That Time I Tried Porting Zig to SerenityOS

sin-ack

Software You Can Love 2022
`https://sycl.it/`

2022-10-07 Fri

Outline

Motivation

- Why port Zig to SerenityOS?
  - It's fun!
  - A nice break away from own projects
  - Wanted to see ZigSelf run on SerenityOS

SerenityOS (1/2)

- Hobby operating system, developed by Andreas Kling since 2018
- Grew to over 700 contributors
- *Everything* is from scratch - kernel, GUI, coreutils, browser
- The whole OS is a monorepo

SerenityOS (2/2)

- I started getting involved in April 2021
- Started contributing various fixes soon after
- Not much of an active contributor anymore, but it's still a project I really enjoy

So...

- In February 2022, I wanted to see ZigSelf run on SerenityOS
- There is no Zig target for SerenityOS!

So...

Humble Beginnings (1/3)

- What do you need for Zig to target something?
  - Just write a target and compile to it
  - *Boring!*
- Let's make *Zig* run on SerenityOS
  - Helps improve POSIX compatibility of SerenityOS, helps find bugs, etc.

Humble Beginnings (2/3)

- Zig currently needs LLVM to build stuff
    - If you want Zig on a target, you need LLVM on it first
    - Self-hosted backends are on the way, but still not ready yet
        - + Not even close to usable back in February
- Thankfully, SerenityOS already has LLVM!
    - Just need to handle the Zig bits

Humble Beginnings (3/3)

- What else do we need for Zig on a target?
    - Zig compiler
    - Wait what?
- Chicken and egg situation
    - zig-bootstrap solves this

Making `zig cc` work (1/2)

- Need to verify Zig can produce valid SerenityOS binaries
  - Don't want to write a bunch of code that would never work
- Let's compile a simple C program for SerenityOS via `zig cc`
  - And then try to build stage1 with that

Making `zig cc` work (2/2)

- ☐ Host Zig
  - ☐ Host LLVM
- ☐ Target Zig

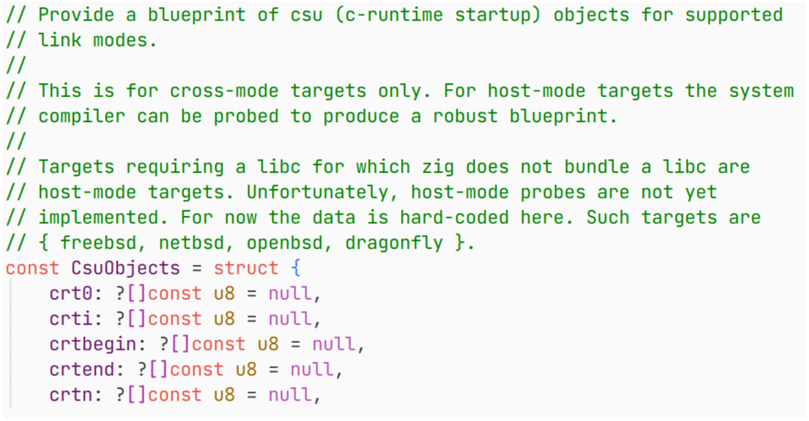
Patching non-monorepo LLVM

- Serenity patches expect monorepo
- `zig-bootstrap` is not the LLVM monorepo
- Daniel split up the patches, thanks Daniel!

Synchronizing Zig with LLVM

- Zig wants its target enums to match LLVM's perfectly
  - Static assertions in C++ code
  - Compiler guides you on where the new value isn't handled
- If you add them there, you have to add them to the Zig side too
  - `serenity` has become an official Zig OS target at this point

A maze of CRT objects, all different (1/2)



```
// Provide a blueprint of csu (c-runtime startup) objects for supported
// link modes.
//
// This is for cross-mode targets only. For host-mode targets the system
// compiler can be probed to produce a robust blueprint.
//
// Targets requiring a libc for which zig does not bundle a libc are
// host-mode targets. Unfortunately, host-mode probes are not yet
// implemented. For now the data is hard-coded here. Such targets are
// { freebsd, netbsd, openbsd, dragonfly }.
const CsuObjects = struct {
    crt0: ?[]const u8 = null,
    crti: ?[]const u8 = null,
    crtbegin: ?[]const u8 = null,
    crtend: ?[]const u8 = null,
    crtn: ?[]const u8 = null,
```

A maze of CRT objects, all different (1/2)

- You are not expected to understand this.
- The Clang driver for SerenityOS figures out some of it
- Just looked at what we have and put in what makes most sense

The Ununited State of LibC (1/3)

- ☒ Host Zig
  - ☒ Host LLVM
- ☐ Target Zig

The Ununited State of LibC (2/3)

- Building a C binary against SerenityOS = knowing which symbols it needs
- At the time, libc was being statically linked
    - Trial and error until the right combination is found
    - `-lpthread -ldl -lm -lunwind`

