LLVMLinux: Compiling Android with LLVM

Presented by:

Behan Webster

Presentation Date: 2013.02.21

Original idea for the talk

- Compiling complete Android system with LLVM

  However...

- It took a lot more time and work than anticipated

Why Would I Want to Use Clang/LLVM to Compile Android?

Fast Moving Project

- In just a few years LLVM and Clang have reached and in some cases surpassed what other toolchains can do
- Easy to follow source code
- Inclusive community of developers
- Similar size and speed of resulting binaries to gcc

Faster Compiles

- Clang is known to compile code faster and use less memory than other toolchains
- This can make the debug/compile/test loop take a lot less time

Toolchain Toolkit

- LLVM is a set of libraries which can build in to:
  - Compiler, linker, JIT
  - Executables, virtual machines
  - Source code analysis tools
  - Meta data extraction from code

One Toolchain

- LLVM is already being used in a lot of domains:
  - DSP, GPU, CPU, JIT, etc.
  - Camera, audio, video, CUDA, Renderscript, kernel, Android, applications, documentation
- Compiler extensions only need to be written once
- For companies working on a range of technologies it's convenient to only need maintain/test a single toolchain

LLVM License

- Licensed under the "UIUC" BSD-Style license
- LLVM technology can be embedded into into non-GPL software
- Allows open and proprietary extensions
  - This is attractive to some companies
- Wider development audience
- Even more full-time developers making it better

Fix-it Hints

- "Fix-it" hints provide advice for fixing small, localized problems in source code.

```
$ clang t.c
t.c:5:28: warning: use of GNU old-style field designator extension struct
point origin = { x: 0.0, y: 0.0 };
                   ~~ ^
                   .x =

t.c:5:36: warning: use of GNU old-style field designator extension struct
point origin = { x: 0.0, y: 0.0 };
                         ~~ ^
                         .y =
```

- gcc 4.8 does similar things now
- This is an example of clang driving improvements to gcc

Static Analyzer

```
2919
2920        for_each_opt(opt, lecup_options, NULL) {
2921                if (optarg && strncasecmp("0x", optarg, 2) == 0)
```

1 Taking false branch

```
2922                        base = 16;
2923                else
2924                        base = 10;
2925
2926                switch (opt) {
```

2 Control jumps to 'case 116:' at line 2939

```
2927                case 'H':
2928                        handle = strtoul(optarg, NULL, base);
2929                        break;
2930                case 'm':
2931                        min = strtoul(optarg, NULL, base);
2932                        break;
2933                case 'M':
2934                        max = strtoul(optarg, NULL, base);
2935                        break;
2936                case 'l':
2937                        latency = strtoul(optarg, NULL, base);
2938                        break;
2939                case 't':
2940                        timeout = strtoul(optarg, NULL, base);
```

3 Null pointer passed as an argument to a 'nonnull' parameter

```
2941                        break;
```

http://littlechina.org/~vcgomes/bluez-static-analysis/2012-02-10-1/report-n7KJtW.html#EndPath

Other kinds of things

- Google is using LLVM to look for common bugs in their vast library of source code
- Once found bugs are found they can be fixed automatically with minimal human involvement
- Wouldn't the possibility for something like automatic code refactoring be a nice option when APIs changed?
- Checker can be extended to look for common bugs in kernel code so that bugs can be found earlier

Clang/LLVM already used by Linux Projects

- LLVM part of Renderscript compiler in Android
  - Supported on ARM, MIPS and x86

- LLVM is a hard dependency for Gallium3D
  - llvm-pipe driver, Clover (Open CL)
  - May be used for GLSL shader optimizer

- Clang built Debian - Sylvestre Ledru

LLVM toolchain suite

- Clang (C/C++/Objective-C)
- Libc++ (C++ library)
- Static Analyzer (Checker)
- LLDB (debugger)
- MC Linker and LLD (Linkers)

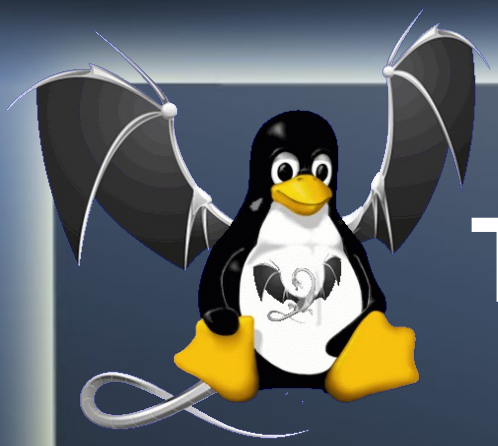
Commercial Deployment

- Clang is being used selectively in place of gcc when it is able to produce more optimal code. Now part of Android NDK
- Clang is commercially deployed in XCode and now Android

