A Pass, a Plugin & a Python Hack

Stefan Gränitz // LLVM Meetup Berlin // 11 December 2024

"The Golden Age of Compilers in an Era of ...

— Chris Lattner, ASPLOS 2021

"The Golden Age of Compilers in an Era of HW/SW Co-design"

"The Golden Age of Compilers
in an Era of ~~HW/SW Co-design~~"

Fragmentation?

Compiler pass for a niche audience

What are the options for distribution?

Get it into the mainline compiler?

This is a niche topic

No mainline maintainer cares about the PR

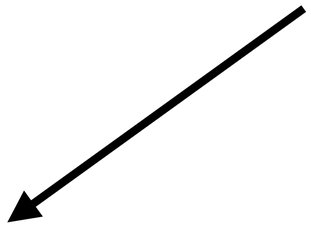
No-one understands the point

Few people will use it, so

why must everyone have it?

Compiler pass for a niche audience

What are the options for distribution?



~~Get it into the machine compiler?~~

Ship your own fork of the compiler?

Compiler pass for a niche audience

What are the options for distribution?

↓

Ship your own fork of the compiler?

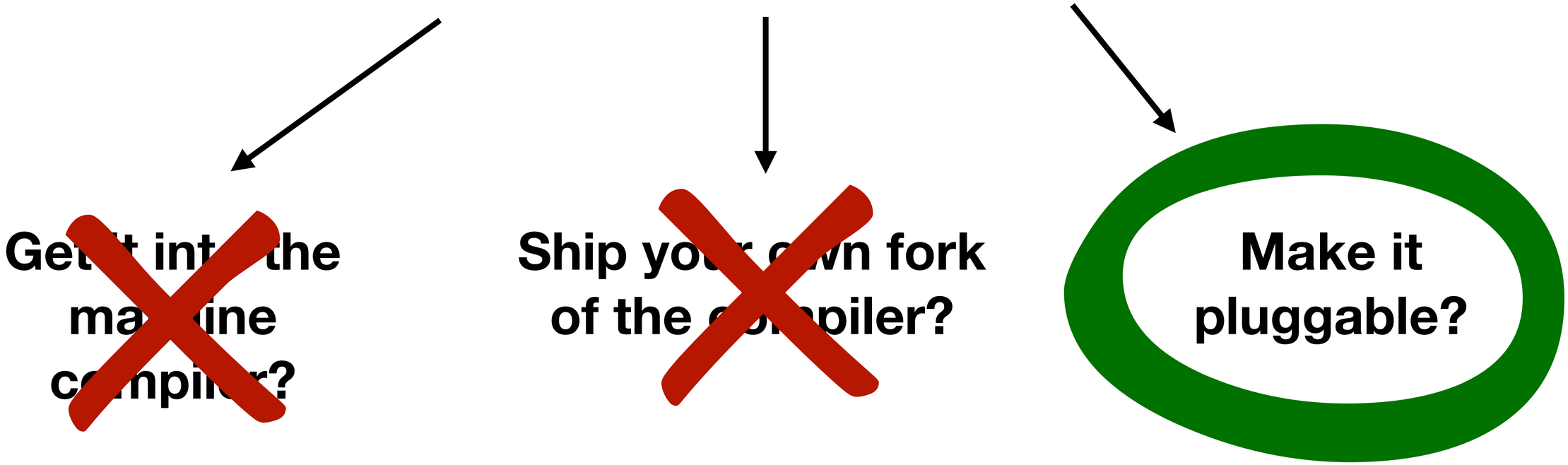
Good luck surviving downstream

Non-signed compilers are a security issue

Compilers are monolithic and distributed as variants: lang × arch × OS × SDK version

Compiler pass for a niche audience

What are the options for distribution?



Quick tour of what might happen next

Find the docs https://llvm.org/docs/WritingAnLLVMNewPMPass.html

Registering passes as plugins

LLVM provides a mechanism to register pass plugins within various tools like `clang` or `opt`. A pass plugin can add passes to default optimization pipelines or to be manually run via tools like `opt`. For more information, see Using the New Pass Manager.

Create a CMake project at the root of the repo alongside other projects. This project must contain the following minimal `CMakeLists.txt`:

```
add_llvm_pass_plugin(MyPassName source.cpp)
```

See the definition of `add_llvm_pass_plugin` for more CMake details.

Quick tour of what might happen next

Find the docs https://llvm.org/docs/WritingAnLLVMNewPMPass.html

The pass must provide at least one of the entry points for the new pass manager, one for static registration and one for dynamically loaded plugins:

- `llvm::PassPluginLibraryInfo get##Name##PluginInfo();`
- `extern "C" ::llvm::PassPluginLibraryInfo llvmGetPassPluginInfo() LLVM_ATTRIBUTE_WEAK;`

Pass plugins are compiled and linked dynamically by default. Setting `LLVM_${NAME}_LINK_INTO_TOOLS` to `ON` turns the project into a statically linked extension.

For an in-tree example, see `llvm/examples/Bye/`.

Quick tour of what might happen next

Find the Bye 👋 example plugin in llvm-project

Search: Bye

Bye

files to include

llvm

files to exclude

```
23 results - 14 files

llvm/examples/CMakeLists.txt:
  9  add_subdirectory(SpeculativeJIT)
 10: add_subdirectory(Bye)
 11

llvm/examples/Bye/Bye.cpp:
 18    if (Wave) {
 19:     errs() << "Bye: ";
 20      errs().write_escaped(F.getName()) << '\n';

 30
 31: struct Bye : PassInfoMixin<Bye> {
 32    PreservedAnalyses run(Function &F, FunctionAnalysisManager &) {

 42
 43: static RegisterPass<LegacyBye> X("goodbye", "Good Bye World Pass",
 44                                   false /* Only looks at CFG */,
```