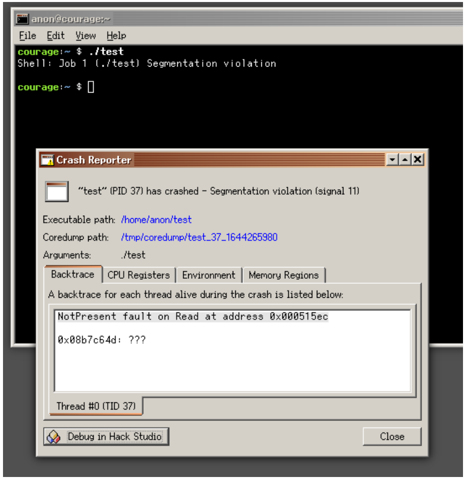
The Ununited State of LibC (3/3)

- And `-lc++` too!
    - `LibC` needs C++ RTTI!
    - Unfortunate, but doesn't seem to create much of a problem
- This has mostly been fixed thanks to Tim Schumacher's work
    - Thanks Tim!

So it built... (1/7)

- We finally have a C binary targeted to SerenityOS!
  - Let's try to run it

So it built... (2/7)



```
courage:~ $ ./test
Shell: Job 1 (./test) Segmentation violation

courage:~ $
```

"test" (PID 37) has crashed - Segmentation violation (signal 11)

Executable path: /home/anon/test
Coredump path: /tmp/coredump/test_37_1644265980
Arguments: ./test

A backtrace for each thread alive during the crash is listed below:

```
NotPresent fault on Read at address 0x000515ec

0x08b7c64d: ???
```

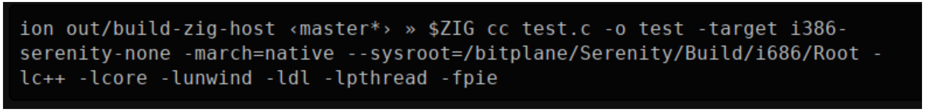
So it built... (3/7)

- The dynamic loader needs to relocate stuff for you
- On other OSes, libc does this for you
    - Serenity chose to put it in DynamicLoader
- We don't support edgy static binaries yet

So it built... (4/7)

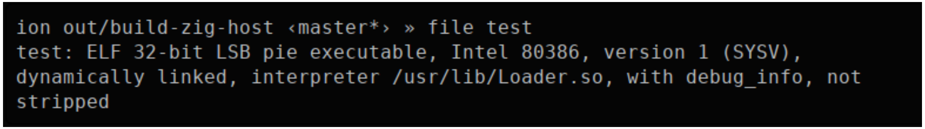
- How to make Zig output a dynamic binary?
  - `-lcore` lol

So it built... (5/7)



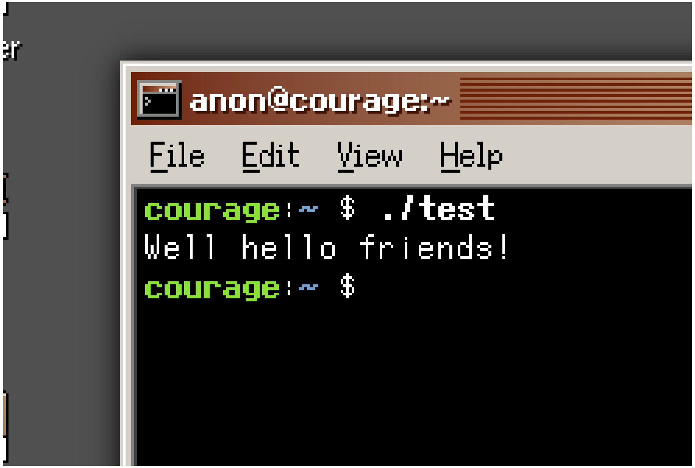
```
ion out/build-zig-host ‹master*› » $ZIG cc test.c -o test -target i386-
serenity-none -march=native --sysroot=/bitplane/Serenity/Build/i686/Root -
lc++ -lcore -lunwind -ldl -lpthread -fpie
```

So it built... (6/7)



```
ion out/build-zig-host ‹master*› » file test
test: ELF 32-bit LSB pie executable, Intel 80386, version 1 (SYSV),
dynamically linked, interpreter /usr/lib/Loader.so, with debug_info, not
stripped
```

So it built... (7/7)



```
courage:~ $ ./test
Well hello friends!
courage:~ $
```

The Big Guns™ (1/2)

- Zig can build a SerenityOS binary
    - But we haven't built any Zig code, just C code via Zig
- For Zig code to run, it needs to be able to talk to the OS
    - Two approaches:
        - Use direct syscalls (Linux)
        - Go through libc (Everything else?)

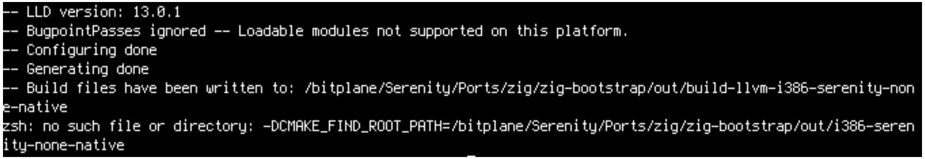
The Big Guns™ (2/2)

- ☒ Host Zig
  - ☒ Host LLVM
- ☐ Target Zig
  - ☐ Target Zlib
  - ☐ Target LLVM

Building Zlib and LLVM (1/3)

- zlib worked, just needed a hack to add -fPIC
    - Since fixed on the Serenity side
- Let's build LLVM for Serenity
    - Can't find compress2 from zlib... 🤔
    - Let's add `CMAKE_FIND_ROOT_PATH`
        - Still doesn't work?

