# User

dll created using rust far bigger in size than g++

# Bing

The size of DLLs created using Rust can be larger than those created with G++ due to a few reasons:

1. **Static Linking**: By default, Rust statically links its standard library[**1**](https://stackoverflow.com/questions/29008127/why-are-rust-executables-so-huge)[**3**](https://stackoverflow.com/questions/28652031/why-does-rust-export-its-whole-standard-library-when-building-a-dll). This means that the entire standard library is included in the DLL, even if only a small part of it is used. This can significantly increase the size of the DLL[**2**](https://stackoverflow.com/questions/27325246/why-does-the-rust-compiler-generate-huge-executables).
2. **Optimization**: The Rust compiler optimizes for execution speed, compilation speed, and ease of debugging (by including symbols), rather than minimal binary size[**1**](https://stackoverflow.com/questions/29008127/why-are-rust-executables-so-huge).

However, there are ways to reduce the size of Rust binaries:

* Use Rust 1.32.0 or newer, which doesn't include jemalloc by default[**1**](https://stackoverflow.com/questions/29008127/why-are-rust-executables-so-huge).
* Add the following to Cargo.toml[**1**](https://stackoverflow.com/questions/29008127/why-are-rust-executables-so-huge):

[profile.release]

opt-level = 'z' # Optimize for size

lto = true # Enable link-time optimization

codegen-units = 1 # Reduce number of codegen units to increase optimizations

panic = 'abort' # Abort on panic

strip = true # Strip symbols from binary

* Compile with the -C prefer-dynamic option to dynamically link the standard library[**3**](https://stackoverflow.com/questions/28652031/why-does-rust-export-its-whole-standard-library-when-building-a-dll).
* Use the cdylib crate type when creating DLLs[**2**](https://stackoverflow.com/questions/27325246/why-does-the-rust-compiler-generate-huge-executables).

Remember, these methods might not be suitable for all use cases, and you should consider the trade-offs before applying them. For example, stripping symbols from the binary can make debugging more difficult. Also, dynamic linking can reduce the size of the binary, but it requires the linked libraries to be present on the system where the binary is run[**3**](https://stackoverflow.com/questions/28652031/why-does-rust-export-its-whole-standard-library-when-building-a-dll).

Source(s)

1. [Why are Rust executables so huge? - Stack Overflow](https://stackoverflow.com/questions/29008127/why-are-rust-executables-so-huge)

2. [Why does the Rust compiler generate huge executables?](https://stackoverflow.com/questions/27325246/why-does-the-rust-compiler-generate-huge-executables)

3. [Why does Rust export its whole standard library when building a DLL?](https://stackoverflow.com/questions/28652031/why-does-rust-export-its-whole-standard-library-when-building-a-dll)

4. [Creating A DLL With Rust - Sam Rambles](https://samrambles.com/guides/window-hacking-with-rust/creating-a-dll-with-rust/index.html)

5. <http://pastebin.com/LsG1u96C>

6. <http://pastebin.com/G0AYZrpF%29>

# User

But I am using cargo to build dll. using cargo buid --release -C prefer-dynamic produces errors:

error: unexpected argument '-C' found

Usage: cargo.exe build [OPTIONS]

For more information, try '--help'.

# Bing

I see, the -C prefer-dynamic option is a rustc flag, not a cargo flag. That's why you're seeing an error when you try to use it with cargo build.

To pass this flag to rustc through cargo, you need to use the RUSTFLAGS environment variable or specify it in a .cargo/config file[**1**](https://stackoverflow.com/questions/75022896/is-there-a-way-to-use-rust-cargo-to-link-some-of-the-dependency-crates-dynamical)[**3**](http://www.elebihan.com/posts/generating-dynamically-linked-programs-with-cargo.html)[**4**](https://stackoverflow.com/questions/47441765/why-is-the-rust-libstd-linked-dynamically-without-using-c-prefer-dynamic-flag-a). Here's how you can do it:

1. **Using RUSTFLAGS**: You can set the RUSTFLAGS environment variable in your terminal before running cargo build --release. Here's how you can do it in a Unix-like shell:

RUSTFLAGS="-C prefer-dynamic" cargo build --release

1. **Using .cargo/config**: You can create a .cargo/config file in your project root and specify the rustflags there[**1**](https://stackoverflow.com/questions/75022896/is-there-a-way-to-use-rust-cargo-to-link-some-of-the-dependency-crates-dynamical)[**3**](http://www.elebihan.com/posts/generating-dynamically-linked-programs-with-cargo.html)[**4**](https://stackoverflow.com/questions/47441765/why-is-the-rust-libstd-linked-dynamically-without-using-c-prefer-dynamic-flag-a). Here's an example:

[build]

rustflags = ["-C", "prefer-dynamic"]

After setting this, you can just run cargo build --release, and cargo will pass the -C prefer-dynamic option to rustc.