Quick tour of what might happen next

Find the Bye 👋 example plugin in llvm-project

```
43: static RegisterPass<LegacyBye> X("goodbye", "Good Bye World Pass",
44                                   false /* Only looks at CFG */,

54  llvm::PassPluginLibraryInfo getByePluginInfo() {
55:   return {LLVM_PLUGIN_API_VERSION, "Bye", LLVM_VERSION_STRING,
56            [](PassBuilder &PB) {

58                  [](llvm::FunctionPassManager &PM, OptimizationLevel Level) {
59:                   PM.addPass(Bye());
60                  });

64                    if (Name == "goodbye") {
65:                     PM.addPass(Bye());
66                      return true;
```

llvm/examples/Bye/CMakeLists.txt:

```
9   if (NOT WIN32)
10:   add_llvm_pass_plugin(Bye
11:     Bye.cpp
12      DEPENDS
```

llvm/test/CMakeLists.txt:

```
182      list(APPEND LLVM_TEST_DEPENDS
183:       Bye
184        )
```

llvm/test/lit.cfg.py:

```
230      config.substitutions.append(('%loadbye',
231:                                  '-load={}/Bye{}'.format(config.llvm_shlib_dir,
```

Quick tour of what might happen next

Find the Bye 👋 example plugin in llvm-project

llvm/test/CMakeLists.txt:

```
182     list(APPEND LLVM_TEST_DEPENDS
183:      Bye
184       )
```

llvm/test/lit.cfg.py:

```
230     config.substitutions.append(('%loadbye',
231:                                 '-load={}/Bye{}'.format(config.llvm_shlib_dir,
232                                                         config.llvm_shlib_ext)))
233     config.substitutions.append(('%loadnewpmbye',
234:                                 '-load-pass-plugin={}/Bye{}'
235                                  .format(config.llvm_shlib_dir,
```

llvm/test/Feature/load_extension.ll:

```
9   ; UNSUPPORTED: windows
10: ; CHECK: Bye
11
```

Quick tour of what might happen next

What libs to link?

```
971  function(add_llvm_pass_plugin name)
972    cmake_parse_arguments(ARG
973      "NO_MODULE" "SUBPROJECT" ""
974      ${ARGN})
975
976    string(TOUPPER ${name} name_upper)
977
978    # Enable the plugin by default if it was explicitly enabled by the user.
979    # Note: If was set to "all", LLVM's CMakeLists.txt replaces it with a
980    # list of all projects, counting as explicitly enabled.
981    set(link_into_tools_default OFF)
982    if (ARG_SUBPROJECT AND LLVM_TOOL_${name_upper}_BUILD)
983      set(link_into_tools_default ON)
984    endif()
985    option(LLVM_${name_upper}_LINK_INTO_TOOLS "Statically link ${name} into tools
986
987    # If we statically link the plugin, don't use llvm dylib because we're going
988    # to be part of it.
989    if(LLVM_${name_upper}_LINK_INTO_TOOLS)
990        list(APPEND ARG_UNPARSED_ARGUMENTS DISABLE_LLVM_LINK_LLVM_DYLIB)
991    endif()
992
993    if(LLVM_${name_upper}_LINK_INTO_TOOLS)
994      list(REMOVE_ITEM ARG_UNPARSED_ARGUMENTS BUILDTREE_ONLY)
```