Building Zlib and LLVM (2/3)



```
-- LLD version: 13.0.1
-- BugpointPasses ignored -- Loadable modules not supported on this platform.
-- Configuring done
-- Generating done
-- Build files have been written to: /bitplane/Serenity/Ports/zig/zig-bootstrap/out/build-llvm-i386-serenity-non
e-native
zsh: no such file or directory: -DCMAKE_FIND_ROOT_PATH=/bitplane/Serenity/Ports/zig/zig-bootstrap/out/i386-seren
ity-none-native
```

Building Zlib and LLVM (3/3)

- Can't find dlopen and dladdr
    - Throwback to the libc weirdness from earlier
- But beyond that, seems to compile fine with zig cc!

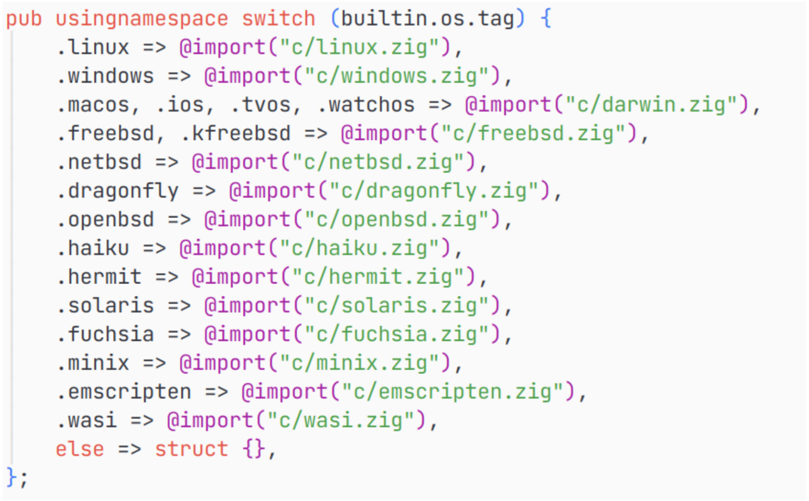
The Holy Yak (1/6)

- ☒ Host Zig
  - ☒ Host LLVM
- ☐ Target Zig
  - ☒ Target Zlib
  - ☒ Target LLVM

The Holy Yak (2/6)

- What does it take to add a new OS to `std`?
    - SerenityOS' “stable” syscall interface is libc
- Need to add it to the target-specific switch in `os.zig`

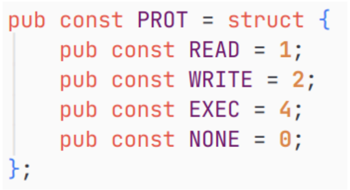
The Holy Yak (3/6)



```zig
pub usingnamespace switch (builtin.os.tag) {
    .linux => @import("c/linux.zig"),
    .windows => @import("c/windows.zig"),
    .macos, .ios, .tvos, .watchos => @import("c/darwin.zig"),
    .freebsd, .kfreebsd => @import("c/freebsd.zig"),
    .netbsd => @import("c/netbsd.zig"),
    .dragonfly => @import("c/dragonfly.zig"),
    .openbsd => @import("c/openbsd.zig"),
    .haiku => @import("c/haiku.zig"),
    .hermit => @import("c/hermit.zig"),
    .solaris => @import("c/solaris.zig"),
    .fuchsia => @import("c/fuchsia.zig"),
    .minix => @import("c/minix.zig"),
    .emscripten => @import("c/emscripten.zig"),
    .wasi => @import("c/wasi.zig"),
    else => struct {},
};
```

The Holy Yak (4/6)

- For the most part, this is just POSIX constants
  - Grouped by the prefix
  - `errno` numbers are an enum though
- Need a couple function/struct definitions
- A few functions to support various OS features



```zig
pub const PROT = struct {
    pub const READ = 1;
    pub const WRITE = 2;
    pub const EXEC = 4;
    pub const NONE = 0;
};
```

The Holy Yak (5/6)

- `errno` strikes back
  - Global value that `libc` functions write to
  - Threads? What are those?
- On most OSes `errno` is fake!
  - It's actually a macro that takes the value from `__errno_location`
- SerenityOS had thread-local `errno`, but it was still public
  - Caused bizarre 0-sized TLS symbol
  - Just patch the check out, lol works
  - Since fixed by doing it like the other OSes

