Fixed width integer types (since C99)

Types

Defined in header <stdint.h>

int8_t **int16_t** (optional) **int32_t** **int64_t**	signed integer type with width of exactly 8, 16, 32 and 64 bits respectively with no padding bits and using 2's complement for negative values (provided if and only if the implementation directly supports the type) (typedef)
int_fast8_t **int_fast16_t** **int_fast32_t** **int_fast64_t**	fastest signed integer type with width of at least 8, 16, 32 and 64 bits respectively (typedef)
int_least8_t **int_least16_t** **int_least32_t** **int_least64_t**	smallest signed integer type with width of at least 8, 16, 32 and 64 bits respectively (typedef)
intmax_t	maximum-width signed integer type (typedef)
intptr_t (optional)	signed integer type capable of holding a pointer to `void` (typedef)
uint8_t **uint16_t** (optional) **uint32_t** **uint64_t**	unsigned integer type with width of exactly 8, 16, 32 and 64 bits respectively (provided if and only if the implementation directly supports the type) (typedef)
uint_fast8_t **uint_fast16_t** **uint_fast32_t** **uint_fast64_t**	fastest unsigned integer type with width of at least 8, 16, 32 and 64 bits respectively (typedef)
uint_least8_t **uint_least16_t** **uint_least32_t** **uint_least64_t**	smallest unsigned integer type with width of at least 8, 16, 32 and 64 bits respectively (typedef)
uintmax_t	maximum-width unsigned integer type (typedef)
uintptr_t (optional)	unsigned integer type capable of holding a pointer to `void` (typedef)

The implementation may define typedef names `int`*N*`_t`, `int_fast`*N*`_t`, `int_least`*N*`_t`, `uint`*N*`_t`, `uint_fast`*N*`_t`, and `uint_least`*N*`_t` when *N* is not 8, 16, 32 or 64. Typedef names of the form `int`*N*`_t` may only be defined if the implementation supports an integer type of that width with no padding. Thus, `uint24_t` denotes an unsigned integer type with a width of exactly 24 bits.

Each of the macros listed in below is defined if and only if the implementation defines the corresponding typedef name. The macros `INT`*N*`_C` and `UINT`*N*`_C` correspond to the typedef names `int_least`*N*`_t` and `uint_least`*N*`_t`, respectively.

Macro constants

Defined in header <stdint.h>

Signed integers : width

INT8_WIDTH **INT16_WIDTH** (C23)(optional) **INT32_WIDTH** **INT64_WIDTH**	bit width of an object of type `int8_t`, `int16_t`, `int32_t`, `int64_t` (exactly 8, 16, 32, 64) (macro constant)
INT_FAST8_WIDTH **INT_FAST16_WIDTH** (C23) **INT_FAST32_WIDTH** **INT_FAST64_WIDTH**	bit width of an object of type `int_fast8_t`, `int_fast16_t`, `int_fast32_t`, `int_fast64_t` (macro constant)
INT_LEAST8_WIDTH **INT_LEAST16_WIDTH** (C23) **INT_LEAST32_WIDTH** **INT_LEAST64_WIDTH**	bit width of an object of type `int_least8_t`, `int_least16_t`, `int_least32_t`, `int_least64_t` (macro constant)
INTPTR_WIDTH (C23)(optional)	bit width of an object of type `intptr_t` (macro constant)
INTMAX_WIDTH (C23)	bit width of an object of type `intmax_t` (macro constant)

Signed integers : minimum value

INT8_MIN **INT16_MIN** (optional) **INT32_MIN** **INT64_MIN**	minimum value of an object of type `int8_t`, `int16_t`, `int32_t`, `int64_t` (macro constant)
INT_FAST8_MIN **INT_FAST16_MIN** **INT_FAST32_MIN** **INT_FAST64_MIN**	minimum value of an object of type `int_fast8_t`, `int_fast16_t`, `int_fast32_t`, `int_fast64_t` (macro constant)
INT_LEAST8_MIN **INT_LEAST16_MIN** **INT_LEAST32_MIN** **INT_LEAST64_MIN**	minimum value of an object of type `int_least8_t`, `int_least16_t`, `int_least32_t`, `int_least64_t` (macro constant)
INTPTR_MIN (optional)	minimum value of an object of type `intptr_t` (macro constant)
INTMAX_MIN	minimum value of an object of type `intmax_t` (macro constant)