Remember, -C prefer-dynamic will cause rustc to prefer dynamic linking for all dependencies if possible[**1**](https://stackoverflow.com/questions/75022896/is-there-a-way-to-use-rust-cargo-to-link-some-of-the-dependency-crates-dynamical). If you want to link only some dependencies dynamically, you might need to adjust your project structure and use crate-type = ["dylib"] in the Cargo.toml of the crates you want to link dynamically[**1**](https://stackoverflow.com/questions/75022896/is-there-a-way-to-use-rust-cargo-to-link-some-of-the-dependency-crates-dynamical).

Source(s)

1. [dll - Is there a way to use Rust-Cargo to link SOME OF THE dependency ...](https://stackoverflow.com/questions/75022896/is-there-a-way-to-use-rust-cargo-to-link-some-of-the-dependency-crates-dynamical)

2. [rust - How can I use Cargo to build a library as both an rlib and a ...](https://stackoverflow.com/questions/40514793/how-can-i-use-cargo-to-build-a-library-as-both-an-rlib-and-a-dylib-but-with-diff)

3. [Generating Dynamically Linked Programs with Cargo](http://www.elebihan.com/posts/generating-dynamically-linked-programs-with-cargo.html)

4. [Why is the Rust libstd linked dynamically without using -C prefer ...](https://stackoverflow.com/questions/47441765/why-is-the-rust-libstd-linked-dynamically-without-using-c-prefer-dynamic-flag-a)

5. [Cargo prefers dynamic linking for a dylib, if in a workspace #7226 - GitHub](https://github.com/rust-lang/cargo/issues/7226)

6. <https://github.com/rust-lang/libc.git>

# User

I am on windows and it does not have .cargo folder in my user profile. Where to include above option?

# Bing

On Windows, the .cargo directory is typically located in your user profile directory, which is represented by the %USERPROFILE% environment variable[**1**](https://stackoverflow.com/questions/46649992/where-can-i-find-cargo-config)[**2**](https://doc.rust-lang.org/cargo/reference/config.html). If the .cargo directory doesn't exist, you can create it yourself[**1**](https://stackoverflow.com/questions/46649992/where-can-i-find-cargo-config).

Here's how you can create the .cargo directory and the config file:

1. Open a command prompt.
2. Run cd %USERPROFILE% to navigate to your user profile directory.
3. Run mkdir .cargo to create the .cargo directory.
4. Run cd .cargo to navigate into the .cargo directory.
5. Run echo [build] > config to create the config file with the [build] section.
6. Run echo rustflags = ["-C", "prefer-dynamic"] >> config to add the rustflags line to the config file.

After these steps, your config file in the .cargo directory should look like this:

[build]

rustflags = ["-C", "prefer-dynamic"]

Now, when you run cargo build --release in your project directory, cargo will pass the -C prefer-dynamic option to rustc[**1**](https://stackoverflow.com/questions/46649992/where-can-i-find-cargo-config)[**2**](https://doc.rust-lang.org/cargo/reference/config.html).

Remember, this will cause rustc to prefer dynamic linking for all dependencies if possible[**1**](https://stackoverflow.com/questions/46649992/where-can-i-find-cargo-config). If you want to link only some dependencies dynamically, you might need to adjust your project structure and use crate-type = ["dylib"] in the Cargo.toml of the crates you want to link dynamically[**1**](https://stackoverflow.com/questions/46649992/where-can-i-find-cargo-config).

Source(s)

1. [rust - Where can I find .cargo/config? - Stack Overflow](https://stackoverflow.com/questions/46649992/where-can-i-find-cargo-config)

2. [Configuration - The Cargo Book - Learn Rust](https://doc.rust-lang.org/cargo/reference/config.html)

3. [GitHub - microsoft/cargo-pkgconfig: pkg-config like interface for ...](https://github.com/microsoft/cargo-pkgconfig)

# User

Can I not include the -C flag at the time of compilation with Cargo Build --Release?

# Bing

The cargo build command does not directly accept the -C flag because it's a flag for the rustc compiler, not cargo. However, you can pass this flag to rustc through cargo using the RUSTFLAGS environment variable or by specifying it in a .cargo/config file.