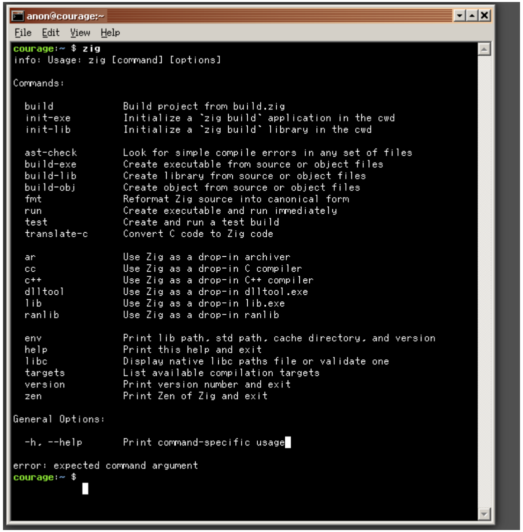
The Holy Yak (6/6)

- Did all that, implemented a directory iterator, etc.
- Semantic Analysis successful! 🤩
- Link not successful! 😢

“Implementing” POSIX support

- Link errors, but this time stuff we’re actually missing
    - Let’s “implement” them by adding stubs to `LibC`
    - What could go wrong?
- Fixing this and a couple more errors and...

zig help On SerenityOS

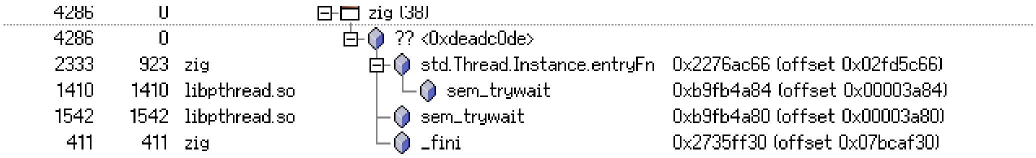


```
courage:~ $ zig
info: Usage: zig [command] [options]

Commands:

  build            Build project from build.zig
  init-exe         Initialize a `zig build` application in the cwd
  init-lib         Initialize a `zig build` library in the cwd

  ast-check        Look for simple compile errors in any set of files
  build-exe        Create executable from source or object files
  build-lib        Create library from source or object files
  build-obj        Create object from source or object files
  fmt              Reformat Zig source into canonical form
  run              Create executable and run immediately
  test             Create and run a test build
  translate-c      Convert C code to Zig code

  ar               Use Zig as a drop-in archiver
  cc               Use Zig as a drop-in C compiler
  c++              Use Zig as a drop-in C++ compiler
  dlltool          Use Zig as a drop-in dlltool.exe
  lib              Use Zig as a drop-in lib.exe
  ranlib           Use Zig as a drop-in ranlib

  env              Print lib path, std path, cache directory, and version
  help             Print this help and exit
  libc             Display native libc paths file or validate one
  targets          List available compilation targets
  version          Print version number and exit
  zen              Print Zen of Zig and exit

General Options:

  -h, --help       Print command-specific usage

error: expected command argument
courage:~ $
```

Oops, non-working semaphores (1/3)

- Waiting on `zig build-exe`
  - Uh, waiting for a while..?
  - Let's profile it

Oops, non-working semaphores (2/3)



```
4286     0                      zig (38)
4286     0                        ?? <0xdeadc0de>
2333   923  zig                     std.Thread.Instance.entryFn  0x2276ac66 (offset 0x02fd5c66)
1410  1410  libpthread.so             sem_trywait                0xb9fb4a84 (offset 0x00003a84)
1542  1542  libpthread.so           sem_trywait                  0xb9fb4a80 (offset 0x00003a80)
 411   411  zig                     _fini                        0x2735ff30 (offset 0x07bcaf30)
```

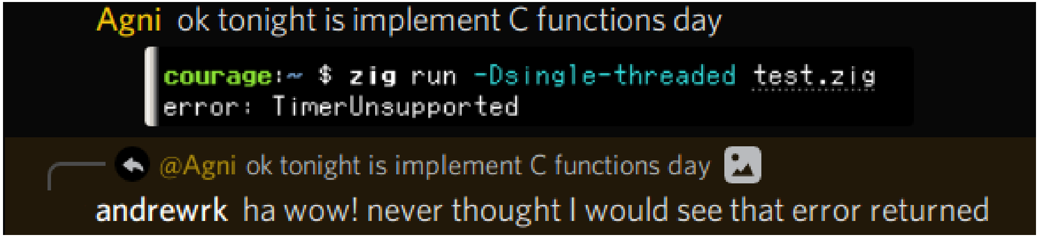
Oops, non-working semaphores (3/3)

- Okay, semaphores don't work on Serenity
  - Andrew suggested rebuilding Zig with `-fsingle-threaded`
  - No longer stuck there

Actually implementing POSIX support (1/3)

- Turns out the functions Zig links to are needed by Zig (who could've guessed?)
    - Most of them are stuff like `symlinkat`, `unlinkat`, etc. so rather quick to implement

Actually implementing POSIX support (2/3)