Signed integers : maximum value

INT8_MAX **INT16_MAX** (optional) **INT32_MAX** **INT64_MAX**	maximum value of an object of type `int8_t`, `int16_t`, `int32_t`, `int64_t` (macro constant)
INT_FAST8_MAX **INT_FAST16_MAX** **INT_FAST32_MAX** **INT_FAST64_MAX**	maximum value of an object of type `int_fast8_t`, `int_fast16_t`, `int_fast32_t`, `int_fast64_t` (macro constant)
INT_LEAST8_MAX **INT_LEAST16_MAX** **INT_LEAST32_MAX** **INT_LEAST64_MAX**	maximum value of an object of type `int_least8_t`, `int_least16_t`, `int_least32_t`, `int_least64_t` (macro constant)
INTPTR_MAX (optional)	maximum value of an object of type `intptr_t` (macro constant)
INTMAX_MAX	maximum value of an object of type `intmax_t` (macro constant)

Unsigned integers : width

UINT8_WIDTH **UINT16_WIDTH** (C23)(optional) **UINT32_WIDTH** **UINT64_WIDTH**	bit width of an object of type `uint8_t`, `uint16_t`, `uint32_t`, `uint64_t` (exactly 8, 16, 32, 64) (macro constant)
UINT_FAST8_WIDTH **UINT_FAST16_WIDTH** **UINT_FAST32_WIDTH** (C23) **UINT_FAST64_WIDTH**	bit width of an object of type `uint_fast8_t`, `uint_fast16_t`, `uint_fast32_t`, `uint_fast64_t` (macro constant)
UINT_LEAST8_WIDTH **UINT_LEAST16_WIDTH** **UINT_LEAST32_WIDTH** (C23) **UINT_LEAST64_WIDTH**	bit width of an object of type `uint_least8_t`, `uint_least16_t`, `uint_least32_t`, `uint_least64_t` (macro constant)
UINTPTR_WIDTH (C23)(optional)	bit width of an object of type `uintptr_t` (macro constant)
UINTMAX_WIDTH (C23)	bit width of an object of type `uintmax_t` (macro constant)

Unsigned integers : maximum value

UINT8_MAX **UINT16_MAX** **UINT32_MAX** (optional) **UINT64_MAX**	maximum value of an object of type `uint8_t`, `uint16_t`, `uint32_t`, `uint64_t` (macro constant)
UINT_FAST8_MAX **UINT_FAST16_MAX** **UINT_FAST32_MAX** **UINT_FAST64_MAX**	maximum value of an object of type `uint_fast8_t`, `uint_fast16_t`, `uint_fast32_t`, `uint_fast64_t` (macro constant)
UINT_LEAST8_MAX **UINT_LEAST16_MAX** **UINT_LEAST32_MAX** **UINT_LEAST64_MAX**	maximum value of an object of type `uint_least8_t`, `uint_least16_t`, `uint_least32_t`, `uint_least64_t` (macro constant)

UINTPTR_MAX (optional)	maximum value of an object of type `uintptr_t` (macro constant)
UINTMAX_MAX	maximum value of an object of type `uintmax_t` (macro constant)

Function macros for minimum-width integer constants

Defined in header `<stdint.h>`

INT8_C **INT16_C** **INT32_C** **INT64_C**	expands to an integer constant expression having the value specified by its argument and whose type is the promoted type of `int_least8_t`, `int_least16_t`, `int_least32_t`, `int_least64_t` respectively (function macro)
INTMAX_C	expands to an integer constant expression having the value specified by its argument and the type `intmax_t` (function macro)
UINT8_C **UINT16_C** **UINT32_C** **UINT64_C**	expands to an integer constant expression having the value specified by its argument and whose type is the promoted type of `uint_least8_t`, `uint_least16_t`, `uint_least32_t`, `uint_least64_t` respectively (function macro)
UINTMAX_C	expands to an integer constant expression having the value specified by its argument and the type `uintmax_t` (function macro)

```
#include <stdint.h>
UINT64_C(0x123) // might expand to 0x123ULL or 0x123UL
```

Format macro constants

Defined in header `<inttypes.h>`

Format constants for the `fprintf` family of functions

Each of the PRI macros listed here is defined if and only if the implementation defines the corresponding typedef name.