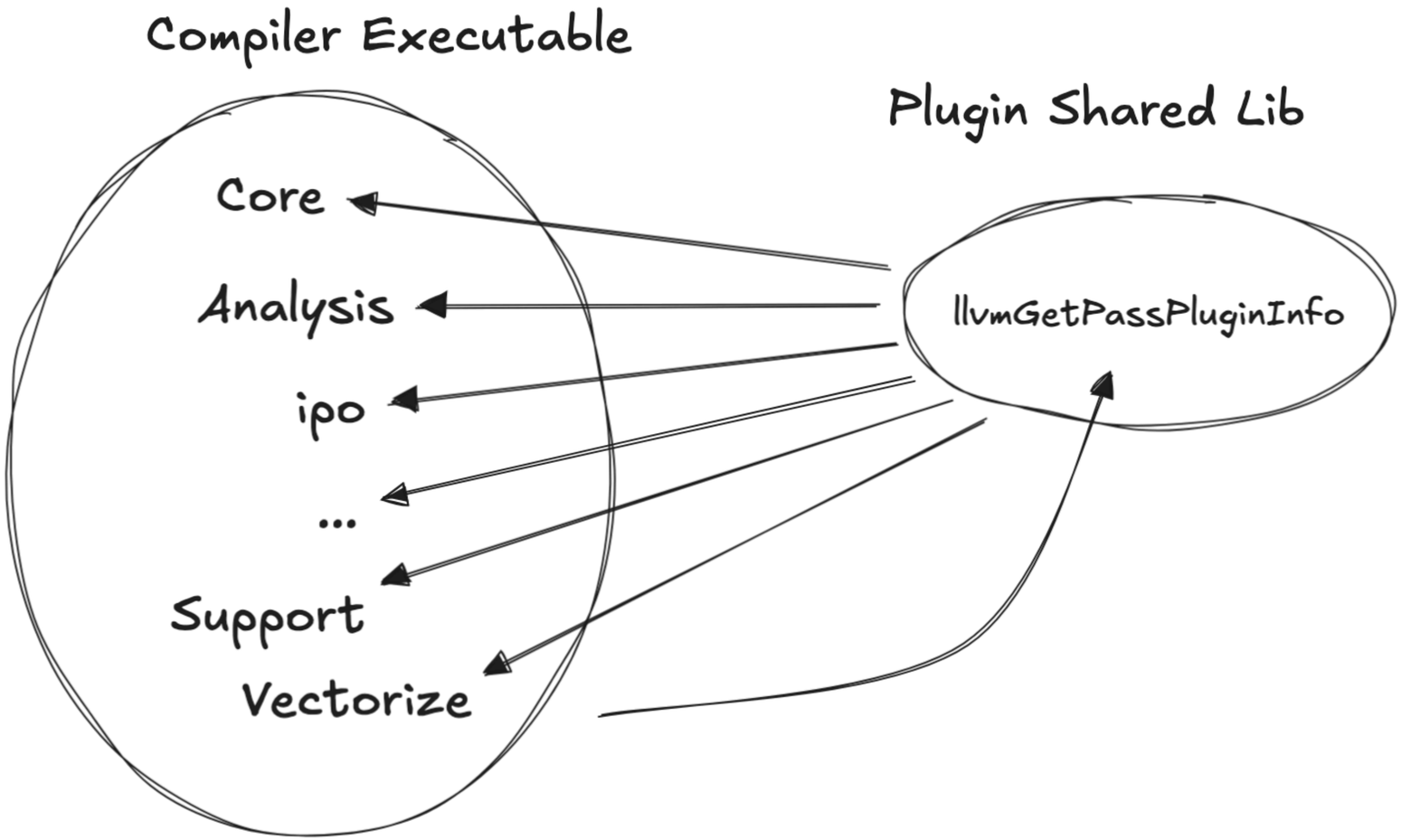
Quick tour of what might happen next

What libs to link?

```
build → ninja -t commands Bye.dylib | tail -1
: && /Applications/Xcode_15.2.app/Contents/Developer/Toolchains/
XcodeDefault.xctoolchain/usr/bin/c++ -fPIC -fvisibility-inlines-
hidden -Werror=date-time -Werror=unguarded-availability-new
-Wall -Wextra -Wno-unused-parameter -Wwrite-strings -Wcast-qual
-Wmissing-field-initializers -pedantic -Wno-long-long -Wc++98-
compat-extra-semi -Wimplicit-fallthrough -Wcovered-switch-
default -Wno-noexcept-type -Wnon-virtual-dtor -Wdelete-non-
virtual-dtor -Wsuggest-override -Wstring-conversion
-Wmisleading-indentation -Wctad-maybe-unsupported -fdiagnostics-
color -g -isysroot /Applications/Xcode_15.2.app/Contents/
Developer/Platforms/MacOSX.platform/Developer/SDKs/
MacOSX14.2.sdk -bundle -Wl,-headerpad_max_install_names -Wl,-
flat_namespace -Wl,-undefined -Wl,suppress   -Wl,-
no_warn_duplicate_libraries -o lib/Bye.dylib examples/Bye/
CMakeFiles/Bye.dir/Bye.cpp.o  -Wl,-rpath,@loader_path/../lib
&& :
```

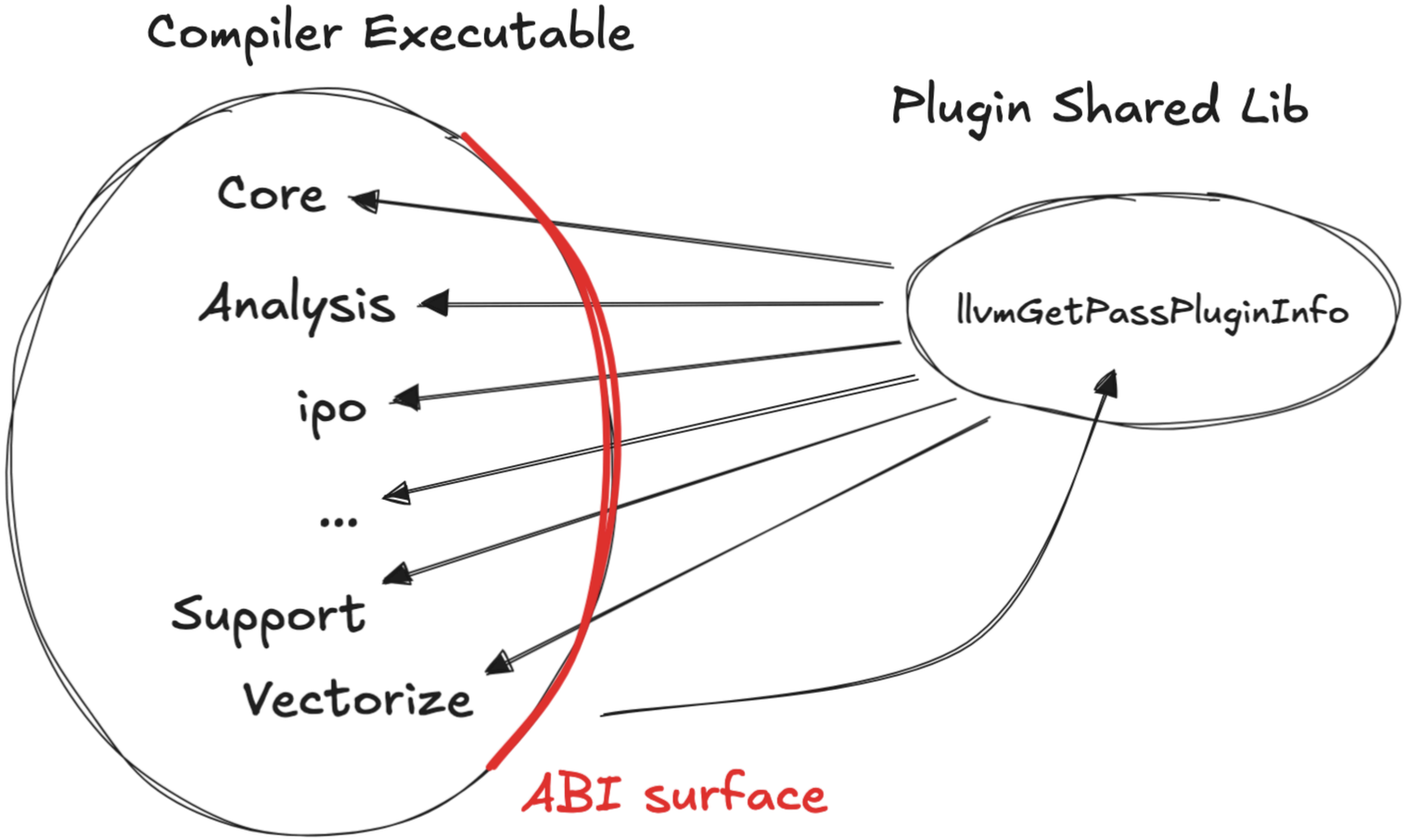
Quick tour of what might happen next

Link no libs at all



Quick tour of what might happen next

Link no libs at all



DEMO: example pass

Quick tour of what might happen next

Build and run

Works!