Here's how you can do it:

**Using RUSTFLAGS**:
In a command prompt, you can set the RUSTFLAGS environment variable before running cargo build --release. Here's how you can do it:

set RUSTFLAGS=-C prefer-dynamic

cargo build --release

**Using .cargo/config**:
You can create a .cargo/config file in your project root and specify the rustflags there. Here's an example:

[build]

rustflags = ["-C", "prefer-dynamic"]

After setting this, you can just run cargo build --release, and cargo will pass the -C prefer-dynamic option to rustc.

Remember, -C prefer-dynamic will cause rustc to prefer dynamic linking for all dependencies if possible. If you want to link only some dependencies dynamically, you might need to adjust your project structure and use crate-type = ["dylib"] in the Cargo.toml of the crates you want to link dynamically.

# User

At present I have crate-type = ["cdylib"]. Do I change that?

# Bing

The crate-type you should use depends on your specific use case[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847)[**2**](https://doc.rust-lang.org/reference/linkage.html)[**3**](https://users.rust-lang.org/t/sharing-a-crate-with-crate-type-dylib/66768)[**4**](https://users.rust-lang.org/t/why-do-i-need-to-set-the-crate-type-to-cdylib-to-build-a-wasm-binary/93247)[**5**](https://stackoverflow.com/questions/44169635/rust-cargo-dylib-clarification).

* cdylib: This crate type is primarily designed for building shared libraries that can be linked into C/C++ programs[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847). It statically links the Rust dependencies, including the standard library[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847)[**5**](https://stackoverflow.com/questions/44169635/rust-cargo-dylib-clarification). This results in larger binary sizes, but the produced dynamic library is self-contained and can be used on systems that don't have the Rust standard library installed[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847)[**5**](https://stackoverflow.com/questions/44169635/rust-cargo-dylib-clarification).
* dylib: This crate type is used for dynamic linking with other Rust projects[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847)[**5**](https://stackoverflow.com/questions/44169635/rust-cargo-dylib-clarification). It does not statically link the Rust dependencies, so the produced dynamic library is smaller[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847). However, it requires the Rust standard library and any other dependencies to be present on the system where the binary is run[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847)[**5**](https://stackoverflow.com/questions/44169635/rust-cargo-dylib-clarification).

If you're building a shared library to be used from another language (like C or C++), cdylib is usually the better choice[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847)[**3**](https://users.rust-lang.org/t/sharing-a-crate-with-crate-type-dylib/66768)[**4**](https://users.rust-lang.org/t/why-do-i-need-to-set-the-crate-type-to-cdylib-to-build-a-wasm-binary/93247)[**5**](https://stackoverflow.com/questions/44169635/rust-cargo-dylib-clarification). If you're building a shared library to be used from other Rust projects, you might want to use dylib, but be aware that it can be a headache to work with outside of that[**3**](https://users.rust-lang.org/t/sharing-a-crate-with-crate-type-dylib/66768).

Remember, changing the crate-type can have significant effects on your project, so you should carefully consider the trade-offs before making a decision[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847)[**2**](https://doc.rust-lang.org/reference/linkage.html)[**3**](https://users.rust-lang.org/t/sharing-a-crate-with-crate-type-dylib/66768)[**4**](https://users.rust-lang.org/t/why-do-i-need-to-set-the-crate-type-to-cdylib-to-build-a-wasm-binary/93247)[**5**](https://stackoverflow.com/questions/44169635/rust-cargo-dylib-clarification). If you're unsure, you might want to ask for advice on a Rust community forum or Stack Overflow[**1**](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847)[**2**](https://doc.rust-lang.org/reference/linkage.html)[**3**](https://users.rust-lang.org/t/sharing-a-crate-with-crate-type-dylib/66768)[**4**](https://users.rust-lang.org/t/why-do-i-need-to-set-the-crate-type-to-cdylib-to-build-a-wasm-binary/93247)[**5**](https://stackoverflow.com/questions/44169635/rust-cargo-dylib-clarification).

Source(s)

1. [What is the Difference Between `dylib` and `cdylib`](https://users.rust-lang.org/t/what-is-the-difference-between-dylib-and-cdylib/28847)

2. [Linkage - The Rust Reference](https://doc.rust-lang.org/reference/linkage.html)

3. [Sharing a crate with crate-type = dylib - help - The Rust Programming ...](https://users.rust-lang.org/t/sharing-a-crate-with-crate-type-dylib/66768)

4. [Why do I need to set the crate-type to cdylib to build a wasm binary?](https://users.rust-lang.org/t/why-do-i-need-to-set-the-crate-type-to-cdylib-to-build-a-wasm-binary/93247)

5. [macos - Rust cargo dylib clarification - Stack Overflow](https://stackoverflow.com/questions/44169635/rust-cargo-dylib-clarification)