C Standard Library

`Base.Libc.malloc` — Function

```
malloc(size::Integer) -> Ptr{Cvoid}
```

Call `malloc` from the C standard library.

`Base.Libc.calloc` — Function

```
calloc(num::Integer, size::Integer) -> Ptr{Cvoid}
```

Call `calloc` from the C standard library.

`Base.Libc.realloc` — Function

```
realloc(addr::Ptr, size::Integer) -> Ptr{Cvoid}
```

Call `realloc` from the C standard library.

See warning in the documentation for `free` regarding only using this on memory originally obtained from `malloc`.

`Base.Libc.free` — Function

```
free(addr::Ptr)
```

Call `free` from the C standard library. Only use this on memory obtained from `malloc`, not on pointers retrieved from other C libraries. `Ptr` objects obtained from C libraries should be freed by the free functions defined in that library, to avoid assertion failures if multiple `libc` libraries exist on the system.

`Base.Libc.errno` — Function

```
errno([code])
```

Get the value of the C library's `errno`. If an argument is specified, it is used to set the value of `errno`.

The value of `errno` is only valid immediately after a `ccall` to a C library routine that sets it. Specifically, you cannot call `errno` at the next prompt in a REPL, because lots of code is executed between prompts.

`Base.Libc.strerror` — Function

```
strerror(n=errno())
```

Convert a system call error code to a descriptive string

`Base.Libc.GetLastError` — Function

```
GetLastError()
```

Call the Win32 `GetLastError` function [only available on Windows].

`Base.Libc.FormatMessage` — Function

```
FormatMessage(n=GetLastError())
```

Convert a Win32 system call error code to a descriptive string [only available on Windows].

`Base.Libc.time` — Method

```
time(t::TmStruct)
```

Converts a `TmStruct` struct to a number of seconds since the epoch.

`Base.Libc.strftime` — Function

```
strftime([format], time)
```

Convert time, given as a number of seconds since the epoch or a `TmStruct`, to a formatted string using the given format. Supported formats are the same as those in the standard C library.

`Base.Libc.strptime` — Function

```
strptime([format], timestr)
```

Parse a formatted time string into a `TmStruct` giving the seconds, minute, hour, date, etc. Supported formats are the same as those in the standard C library. On some platforms, timezones will not be parsed correctly. If the result of this function will be passed to `time` to convert it to seconds since the epoch, the `isdst` field should be filled in manually. Setting it to `-1` will tell the C library to use the current system settings to determine the timezone.

`Base.Libc.TmStruct` — Type

```
TmStruct([seconds])
```

Convert a number of seconds since the epoch to broken-down format, with fields `sec`, `min`, `hour`,

`mday`, `month`, `year`, `wday`, `yday`, and `isdst`.

`Base.Libc.flush_cstdio` — Function

```
flush_cstdio()
```

Flushes the C `stdout` and `stderr` streams (which may have been written to by external C code).

`Base.Libc.systemsleep` — Function

```
systemsleep(s::Real)
```

Suspends execution for `s` seconds. This function does not yield to Julia's scheduler and therefore blocks the Julia thread that it is running on for the duration of the sleep time.

See also: `sleep`

« C Interface StackTraces »