

АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

**Лекция 10:
Языки Erlang и Go**

Erlang

Примеры из <http://www.erlang.org/course/course.html>



Константы

Числовые

- **Integers**

-234

16#AB10F – 16-ричные

2#110111010 – двоичные

\$A – символ

- **Floats**

•6.654

12.34E-10

Атомы

abcef

atom_with_underscores

'Blanks can be quoted'



Структуры данных

Кортежи

{123, bcd}

{123, def, abc}

{person, 'Joe', 'Armstrong'}

{abc, {def, 123}, jkl}

{}

Списки

[123, xyz]

[123, def, abc]

[{person, 'Joe', 'Armstrong'},

{person, 'Robert', 'Virding'},

{person, 'Mike', 'Williams'}]

"abcd" == [97,98,99,100]

"" == []



Структуры данных

Сложные структуры:

```
[{{person,'Joe', 'Armstrong'},  
  {telephoneNumber, [3,5,9,7]},  
  {shoeSize, 42},  
  {pets, [{cat, tubby},{cat, tiger}]},  
  {children,[{thomas, 5},{claire,1}]}},  
{{person,'Mike','Williams'},  
 {shoeSize,41},  
 {likes,[boats, beer]},  
 ...
```



Переменные

Abc

A_long_variable_name

AnObjectOrientatedVariableName

- С заглавной буквы
- Однократно инициализируются, значение не изменяется



Сопоставление с образцом

(Pattern Matching)

$A = 10$ % Succeeds - binds A to 10

$\{B, C, D\} = \{10, \text{foo}, \text{bar}\}$

 % Succeeds - $B := 10$, $C := \text{foo}$, $D := \text{bar}$

$\{A, A, B\} = \{\text{abc}, \text{abc}, \text{foo}\}$

 % Succeeds - binds A to abc, B to foo

$[A, B, C] = [1, 2, 3]$

 % Succeeds - binds A to 1, B to 2, C to 3

$\{A, A, B\} = \{\text{abc}, \text{def}, 123\}$ % **Fails**

$[A, B, C, D] = [1, 2, 3]$ % **Fails**



Сопоставление с образцом

$[A,B|C] = [1,2,3,4,5,6,7]$ % OK, $A := 1$, $B := 2$, $C := [3,4,5,6,7]$

$[H|T] = [1,2,3,4]$ % OK - binds $H = 1$, $T = [2,3,4]$

$[H|T] = [abc]$ % OK - binds $H = abc$, $T = []$

$[H|T] = []$ % Fails

$\{A, _, [B|_], \{B\}\} = \{abc, 23, [22, x], \{22\}\}$
% OK- binds $A = abc$, $B = 22$



Функции

ВЫЗОВЫ:

- `module:func(Arg1, Arg2, ... Argn)`
- `func(Arg1, Arg2, .. Argn)`

Объявление:

`func(Pattern1, Pattern2, ...) -> ... ;`

`func(Pattern1, Pattern2, ...) -> ... ;`

`...`

`func(Pattern1, Pattern2, ...) -> ... .`



Функции

`factorial(0) -> 1;`

`factorial(N) -> N * factorial(N-1).`

`area({square, Side}) -> Side * Side;`

`area({circle, Radius}) -> % almost :-)`

`3.14 * Radius * Radius;`



Модули

- module(demo).
- export([double/1]).
- import(Module,Functions).

double(X) -> times(X, 2).

times(X, N) -> X * N.



Сторожевые выражения

factorial(N) when $N > 0 \rightarrow$
 $N * \text{factorial}(N - 1);$
factorial(0) $\rightarrow 1$.

Примеры сторожевых выражений:

- $\text{number}(X)$ - X is a number
- $\text{atom}(X)$ - X is an atom
- $\text{list}(X)$ - X is a list
- $\text{length}(X) == 3$ - X is a list of length 3
- $X > Y + Z$ - X is $> Y + Z$



Рекурсия

average(X) ->
sum(X) / len(X).

sum([H|T]) -> H +
sum(T);
sum([]) -> 0.

len([_|T]) -> 1 + len(T);
len([]) -> 0.

double([H|T]) ->
[2*H|double(T)];
double([]) -> [].

member(H, [H|_]) ->
true;
member(H, [_|T]) ->
member(H, T);
member(_, []) -> false.