Driving Change in gcc

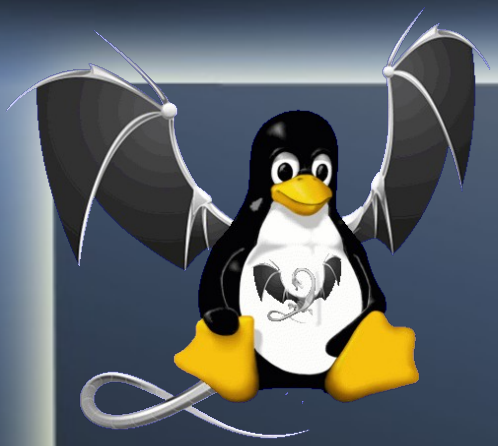
- Macro expansion
- Better error reporting
- Fix-it hints
- Address Sanitizer

The LLVMLinux Project



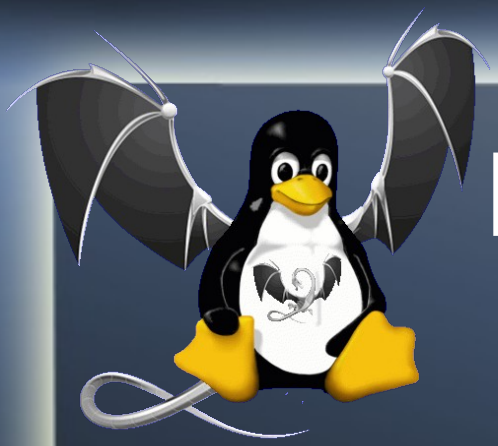
The LLVMProject Goals

- To fully build the Linux kernel for multiple architectures, using the Clang/LLVM toolchain
- Discover any blocking issues via testing and make patches
- Upstream any patches to the Linux Kernel and Clang/LLVM to make this possible
- Bring together like-minded developers



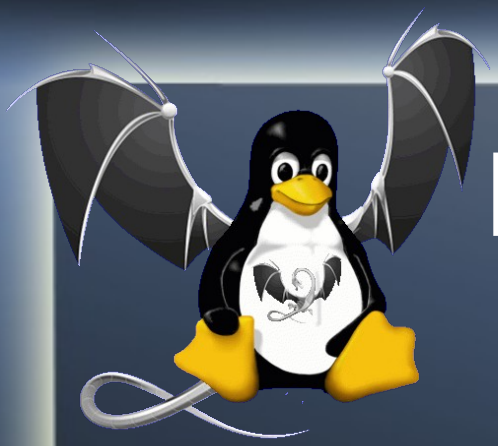
Project Website

- Project wiki page
  - http://llvm.linuxfoundation.org
- Project Status, Road map, Bugs
- Information about Architecture support
- Documentation, Howtos, Notes



LLVMLinux Automated Build Framework

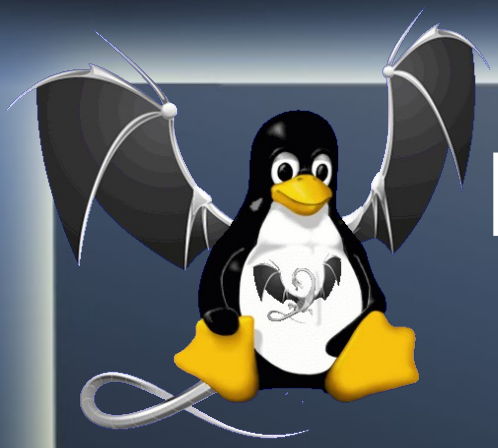
- git clone http://git.linuxfoundation.org/llvmlinux.git
- The framework consists of scripts and patches
- Automates fetching, patching, and building
  - LLVM, Clang,
  - Toolchains for cross assembler, linker
  - Linux Kernel
  - QEMU, and test images



LLVMLinux Automated Build Framework

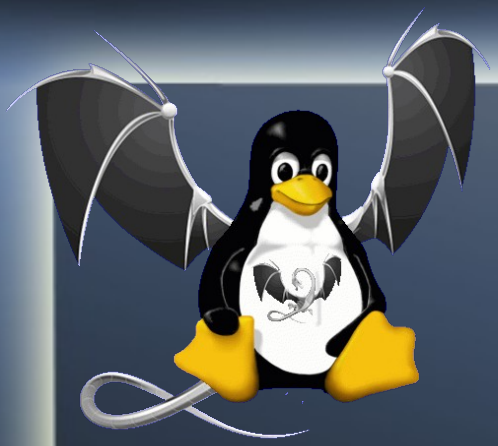
- Patch management using quilt
- Choice of cross- toolchain (gcc, as, ld)
  - Codesourcery (Default)
  - Linaro/Ubuntu
  - Android

```
$ make CROSS_ARM_TOOLCHAIN=android kernel-gcc-build
```



