Erlang Solutions Ltd.

Binaries

Binaries

- A binary is a reference to a chunk of untyped memory
- Originally used by the ERTS for code loading over the network
- Effective when moving large amounts of data among processes
- BIFs
  - **binary_to_term/1, term_to_binary/1, binary_to_list/1, split_binary/2, concat_binary/1**
  - **is_binary/1** as a guard

Binaries

```
Bin = <<E1, E2, …, En>>
<<E1, E2, …, En>> = Bin

Bin = <<1, 2, 3>>
binary_to_list(Bin) == [1,2,3]
```

- A **Bin** is a low level sequence of bytes
- They can be used to construct and pattern match Binaries
- Each element specifies a segment of the binary
- A segment is a set of bits, not necessarily a byte

Binaries

```
Bin = <<E1:[Size/Type], E2:[Size/Type], ..., En>>

Bin = <<1,2,3:16>> == list_to_binary([1,2,0,3])
```

- **Size** and **Type**s can be specified or omitted
- **Size** is in bits. Total size must be a multiple of 8 (a byte)
- **Type** is a list of type specifiers separated by hyphens
- Valid types are **integer**, **float** or **binary**

Binaries

- Valid signs are **signed** and **unsigned** (Default)
  - If the first bit is 0, the integer is positive
  - If it is 1, it is negative
- Valid endian values are **big** (Default) and **little**
  - Little endian, the first byte is the most significant
  - Big endian, the first byte is the least significant
- Default size of
  - **integers** is 8
  - **floats** is 64
  - **binaries** is the size of the binary

Binaries

```
<< 5:4/little-signed-integer-unit:8>> == <<5,0,0,0>>

<< 5:4/big-signed-integer-unit:8>> == <<0,0,0,5>>
```

- **unit:Val** is the default size of the type times the unit
  - Val is an integer between 1 – 255
  - Default unit is type dependent. It is 1 for float / integer, 8 for binary
- The element created has a size of (4*8) 32 bits

Binaries

```
A = 1, Bin = <<A, 17, 42:16>>
<<D:16, E, F/binary>> = Bin

D = 273, E = 0, binary_to_list(F) = [42]
```

- A Bin can be used to pattern match binaries
- Length and types can be specified or omitted
- Default types are unsigned integers
- Binary segments types must have a size divisible by 8

Binaries

- `B=<<1>>` will not compile. It is interpreted as `B =< <1>>`. Write `B = <<1>>`
- `<<X+1:8>>` will not compile. Write `<<(X+1):8>>`
- `<<"hello">>` is the same as writing `<<$h,$e,$l,$l,$o>>`
- `foo(N, <<X:N, T/binary>>) -> ...` will not compile. The two instances of N are unrelated
- `<<X:7/binary-unit:1, Y/binary>>` will never match. The size and unit, when multiplied, must be divisible by 8.

Binaries

- `<<X:7/binary, Y:1/binary>> = Z` will match because binaries have a default unit of 8. It is equivalent to `<<X:7/binary-unit:8, Y:1/binary-unit:8>> = Z`. 7*8 = 56 bits binary.
- `<<X:7/integer, Y/binary>> = Z` will fail because integers have a default unit of 1. When multiplied by 7, it isn't divisible by 8.
- `<<X:7/bitstring-unit:1, Y/bitstring>> = Z` will work, because the **bitstring** type doesn't require 8-bit alignment.

Binaries: examples

```
1> <<5:4, 5:4>>.
<<"U">>
2> <<Int1:2, Int2:6>> = <<128>>.
<<128>>
3> {Int1, Int2}.
{2,0}
4> <<5:4/little-signed-integer-unit:4>>.
<<5,0>>
5> <<5:4/big-signed-integer-unit:4>>.
<<0,5>>
6> <<5:2,5:8>>.
<<65,1:2>>
```


Binaries: examples

```
1> A = 1.
1
2> Bin = <<A, 17, 42:16>>.
<<1,17,0,42>>
3> <<D:16, E, F/binary>> = Bin.
<<1,17,0,42>>
4> [D,E,F].
[273,0,<<"*">>]
5> <<X:7/bitstring,Y:1/bitstring>> = <<42:8>>.
<<"*">>
6> {X, Y}.
{<<21:7>>,<<0:1>>}
```


Binaries: examples

```
<<?IP_VERSION:4, HLen:4, SrvcType:8, TotLen:16,
  ID:16, Flgs:3, FragOff:13,
  TTL:8, Proto:8, HdrChkSum:16,
  SrcIP:32, DestIP:32,
  RestDgram/binary>>
```

- IP datagram of IP protocol version 4

Binaries

- Concatenating binaries will copy the chunks and create a completely new binary
- Concatenate your binary once you have all the chunks
- There is no need to append binary chunks sent to a port
- There is no need to flatten lists of binary chunks