LLVMLinux: Compiling the Linux Kernel with LLVM

Presented by:

Behan Webster

Presentation Date: 2013.02.21

What is Clang/LLVM?

LLVM is a Toolchain Toolkit

- LLVM is a modular set of libraries which can be used to build things such as:
  - Compiler, linker, JIT
  - Source code analysis tools
  - Meta data extraction from code
  - Code refactoring tools
  - Tight integration with IDEs

LLVM Toolchain Suite

- Clang (C/C++/Objective-C compiler)
- Libc++ (C++ library)
- Compiler-rt (highly tuned low level operations)
- Static Analyzer (Checker)
- LLDB (debugger)
- MC Linker and LLD (Linkers)
- And more...

Why Would I Want to Use Clang/LLVM to Compile the Linux Kernel?

Fast Moving Project

- In just a few years Clang has 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

Fast 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

One Toolchain

- LLVM is already being used in a lot of domains:
  - DSP, GPU, CPU, JIT, etc.
  - Camera, audio, video, CUDA, Renderscript, kernel, userspace, 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 both open and proprietary extensions
- Wide development audience
- Wide range of 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
- Clang part of the Android NDK
- 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

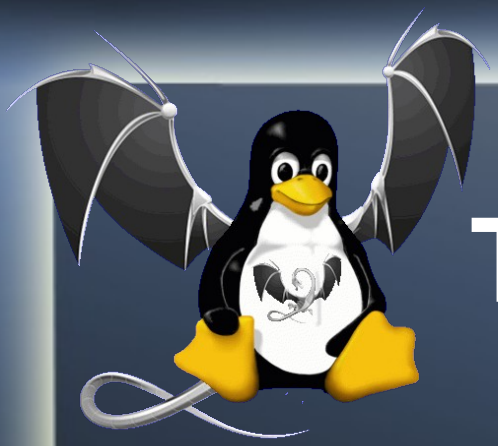
Commercial Deployment

- 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

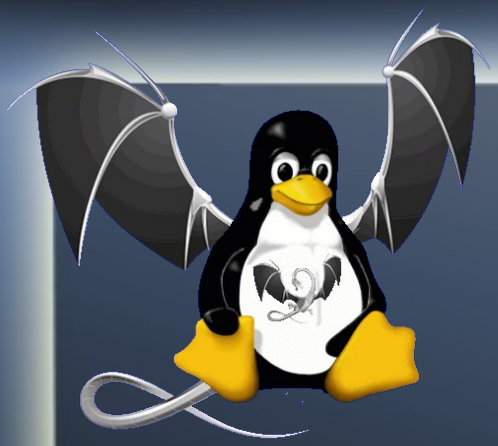
- Macro expansion
- Better error reporting
- Fix-it hints
- Address Sanitizer

The LLVMLinux Project



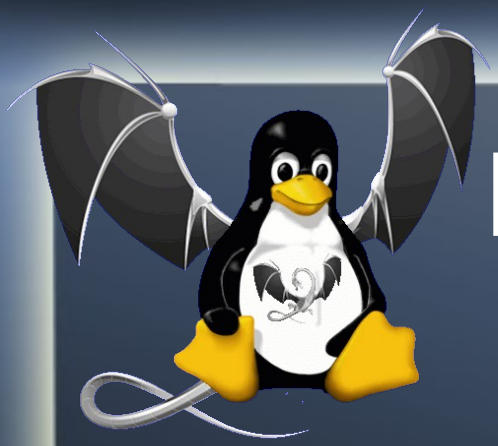
The LLVMProject Goals

- “Meta project” between Kernel and LLVM
- 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