Рекурсия: оптимизация

`average(X) -> average(X, 0, 0).`

`average([H|T], Length, Sum) ->`

`average(T, Length + 1, Sum + H);`

`average([], Length, Sum) -> Sum / Length.`

- Однократный проход
- Используются аккумуляторы `Length` и `Sum`
- Константное требование к памяти:
оптимизация хвостовой рекурсии



Специальные формы

case lists:member(a, X) **of**

 true ->

 ... ;

 false ->

 ...

end,

...

if

 integer(X) -> ... ;

 tuple(X) -> ...

end,

...



Многозадачность

Process – конкурирующая задача, работает в контексте виртуальной машины

Message – механизм взаимодействия между процессами

Registered Process – процесс, зарегистрированный под именем

Client/Server Model – стандартная модель построения многозадачных систем



Порождение процесса

```
Pid2 = spawn(Mod, Func, Args)
```

```
% Pid2 – новый процесс в системе
```

```
% Отправка сообщения
```

```
Pid2 ! {self(), foo} % свой PID и атом
```

```
% Получение сообщения
```

```
receive
```

```
    {A, foo} -> ...;
```

```
end
```



Пример: echo-процесс

```
-module(echo).  
-export([go/0, loop/0]).
```

```
go() ->  
    Pid2 = spawn(echo, loop, []),  
    Pid2 ! {self(), hello},  
    receive  
        {Pid2, Msg} -> io:format("P1 ~w~n",[Msg])  
    end,  
    Pid2 ! stop.
```

```
loop() ->  
    receive  
        {From, Msg} ->  
            From ! {self(), Msg},  
            loop();  
        stop -> true  
    end.
```



Регистрация процесса

start() ->

```
Pid = spawn(num_anal, server, [])  
register(analyser, Pid).
```

analyse(Seq) ->

```
analyser ! {self(), {analyse, Seq}},
```

```
receive
```

```
    {analysis_result, R} ->
```

```
        R
```

