Fuchsia

LLVM Multicall Driver

Alex Brachet (*abrachet@google.com*)

What is the multicall driver

- Many tools linked into one binary
  - Like “busybox”
- One binary called just `llvm` and we install tools like `clang` as a symlink to it
- Convenience of static linking with the size of a dynamically linked toolchain
  - Some tools have a lot of overlap. `clang-scan-deps` was 112M standalone and less than 1M added size to the driver binary
- We aim to a ship a new toolchain weekly
  - For some users this cadendence was high to be a downloading an entire toolchain
- Saved over 860M in binary size
  - 1.5G to 640M

What is the multicall driver

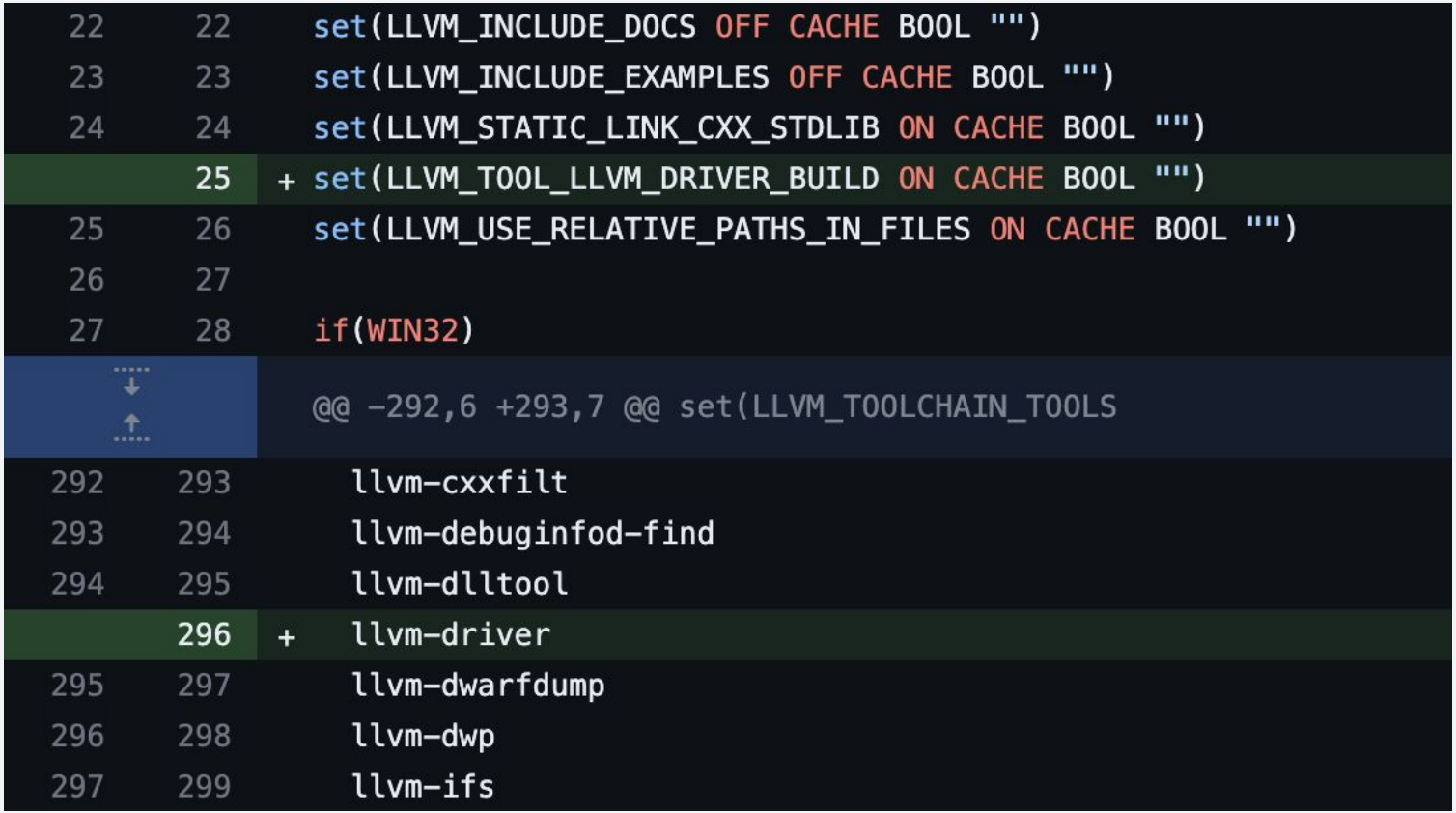
llvm/tools/sancov/CMakeLists.txt

```
@@ -20,4 +20,5 @@ add_llvm_tool(sancov
20  20
21  21    DEPENDS
22  22    SancovOptsTableGen
    23  +  GENERATE_DRIVER
23  24    )
```