Quick tour of what might happen next

ABI surface

```
                 U __ZN4llvm11raw_ostream5writeEPKcm
                 U __ZN4llvm15SmallVectorBaseIjE13mallocForGrowEmmRm
                 U __ZN4llvm17PreservedAnalyses14AllAnalysesKeyE
0000000000001460 t __ZN4llvm23SmallVectorTemplateBaseINSt3__18functionIFbNS_9StringRefERNS_11PassManagerINS_
00000000000013b0 t __ZN4llvm23SmallVectorTemplateBaseINSt3__18functionIFbNS_9StringRefERNS_11PassManagerINS_
0000000000001460 t __ZN4llvm23SmallVectorTemplateBaseINSt3__18functionIFvRNS_11PassManagerINS_6ModuleENS_15A
00000000000013b0 t __ZN4llvm23SmallVectorTemplateBaseINSt3__18functionIFvRNS_11PassManagerINS_6ModuleENS_15A
                 U __ZN4llvm24DisableABIBreakingChecksE
0000000000003068 d __ZN4llvm30VerifyDisableABIBreakingChecksE
                 U __ZN4llvm5Value7setNameERKNS_5TwineE
0000000000001b10 t __ZN4llvm6detail9PassModelINS_6ModuleEN12_GLOBAL__N_16PyPassENS_17PreservedAnalysesENS_15
00000000000018a0 t __ZN4llvm6detail9PassModelINS_6ModuleEN12_GLOBAL__N_16PyPassENS_17PreservedAnalysesENS_15
0000000000001890 t __ZN4llvm6detail9PassModelINS_6ModuleEN12_GLOBAL__N_16PyPassENS_17PreservedAnalysesENS_15
0000000000001880 t __ZN4llvm6detail9PassModelINS_6ModuleEN12_GLOBAL__N_16PyPassENS_17PreservedAnalysesENS_15
                 U __ZNK4llvm5Value7getNameEv
0000000000001ca0 t __ZNK4llvm6detail9PassModelINS_6ModuleEN12_GLOBAL__N_16PyPassENS_17PreservedAnalysesENS_1
0000000000001c00 t __ZNK4llvm6detail9PassModelINS_6ModuleEN12_GLOBAL__N_16PyPassENS_17PreservedAnalysesENS_1
                 U __ZNK4llvm9StringRef4findES0_m
0000000000001cf0 t __ZNKSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEU
0000000000001cd0 t __ZNKSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEU
00000000000015c0 t __ZNKSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEU
00000000000015a0 t __ZNKSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEU
0000000000001d10 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001d00 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001cc0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001cb0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001d20 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
00000000000015e0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
00000000000015d0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001590 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001580 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
00000000000015f0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001670 t __ZNSt3__16vectorINS_10unique_ptrIN4llvm6detail11PassConceptINS2_6ModuleENS2_15AnalysisMa
0000000000001660 t __ZSt28__throw_bad_array_new_lengthB8nn180100v
0000000000002078 s __ZTVN4llvm6detail9PassModelINS_6ModuleEN12_GLOBAL__N_16PyPassENS_17PreservedAnalysesENS_
00000000000020b8 s __ZTVNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEE
0000000000002030 s __ZTVNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEE
00000000000012e0 t __ZZ21llvmGetPassPluginInfoEN3$_08__invokeERN4llvm11PassBuilderE
                 U __ZdlPv
                 U __Znwm
```

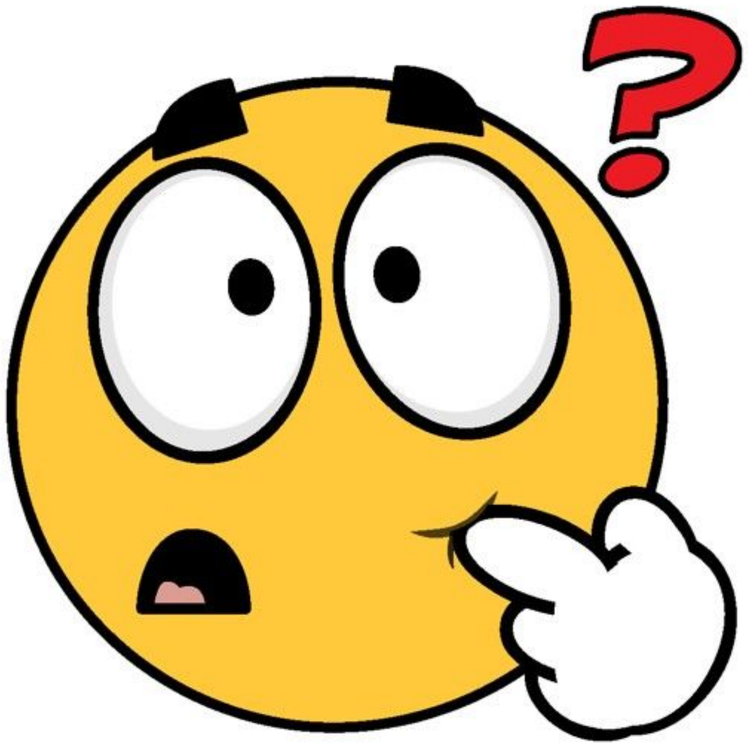
Quick tour of what might happen next

ABI surface

```
                 U __ZNK4llvm9StringRef4findES0_m
0000000000001cf0 t __ZNKSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEU
0000000000001cd0 t __ZNKSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEU
00000000000015c0 t __ZNKSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEU
00000000000015a0 t __ZNKSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEU
0000000000001d10 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001d00 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001cc0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001cb0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001d20 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
00000000000015e0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
00000000000015d0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001590 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001580 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
00000000000015f0 t __ZNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEEUl
0000000000001670 t __ZNSt3__16vectorINS_10unique_ptrIN4llvm6detail11PassConceptINS2_6ModuleENS2_15AnalysisMa
0000000000001660 t __ZSt28__throw_bad_array_new_lengthB8nn180100v
0000000000002078 s __ZTVN4llvm6detail9PassModelINS_6ModuleEN12_GLOBAL__N_16PyPassENS_17PreservedAnalysesENS_
00000000000020b8 s __ZTVNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEE
0000000000002030 s __ZTVNSt3__110__function6__funcIZZ21llvmGetPassPluginInfoENK3$_0clERN4llvm11PassBuilderEE
00000000000012e0 t __ZZ21llvmGetPassPluginInfoEN3$_08__invokeERN4llvm11PassBuilderE
                 U __ZdlPv
                 U __Znwm
                 U ___stack_chk_fail
                 U ___stack_chk_guard
                 U ___stderrp
0000000000003060 d __dyld_private
                 U _abort
                 U _fprintf
                 U _free
                 U _fwrite
00000000000012b0 T _llvmGetPassPluginInfo
                 U _memcpy
                 U _memmove
                 U dyld_stub_binder
```

