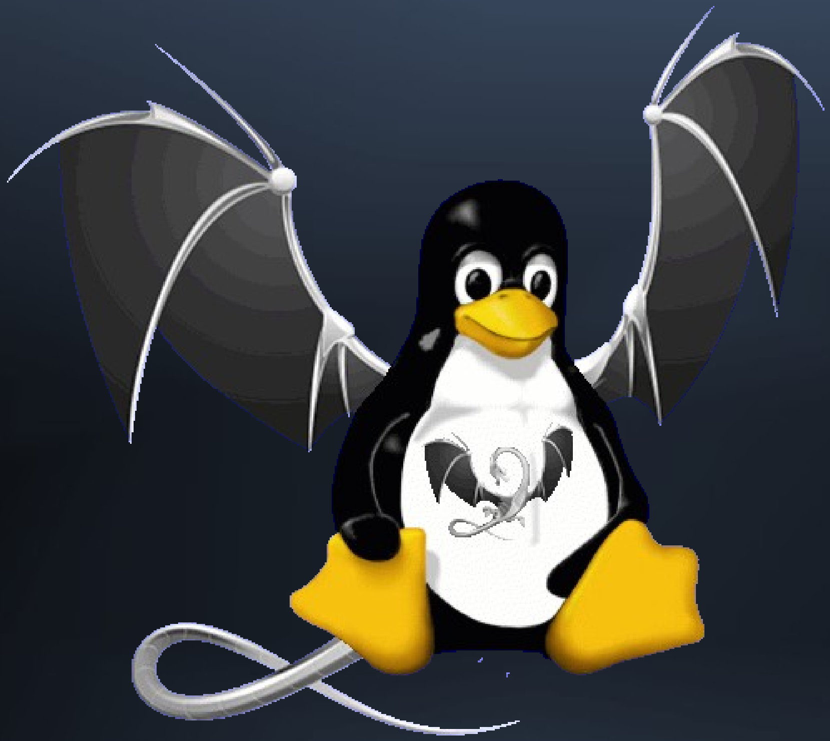
LLVMLinux: The Linux Kernel with Dragon Wings



Presented by:

Behan Webster

(LLVMLinux project lead)

Presentation Date: 2013.10.24

What is Clang/LLVM?

LLVM is a Toolchain Toolkit

- A modular set of libraries for building tools
  - Compiler, linker
  - Source code analysis tools
  - Meta data extraction from code
  - Code refactoring tools
  - Tight integration with IDEs