llvm/tools/sancov/sancov.cpp

```
@@ -38,6 +38,7 @@
38  38  #include "llvm/Support/FileSystem.h"
39  39  #include "llvm/Support/InitLLVM.h"
40  40  #include "llvm/Support/JSON.h"
    41  + #include "llvm/Support/LLVMDriver.h"
41  42  #include "llvm/Support/MD5.h"
42  43  #include "llvm/Support/MemoryBuffer.h"
43  44  #include "llvm/Support/Path.h"
@@ -1221,7 +1222,7 @@ static void parseArgs(int Argc, char **Argv) {
1221 1222    ClIgnorelist = Args.getLastArgValue(OPT_ignorelist_EQ);
1222 1223  }
1223 1224
1224     - int main(int Argc, char **Argv) {
        1225  + int sancov_main(int Argc, char **Argv, const llvm::ToolContext &) {
```

How to use


22 22 set(LLVM_INCLUDE_DOCS OFF CACHE BOOL "")
23 23 set(LLVM_INCLUDE_EXAMPLES OFF CACHE BOOL "")
24 24 set(LLVM_STATIC_LINK_CXX_STDLIB ON CACHE BOOL "")
25 + set(LLVM_TOOL_LLVM_DRIVER_BUILD ON CACHE BOOL "")
25 26 set(LLVM_USE_RELATIVE_PATHS_IN_FILES ON CACHE BOOL "")
26 27
27 28 if(WIN32)
@@ -292,6 +293,7 @@ set(LLVM_TOOLCHAIN_TOOLS
292 293 llvm-cxxfilt
293 294 llvm-debuginfod-find
294 295 llvm-dlltool
296 + llvm-driver
295 297 llvm-dwarfdump
296 298 llvm-dwp
297 299 llvm-ifs

How to use

```
32      if(WIN32)
33        set(FUCHSIA_DISABLE_DRIVER_BUILD ON)
34      endif()
35
36      if (NOT FUCHSIA_DISABLE_DRIVER_BUILD)
37        set(LLVM_TOOL_LLVM_DRIVER_BUILD ON CACHE BOOL "")
38        set(LLVM_DRIVER_TARGET llvm-driver)
39      endif()
```

Issues we ran into rolling out

- Hardcoded expectation that `clang` is a symlink to `clang-{ver}`
- `clang -### -canonical-prefixes`
  - The output of this looks like `{full_path}/llvm clang …`
  - `-canonical-prefixes` instructs clang to take the realpath of itself, which in the driver build will point to `llvm`
  - Another hardcoded expectation that the binary clang invokes will be called `clang`

Remaining tools

- Not all tools are part of the multicall driver
- Global `cl::opt`'s with the same value can't be linked together
  - We need to first transition tools to use `OptTable` before we can add them to the driver build
  - Clang's use of `cl::opt` objects in passes is fine because these names are unique and will not conflict with other tools
- Many clang tools left
  - These use `clang::tooling::CommonOptionsParser` which expects that options are parsed with `llvm::cl`
  - These tools are not generally large so we haven't prioritized this work

Special thanks to

chris.bieneman@me.com

maskray@google.com

phosek@google.com

andresvi@google.com