

# The problem with integer types

C defines integer types `int`, `long`, `short`, `char`, `signed` or `unsigned`.

These types raise numerous problems in portable code:

- They change size from one arch/cpu to another
- There is no integer type which does the size of a pointer

## C99 Types

C99 defines portable fixed-size types:

`[u]intXX_t` where

- `u` means unsigned,
- `XX` is the size in bits. It can range from 8 to 64, and is power of 2.

Moreover, C99 defines:

- `[u]intptr_t`, which is the size of an address, thus can contain a pointer
- `[u]int_fastXX_t`, which are **at least** `XX` bits long, but may be larger if the larger implementation is cheaper

These types are available from the standard `stdint.h` header and `hexo/types.h` header.

See [?stdint.h wikipedia page](#) for details.

## Legacy compiler types in MutekH

Using legacy integer types in MutekH's core code is not wanted.

Default settings in configuration of the build system makes the legacy types emit a warning, or even dont compile at all.

Sometimes, because you are using huge amounts of code that must be used without a complete rewrite, you may want to disable this limitation adding the following line in the build configuration file:

```
CONFIG_HEXO_INTTYPES_DEPRECATED undefined
```