Platform Support

Support for different platforms ("targets") are organized into three tiers, each with a different set of guarantees. For more information on the policies for targets at each tier, see the Target Tier Policy.

Targets are identified by their "target triple" which is the string to inform the compiler what kind of output should be produced.

Tier 1 with Host Tools

Tier 1 targets can be thought of as "guaranteed to work". The Rust project builds official binary releases for each tier 1 target, and automated testing ensures that each tier 1 target builds and passes tests after each change.

Tier 1 targets with host tools additionally support running tools like `rustc` and `cargo` natively on the target, and automated testing ensures that tests pass for the host tools as well. This allows the target to be used as a development platform, not just a compilation target. For the full requirements, see Tier 1 with Host Tools in the Target Tier Policy.

All tier 1 targets with host tools support the full standard library.

target	notes
`aarch64-unknown-linux-gnu`	ARM64 Linux (kernel 4.2, glibc 2.17+) [^missing-stack-probes]
`i686-pc-windows-gnu`	32-bit MinGW (Windows 7+)
`i686-pc-windows-msvc`	32-bit MSVC (Windows 7+)
`i686-unknown-linux-gnu`	32-bit Linux (kernel 2.6.32+, glibc 2.11+)
`x86_64-apple-darwin`	64-bit macOS (10.7+, Lion+)
`x86_64-pc-windows-gnu`	64-bit MinGW (Windows 7+)
`x86_64-pc-windows-msvc`	64-bit MSVC (Windows 7+)
`x86_64-unknown-linux-gnu`	64-bit Linux (kernel 2.6.32+, glibc 2.11+)

[^missing-stack-probes]: Stack probes support is missing on `aarch64-unknown-linux-gnu`, but it's planned to be implemented in the near future. The implementation is tracked on issue #77071.

Tier 1

Tier 1 targets can be thought of as "guaranteed to work". The Rust project builds official binary releases for each tier 1 target, and automated testing ensures that each tier 1 target builds and passes tests after each change. For the full requirements, see Tier 1 target policy in the Target Tier Policy.

At this time, all Tier 1 targets are Tier 1 with Host Tools.

Tier 2 with Host Tools

Tier 2 targets can be thought of as "guaranteed to build". The Rust project builds official binary releases for each tier 2 target, and automated builds ensure that each tier 2 target builds after each change. Automated tests are not

always run so it's not guaranteed to produce a working build, but tier 2 targets often work to quite a good degree and patches are always welcome!

Tier 2 targets with host tools additionally support running tools like `rustc` and `cargo` natively on the target, and automated builds ensure that the host tools build as well. This allows the target to be used as a development platform, not just a compilation target. For the full requirements, see Tier 2 with Host Tools in the Target Tier Policy.

All tier 2 targets with host tools support the full standard library.

NOTE: The `rust-docs` component is not usually built for tier 2 targets, so Rustup may install the documentation for a similar tier 1 target instead.

target	notes
`aarch64-apple-darwin`	ARM64 macOS (11.0+, Big Sur+)
`aarch64-pc-windows-msvc`	ARM64 Windows MSVC
`aarch64-unknown-linux-musl`	ARM64 Linux with MUSL
`arm-unknown-linux-gnueabi`	ARMv6 Linux (kernel 3.2, glibc 2.17)
`arm-unknown-linux-gnueabihf`	ARMv6 Linux, hardfloat (kernel 3.2, glibc 2.17)
`armv7-unknown-linux-gnueabihf`	ARMv7 Linux, hardfloat (kernel 3.2, glibc 2.17)
`mips-unknown-linux-gnu`	MIPS Linux (kernel 4.4, glibc 2.23)
`mips64-unknown-linux-gnuabi64`	MIPS64 Linux, n64 ABI (kernel 4.4, glibc 2.23)
`mips64el-unknown-linux-gnuabi64`	MIPS64 (LE) Linux, n64 ABI (kernel 4.4, glibc 2.23)
`mipsel-unknown-linux-gnu`	MIPS (LE) Linux (kernel 4.4, glibc 2.23)
`powerpc-unknown-linux-gnu`	PowerPC Linux (kernel 2.6.32, glibc 2.11)
`powerpc64-unknown-linux-gnu`	PPC64 Linux (kernel 2.6.32, glibc 2.11)
`powerpc64le-unknown-linux-gnu`	PPC64LE Linux (kernel 3.10, glibc 2.17)
`riscv64gc-unknown-linux-gnu`	RISC-V Linux (kernel 4.20, glibc 2.29)
`s390x-unknown-linux-gnu`	S390x Linux (kernel 2.6.32, glibc 2.12)
`x86_64-unknown-freebsd`	64-bit FreeBSD
`x86_64-unknown-illumos`	illumos
`x86_64-unknown-linux-musl`	64-bit Linux with MUSL
`x86_64-unknown-netbsd`	NetBSD/amd64

Tier 2

Tier 2 targets can be thought of as "guaranteed to build". The Rust project builds official binary releases for each tier 2 target, and automated builds ensure that each tier 2 target builds after each change. Automated tests are not always run so it's not guaranteed to produce a working build, but tier 2 targets often work to quite a good degree and patches are always welcome! For the full requirements, see Tier 2 target policy in the Target Tier Policy.

