```
int_least32_t   uint_least32_t
int_least64_t   uint_least64_t
```

- *Fastest minimum-width integer types.* Each name of the form `int_fastN_t` represents the fastest signed integer type with at least *N* bits. (The meaning of "fastest" is up to the implementation. If there's no reason to classify a particular type as the fastest, the implementation may choose any signed integer type with at least *N* bits.) Each name of the form `uint_fastN_t` represents the fastest unsigned integer type with *N* or more bits. `<stdint.h>` is required to provide at least the following fastest minimum-width types:

```
int_fast8_t     uint_fast8_t
int_fast16_t    uint_fast16_t
int_fast32_t    uint_fast32_t
int_fast64_t    uint_fast64_t
```

- *Integer types capable of holding object pointers.* The `intptr_t` type represents a signed integer type that can safely store any `void *` value. More precisely, if a `void *` pointer is converted to `intptr_t` type and then back to `void *`, the resulting pointer and the original pointer will compare equal. The `uintptr_t` type is an unsigned integer type with the same property as `intptr_t`. The `<stdint.h>` header isn't required to provide either type.
- *Greatest-width integer types.* `intmax_t` is a signed integer type that includes all values that belong to any signed integer type. `uintmax_t` is an unsigned integer type that includes all values that belong to any unsigned integer type. `<stdint.h>` is required to provide both types, which might be wider than `long long int`.

The names in the first three groups are declared using `typedef`.

An implementation may provide exact-width integer types, minimum-width integer types, and fastest minimum-width integer types for values of *N* in addition to the ones listed above. Also, *N* isn't required to be a power of 2 (although it will normally be a multiple of 8). For example, an implementation might provide types named `int24_t` and `uint24_t`.

Limits of Specified-Width Integer Types

For each signed integer type declared in `<stdint.h>`, the header defines macros that specify the type's minimum and maximum values. For each unsigned integer type, `<stdint.h>` defines a macro that specifies the type's maximum value. The first three rows of Table 27.1 show the values of these macros for the exact-width integer types. The remaining rows show the constraints imposed by the C99 standard on the minimum and maximum values of the other `<stdint.h>` types. (The precise values of these macros are implementation-defined.) All macros in the table represent constant expressions.