Supported platforms

System	Support type	Supported versions	Notes
GNU/Linux	Tier 1	Linux >= 2.6.32 with glibc >= 2.12	
macOS	Tier 1	macOS >= 10.7	
Windows	Tier 1	>= Windows 8	VS 2015 and later are supported
FreeBSD	Tier 1	>= 10	
AIX	Tier 2	>= 6	Maintainers: @libuv/aix
IBM i	Tier 2	>= IBM i 7.2	Maintainers: @libuv/ibmi
z/OS	Tier 2	>= V2R2	Maintainers: @libuv/zos
Linux with musl	Tier 2	musl >= 1.0	
SmartOS	Tier 3	>= 14.4	
Android	Tier 3	NDK >= r15b	
MinGW	Tier 3	MinGW32 and MinGW-w64	
SunOS	Tier 3	Solaris 121 and later	
Other	Tier 3	N/A	

Support types

- **Tier 1**: Officially supported and tested with CI. Any contributed patch MUST NOT break such systems. These are supported by @libuv/collaborators.
- **Tier 2**: Officially supported, but not necessarily tested with CI. These systems are maintained to the best of @libuv/collaborators ability, without being a top priority.
- **Tier 3**: Community maintained. These systems may inadvertently break and the community and interested parties are expected to help with the maintenance.

Adding support for a new platform

IMPORTANT: Before attempting to add support for a new platform please open an issue about it for discussion.

Unix

I/O handling is abstracted by an internal `uv__io_t` handle. The new platform will need to implement some of the functions, the prototypes are in `src/unix/internal.h`.

If the new platform requires extra fields for any handle structure, create a new include file in `include/` with the name `uv-theplatform.h` and add the appropriate defines there.

All functionality related to the new platform must be implemented in its own file inside `src/unix/` unless it's already done in a common file, in which case adding an `ifdef` is fine.

Two build systems are supported: autotools and cmake. Ideally both need to be supported, but if one of the two does not support the new platform it can be left out.

Windows

Windows is treated as a single platform, so adding support for a new platform would mean adding support for a new version.

Compilation and runtime must succeed for the minimum supported version. If a new API is to be used, it must be done optionally, only in supported versions.

Common

Some common notes when adding support for new platforms:

- Generally libuv tries to avoid compile time checks. Do not add any to the autotools based build system or use version checking macros. Dynamically load functions and symbols if they are not supported by the minimum supported version.