Quick tour of what might happen next

Clang doesn't link all of LLVM



Quick tour of what might happen next

Clang doesn't link all of LLVM

```
lib/libLLVMAArch64AsmParser.a
lib/libLLVMAArch64CodeGen.a
lib/libLLVMAArch64Desc.a
lib/libLLVMAArch64Disassembler.a
lib/libLLVMAArch64Info.a
lib/libLLVMAArch64Utils.a
lib/libLLVMARMAsmParser.a
lib/libLLVMARMCodeGen.a
lib/libLLVMARMDesc.a
lib/libLLVMARMDisassembler.a
lib/libLLVMARMInfo.a
lib/libLLVMARMUtils.a
lib/libLLVMAggressiveInstCombine.a
lib/libLLVMAnalysis.a
lib/libLLVMAsmParser.a
lib/libLLVMAsmPrinter.a
lib/libLLVMBinaryFormat.a
lib/libLLVMBitReader.a
lib/libLLVMBitWriter.a
lib/libLLVMBitstreamReader.a
lib/libLLVMCFGuard.a
lib/libLLVMCFIVerify.a
lib/libLLVMCodeGen.a
lib/libLLVMCore.a
lib/libLLVMCoroutines.a
lib/libLLVMCoverage.a
lib/libLLVMDWARFLinker.a
lib/libLLVMDWP.a
lib/libLLVMDebugInfoCodeView.a
lib/libLLVMDebugInfoDWARF.a
lib/libLLVMDebugInfoGSYM.a
lib/libLLVMDebugInfoMSF.a
lib/libLLVMDebugInfoPDB.a
lib/libLLVMDebuginfod.a
lib/libLLVMDemangle.a
lib/libLLVMDiff.a
lib/libLLVMDlltoolDriver.a
lib/libLLVMExecutionEngine.a
```

```
lib/libLLVMAArch64AsmParser.a
lib/libLLVMAArch64CodeGen.a
lib/libLLVMAArch64Desc.a
lib/libLLVMAArch64Disassembler.a
lib/libLLVMAArch64Info.a
lib/libLLVMAArch64Utils.a
lib/libLLVMARMAsmParser.a
lib/libLLVMARMCodeGen.a
lib/libLLVMARMDesc.a
lib/libLLVMARMDisassembler.a
lib/libLLVMARMInfo.a
lib/libLLVMARMUtils.a
lib/libLLVMAggressiveInstCombine.a
lib/libLLVMAnalysis.a
lib/libLLVMAsmParser.a
lib/libLLVMAsmPrinter.a
lib/libLLVMBinaryFormat.a
lib/libLLVMBitReader.a
lib/libLLVMBitWriter.a
lib/libLLVMBitstreamReader.a
lib/libLLVMCFGuard.a
lib/libLLVMCodeGen.a
lib/libLLVMCore.a
lib/libLLVMCoroutines.a
lib/libLLVMCoverage.a
lib/libLLVMDebugInfoCodeView.a
lib/libLLVMDebugInfoDWARF.a
lib/libLLVMDebugInfoMSF.a
lib/libLLVMDemangle.a
```

Quick tour of what might happen next

Clang doesn't link all of LLVM

```
36 - lib/libLLVMDiff.a
37 - lib/libLLVMDlltoolDriver.a
38 - lib/libLLVMExecutionEngine.a
39 - lib/libLLVMExegesis.a
40 - lib/libLLVMExegesisAArch64.a
41 - lib/libLLVMExegesisX86.a
42   lib/libLLVMExtensions.a
43 - lib/libLLVMFileCheck.a
44 - lib/libLLVMFrontendOpenACC.a
45   lib/libLLVMFrontendOpenMP.a
46 - lib/libLLVMFuzzMutate.a
47   lib/libLLVMGlobalISel.a
48   lib/libLLVMIRReader.a
49   lib/libLLVMInstCombine.a
50   lib/libLLVMInstrumentation.a
51 - lib/libLLVMInterfaceStub.a
52 - lib/libLLVMInterpreter.a
53 - lib/libLLVMJITLink.a
54   lib/libLLVMLTO.a
55 - lib/libLLVMLibDriver.a
56 - lib/libLLVMLineEditor.a
57   lib/libLLVMLinker.a
58   lib/libLLVMMC.a
59 - lib/libLLVMMCA.a
60   lib/libLLVMMCDisassembler.a
61 - lib/libLLVMMCJIT.a
62   lib/libLLVMMCParser.a
63 - lib/libLLVMMIRParser.a
64   lib/libLLVMObjCARCOpts.a
65   lib/libLLVMObject.a
66 - lib/libLLVMObjectYAML.a
67   lib/libLLVMOption.a
68 - lib/libLLVMOrcJIT.a
69 - lib/libLLVMOrcShared.a
70 - lib/libLLVMOrcTargetProcess.a
71   lib/libLLVMPasses.a
72   lib/libLLVMProfileData.a
73   lib/libLLVMRemarks.a
74 - lib/libLLVMRuntimeDyld.a
```

```
30   lib/libLLVMExtensions.a
31   lib/libLLVMFrontendOpenMP.a
32   lib/libLLVMGlobalISel.a
33   lib/libLLVMIRReader.a
34   lib/libLLVMInstCombine.a
35   lib/libLLVMInstrumentation.a
36   lib/libLLVMLTO.a
37   lib/libLLVMLinker.a
38   lib/libLLVMMC.a
39   lib/libLLVMMCDisassembler.a
40   lib/libLLVMMCParser.a
41   lib/libLLVMObjCARCOpts.a
42   lib/libLLVMObject.a
43   lib/libLLVMOption.a
44   lib/libLLVMPasses.a
45   lib/libLLVMProfileData.a
46   lib/libLLVMRemarks.a
```

