Cheatography

Erlang Binaries Cheat Sheet

by Fylke via cheatography.com/93081/cs/20609/

Overview

Erlang can match binaries just as any list of things. `<<E1, E2, E3>> = Bin` divides the binary `Bin` into three elements of type `integer` (which is default) of one byte each. This means that `Bin` has to be 24 bits long, or we get a `badmatch`. You can also make partial matches, in a `[Head | Tail]` fashion, by putting `/bitstring` on the last element, like so: `<<E1, E2, E3/bitstring>> = Bin`. This is a **type modifier** and tells Erlang that there are two 8-bit elements, in `E1` and `E2` respectively, and then an undetermined amount of bits stored in `E3`..

Type Modifiers

Type	Size in bits	Remarks
`integer`	8	Default type
`float`	64	
`binary|bytes`	8 per chunk	Anything matched must be of size evenly divisible by 8
`bitstring|bits`	1 per chunk	Will always match
`utf8|-utf16|utf32`	8-32, 16-32, and 32	`<<"abc"/utf8>>` is the same as `<<-$a/utf8, $b/utf8, $c/utf8>>`

Other Type Modifiers

Kinds	Values	Remarks
Signedness	`signed|`**`unsigned`**	Unsigned is default
Endianness	**`big`**`|little|native`	`native` is resolved at load time to whatever the CPU uses
Custom unit	`unit:IntLiteral`	Define a custom unit of length 1..256

Segments

Each segment in a binary has the following general syntax: `Value:Size/TypeSpecifierList`. The `Size` or the `TypeSpecifier`, or both, can be omitted.
`Value` is either a literal or a variable, `Size` is multiplied by the unit in `TypeSpecifierList`, and can be any expression that evaluates to an `integer`.

Contrived example: `<<X:4/little-signed-integer-unit:8>>` has a total size of 4*8 = 32 bits, and it contains a signed integer in little endian byte order.

Examples

Expression	Result
`<<"abc">>`	`<<97,98,99>>`
`<<1,17,42:16>>`	`<<1,17,0,42>>`
`<<A,B,C:16>> = <<1,17,42:16->>`	`C = 42`
`<<D:16,E,F>> = <<1,17,42:16->>`	`D = 273` and `F = 42`
`<<G,Rest/binary>> = <<1,17,42:12>>`	`<<1,17,0,42>>`
`<<G,Rest/bitstring>> = <<1,17,42:12>>`	`<<1,17,2,10:4>>`
`<<IntAsLittleEndian>> = <<IntAsBigEndian/little>>`	`IntAsBigEndian = 9494` and `IntAsLittleEndian = 22`
`<<"pöpcörn"/utf8>>`	This is how Erlang handles unicode

When constructing a binary, if the size of an integer `N` is too large to fit inside the given segment, the most significant bits are silently discarded and only the `N` least significant bits kept.

Binary Comprehension Examples

Just like with lists, there is a notation for binary comprehension. It works almost the same, you just exchange the <- before the generator for a <=. Here is an example of how to use this to convert a 32 bit integer into a hex representation:

```
int_as_hex(Int) ->
    IntAsBin = <<Int:32>>,
    "0x" ++ lists:flatten([byte_to_hex(<<Byte>>)
|| <<Byte:8>> <= IntAsBin]).
byte_to_hex(<<Nibble1:4, Nibble2:4>>) ->`
    [integer_to_list(Nibble1, 16), integer_to_l-
ist(Nibble2, 16))].
```

Depending on what you put around the comprehension expression, the end result is either a binary, or as above, a list.

By **Fylke**
cheatography.com/fylke/

Published 25th September, 2019.
Last updated 27th September, 2019.