
University of
Kent

University of Kent

Going further with Erlang

Macros

Records

Maps

Binaries

University of Kent

Macros

Macros expanded by the Erlang Preprocessor, EPP.

Macros can take parameters.

Macros invoked with `?`

Macros often stored in `include` files

`??X` is the argument `X` as a string.

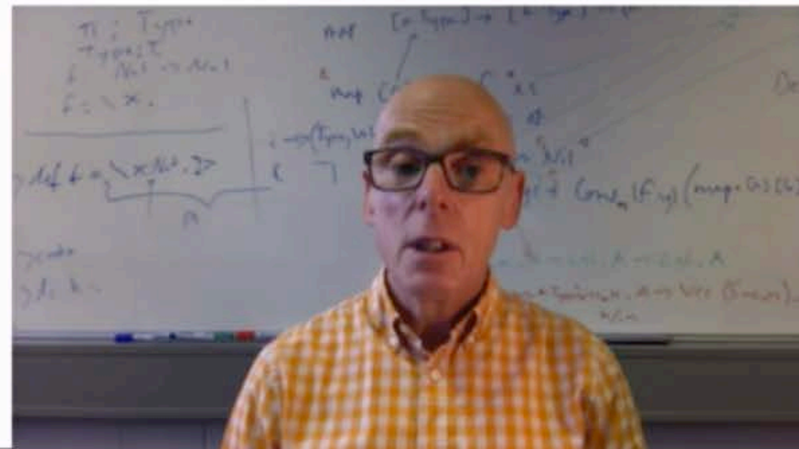
```
%% defs.hrl

-define(ONE, 1).

-define(TWICE(F),(fun(P)->F(F(P))end)).

-define(FMT(S,As),io:format(S,As)).

-define(SHOW(X),?FMT("~p=~p~n",[??X,X])).
```



University of Kent

Macros

Macros expanded by the Erlang Preprocessor, EPP.

Macros can take parameters.

Macros invoked with `?`

Macros often stored in `include` files
`??X` is the argument `X` as a string.

```
%% defs.hrl
-define(ONE, 1).
-define(TWICE(F),(fun(P)->F(F(P))end)).
-define(FMT(S,As),io:format(S,As)).
-define(SHOW(X),?FMT("~p=~p~n",[??X,X])).
```

```
-module(foo).
-include("defs.hrl").
succ(X)     -> X+?ONE.
add_two(X) -> ?TWICE(succ)(X).
show()      -> ?SHOW(add_two(7)).
```

University of Kent

Records

```
%% recs.hrl

-record(twitter, {name,handle}).
```

Records expanded by the EPP.

Record use with #

Notations for
- construction,
- selection,
- pattern matching,
- update.

University of Kent

Records

Records expanded by the EPP.

Record use with #

Notations for
- construction,
- selection,
- pattern matching,
- update.

```
%% recs.hrl

-record(twitter, {name,handle}).
```

```
-module(foo).
-include("recs.hrl").

r1() -> #twitter{name="simon",
                 handle="@si"}.

get_name(X) -> X#twitter.name.

update_si(#twitter{name="simon"} = X) ->
  X#twitter{name="si"};
update_si(X) ->
  X.
```

University of Kent

Maps

Standard `map:xxx` notation plus some extra syntax.

Notations for
- construction,
- selection,
- pattern matching,
- update.

`=>` can add a field,
`:=` updates only.

```
-module(foo).

map1() -> #{name => "simon",
            handle => "@si"}.

get_name(#{name := Y}) -> Y.

update_si(#{name := "simon"} = X) ->
  X#{name := "si"};
update_si(X) ->
  X.
```

University of Kent

Binaries

Notations for binary data structures, and pattern mapping over them.

Ideal for protocol analysis and implementation.

`E:5` matches 5 bits,
`E` matches n×8 bits.

```
1> Bin = <<22,23,24>>.
<<22,23,24>>
2> <<C:4,D:4,E:5,F:4,G:7>> = Bin.
<<22,23,24>>
3> C.
1
4> D.
6
5> E.
2
6> F.
14
7> G.
24
```

University of Kent

Binaries

Notations for binary data structures, and pattern mapping over them.

Ideal for protocol analysis and implementation.

`E:5` matches 5 bits,
`E` matches n×8 bits.

```
1> Bin = <<22,23,24>>.
<<22,23,24>>
2> <<C:4,D:4,E:5,F:4,G:7>> = Bin.
<<22,23,24>>
3> C.
1
4> D.
6
5> E.
2
6> F.
14
7> G.
24
```

University of Kent

Binaries

Notations for binary data structures, and pattern mapping over them.

Ideal for protocol analysis and implementation.

`E:5` matches 5 bits,
`E` matches n×8 bits.

```
1> Bin = <<22,23,24>>.
<<22,23,24>>
2> <<C:4,D:4,E:5,F:4,G:7>> = Bin.
<<22,23,24>>
3> C.
1
4> D.
6
5> E.
2
6> F.
14
7> G.
24
```


University of
Kent