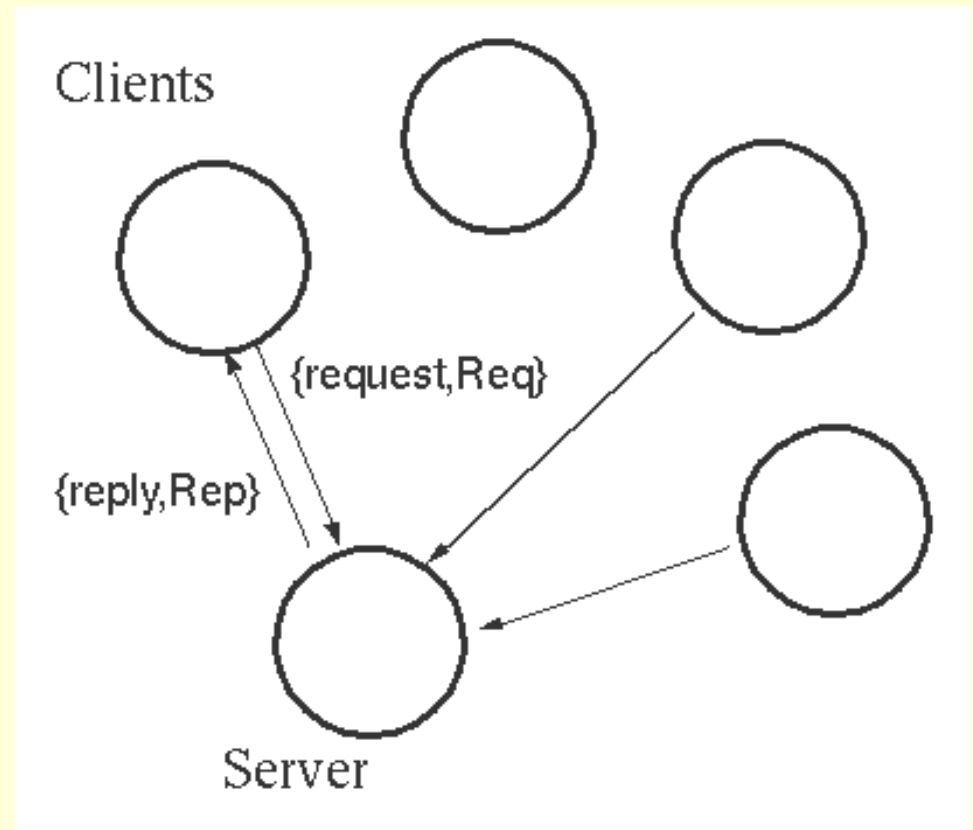
```
end.
```



Клиент-сервер

```
%Server code
-module(myserver).
server(Data) ->
  receive
    {From,{request,X}} ->
      {R, Data1} = fn(X, Data),
      From ! {myserver,{reply, R}},
      server(Data1)
  end.

% Interface library
-export([request/1]).
request(Req) ->
  myserver ! {self(),{request,Req}},
  receive
    {myserver,{reply,Rep}} -> Rep
  end.
```

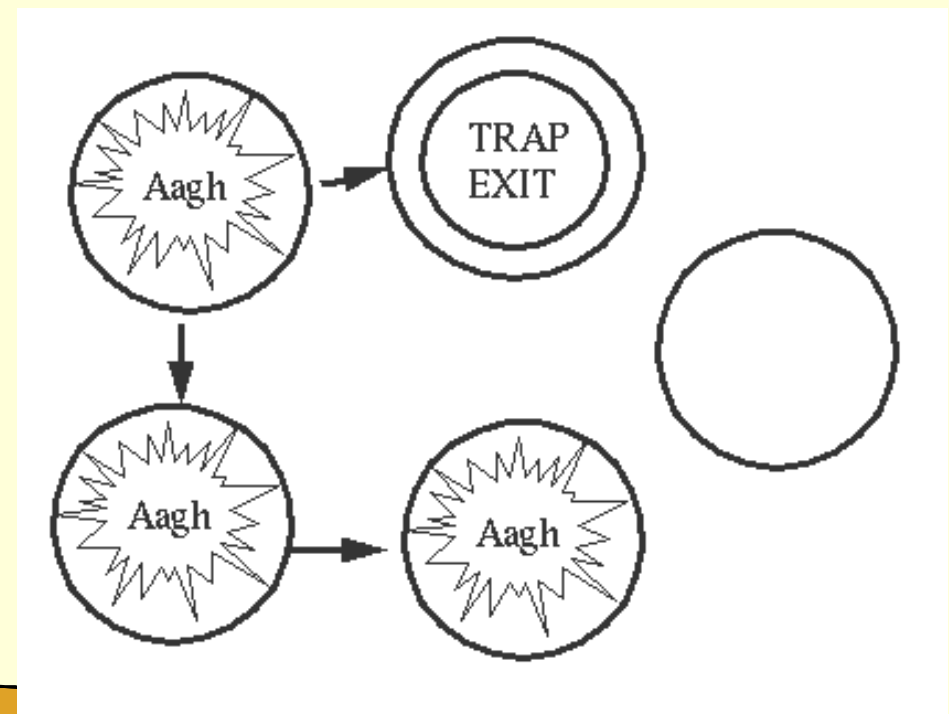


Обработка ошибок

Exit Signal – передача информации о завершении процесса

Link – двунаправленная связь для распространения Exit Signals

Error trapping – способность процесса обрабатывать сигналы завершения подобно обычным сообщениям



Примитивы

link(Pid) – установление двунаправленной связи текущего процесса и **Pid**

process_flag(trap_exit, true) – превращает выходные сообщения данного процесса в выходные сообщения, принимаемые в обычном выражении **receive ... end**

exit(Reason) – завершает процесс и генерирует сигнал завершения с информацией о завершении - **Reason**



Обработка

link(P1), % связали с P1
receive

% Перехват завершения:

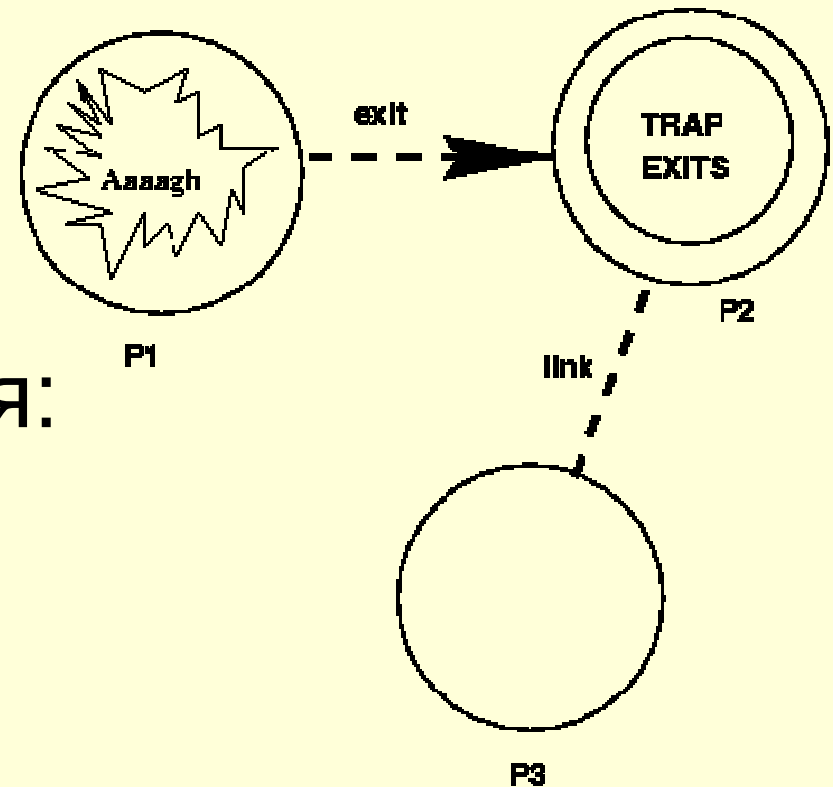
{'EXIT', P1, Why} ->

... exit signals ...

{P3, Msg} ->

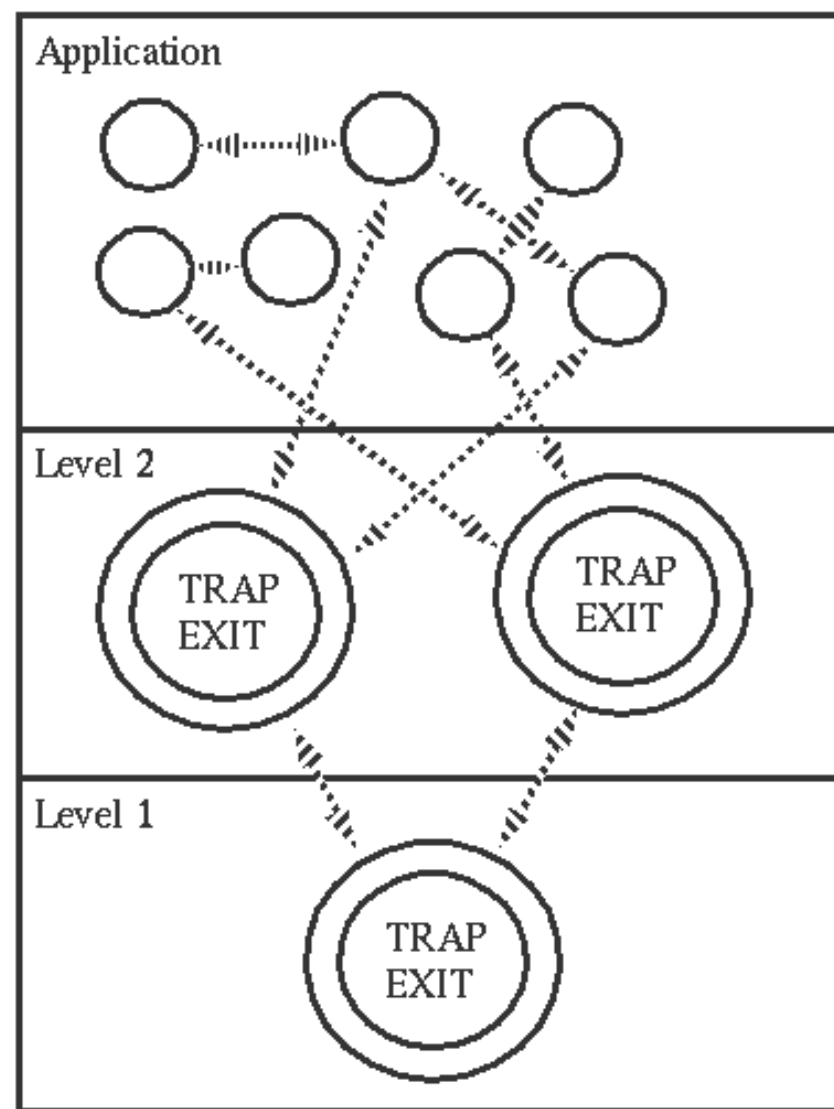
... normal messages ...