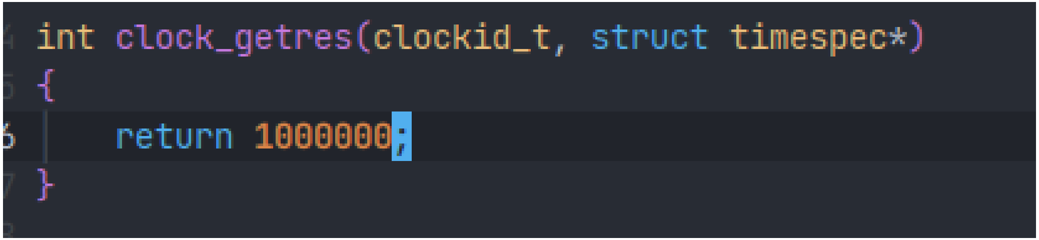
Agni ok tonight is implement C functions day

```
courage:~ $ zig run -Dsingle-threaded test.zig
error: TimerUnsupported
```

@Agni ok tonight is implement C functions day

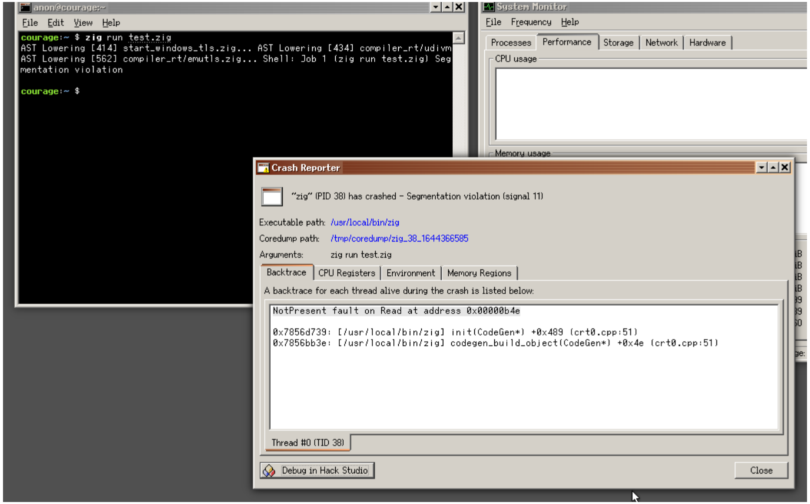
andrewrk ha wow! never thought I would see that error returned

Actually implementing POSIX support (3/3)



```
int clock_getres(clockid_t, struct timespec*)
{
    return 1000000;
}
```

Hmm... weird crash (1/7)



```
courage:~ $ zig run test.zig
AST Lowering [414] start_windows_tls.zig... AST Lowering [434] compiler_rt/udivm
AST Lowering [562] compiler_rt/emutls.zig... Shell: Job 1 (zig run test.zig) Seg
mentation violation

courage:~ $
```

"zig" (PID 38) has crashed - Segmentation violation (signal 11)

Executable path: /usr/local/bin/zig
Coredump path: /tmp/coredump/zig_38_1644366585
Arguments: zig run test.zig

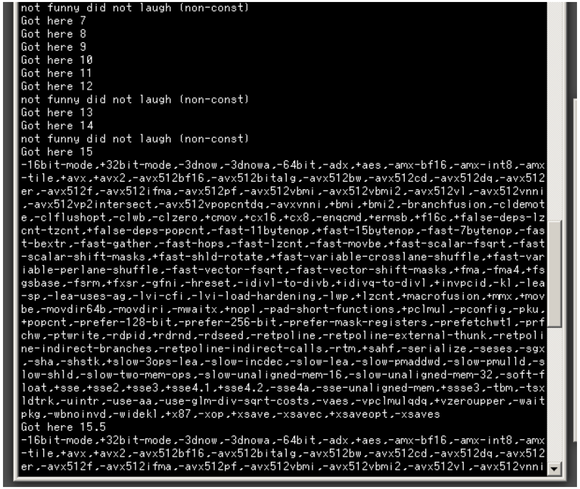
A backtrace for each thread alive during the crash is listed below:

```
NotPresent fault on Read at address 0x00000b4e

0x7856d739: [/usr/local/bin/zig] init(CodeGen*) +0x489 (crt0.cpp:51)
0x7856bb3e: [/usr/local/bin/zig] codegen_build_object(CodeGen*) +0x4e (crt0.cpp:51)
```

Hmm... weird crash (2/6)

- If you change the location or arguments of the prints, the error changes location!
- Had to continue the next day
    - Just keep inserting debug prints until something happens

Hmm... weird crash (3/6)

