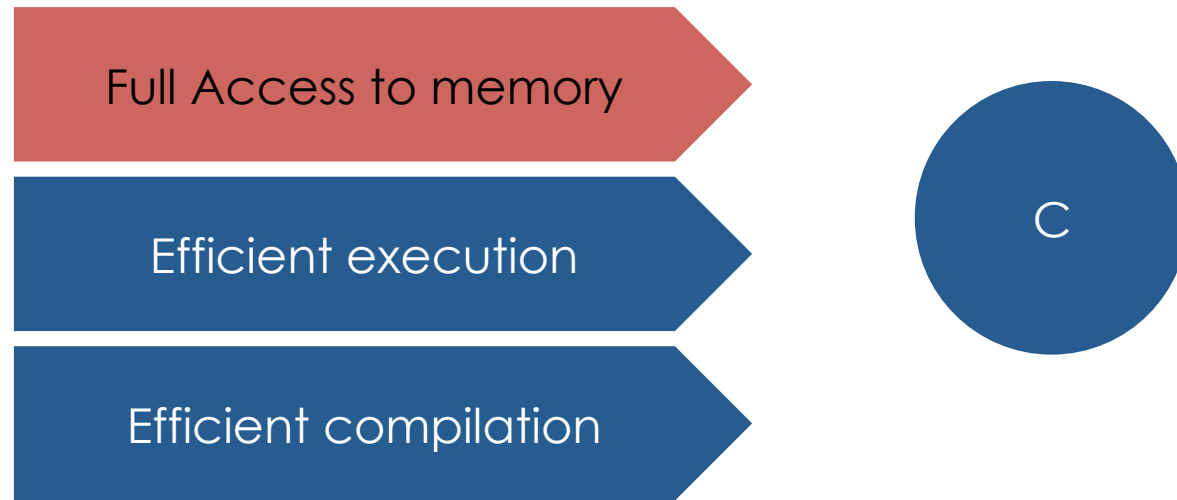


System programming languages



Golang, Erlang, RUST, C

Systeemprogrammeren



Voor systeemprogrammeren wordt nog steeds de oude taal C gebruikt (ontwikkeld van 1973 - 1978).

C is heel snel maar mist moderne features. Objectge-oriënteerde variant C++ is moeilijk te leren en daardoor foutgevoelig.

Full access to memory heeft nadelen

```
void main(  
{  
    int* p =2000;  
    *p=12;  
}
```

- Met C kun elk programma elke geheugenplaats aanspreken en zetten.
- Dit programma zet de waarde '12' neer op geheugenplaats 2000.
- Voor sommige toepassingen heel nuttig, meestal erg gevaarlijk en onveilig.
- Feature is niet uit te schakelen: basisconcepten als arrays zijn hierop gebouwd

Nieuwe, alternatieve talen:

RUST

- RUST: gepubliceerd in 2010, ontwikkeld bij Mozilla research.
- Maakt zoveel mogelijk gebruik van compile time checking om zowel snel als veilig te zijn
- Heeft concurrency ingebakken
- Nog weinig ge-adopteerd: Door docenten nog niet in het wild gezien