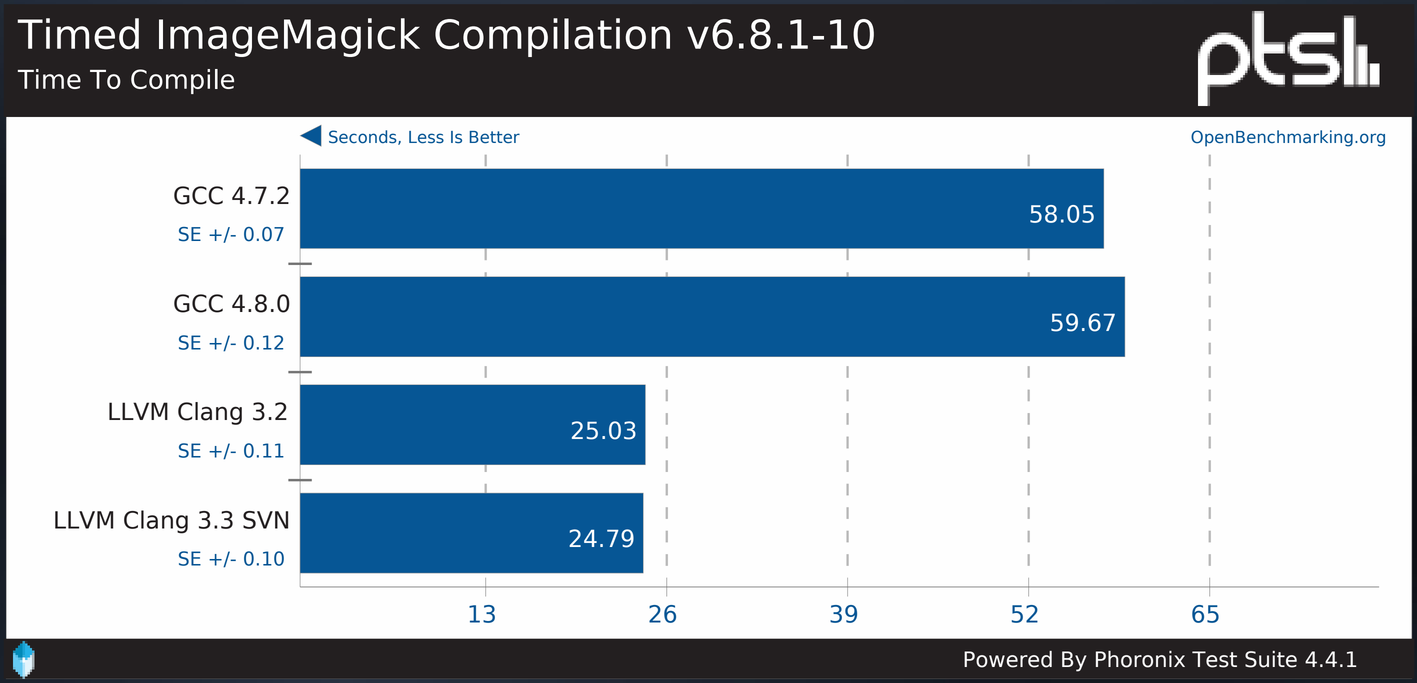
LLVM Toolchain Suite

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

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

Fast Compiles

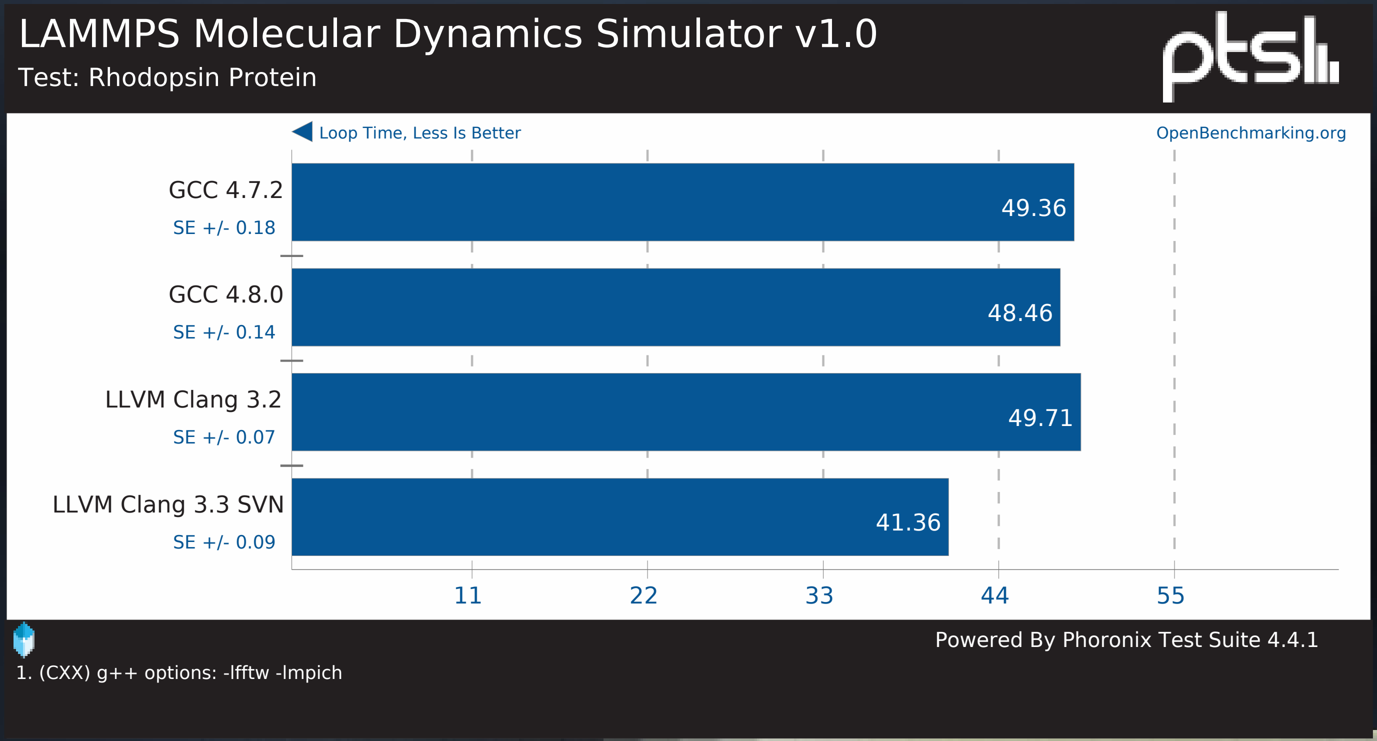
- Clang compiles code faster and use less memory than other toolchains



