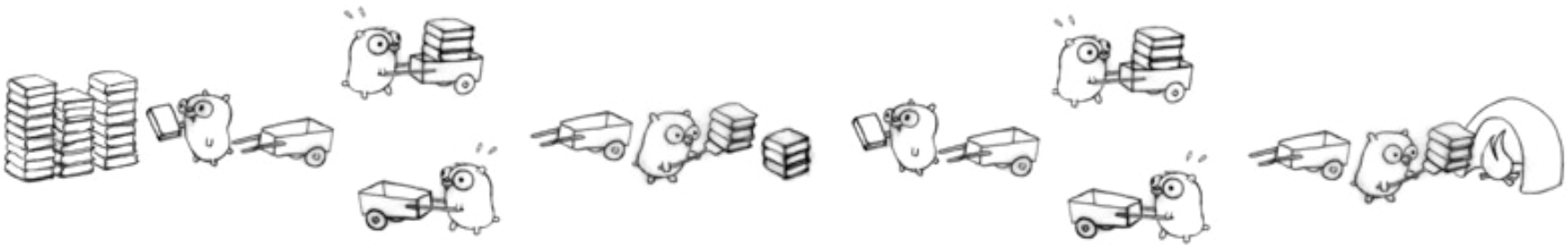
- gofmt, goimports
- godoc, `https://godoc.org/`
- IDE und Editor Unterstützung

vim:

```
1 autocmd filetype go
2   \ autocmd BufWritePre <buffer> Fmt
3 let g:gofmt_command = "goimports"
```

Danke für die Aufmerksamkeit

Fragen, Kommentare?



`https://tour.golang.org` — `https://play.golang.org`

Feedback: `https://glt16-programm.linuxtage.at/`