Quick tour of what might happen next

Clang doesn't link all of LLVM

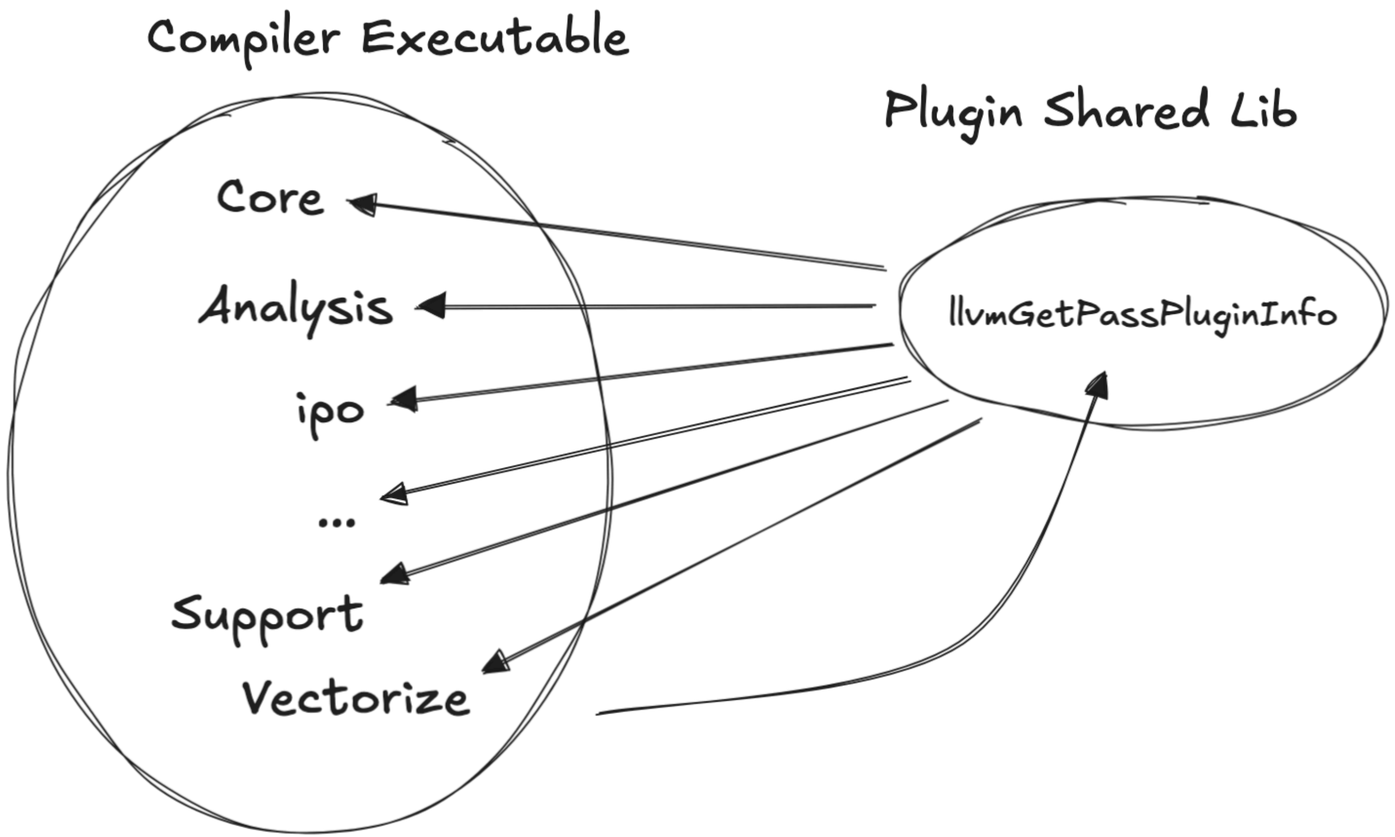
Different targets, different libs..?!

```
63- lib/libLLVMMIRParser.a
64  lib/libLLVMObjCARCOpts.a
65  lib/libLLVMObject.a
66- lib/libLLVMObjectYAML.a
67  lib/libLLVMOption.a
68- lib/libLLVMOrcJIT.a
69- lib/libLLVMOrcShared.a
70- lib/libLLVMOrcTargetProcess.a
71  lib/libLLVMPasses.a
72  lib/libLLVMProfileData.a
73  lib/libLLVMRemarks.a
74- lib/libLLVMRuntimeDyld.a
75  lib/libLLVMScalarOpts.a
76  lib/libLLVMSelectionDAG.a
77  lib/libLLVMSupport.a
78- lib/libLLVMSymbolize.a
79- lib/libLLVMTableGen.a
80- lib/libLLVMTableGenGlobalISel.a
81  lib/libLLVMTarget.a
82- lib/libLLVMTestingSupport.a
83  lib/libLLVMTextAPI.a
84  lib/libLLVMTransformUtils.a
85  lib/libLLVMVectorize.a
86- lib/libLLVMWindowsManifest.a
87  lib/libLLVMX86AsmParser.a
88  lib/libLLVMX86CodeGen.a
89  lib/libLLVMX86Desc.a
90  lib/libLLVMX86Disassembler.a
91  lib/libLLVMX86Info.a
92- lib/libLLVMX86TargetMCA.a
93- lib/libLLVMXRay.a
94  lib/libLLVMipo.a
95
```

```
41  lib/libLLVMObjCARCOpts.a
42  lib/libLLVMObject.a
43  lib/libLLVMOption.a
44  lib/libLLVMPasses.a
45  lib/libLLVMProfileData.a
46  lib/libLLVMRemarks.a
47  lib/libLLVMScalarOpts.a
48  lib/libLLVMSelectionDAG.a
49  lib/libLLVMSupport.a
50  lib/libLLVMTarget.a
51  lib/libLLVMTextAPI.a
52  lib/libLLVMTransformUtils.a
53  lib/libLLVMVectorize.a
54  lib/libLLVMX86AsmParser.a
55  lib/libLLVMX86CodeGen.a
56  lib/libLLVMX86Desc.a
57  lib/libLLVMX86Disassembler.a
58  lib/libLLVMX86Info.a
59  lib/libLLVMipo.a
60
```

