Investigating Third-Party IDE Integration Problems

This thread has been locked by a moderator.

242

I regularly see questions from folks who've run into problems with their third-party IDE on macOS. Specifically, the issue is that their IDE is invoking Apple's command-line tools — things like `clang` and `ld` — and that's failing in some way. This post collects my ideas on how to investigate, and potentially resolve, issues like this.

If you have any questions or comments, please put them in a new thread here on DevForums. Tag it appropriately so that I see it. Good tags include *Compiler*, *Linker*, *LLVM*, and *Command Line Tools*.

Share and Enjoy
—
Quinn "The Eskimo!" @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

Investigating Third-Party IDE Integration Problems

Many third-party IDEs rely on Apple tools. For example, the IDE might run `clang` to compile C code or run `ld` to link object files. These IDEs typically don't include the tools themselves. Rather, they rely on you to install Xcode or Apple's Command Line Tools package. These are available at Apple > Developer > Downloads.

Occasionally I see folks having problems with this. They most typically report that basic stuff, like compiling a simple C program, fails with some mysterious error. If you're having such a problem, follow the steps below to investigate it.

IMPORTANT Some IDEs come with their own tools for compiling and linking. Such IDEs are not the focus of this post. If you have problems with an IDE like that, contact its vendor.

Select Your Tools

macOS has a concept of the **current command-line tools**. This can either point to the tools within Xcode or to an installed Command Line Tools package. To see which tools are currently selected, run `xcode-select` with the `--print-path` argument. This is what you'll see if you have Xcode installed in the Applications folder:

```
% xcode-select --print-path
/Applications/Xcode.app/Contents/Developer
```

Note All of the tools I discuss here are documented in man pages. If you're not familiar with those, see Reading UNIX Manual Pages.

And this is what you'll see with a Command Line Tools package selected:

```
% xcode-select --print-path
/Library/Developer/CommandLineTools
```

There are two common problems with this:

- It points to something you've deleted.
- It points to something unexpected.

Run the command above to see the current state. If necessary, change the state using the `--switch` option. For example:

```
% xcode-select --print-path
/Applications/Xcode.app/Contents/Developer
% clang -v
Apple clang version 14.0.3 (clang-1403.0.22.14.1)
…
% sudo xcode-select --switch ~/XcodeZone/Xcode-beta.app
% clang -v
Apple clang version 15.0.0 (clang-1500.0.38.1)
…
```

I have Xcode 14.3 in the Applications foledr and thus `clang` runs Clang 14.0.3. I have Xcode 15.0b5 in `~/XcodeZone`, so switching to that yields Clang 15.0.0.

It's possible to run one specific command with different tools. See *Select Your Tools Temporarily*, below.

Run a Simple Test

A good diagnostic test is to use the selected command-line tools to compile a trivial test program. Consider this C [1] example:

```
% cat hello.c
#include <stdio.h>

int main(int argc, char ** argv) {
    printf("Hello Cruel World!\n");
    return 0;
}
% clang -o hello hello.c
% ./hello
Hello Cruel World!
```

IMPORTANT If possible, run this from Terminal rather than, say, over SSH.

You may need to expand this test program to exercise your specific case. For example, if your program is hitting an error when it tries to import the Core Foundation framework, add that import to your test program:

```
% cat hello.c
#include <stdio.h>

#include <CoreFoundation/CoreFoundation.h>

int main(int argc, char ** argv) {
    printf("Hello Cruel World!\n");
    return 0;
}
```

When you compile your test program, you might see one of these results:

- Your test program compiles.
- Your test program fails with a similar error.
- Your test program fails with a different error.

I'll explore each case in turn.

[1] For a C++ example, see *C++ Issues*, below.

If your test program compiles…

If your test program compiles from the shell, that proves that your basic command-line tools setup is fine. If the same program fails to compile in your IDE, there's something IDE-specific going on here. I can't help you with that. I recommend that you escalate the issue via the support channel for your IDE.