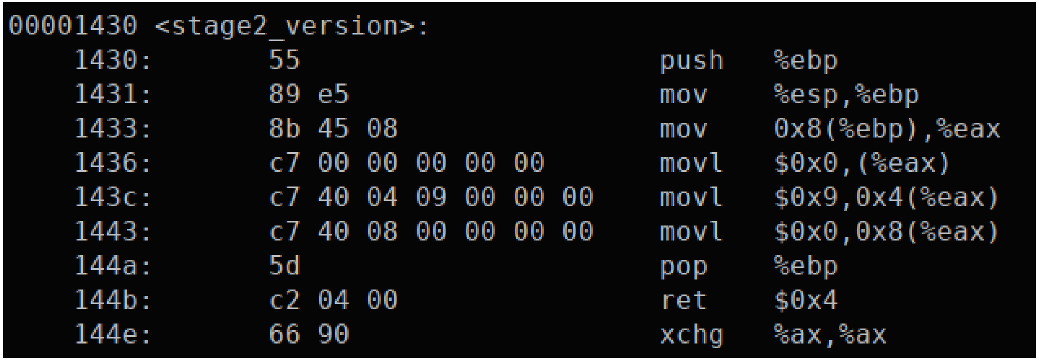


```
not funny did not laugh (non-const)
Got here 7
Got here 8
Got here 9
Got here 10
Got here 11
Got here 12
not funny did not laugh (non-const)
Got here 13
Got here 14
not funny did not laugh (non-const)
Got here 15
-16bit-mode,+32bit-mode,-3dnow,-3dnowa,-64bit,-adx,+aes,-amx-bf16,-amx-int8,-amx
-tile,+avx,+avx2,-avx512bf16,-avx512bitalg,-avx512bw,-avx512cd,-avx512dq,-avx512
er,-avx512f,-avx512ifma,-avx512pf,-avx512vbmi,-avx512vbmi2,-avx512vl,-avx512vnni
,-avx512vp2intersect,-avx512vpopcntdq,-avxvnni,+bmi,+bmi2,-branchfusion,-cldemot
e,-clflushopt,-clwb,-clzero,+cmov,+cx16,+cx8,-enqcmd,+ermsb,+f16c,+false-deps-lz
cnt-tzcnt,+false-deps-popcnt,-fast-11bytenop,+fast-15bytenop,-fast-7bytenop,-fas
t-bextr,-fast-gather,-fast-hops,-fast-lzcnt,-fast-movbe,+fast-scalar-fsqrt,-fast
-scalar-shift-masks,+fast-shld-rotate,+fast-variable-crosslane-shuffle,+fast-var
iable-perlane-shuffle,-fast-vector-fsqrt,-fast-vector-shift-masks,+fma,-fma4,+fs
gsbase,-fsrm,+fxsr,-gfni,-hreset,-idivl-to-divb,+idivq-to-divl,+invpcid,-kl,-lea
-sp,-lea-uses-ag,-lvi-cfi,-lvi-load-hardening,-lwp,+lzcnt,+macrofusion,+mmx,+mov
be,-movdir64b,-movdiri,-mwaitx,+nopl,-pad-short-functions,+pclmul,-pconfig,-pku,
+popcnt,-prefer-128-bit,-prefer-256-bit,-prefer-mask-registers,-prefetchwt1,-prf
chw,-ptwrite,-rdpid,+rdrnd,-rdseed,-retpoline,-retpoline-external-thunk,-retpoli
ne-indirect-branches,-retpoline-indirect-calls,-rtm,+sahf,-serialize,-seses,-sgx
,-sha,-shstk,+slow-3ops-lea,-slow-incdec,-slow-lea,-slow-pmaddwd,-slow-pmulld,-s
low-shld,-slow-two-mem-ops,-slow-unaligned-mem-16,-slow-unaligned-mem-32,-soft-f
loat,+sse,+sse2,+sse3,+sse4.1,+sse4.2,-sse4a,-sse-unaligned-mem,+ssse3,-tbm,-tsx
ldtrk,-uintr,-use-aa,-use-glm-div-sqrt-costs,-vaes,-vpclmulqdq,+vzeroupper,-wait
pkg,-wbnoinvd,-widekl,+x87,-xop,+xsave,-xsavec,+xsaveopt,-xsaves
Got here 15.5
-16bit-mode,+32bit-mode,-3dnow,-3dnowa,-64bit,-adx,+aes,-amx-bf16,-amx-int8,-amx
-tile,+avx,+avx2,-avx512bf16,-avx512bitalg,-avx512bw,-avx512cd,-avx512dq,-avx512
er,-avx512f,-avx512ifma,-avx512pf,-avx512vbmi,-avx512vbmi2,-avx512vl,-avx512vnni
```

Hmm... weird crash (4/6)

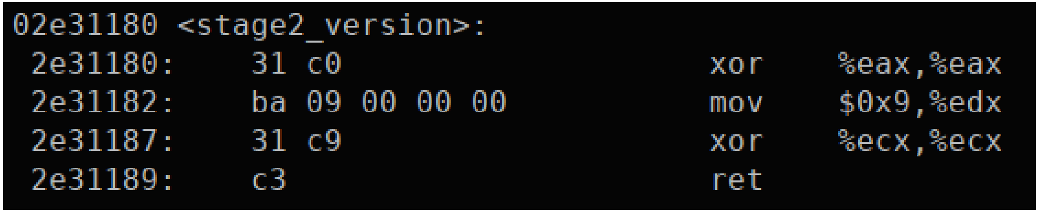
- We start getting weirdness right after `stage2_version`
  - What's special about it?
  - It's a Zig function being called from C++ code!
- What happens if Zig and C++ ABIs don't match?
  - No nice compile errors, only 💥

Hmm... weird crash (5/6)