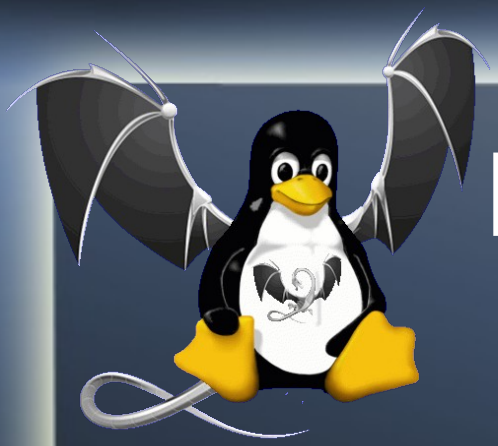
Project Website

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



LLVMLinux Automated Build Framework

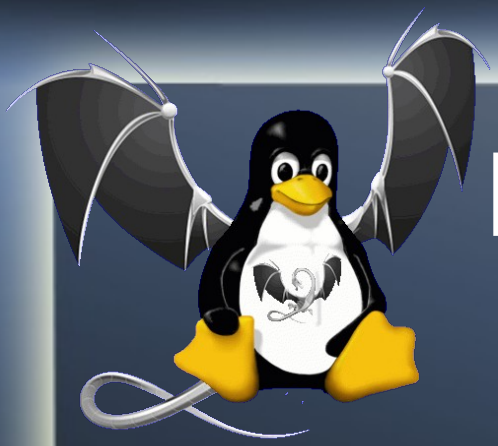
- git clone http://git.linuxfoundation.org/llvmlinux.git
- The framework consists of scripts and patches
- Automates fetching, patching, and building
  - LLVM, Clang,
  - Toolchains for cross assembler, linker
  - Linux Kernel
  - QEMU, and test images



LLVMLinux Automated Build Framework

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

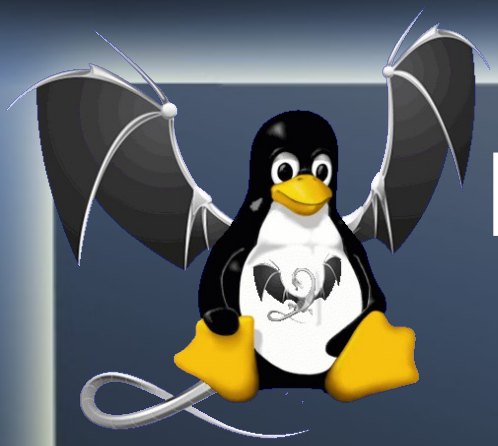
```
$ make CROSS_ARM_TOOLCHAIN=android kernel-gcc-build
```



LLVMLinux Automated Build Framework

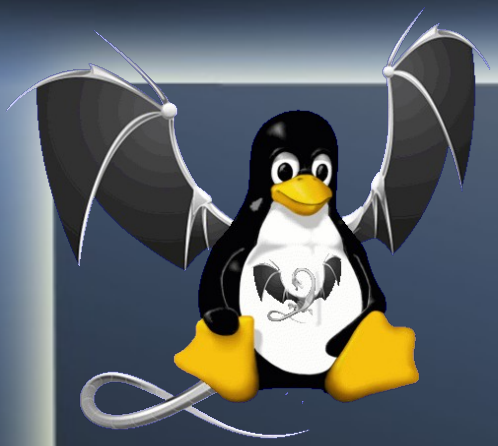
- Example make targets

```
$ cd targets/vexpress
$ make sync-all kernel-build test-boot-poweroff
$ make clean all
$ make llvm-clean clang-build
$ make list-patches-applied
$ make help
```



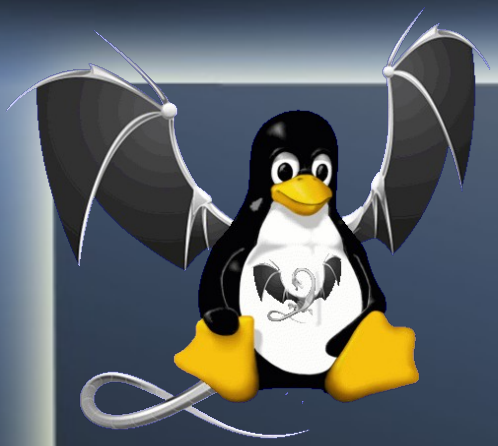
LLVMLinux Automated Build Framework

- 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)
  - BeagleBone (in progress for 3.7)