end



Устойчивые к сбоям системы

- Локализация ошибок по уровням
- На прикладном уровне процессы в случае ошибки просто завершаются
- Фреймворк **ОТР** (Open Telecom Platform)



Полезные ссылки

На русском:

- <http://rdsn.ru/article/erlang/GettingStartedWithErlang.xml>
- <http://erlanger.ru>

На английском:

- <http://www.erlang.org>, в частности,
- <http://www.erlang.org/download.html> - инструменты
- <http://www.erlang.org/course/course.html>
- <http://chimera.labs.oreilly.com/books/12340000000726> -
Этюды на Erlang



Go

Примеры из <http://tour.golang.org>



Пакеты

```
package main
```

```
import (  
    "fmt"  
    "math"  
)
```

```
func main() {  
    fmt.Println("Happy", math.Pi, "Day")  
}
```



Функции

```
package main
```

```
import "fmt"
```

```
func add(x int, y int) int {  
    return x + y  
}
```

```
func main() {  
    fmt.Println(add(42, 13))  
}
```



Функции: возврат кортежа

```
package main
```

```
import "fmt"
```

```
func swap(x, y string) (string, string) {  
    return y, x  
}
```

```
func main() {  
    a, b := swap("hello", "world")  
    fmt.Println(a, b)  
}
```



Функции: именованные результаты

```
package main
```

```
import "fmt"
```

```
func split(sum int) (x, y int) {  
    x = sum * 4 / 9  
    y = sum - x  
    return  
}
```

```
func main() {  
    fmt.Println(split(17))  
}
```



Переменные

```
package main
```

```
import "fmt"
```

```
var x, y, z int = 1, 2, 3
```

```
func main() {  
    c, python, java := true, false, "no!"  
    fmt.Println(x, y, z, c, python, java)  
}
```



Типы данных

- `bool`
- `string`
- `int` `int8` `int16` `int32` `int64`
- `uint` `uint8` `uint16` `uint32` `uint64` `uintptr`
- `byte` // alias для `uint8`
- `rune` // alias для `int32`, Unicode code point
- `float32` `float64`
- `complex64` `complex128`



Циклы

```
package main
```

```
import "fmt"
```

```
func main() {  
    sum := 0  
    for i := 0; i < 10; i++ {  
        sum += i  
    }  
    fmt.Println(sum)  
}
```



Циклы

```
package main
```

```
import "fmt"
```

```
func main() {  
    sum := 1  
    for sum < 1000 {  
        sum += sum  
    }  
    fmt.Println(sum)  
}
```



Условия

```
func pow(x, n, lim float64) float64 {  
    if v := math.Pow(x, n); v < lim {  
        return v  
    } else {  
        fmt.Printf("%g >= %g\n", v, lim)  
    }  
    // can't use v here, though  
    return lim  
}
```



Структуры и указатели

```
package main  
import "fmt"
```

```
type Vertex struct {  
    X int  
    Y int  
}
```

```
func main() {  
    p := Vertex{1, 2}  
    q := &p  
    q.X = 1e9  
    fmt.Println(p)  
}
```



Срезы

```
func main() {  
    a := make([]int, 5) // len=5 [0 0 0 0 0]  
    b := make([]int, 0, 5) // len=0, cap=5 []  
    c := b[:2] // len=2 cap=5 [0 0]  
    d := c[2:5] // len=3 cap=3 [0 0 0]  
  
    // итерирование по срезу:  
    var pow = []int{1, 2, 4, 8, 16, 32, 64, 128}  
    for i, v := range pow {  
        fmt.Printf("2**%d = %d\n", i, v)  
    }  
}
```



