Known Issues

This section informs you about known "gotchas". Keep in mind, that this section is (and always will be) incomplete. For suggestions and amendments, feel free to contribute to this guide.

Target Features

Most target-feature problems arise, when mixing code that have the target-feature *enabled* with code that have it *disabled*. If you want to avoid undefined behavior, it is recommended to build *all code* (including the standard library and imported crates) with a common set of target-features.

By default, compiling your code with the `-C target-feature` flag will not recompile the entire standard library and/or imported crates with matching target features. Therefore, target features are generally considered as unsafe. Using `#[target_feature]` on individual functions makes the function unsafe.

Examples:

Target-Feature	Issue	Seen on	Description	Details
`+soft-float` and `-sse`	Segfaults and ABI mismatches	`x86` and `x86-64`	The `x86` and `x86_64` architecture uses SSE registers (aka `xmm`) for floating point operations. Using software emulated floats ("soft-floats") disables usage of `xmm` registers, but parts of Rust's core libraries (e.g. `std::f32` or `std::f64`) are compiled without soft-floats and expect parameters to be passed in `xmm` registers. This leads to ABI mismatches. Attempting to compile with disabled SSE causes the same error, too.	#63466