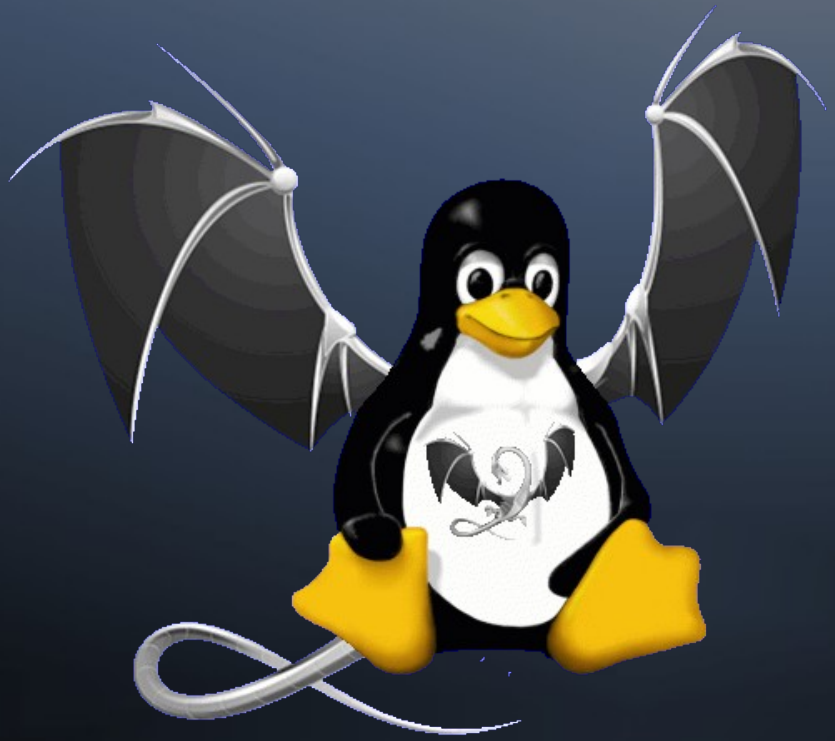
Buildbot

- 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
  - Kernel currently expects some undocumented GCC behavior
  - Unsupported GCC flags, built-in function behavior differences

Kbuild is GCC specific

- GCC returns false for unsupported flag and issues warning
- Clang returns true for unused flag and issues warning
- This means that special versions of things like cc-option macro need to be provided

Unsupported GCC Flags

-fconserve-stack

-fdelete-null-pointer-checks (Bug 9251)

-fno-inline-functions-called-once

-mno-thumb-interwork

- See 2012 LPC talk for more details:

http://www.linuxplumbersconf.org/2012/wp-content/uploads/2012/09/2012-LPC-LLVMLinux-bw2.odp

Unsupported GCC C Language Extentions

- Variable length arrays in structs (VLAIS)
  - A declaration like:

```
void f (int i) {
     struct foo_t {
         char a[i];
     } foo;
 }
```

    - cannot be compiled in Clang, though declarations like:

```
void f (int i) {
     char foo[i];
}
```

    - are perfectly acceptable.
  - Used in the iptables code, the kernel hashing (HMAC) routines, gadget driver, and possibly some other drivers

Nested Functions

- Thinkpad ACPI Driver uses Nested Functions

```
static void hotkey_compare_and_issue_event(
                                struct tp_nvram_state *oldn,
                                struct tp_nvram_state *newn,
                                const u32 event_mask)
{
…
        void issue_volchange(const unsigned int oldvol,
                             const unsigned int newvol)
…
        void issue_brightnesschange(const unsigned int oldbrt,
                                    const unsigned int newbrt)
…
```

- Patch submitted

Unsupported GCC C Language Extentions

- Explicit register variables not supported
  - X86

```
register unsigned long current_stack_pointer asm("esp") __used;
```

  - ARM

```
register unsigned long current_sp asm ("sp");
```

- Use of 'aligned' attribute in cast (Bug 11071)
  - Crypto/shash.c

```
Return len + (mask & ~(__alignof__(u8 __attribute__ ((aligned))) - 1));
                                                    ^~~~~~~
```