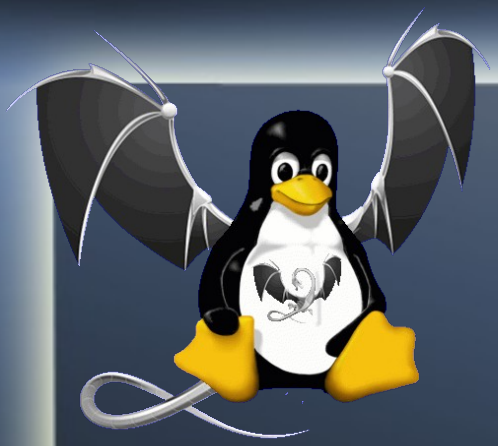
LLVMLinux Automated Build Framework

- Current support for various targets
  - Versatile Express (QEMU testing mainline)
  - Qualcomm MSM (3.4)
  - X86_64 (mainline)
  - Raspberry-pi (3.2 soon 3.6)
  - Nexus 7 (3.1.10)
  - Galaxy S3 (in progress for 3.0.59)



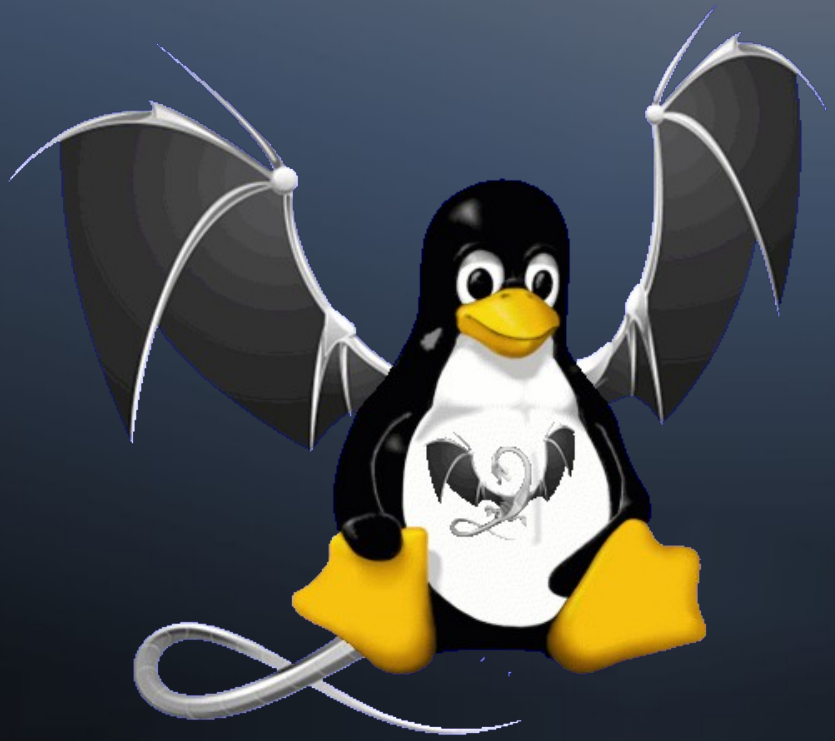
Buildbot

- Buildbot Continuous Integration Server
- Builds and tests LLVMLinux Code
- Builds and retests on every commit to the LLVM, Clang, and the Linux Kernel repos
- Also builds/tests the patched Linux Kernel with gcc to make sure not to break compatibility
- Runs LTP tests in QEMU for Versatile Express



Project Communication

- Project Mailing List
  - http://lists.linuxfoundation.org/mailman/listinfo/llvmlinux
  - http://lists.linuxfoundation.org/pipermail/llvmlinux/
- IRC Channel
  - #llvmlinux on OFTC
  - http://buildbot.llvm.linuxfoundation.org/irclogs/OFTC/%23llvmlinux/



Challenges Using Clang/LLVM to Build the Linux Kernel

Challenges Using Clang for Cross Compilation

- Finding the right triplet for Clang
- IA can't be used everywhere, and furthermore it doesn't support 16-bit code
- Dependence on GNU toolchain for assembly and linking (as and ld)
- Configuring GNU toolchain dependencies (-gcc-toolchain <path>)

Challenges Using Clang for Cross Compilation

- GCC Dependencies:
  - gcc defaults to gnu89, clang to gnu99
  - Code written with gcc often uses gcc extentions (some aren't supported by clang)
  - Kernel currently expects some undocumented GCC behavior
  - Unsupported GCC flags, built-in function behavior differences

Challenges with the Linux Kernel

- Kbuild is currently gcc-centric
- Unsupported gcc options in clang
- VLAIS (Variable Length Arrays In Structs)
- Explicit register variables not supported
- Segment references (problem with clang)
- Inline ASM incompatible with IA
- __builtin_constant_p() fails for Clang