The `std` column in the table below has the following meanings:

- ✓ indicates the full standard library is available.
- * indicates the target only supports `no_std` development.

NOTE: The `rust-docs` component is not usually built for tier 2 targets, so Rustup may install the documentation for a similar tier 1 target instead.

target	std	notes
`aarch64-apple-ios`	✓	ARM64 iOS
`aarch64-apple-ios-sim`	✓	Apple iOS Simulator on ARM64
`aarch64-fuchsia`	✓	ARM64 Fuchsia
`aarch64-linux-android`	✓	ARM64 Android
`aarch64-unknown-none-softfloat`	*	Bare ARM64, softfloat
`aarch64-unknown-none`	*	Bare ARM64, hardfloat
`arm-linux-androideabi`	✓	ARMv7 Android
`arm-unknown-linux-musleabi`	✓	ARMv6 Linux with MUSL
`arm-unknown-linux-musleabihf`	✓	ARMv6 Linux with MUSL, hardfloat
`armebv7r-none-eabi`	*	Bare ARMv7-R, Big Endian
`armebv7r-none-eabihf`	*	Bare ARMv7-R, Big Endian, hardfloat
`armv5te-unknown-linux-gnueabi`	✓	ARMv5TE Linux (kernel 4.4, glibc 2.23)
`armv5te-unknown-linux-musleabi`	✓	ARMv5TE Linux with MUSL
`armv7-linux-androideabi`	✓	ARMv7a Android
`armv7-unknown-linux-gnueabi`	✓	ARMv7 Linux (kernel 4.15, glibc 2.27)
`armv7-unknown-linux-musleabi`	✓	ARMv7 Linux with MUSL
`armv7-unknown-linux-musleabihf`	✓	ARMv7 Linux with MUSL, hardfloat
`armv7a-none-eabi`	*	Bare ARMv7-A
`armv7r-none-eabi`	*	Bare ARMv7-R
`armv7r-none-eabihf`	*	Bare ARMv7-R, hardfloat
`asmjs-unknown-emscripten`	✓	asm.js via Emscripten
`i586-pc-windows-msvc`	*	32-bit Windows w/o SSE
`i586-unknown-linux-gnu`	✓	32-bit Linux w/o SSE (kernel 4.4, glibc 2.23)
`i586-unknown-linux-musl`	✓	32-bit Linux w/o SSE, MUSL
`i686-linux-android`	✓	32-bit x86 Android

`i686-unknown-freebsd`	✓	32-bit FreeBSD
`i686-unknown-linux-musl`	✓	32-bit Linux with MUSL
`mips-unknown-linux-musl`	✓	MIPS Linux with MUSL
`mips64-unknown-linux-muslabi64`	✓	MIPS64 Linux, n64 ABI, MUSL
`mips64el-unknown-linux-muslabi64`	✓	MIPS64 (LE) Linux, n64 ABI, MUSL
`mipsel-unknown-linux-musl`	✓	MIPS (LE) Linux with MUSL
`nvptx64-nvidia-cuda`	*	--emit=asm generates PTX code that runs on NVIDIA GPUs
`riscv32i-unknown-none-elf`	*	Bare RISC-V (RV32I ISA)
`riscv32imac-unknown-none-elf`	*	Bare RISC-V (RV32IMAC ISA)
`riscv32imc-unknown-none-elf`	*	Bare RISC-V (RV32IMC ISA)
`riscv64gc-unknown-none-elf`	*	Bare RISC-V (RV64IMAFDC ISA)
`riscv64imac-unknown-none-elf`	*	Bare RISC-V (RV64IMAC ISA)
`sparc64-unknown-linux-gnu`	✓	SPARC Linux (kernel 4.4, glibc 2.23)
`sparcv9-sun-solaris`	✓	SPARC Solaris 10/11, illumos
`thumbv6m-none-eabi`	*	Bare Cortex-M0, M0+, M1
`thumbv7em-none-eabi`	*	Bare Cortex-M4, M7
`thumbv7em-none-eabihf`	*	Bare Cortex-M4F, M7F, FPU, hardfloat
`thumbv7m-none-eabi`	*	Bare Cortex-M3
`thumbv7neon-linux-androideabi`	✓	Thumb2-mode ARMv7a Android with NEON
`thumbv7neon-unknown-linux-gnueabihf`	✓	Thumb2-mode ARMv7a Linux with NEON (kernel 4.4, glibc 2.23)
`thumbv8m.base-none-eabi`	*	ARMv8-M Baseline
`thumbv8m.main-none-eabi`	*	ARMv8-M Mainline
`thumbv8m.main-none-eabihf`	*	ARMv8-M Mainline, hardfloat
`wasm32-unknown-emscripten`	✓	WebAssembly via Emscripten
`wasm32-unknown-unknown`	✓	WebAssembly
`wasm32-wasi`	✓	WebAssembly with WASI
`x86_64-apple-ios`	✓	64-bit x86 iOS
`x86_64-fortanix-unknown-sgx`	✓	Fortanix ABI for 64-bit Intel SGX
`x86_64-fuchsia`	✓	64-bit Fuchsia
`x86_64-linux-android`	✓	64-bit x86 Android