Incompatibilities with GCC

- Segment references
  - It has just been determined that certain attributes used for __init and __exit are being dropped by cpp stage in clang
  - This is a bug which still needs to be fixed
  - This should solve the “Merged global” issue

Incompatibilities with GCC

- Inline syntax handling
  - GNU89
- __builtin_constant_p() fails for Clang
  - (LLVM Bug 4898)
  - Include/linux/rcupdate.h

Status of Building Linux Kernel With Clang/LLVM

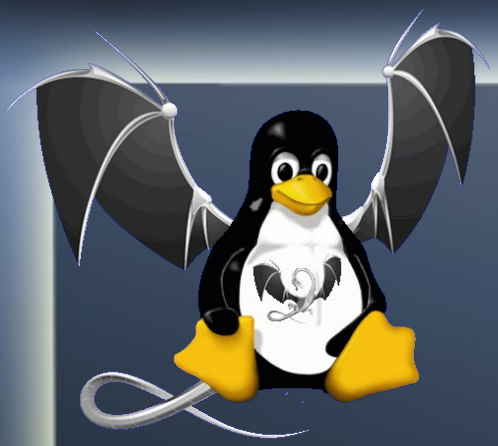
Linux Kernel Patches

- Kbuild support
- Explicit ASM to handle register variables
- Remove the use of VLAIS
- Segment linkage differences related to attributes
- “extern inline” in ARM ftrace.h (GNU89 vs GNU99)
- __builtin_constant_p() workaround
- GCC specific use of aligned attribute in cast

LLVM for Linux Status

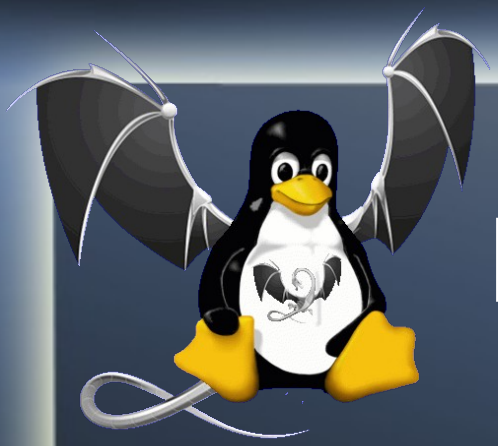
- Only 4 patches for Clang/LLVM (svn)
  - Specific to X86_64 target
- 64-bit type handling for ARM now upstream
- Clang IA not yet enabled by default
- Stripped attribute issue affecting linking needs to be tracked down
- Clang 3.3 will likely work mostly out-of-the-box for the Linux Kernel

What's Left to Do?



Todos

- Enabling Clang IA (Integrated Assembler)
- Upstream VLAIS patches
- Segment linkage difference fix
- Inline differences
- Proper fix for lcak of __builtin_constant_p()
- Getting Checker to work with the Kernel



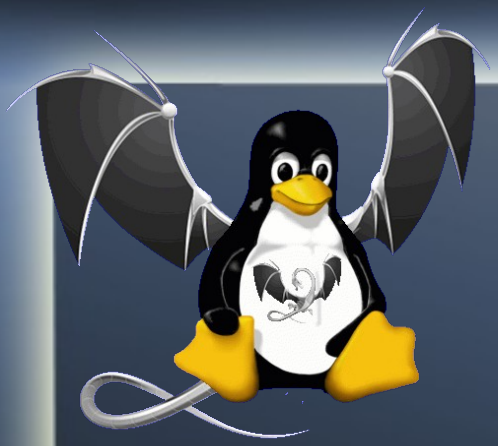
How Can I Help?

- Work on unsupported features and Bugs
- Review patches for Clang/LLVM and Kernel
- Help get patches upstream
- Submit new targets and arch support
- Try the code on your own HW
- Propose new test cases
- Report Bugs

Who wouldn't want a penguin 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/