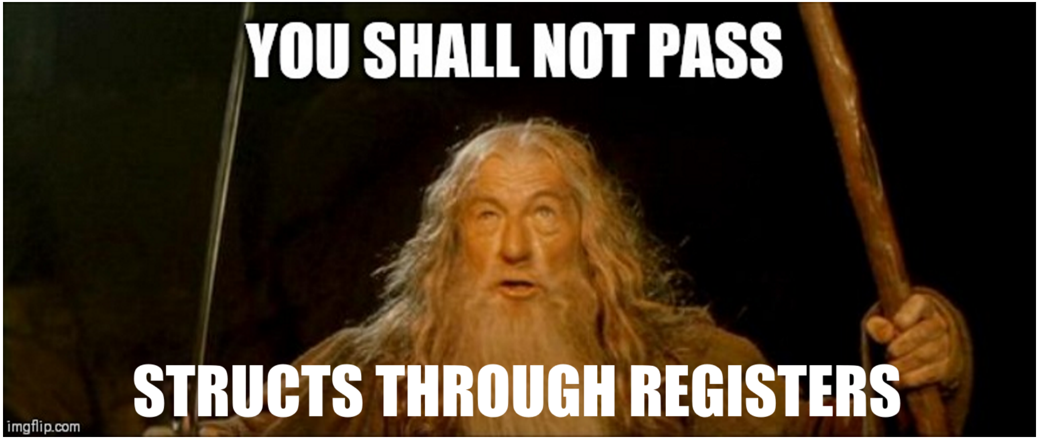
```
00001430 <stage2_version>:
    1430:       55                      push   %ebp
    1431:       89 e5                   mov    %esp,%ebp
    1433:       8b 45 08                mov    0x8(%ebp),%eax
    1436:       c7 00 00 00 00 00       movl   $0x0,(%eax)
    143c:       c7 40 04 09 00 00 00    movl   $0x9,0x4(%eax)
    1443:       c7 40 08 00 00 00 00    movl   $0x0,0x8(%eax)
    144a:       5d                      pop    %ebp
    144b:       c2 04 00                ret    $0x4
    144e:       66 90                   xchg   %ax,%ax
```

Hmm... weird crash (6/6)



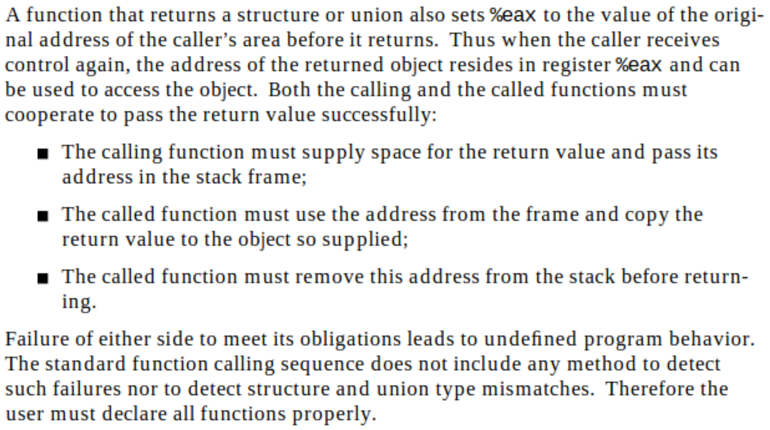
```
02e31180 <stage2_version>:
 2e31180:      31 c0                                  xor      %eax,%eax
 2e31182:      ba 09 00 00 00                         mov      $0x9,%edx
 2e31187:      31 c9                                  xor      %ecx,%ecx
 2e31189:      c3                                     ret
```

THOU SHALT NOT PASS (1/3)



Credit: kleines Filmröllchen

THOU SHALT NOT PASS (2/3)



A function that returns a structure or union also sets %eax to the value of the original address of the caller's area before it returns. Thus when the caller receives control again, the address of the returned object resides in register %eax and can be used to access the object. Both the calling and the called functions must cooperate to pass the return value successfully:

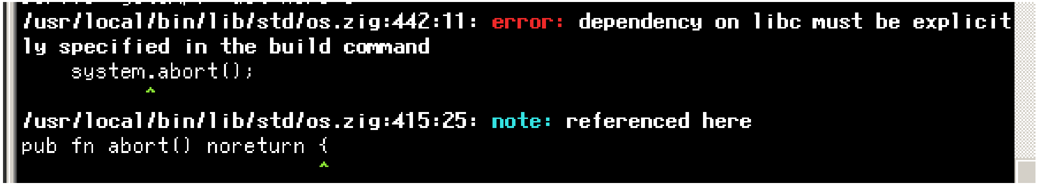
- The calling function must supply space for the return value and pass its address in the stack frame;
- The called function must use the address from the frame and copy the return value to the object so supplied;
- The called function must remove this address from the stack before returning.

Failure of either side to meet its obligations leads to undefined program behavior. The standard function calling sequence does not include any method to detect such failures nor to detect structure and union type mismatches. Therefore the user must declare all functions properly.

THOU SHALT NOT PASS (3/3)

- My terrible “fix” was inserting a check inside the `x86_64` codegen
  - Thankfully since fixed on Zig upstream 😅

Zig compiler actually doing stuff on SerenityOS!