Innovatieve niche-taal: Erlang

```
% Create a process and invoke the function web:start_server(Port, MaxConnections)
ServerProcess = spawn(web, start_server, [Port, MaxConnections]),

% Create a remote process and invoke the function
% web:start_server(Port, MaxConnections) on machine RemoteNode
RemoteProcess = spawn(RemoteNode, web, start_server, [Port, MaxConnections]),

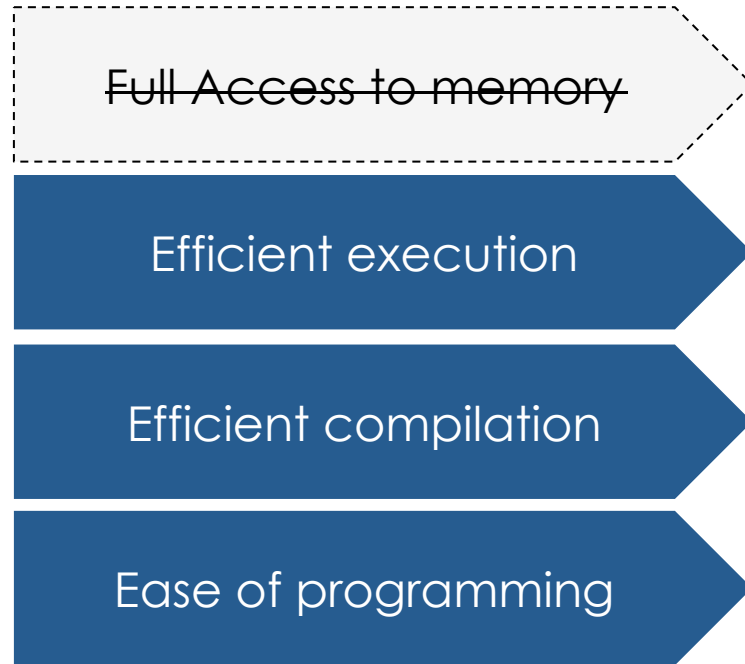
% Send a message to ServerProcess (asynchronously). The message consists of a tuple
% with the atom "pause" and the number "10".
ServerProcess ! {pause, 10},

% Receive messages sent to this process
receive
    a_message -> do_something;
    {data, DataContent} -> handle(DataContent);
    {hello, Text} -> io:format("Got hello message: ~s", [Text]);
    {goodbye, Text} -> io:format("Got goodbye message: ~s", [Text])
end.
```

- Ontwikkeld in 1986 by Ericson, in 1998 open source gemaakt
- Ontwikkeld als programmeertaal voor een zeer betrouwbare Internet-switch
- Code is hot-switchable: code kan gewijzigd worden zonder programma af te sluiten
- Automatisch geheugenbeheer en ondersteuning voor meerdere processen

Bron codevoorbeeld: [http://en.wikipedia.org/wiki/Erlang_\(programming_language\)](http://en.wikipedia.org/wiki/Erlang_(programming_language))

Waarom GO?



Go is een nieuwe taal gericht op systeemprogrammeren, mogelijke vervanger van C

Achtergrond GO



- Ontwikkeld van 2007 (eerste ideeën) tot 2009 (publicatie als open source project).
- Huidige versie 1.0 dateert van 2012.
- Auteurs Robert Griesemer, Rob Pike and Ken Thompson, werknemers van Google
- Go is open source en gratis te gebruiken.

Bronnen:

<http://golang.org/doc/faq>

<http://www.golang-book.com>

Voorbeeld programma Go

```
package main
```

```
import (  
    "fmt"  
    "io/ioutil"  
)
```

Package-management.

```
type Page struct {  
    Title string  
    Body []byte  
}
```

objectdefinitie

```
func (p *Page) save() error {  
    filename := p.Title + ".txt"  
    return ioutil.WriteFile(filename, p.Body, 0600)  
}
```

functiedefinitie

```
func loadPage(title string) (*Page, error) {  
    filename := title + ".txt"  
    body, err := ioutil.ReadFile(filename)  
    if err != nil {  
        return nil, err  
    }  
    return &Page{Title: title, Body: body}, nil  
}
```

```
func main() {  
    p1 := &Page{Title: "TestPage", Body: []byte("This is a sample Page.")}  
    p1.save()  
    p2, _ := loadPage("TestPage")  
    fmt.Println(string(p2.Body))  
}
```

Hoofdfunctie. Maakt een object, bewaart het in een bestand, en drukt de inhoud van het bestand af

Go features:

```
func f() (int, int) {  
    return 5, 6  
}
```

```
x, y := f()
```

Meerdere waarden teruggeven

```
func add(args ...int) int {  
    total := 0  
    for _, v := range args {  
        total += v  
    }  
    return total  
}
```

```
fmt.Println(add(1,2,3))
```