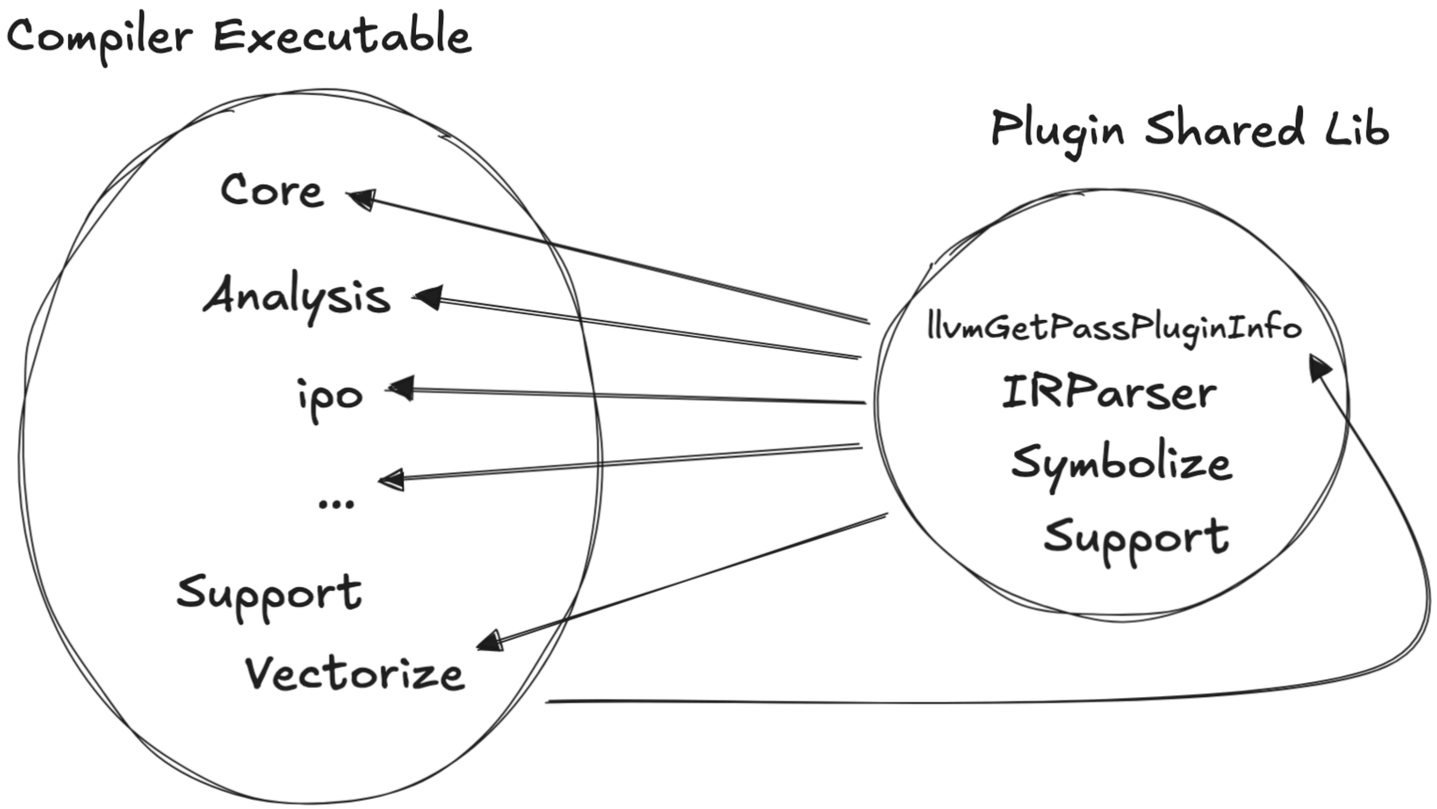
Quick tour of what might happen next

~~Link no libs at all~~



Quick tour of what might happen next

Link some libs?