http://www.phoronix.com/scan.php?page=article&item=llvm_33svn_competes&num=1

Fast Moving Project

- In just a few years Clang has reached and in some cases surpassed what other toolchains can do



http://www.phoronix.com/scan.php?page=article&item=llvm_33svn_competes&num=1

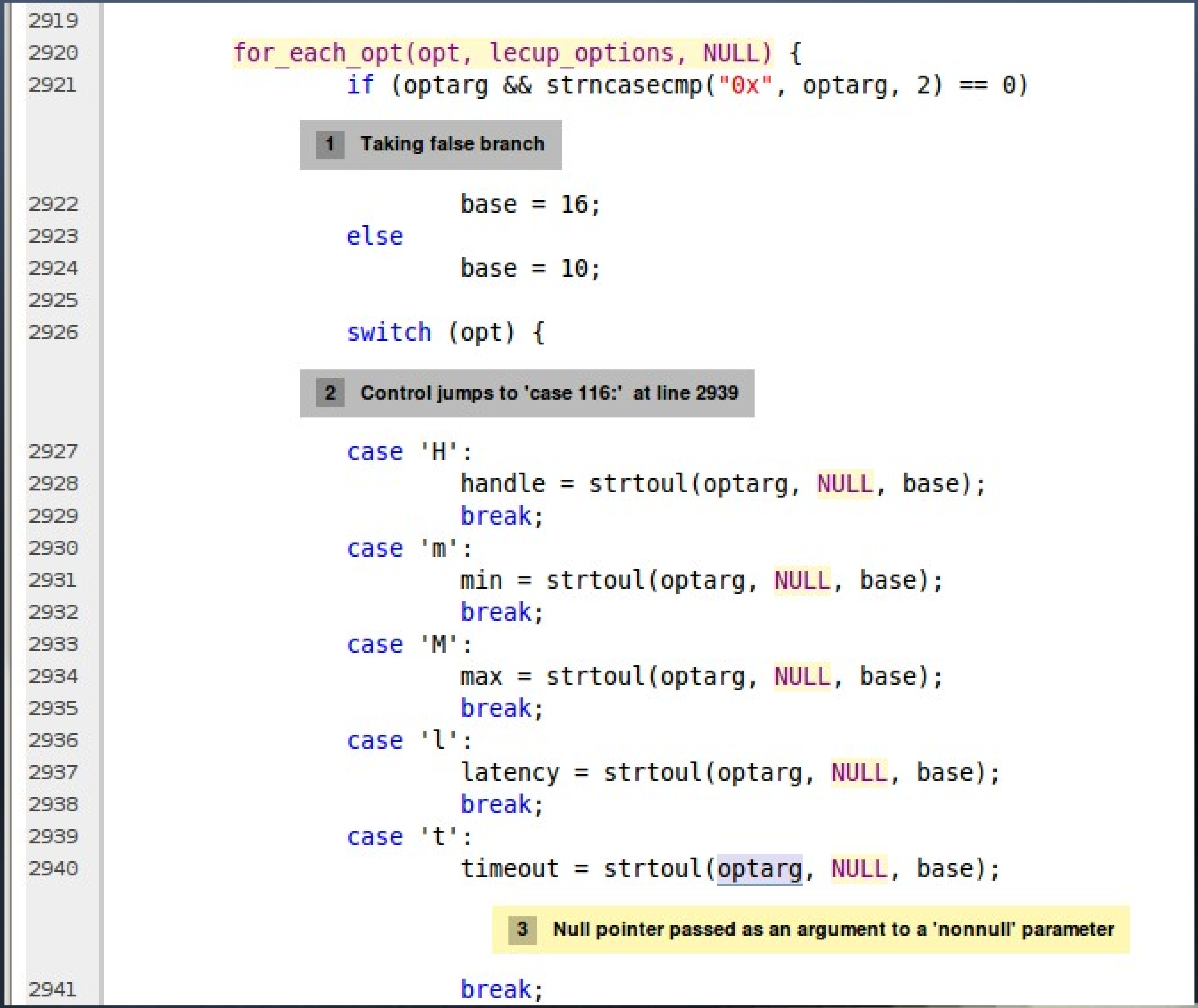
One Toolchain

- Compiler extensions only need to be written once
- LLVM is already being used in a lot of domains:
  - Audio
  - Video (llvmpipe)
  - CUDA
  - Renderscript
  - Kernel
  - Userspace
  - Applications
  - Documentation
  - HPC

LLVM License

- Licensed under the "UIUC" BSD-Style license
- Embeddable into many other projects
- Wide range of full-time developers building the LLVM project and derived technologies
- Wide development audience using LLVM

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

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

Security

Talking about Linux kernel security surrounding recent events involving the NSA...

"I also think this is a reason that having multiple independent compilers that are structurally very different (gcc/llvm) could give a potential security advantage. It's harder in practice to create a "rtt" attack that works simultaneously against two independently moving targets."

- Michael K Johnson

Other Kinds of Things

- Google is using a tool based on LLVM to look for common bugs in their vast library code
- Once bugs are found they are fixed automatically with minimal human involvement
  - http://youtu.be/mVbDzTM21BQ
- Conceivably something similar could be built to look for common bugs in the kernel code so that bugs could 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 used in Gallium3D
  - llvmpipe driver, Clover (Open CL)
  - GLSL shader optimizer
- Clang built Debian - Sylvestre Ledru

The LLVMLinux Project

The LLVMProject Goals

- Fully build the Linux kernel for multiple architectures, using the Clang/LLVM toolchain
- Discover LLVM/Kernel issues early and find fixes quickly across both communities