Отображение (словарь)

```
type Vertex struct {  
    Lat, Long float64  
}
```

```
var m map[string]Vertex
```

```
func main() {  
    m = make(map[string]Vertex)  
    m["Bell Labs"] = Vertex{  
        40.68433, -74.39967,  
    }  
    fmt.Println(m["Bell Labs"])  
}
```



Лямбда-функции

```
package main
```

```
import (  
    "fmt"  
    "math"  
)
```

```
func main() {  
    hypot := func(x, y float64) float64 {  
        return math.Sqrt(x*x + y*y)  
    }
```

```
    fmt.Println(hypot(3, 4))  
}
```



Замыкания

```
package main
```

```
import "fmt"
```

```
func adder() func(int) int {  
    sum := 0  
    return func(x int) int {  
        sum += x  
        return sum  
    }  
}
```



Switch

```
func main() {  
    fmt.Println("When's Saturday?")  
    today := time.Now().Weekday()  
  
    switch time.Saturday {  
    case today + 0:  
        fmt.Println("Today.")  
    case today + 1:  
        fmt.Println("Tomorrow.")  
    case today + 2:  
        fmt.Println("In two days.")  
    default:  
        fmt.Println("Too far away.")  
    }  
}
```



Методы

Go не имеет классов, но можно определять методы для структур и пользовательских типов:

```
type Vertex struct {  
    X, Y float64  
}
```

```
func (v *Vertex) Abs() float64 {  
    return math.Sqrt(v.X*v.X + v.Y*v.Y)  
}
```

```
func main() {  
    v := &Vertex{3, 4}  
    fmt.Println(v.Abs())  
}
```



Интерфейсы

Инструмент обеспечения полиморфизма:

```
type Abser interface {  
    Abs() float64  
}
```

```
type Vertex struct {  
    X, Y float64  
}
```

```
func (v *Vertex) Abs() float64 {  
    return math.Sqrt(v.X*v.X + v.Y*v.Y)  
}
```

```
var a Abser = &Vertex{3, 4}  
result := a.Abs()
```



Многозадачность: goroutines

```
func say(s string) {  
    for i := 0; i < 5; i++ {  
        time.Sleep(100 * time.Millisecond)  
        fmt.Println(s)  
    }  
}
```

```
func main() {  
    go say("world")  
    say("hello")  
}
```



Каналы

```
func sum(a []int, c chan int) {  
    sum := 0  
    for _, v := range a {  
        sum += v  
    }  
    c <- sum // send sum to c  
}
```

```
func main() {  
    a := []int{7, 2, 8, -9, 4, 0}  
  
    c := make(chan int)  
    go sum(a[:len(a)/2], c)  
    go sum(a[len(a)/2:], c)  
    x, y := <-c, <-c // receive from c  
  
    fmt.Println(x, y, x+y)  
}
```



Каналы: буферизация

```
package main
```

```
import "fmt"
```

```
func main() {  
    c := make(chan int, 2)  
    c <- 1  
    c <- 2  
    fmt.Println(<-c)  
    fmt.Println(<-c)  
}
```



Каналы: итерирование и закрытие

```
func fibonacci(n int, c chan int) {  
    x, y := 0, 1  
    for i := 0; i < n; i++ {  
        c <- x  
        x, y = y, x+y  
    }  
    close(c)  
}
```

```
func main() {  
    c := make(chan int, 10)  
    go fibonacci(cap(c), c)  
    for i := range c {  
        fmt.Println(i)  
    }  
}
```



Каналы: выборка

```
func fibonacci(c, quit chan int)
{
    x, y := 0, 1
    for {
        select {
        case c <- x:
            x, y = y, x+y
        case <-quit:
            fmt.Println("quit")
            return
        }
    }
}
```

```
func main() {
    c := make(chan int)
    quit := make(chan int)
    go func() {
        for i := 0; i < 10; i++ {
            fmt.Println(<-c)
        }
        quit <- 0
    }()
    fibonacci(c, quit)
}
```



Полезные ссылки

- Официальный сайт <http://golang.org>
- В том числе, тур по языку (с интерактивной средой)
<http://tour.golang.org>
- Учебник Learning Go
<http://www.miek.nl/files/go/20130414-go.pdf>