Quick tour of what might happen next

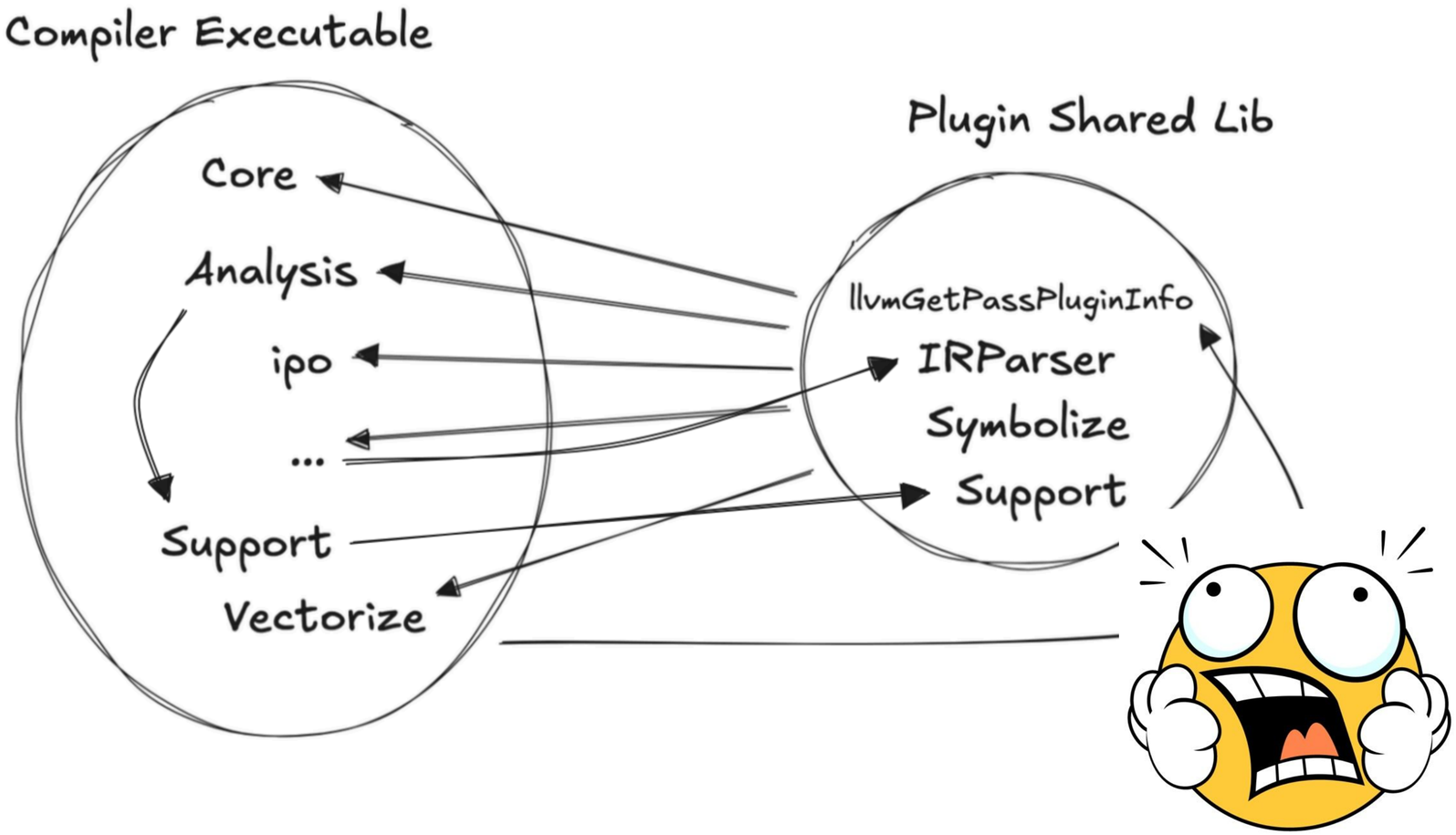
~~Link some libs?~~

Global variables

Static init

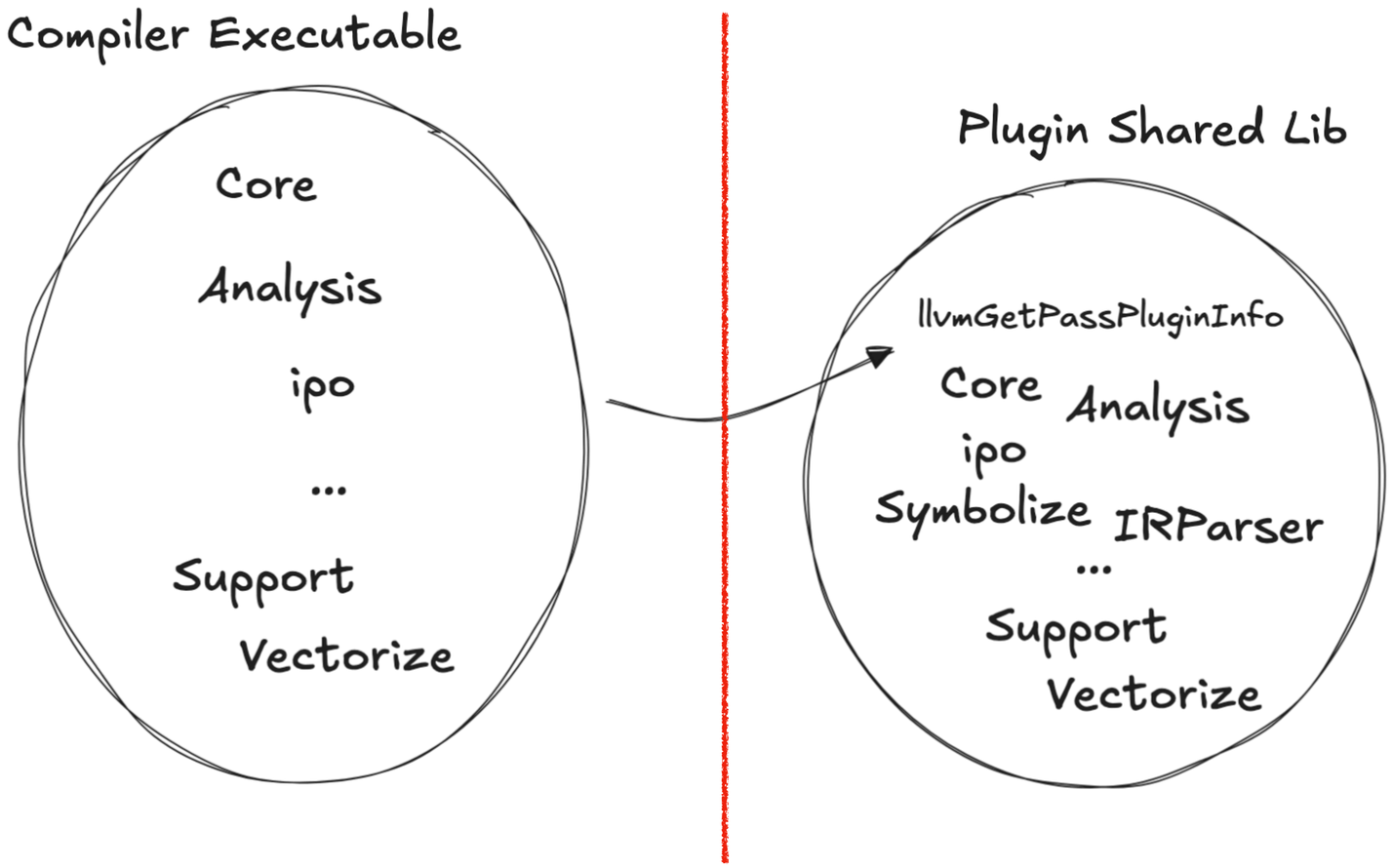
Weak symbols

Duplicate options



Quick tour of what might happen next

Link all of LLVM



Quick tour of what might happen next

All ABI questions solved?

No

☞ **Demo**

Quick tour of what might happen next

What sources to build against?

Linux Windows Android macOS/iOS

https://github.com/llvm/llvm-project

clang_source_info.md

https://android.googlesource.com/toolchain/llvm-project

https://github.com/swiftlang/llvm-project + X

Quick tour of what might happen next

What SDK to build with?

Released compilers are stage-2 builds right?

So, whatever they produce should be compatible..

Quick tour of what might happen next

What SDK to build with?

Well.. random example

```
394  Load command 18
395            cmd LC_LOAD_DYLIB
396        cmdsize 48
397:          name /usr/lib/libc++.1.dylib (offset 24)
398     time stamp 2 Thu Jan  1 01:00:02 1970
399        current version 1700.245.0
400  compatibility version 1.0.0
```

The version clang links against

```
394  Load command 18
395            cmd LC_LOAD_DYLIB
396        cmdsize 48
397:          name /usr/lib/libc++.1.dylib (offset 24)
398     time stamp 2 Thu Jan  1 01:00:02 1970
399        current version 1700.255.0
400  compatibility version 1.0.0
```

The version clang is linking binaries against

Quick tour of what might happen next

What was it again that we wanted to do?

Compiler pass for a niche audience

What are the options for distribution?

~~Get it into the machine compiler?~~

~~Ship your own fork of the compiler?~~

Make it pluggable?

Is it a suitable option?

Compiler pass for a niche audience

What could we do?

A generic plugin container with a Python programming layer?

- build once for each: LLVM version × arch × OS × SDK version
- supported in Clang, Swift, Rust unofficial
- compatible with LLVM tools like opt, clang-repl

DEMO: llvm-py-pass

Questions?

Is that compatible with MLIR?

Why do we use LLVM 14?

Is llvmlite the only option on the Python side?

Why isn't the ABI surface sufficient to determine ABI compatibility?