`x86_64-pc-solaris`	✓	64-bit Solaris 10/11, illumos
`x86_64-unknown-linux-gnux32`	✓	64-bit Linux (x32 ABI) (kernel 4.15, glibc 2.27)
`x86_64-unknown-redox`	✓	Redox OS

Tier 3

Tier 3 targets are those which the Rust codebase has support for, but which the Rust project does not build or test automatically, so they may or may not work. Official builds are not available. For the full requirements, see Tier 3 target policy in the Target Tier Policy.

The `std` column in the table below has the following meanings:

- ✓ indicates the full standard library is available.
- * indicates the target only supports `no_std` development.
- ? indicates the standard library support is unknown or a work-in-progress.

The `host` column indicates whether the codebase includes support for building host tools.

target	std	host	notes
`aarch64-apple-ios-macabi`	?		Apple Catalyst on ARM64
`aarch64-apple-tvos`	*		ARM64 tvOS
`aarch64-kmc-solid_asp3`	✓		ARM64 SOLID with TOPPERS/ASP3
`aarch64-unknown-freebsd`	✓	✓	ARM64 FreeBSD
`aarch64-unknown-hermit`	✓		ARM64 HermitCore
`aarch64-unknown-uefi`	*		ARM64 UEFI
`aarch64-unknown-linux-gnu_ilp32`	✓	✓	ARM64 Linux (ILP32 ABI)
`aarch64-unknown-netbsd`	✓	✓	
`aarch64-unknown-openbsd`	✓	✓	ARM64 OpenBSD
`aarch64-unknown-redox`	?		ARM64 Redox OS
`aarch64-uwp-windows-msvc`	?		
`aarch64-wrs-vxworks`	?		
`aarch64_be-unknown-linux-gnu_ilp32`	✓	✓	ARM64 Linux (big-endian, ILP32 ABI)
`aarch64_be-unknown-linux-gnu`	✓	✓	ARM64 Linux (big-endian)
`armv4t-unknown-linux-gnueabi`	?		
`armv5te-unknown-linux-uclibceabi`	?		ARMv5TE Linux with uClibc
`armv6-unknown-freebsd`	✓	✓	ARMv6 FreeBSD
`armv6-unknown-netbsd-eabihf`	?		
	*		ARMv6K Nintendo 3DS, Horizon (Requires