Status of Building Linux Kernel With Clang/LLVM

Linux Kernel Patches

- Kbuild support
- Explicit ASM to handle register variables
- Remove the use of VLAIS
- Segment linkage differences related to attributes
- “extern inline” in ftrace.h (GNU89 vs GNU/C99)
- __builtin_constant_p() workaround
- GCC specific use of aligned attribute in cast

LLVM for Linux Status

- Only 4 patches for Clang/LLVM (svn)
  - Specific to X86_64 target
- 64-bit type handling for ARM now upstream
- Clang IA not yet enabled by default
- Stripped attribute issue affecting linking needs to be tracked down
- Clang 3.3 will likely work out-of-the-box for the Linux Kernel

Compiling Android (CyanogenMod 10.1)

Android Development on a GalaxyS3

- Samsung S3 AT&T (MSM8960)
- Need to root the phone first
- No fastboot on the Samsung S3
- Samsung phones use “ODIN mode” instead
- Use ODIN to install ClockworkMod Recovery
- (There is also a tool called Heimdal for Linux)
- With custom recovery image installed I can now install whatever I need

Android Kernel for GalaxyS3

- Started with the d2att kernel from CyanogenMod 10.1
- Ported LLVMLinux msm patches
- Build the kernel in LLVMLinux framework
- Added it to the CM 10.1 to build boot.img
- Install with ClockworkMod Recovery
- Reboot loop
- Doh!

Android Kernel for GalaxyS3

- Back to ClockworkMod Recovery
- adb shell
- cat /proc/last_kmsg to see what happened
- Debug oops message
- Rebuild, install, repeat

Android Development for Nexus7

- Nexus 7 (NVIDIA Tegra 3)
- Unlock the bootloader with fastboot (no problems!)
- Install ClockworkMod reovery with fast boot

Android Kernel for Nexus7

- Started with the grouper kernel from CM 10.1
- Ported ARM patches from LLVMLinux
- Build the kernel in LLVMLinux framework
- Added it to the CM 10.1 to build boot.img
- Install with ClockworkMod Recovery
- Worked first time!

Using LLVM to build Android

- Four approaches
  1. Pass in CC Variables to Makefiles
  2. Change CC variables in Makefiles
  3. Wrapper script in PATH to override compilers
  4. Replace compilers with wrapper scripts

Pass in CC

- CC=clang CXX=clang++ brunch d2att
- Result: Ignored settings and used gcc/g++

Change CC in Makefiles

- Replace gcc/gcc+ in Makefiles
- Result: Lots of errors in the build system
  - CC=$(CROSS_COMPILER)gcc
  - CC=clang -target $(CROSS_COMPILE:-=)

Wrapper script in PATH

- Write a wrapper script which gets loaded before all other compiles in the PATH
- Call clang instead of gcc
- Result: PATH is reset in Makefiles and scripts, or tools are called by absolute path
- the wrapper script was never run

Wrapper Replacing Compilers

- Replace the all instances of gcc on the filesystem with wrapper scripts which call clang appropriately
- Ran out of time to test this idea...

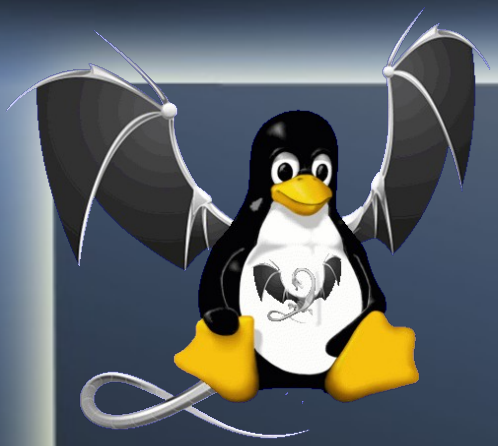
Overall Observations

- Similar to the kernel, the Android build system (CM build system) seems to be gcc-centric
- Gcc compiler options seem to be used which are not supported by clang
- There is likely code which compiles with gcc (gnu89) but doesn't with clang (gnu99)
- Need to use a version of clang which is built with support for bionic
- The version of clang/LLVM in Android is v3.1

Who wouldn't want ~~a penguin~~ an Android with dragon wings?

Thank you

http://llvm.linuxfoundation.org



Contribute to the LLVMLinux Project

- Project wiki page
  - http://llvm.linuxfoundation.org
- Project Mailing List
  - http://lists.linuxfoundation.org/mailman/listinfo/llvmlinux
  - http://lists.linuxfoundation.org/pipermail/llvmlinux/
- IRC Channel
  - #llvmlinux on OFTC
  - http://buildbot.llvm.linuxfoundation.org/irclogs/OFTC/%23llvmlinux/