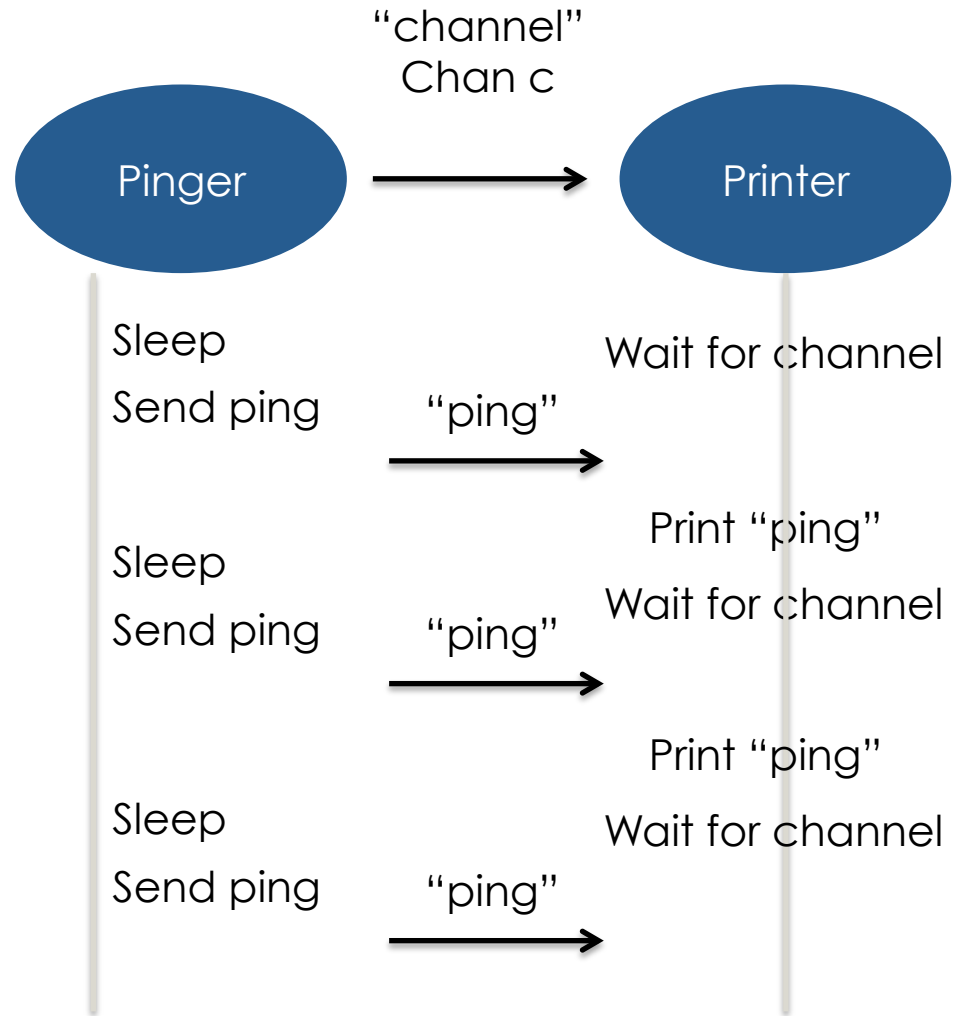
Funcies met variabel aantal argumenten

```
func one(xPtr *int) {  
    *xPtr = 1  
}  
func main() {  
    xPtr := new(int)  
    one(xPtr)  
    fmt.Println(*xPtr) // x is 1  
}
```

Werken met pointers naar variabelen

Go Concurrency

```
...  
func pinger(c chan string) {  
    for i := 0; ; i++ {  
        c <- "ping"  
    }  
}  
func printer(c chan string) {  
    for {  
        msg := <- c  
        fmt.Println(msg)  
        time.Sleep(time.Second * 1)  
    }  
}  
  
func main() {  
    var c chan string = make(chan string)  
    go pinger(c)  
    go printer(c)  
    ...  
}
```



Gebaseerd op Hoare CSP:
http://en.wikipedia.org/wiki/Communicating_sequential_processes

Discussie: voor wie is Go?



- Aanpassingen aan besturings-systeem
- Grote 'totaal'-projecten
- Projecten waar executiesnelheid belangrijk is
- Web-development
- Embedded systemen
- Omgevingen waar Go nieuw is
- Kleine systemen in omgevingen waar niemand Go kent
- .NET / microsoft omgevingen