armv6k-nintendo-3ds			devkitARM toolchain)
armv7-apple-ios	✓		ARMv7 iOS, Cortex-a8
armv7-unknown-linux-uclibceabi	✓	✓	ARMv7 Linux with uClibc, softfloat
armv7-unknown-linux-uclibceabihf	✓	?	ARMv7 Linux with uClibc, hardfloat
armv7-unknown-freebsd	✓	✓	ARMv7 FreeBSD
armv7-unknown-netbsd-eabihf	✓	✓	
armv7-wrs-vxworks-eabihf	?		
armv7a-kmc-solid_asp3-eabi	✓		ARM SOLID with TOPPERS/ASP3
armv7a-kmc-solid_asp3-eabihf	✓		ARM SOLID with TOPPERS/ASP3, hardfloat
armv7a-none-eabihf	*		ARM Cortex-A, hardfloat
armv7s-apple-ios	✓		
avr-unknown-gnu-atmega328	*		AVR. Requires `-Z build-std=core`
bpfeb-unknown-none	*		BPF (big endian)
bpfel-unknown-none	*		BPF (little endian)
hexagon-unknown-linux-musl	?		
i386-apple-ios	✓		32-bit x86 iOS
i686-apple-darwin	✓	✓	32-bit macOS (10.7+, Lion+)
i686-pc-windows-msvc	*		32-bit Windows XP support
i686-unknown-haiku	✓	✓	32-bit Haiku
i686-unknown-netbsd	✓	✓	NetBSD/i386 with SSE2
i686-unknown-openbsd	✓	✓	32-bit OpenBSD
i686-unknown-uefi	*		32-bit UEFI
i686-uwp-windows-gnu	?		
i686-uwp-windows-msvc	?		
i686-wrs-vxworks	?		
m68k-unknown-linux-gnu	?		Motorola 680x0 Linux
mips-unknown-linux-uclibc	✓		MIPS Linux with uClibc
mips64-openwrt-linux-musl	?		MIPS64 for OpenWrt Linux MUSL
mipsel-sony-psp	*		MIPS (LE) Sony PlayStation Portable (PSP)
mipsel-unknown-linux-uclibc	✓		MIPS (LE) Linux with uClibc
mipsel-unknown-none	*		Bare MIPS (LE) softfloat

mipsisa32r6-unknown-linux-gnu	?		
mipsisa32r6el-unknown-linux-gnu	?		
mipsisa64r6-unknown-linux-gnuabi64	?		
mipsisa64r6el-unknown-linux-gnuabi64	?		
msp430-none-elf	*		16-bit MSP430 microcontrollers
powerpc-unknown-linux-gnuspe	✓		PowerPC SPE Linux
powerpc-unknown-linux-musl	?		
powerpc-unknown-netbsd	✓	✓	
powerpc-unknown-openbsd	?		
powerpc-wrs-vxworks-spe	?		
powerpc-wrs-vxworks	?		
powerpc64-unknown-freebsd	✓	✓	PPC64 FreeBSD (ELFv1 and ELFv2)
powerpc64le-unknown-freebsd			PPC64LE FreeBSD
powerpc-unknown-freebsd			PowerPC FreeBSD
powerpc64-unknown-linux-musl	?		
powerpc64-wrs-vxworks	?		
powerpc64le-unknown-linux-musl	?		
riscv32gc-unknown-linux-gnu			RISC-V Linux (kernel 5.4, glibc 2.33)
riscv32gc-unknown-linux-musl			RISC-V Linux (kernel 5.4, musl + RISCV32 support patches)
riscv32im-unknown-none-elf	*		Bare RISC-V (RV32IM ISA)
riscv32imc-esp-espidf	✓		RISC-V ESP-IDF
riscv64gc-unknown-freebsd			RISC-V FreeBSD
riscv64gc-unknown-linux-musl			RISC-V Linux (kernel 4.20, musl 1.2.0)
s390x-unknown-linux-musl			S390x Linux (kernel 2.6.32, MUSL)
sparc-unknown-linux-gnu	✓		32-bit SPARC Linux
sparc64-unknown-netbsd	✓	✓	NetBSD/sparc64
sparc64-unknown-openbsd	✓	✓	OpenBSD/sparc64
thumbv4t-none-eabi	*		ARMv4T T32
thumbv7a-pc-windows-msvc	?		
thumbv7a-uwp-windows-msvc	✓		

thumbv7neon-unknown-linux-musleabihf	?		Thumb2-mode ARMv7a Linux with NEON, MUSL
wasm64-unknown-unknown	?		WebAssembly
x86_64-apple-ios-macabi	✓		Apple Catalyst on x86_64
x86_64-apple-tvos	*		x86 64-bit tvOS
x86_64-pc-windows-msvc	*		64-bit Windows XP support
x86_64-sun-solaris	?		Deprecated target for 64-bit Solaris 10/11, illumos
x86_64-unknown-dragonfly	✓	✓	64-bit DragonFlyBSD
x86_64-unknown-haiku	✓	✓	64-bit Haiku
x86_64-unknown-hermit	✓		HermitCore
x86_64-unknown-l4re-uclibc	?		
x86_64-unknown-none	*		Freestanding/bare-metal x86_64, softfloat
x86_64-unknown-none-linuxkernel	*		Linux kernel modules
x86_64-unknown-openbsd	✓	✓	64-bit OpenBSD
x86_64-unknown-uefi	*		64-bit UEFI
x86_64-uwp-windows-gnu	✓		
x86_64-uwp-windows-msvc	✓		
x86_64-wrs-vxworks	?		