Equivalent for `int` or `unsigned int`	Description	Macros for data types				
		[u]intx_t	[u]int_leastx_t	[u]int_fastx_t	[u]intmax_t	[u]intptr_t
d	output of a signed decimal integer value	PRId**x**	PRIdLEAST**x**	PRIdFAST**x**	PRIdMAX	PRIdPTR
i		PRIi**x**	PRIiLEAST**x**	PRIiFAST**x**	PRIiMAX	PRIiPTR
u	output of an unsigned decimal integer value	PRIu**x**	PRIuLEAST**x**	PRIuFAST**x**	PRIuMAX	PRIuPTR
o	output of an unsigned octal integer value	PRIo**x**	PRIoLEAST**x**	PRIoFAST**x**	PRIoMAX	PRIoPTR
x	output of an unsigned lowercase hexadecimal integer value	PRIx**x**	PRIxLEAST**x**	PRIxFAST**x**	PRIxMAX	PRIxPTR
X	output of an unsigned uppercase hexadecimal integer value	PRIX**x**	PRIXLEAST**x**	PRIXFAST**x**	PRIXMAX	PRIXPTR

Format constants for the `fscanf` family of functions

Each of the SCN macros listed in here is defined if and only if the implementation defines the corresponding typedef name and has a suitable `fscanf` length modifier for the type.

Equivalent for `int` or `unsigned int`	Description	Macros for data types				
		[u]intx_t	[u]int_leastx_t	[u]int_fastx_t	[u]intmax_t	[u]intptr_t
d	input of a signed decimal integer value	SCNd**x**	SCNdLEAST**x**	SCNdFAST**x**	SCNdMAX	SCNdPTR
i	input of a signed integer value (base is determined by the first characters parsed)	SCNi**x**	SCNiLEAST**x**	SCNiFAST**x**	SCNiMAX	SCNiPTR
u	input of an unsigned decimal integer value	SCNu**x**	SCNuLEAST**x**	SCNuFAST**x**	SCNuMAX	SCNuPTR
o	input of an unsigned octal integer value	SCNo**x**	SCNoLEAST**x**	SCNoFAST**x**	SCNoMAX	SCNoPTR
x	input of an unsigned hexadecimal integer value	SCNx**x**	SCNxLEAST**x**	SCNxFAST**x**	SCNxMAX	SCNxPTR

Example

Run this code

```
#include <stdio.h>
#include <inttypes.h>

int main(void)
{
    printf("%zu\n", sizeof(int64_t));
    printf("%s\n", PRId64);
    printf("%+"PRId64"\n", INT64_MIN);
    printf("%+"PRId64"\n", INT64_MAX);

    int64_t n = 7;
    printf("%+"PRId64"\n", n);
}
```

Possible output:

```
8
lld
-9223372036854775808
+9223372036854775807
+7
```

References

- C17 standard (ISO/IEC 9899:2018):
  - 7.8.1 Macros for format specifiers (p: 158-159)
  - 7.18 Integer types <stdint.h> (p: 212-216)
- C11 standard (ISO/IEC 9899:2011):
  - 7.8.1 Macros for format specifiers (p: 217-218)
  - 7.18 Integer types <stdint.h> (p: 289-295)
- C99 standard (ISO/IEC 9899:1999):
  - 7.8.1 Macros for format specifiers (p: 198-199)
  - 7.18 Integer types <stdint.h> (p: 255-261)

See also

- Arithmetic types

C++ documentation for **Fixed width integer types**