

Null safety in C/C++

Adding null-pointer dereference warnings to Clang

Chad Smith · github.com/cs01/llvm-project
2026 LLVM Developers' Meeting, lightning talk

Should this warn?

```
1 int get(int *p) {  
2     return *p;  
3 }
```

- p may or may not be NULL, we don't know

Clang has nullability annotations

Spelling	Means
<code>int *_Nonnull p</code>	never null
<code>int *_Nullable p</code>	may be null: check before use
<code>int *_Null_unspecified p</code>	unknown, same as no annotation
<code>#pragma clang assume_nonnull begin ... end</code>	unannotated pointers in the region are <code>_Nonnull</code>

- In Clang since 2015
- They set clear expectations around pointer values

Now it's marked `_Nullable`. Should it warn?

```
1 int get(int *_Nullable p) {  
2     return *p;  
3 }
```

← Clang today: no warning

- The annotation says p can be NULL
- Still no warnings

Clang today checks types

```
1 void use(int *_Nonnull p);  
2 int get(int *_Nullable p) {  
3     if (p) use(p);  
4     return *p;  
5 }
```

warns, even though p was checked

silent: the actual crash

- -Wnullable-to-nonnull-conversion (opt-in) compares the two types
- It does not take into account `if (p)` control flow
- It does not warn if dereferencing a `_Nullable` value

Proposed behavior in fork

```
1 void use(int *_Nonnull p);  
2 int get(int *_Nullable p) {  
3     if (p) use(p);  
4     return *p;  
5 }
```

OK: p is non-null here

warning: nullable dereference

- `-fnullability-safety`: dataflow over the CFG
- Checks only code that opts in; Can adopt incrementally

Comparison

	This proposal	Static Analyzer	clang-tidy (Google RFC)
Technique	dataflow	symbolic execution	dataflow + SAT solver
Runs	in every compile	separate run	separate run
Extra time, Clang's Sema	+70 s	+2,533 s	+442 s

Summary

- No new language features or syntax
- Increases safety via compile-time checks
- Works like existing Clang analyses: -Wuninitialized, -Wthread-safety, Lifetime Safety
- Low overhead: 1.03x to 1.14x a full build, in the normal compile and in clangd

Try it

- Playground: cs01.github.io/llvm-project
- Code: github.com/cs01/llvm-project (branch nullability-safety)
- RFC: discourse.llvm.org/t/89042