

# Forum:[Hack Forums](#)

Thread: [Create a Library in Zig and use it in Python or PHP! - Part 1](#)

Infinity. - August 26, 2024, 11:14 am

Hi everyone! Recently, I've been exploring Zig and I have to say, I like it even more than Rust or C. In fact, you can even compile your C programs using Zig!

Note: This isn't an introduction to the programming language—there are better tutorials linked above for that.

In this guide, you'll learn how to create a Zig library and use it in Python or PHP. We'll start with a basic 'Hello, World!' example, and in Part 2 of this series, we'll dive into more complex Windows NT functionality.

First, let's write a simple Zig program that creates a DLL. This DLL will expose a function that either returns a string or performs some computation."

Btw, Zig competes with C instead of depending on it:

<https://ziglang.org/learn/overview/#zig-competes-with-c-instead-of-depending-on-it>

Zig site: <https://ziglang.org/>

Learn zig: <https://zig.guide/>

Step 1: Create a directory and run

Code \$ zig init-lib

Following things will be generated (note: Ignore shell.nix file, I am on nixos(<https://nixos.org/>) so my Zig installation option differs.

Our src/main.zig file looks like this

```
Code const std = @import("std");
const testing = std.testing;
export fn add(a: i32, b: i32) i32 {
    return a + b;
}
```

```
test "basic add func" {
    try testing.expect(add(3, 7) == 10);
}
```

Step 2

Build the library

Code zig build-lib -dynamic -target x86\_64-windows-gnu -O ReleaseSmall src/main.zig

Now our directory will look something like this

Now that we have our main.dll lets use it in a python environment

Code import ctypes

```
# Load the DLL
mylib = ctypes.CDLL('./main.dll')

# Call the add function
result = mylib.add(5, 3)
print(f"Result of add: {result}")
PHP
```

Code <?php

```
// Define the FFI interface
$ffi = FFI::cdef("
    int add(int, int);
", "./main.dll");

// Call the add function
$result = $ffi->add(5, 3);
echo "Result of add: $result\n";

?>
```

**Klonopin** - August 26, 2024, 7:08 pm

This is very interesting, makes me want to learn zig and give it a try.  
This has me thinking of all the possibilities with this.

Cool little write up man

**DeltaNox** - August 27, 2024, 5:52 am

That's interesting. I hadn't heard of zig before. I'll definitely look into it.

**Infinity.** - August 27, 2024, 10:12 pm

(Aug 26, 2024 07:08 PM)Klonopin Wrote: [\[CITE\]](#) This is very interesting, makes me want to learn zig and give it a try.  
This has me thinking of all the possibilities with this.

Cool little write up man

Thanks man, hoping to create more tutorials around Zig.

**HFDone** - August 28, 2024, 10:12 am

didnt understand why using it is better than using the C libraries

**Infinity.** - August 28, 2024, 9:26 pm

(Aug 28, 2024 10:12 AM)HFDone Wrote: [\[CITE\]](#) didnt understand why using it is better than using the C libraries  
Yes there's references mentioned but here's some point.

This Zig code is significantly simpler than the equivalent C code, as well as having more safety protections, and all this is accomplished by directly importing the C header file - no API bindings. Zig is better at using C libraries than C is at using C libraries.

Zig uses undefined behavior as a razor sharp tool for both bug prevention and performance enhancement. Speaking of performance, Zig is faster than C. The reference implementation uses LLVM as a backend for state of the art optimizations. What other projects call "Link Time Optimization" Zig does automatically.

Above is taken from their site.

For more: <https://ziglang.org/learn/overview/>

[https://www.reddit.com/r/Zig/comments/17le1rs/is\\_zig\\_actually\\_lower\\_level\\_compared\\_to\\_c\\_and\\_is/](https://www.reddit.com/r/Zig/comments/17le1rs/is_zig_actually_lower_level_compared_to_c_and_is/)

**HFDone** - August 29, 2024, 11:17 am

so its sound as zig is a better compiler for C

**Infinity.** - October 4, 2024, 6:18 am

(Aug 29, 2024 11:17 AM)HFDone Wrote: [\[CITE\]](#) so its sound as zig is a better compiler for C

Hi, I think you're missing major things here brother. Have you checked the resources I mentioned?

Zig is more than just a "better compiler for C." While it is designed to be compatible with C (you can use C libraries and even compile C code directly with Zig), but it's more than that.

Memory Safety: Zig has safety features like optional pointers which are safer than raw pointers and undefined behavior detection, while still giving you the control that C provides.

Error Handling: Instead of using error codes like in C, Zig has a built-in error handling system similar to Rust's Result type but more lightweight.

In C, if you want to compile code for different platforms you need to configure multiple compilers in the toolchain, Zig provides easy cross-compilation feature.

C has hidden control flow mechanism like implicit type casting or function calls in macros :(. Zig has no such thing.

Far better "STD" library than C. (you can google how)

C is known for performance, yes, but no language other than Zig has matched or even exceeded C's performance in certain cases.

You can easily integrate Zig in existing C codebase.

**HFDone** - October 5, 2024, 9:47 am

i got it from your first reply by saying "better compiler for C" im reffering to all above too

Ok i Add it to my multi compiler

(Aug 26, 2024 11:14 AM)Infinity. Wrote: [\[CITE\]](#)

Hi everyone! Recently, I've been exploring Zig and I have to say, I like it even more than Rust or C. In fact, you can even compile your C programs using Zig!

Note: This isn't an introduction to the programming language—there are better tutorials linked above for that.

In this guide, you'll learn how to create a Zig library and use it in Python or PHP. We'll start with a basic 'Hello, World!' example, and in Part 2 of this series, we'll dive into more complex Windows NT functionality.

First, let's write a simple Zig program that creates a DLL. This DLL will expose a function that either returns a string or performs some computation."

Btw, Zig competes with C instead of depending on it:

<https://ziglang.org/learn/overview/#zig-competes-with-c-instead-of-depending-on-it>

Zig site: <https://ziglang.org/>

Learn zig: <https://zig.guide/>

Step 1: Create a directory and run

Code \$ zig init-lib

Following things will be generated (note: Ignore shell.nix file, I am on nixos(<https://nixos.org/>) so my Zig installation option differs.

Our src/main.zig file looks like this

```
Code const std = @import("std");
const testing = std.testing;
export fn add(a: i32, b: i32) i32 {
    return a + b;
}
```

```
test "basic add func" {
    try testing.expect(add(3, 7) == 10);
}
```

Step 2

Build the library

Code zig build-lib -dynamic -target x86\_64-windows-gnu -O ReleaseSmall src/main.zig  
Now our directory will look something like this

Now that we have our main.dll lets use it in a python environment

Code import ctypes

```
# Load the DLL
mylib = ctypes.CDLL('./main.dll')
```

```
# Call the add function
result = mylib.add(5, 3)
print(f"Result of add: {result}")
PHP
```

Code <?php

```
// Define the FFI interface
$ffi = FFI::cdef("
    int add(int, int);
", "./main.dll");
```

```
// Call the add function
$result = $ffi->add(5, 3);
echo "Result of add: $result\n";
```

?>

Huh looks cool. My understanding of this is that its like C with the way you write it. Handy you dont really need to learn a whole new lang.