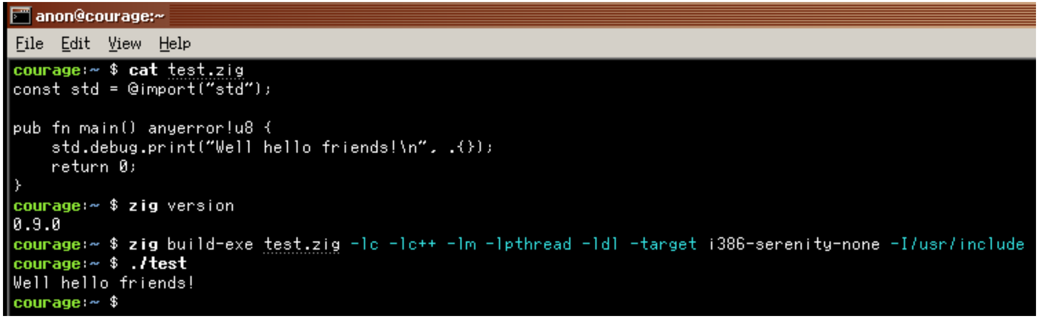


```
/usr/local/bin/lib/std/os.zig:442:11: error: dependency on libc must be explicit
ly specified in the build command
    system.abort();
          ^
/usr/local/bin/lib/std/os.zig:415:25: note: referenced here
pub fn abort() noreturn {
                        ^
```

Remaining fixes and LibC stuff

- Just need to fix compiler errors until it works now
    - Add more missing POSIX functions
    - Add in stack trace support
    - flock broken? Just make it `return 0` lol
    - Need to tell Zig about the `libc` installation

Zig code built and running on SerenityOS

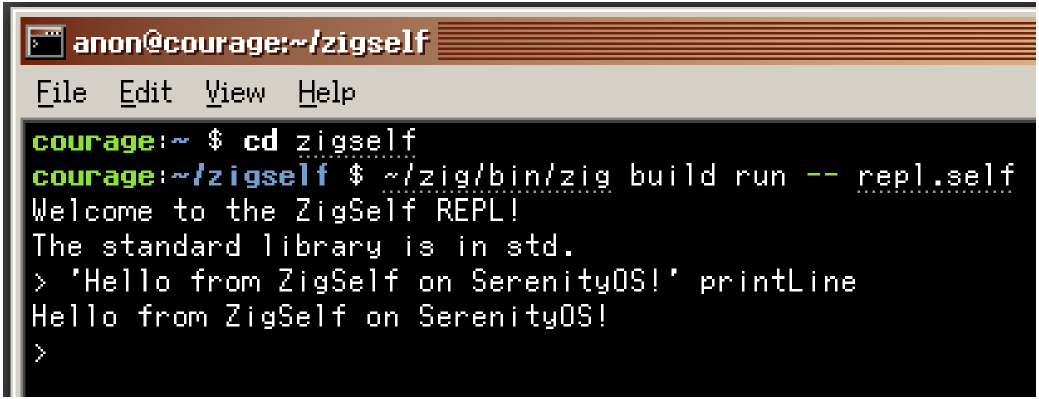


```
courage:~ $ cat test.zig
const std = @import("std");

pub fn main() anyerror!u8 {
    std.debug.print("Well hello friends!\n", .{});
    return 0;
}
courage:~ $ zig version
0.9.0
courage:~ $ zig build-exe test.zig -lc -lc++ -lm -lpthread -ldl -target i386-serenity-none -I/usr/include
courage:~ $ ./test
Well hello friends!
courage:~ $
```

Just a couple more hacks... (1/2)

- Let's see if I can get ZigSelf working too...
  - Actually helped me find some portability issues
  - And yes, it's running

Just a couple more hacks... (2/2)



```
courage:~ $ cd zigself
courage:~/zigself $ ~/zig/bin/zig build run -- repl.self
Welcome to the ZigSelf REPL!
The standard library is in std.
> 'Hello from ZigSelf on SerenityOS!' printLine
Hello from ZigSelf on SerenityOS!
>
```

Aftermath

- This work was not upstreamed
  - stage2 was "very close"
  - Too many hacks on the SerenityOS side
- I had always intended to port again once stage2 was official

Now for something different...

Now for something different...

Future work

- What's next? Upstreaming!
  - Need to upstream the LLVM patches
  - Find a solution to Serenity's `libc` instability
    - `generate-serenity-constants.zig`
  - Upstream Zig target!
  - Possible to pop up a `LibGUI` window via Zig? 🤔

Closing thoughts

- My work was nowhere close to the scale of both projects
  - But it wasn't done since nobody tried it
- Any task is achievable, just gotta yakshave!

Thanks

Andreas Kling 🐞	Andrew Kelley ⚡	Andrew Kaster
Ali Mohammad Pur	Daniel Bertalan	Nico Weber
Tim Schumacher	Gunnar Beutner	Idan Horowitz
Egor Ananyin	Peter Bindels	

...and all contributors!

And You...

Challenge Extra Stage: SerenityOS Kernel!