If your test program fails with a similar error…

If your test program fails with an error similar to the one you're seeing in your IDE, there are two possibilities:

- There's a bug in your test program's code.
- There's an environmental issue that's affecting your command-line tools setup.

Don't rule out the first possibility. I regularly see folks bump into problems like this, where it turns out to be a bug in their code. For a specific example, see *C++ Issues*, below.

Assuming, however, that your test program's code is OK, it's time to investigate environmental issues. See *Vary Your Environment*, below.

If your test program fails with a different error…

If your test program fails with a different error, look at the test program's code to confirm that it's correct, and that it accurately reflects the code you're trying to run in your IDE.

Vary Your Environment

If your test program fails with the same error as you're seeing in your IDE, and you are sure that the code is correct, it's time to look for environmental factors. I typically do this with the steps described in the next sections, which are listed from most to least complex.

These steps only tell you *where* things are going wrong, not what is going wrong. However, that's often enough to continue the investigation of your issue.

Vary Your Shell

Try running your commands in a different shell. macOS's default shell is `zsh`. Try running your commands in `bash` instead:

```
% bash
…
bash-3.2$ clang -o hello hello.c
bash-3.2$ ./hello
Hello Cruel World!
```

Or if you've switched your shell to `bash`, try it in `zsh`.

Vary Your User Account

Some problems are caused by settings tied to your user account. To investigate whether that's an issue here:

1. Use System Settings > Users & Groups to create a new user.
2. Log in as that user.
3. Run your test again.

Vary Your Mac

Some problems are system wide, so you need to test on a different Mac. The easiest way to do that is to set up a virtual machine (VM) and run your test there. Or, if you have a separate physical Mac, run your test on that.

Vary Your Site

If you're working for an organisation, they may have installed software on your Mac that causes problems. If you have a Mac at home, try running your test there.

It's also possible that your network is causing problems [1]. If you have a laptop, try taking it to a different location to see if that changes things.

[1] I rarely see this when building a simple test program, but it do see it with other stuff, like code signing.

C++ Issues

If you're using C++, here's a simple test you can try:

```
% cat hello.cpp
#include <iostream>

int main()
{
    std::cout << "Hello Cruel World!\n";
}
% clang++ -o hello hello.cpp
% ./hello
Hello Cruel World!
```

A classic problem with C++ relates to name mangling. Consider this example:

```
% cat hello.c
#include <stdio.h>

#include "hello-core.h"

int main(int argc, char ** argv) {
    HCSayHello();
    return 0;
}
% cat hello-core.cpp
#include "hello-core.h"

#include <iostream>

extern void HCSayHello() {
    std::cout << "Hello Cruel World!\n";
}
% cat hello-core.h
extern void HCSayHello();
% clang -c hello.c
% clang++ -c hello-core.cpp
% clang++ -o hello hello.o hello-core.o
Undefined symbols for architecture x86_64:
  "_HCSayHello", referenced from:
      _main in hello.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

The issue here is that C++ generates a mangled name for `HCSayHello`:

```
% nm hello-core.o | grep HCSayHello
0000000000000000 T __Z10HCSayHellov
```

whereas C uses the non-mangled name:

```
% nm hello.o | grep HCSayHello
                 U _HCSayHello
```

The fix is an appropriate application of `extern "C"`:

```
% cat hello-core.h
extern "C" {

extern void HCSayHello();

}
```

Select Your Tools Temporarily

Sometimes you want to temporarily run a command from a particular tools package. To continue my earlier example, I currently have Xcode 14.3 installed in the Applications folder and Xcode 15.0b5 in `~/XcodeZone`. Xcode 14.3 is the default but I can override that with the `DEVELOPER_DIR` environment variable:

```
% clang -v
Apple clang version 14.0.3 (clang-1403.0.22.14.1)
…
% DEVELOPER_DIR=~/XcodeZone/Xcode-beta.app/Contents/Developer clang -v
Apple clang version 15.0.0 (clang-1500.0.38.1)
…
```

xcselect Compiler Linker Command Line Tools

Reply

Posted 2 months ago by eskimo

Add a Comment