- Upstream patches to the Linux Kernel and LLVM projects
- Bring together like-minded developers

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 clang compiler
  - From-source, prebuilt, native
- Choice of gnu cross-toolchain (as, ld)
  - Codesourcery, Linaro, Android, native

```
$ cd targets/vexpress
$ make CLANG_TOOLCHAIN=prebuilt kernel-build
$ make CROSS_ARM_TOOLCHAIN=linaro kernel-build
```

LLVMLinux Automated Build Framework

- Current support for various targets
  - X86_64 (mainline)
  - Versatile Express (QEMU testing mainline)
  - Qualcomm MSM (3.4)
  - Raspberry-pi (3.2 and 3.6)
  - Nexus 7 (3.1.10), Galaxy S3 (3.0.59 in progress)
  - BeagleBone (3.8 in progress)
  - Arm64 (mainline in progress)

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

Status of Building Linux Kernel With Clang/LLVM

LLVM for Linux Status

- All required patches are now upstream
- The kernel can be compiled with Clang 3.3 (with the LLVMLinux kernel patches)
- Any new issues introduced to LLVM which break the Kernel are being fixed as they are being found by the LLVMLinux team with help from LLVM developers
- Some further fixes have made it into what will be released as 3.4

Challenges Using Clang/LLVM to Build the Linux Kernel

Challenges Using Clang for Cross Compilation

- GCC Dependencies:
  - gcc conforms to gnu90, clang to gnu99
  - Kernel currently expects some undocumented GCC behavior
  - Unsupported GCC extensions and flags
  - __builtin function 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
- Kbuild requires patches to support clang
- New in clang 3.4svn, follows gcc behaviour

Unsupported GCC Language Extentions

- Named register variables are not supported

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

Proposed by LLVMLinux project

- __builtin_stack_pointer()
- Arch independent, in line with existing __builtin_frame_pointer()
- Patch for LLVM available, looking to have a similar patch for gcc

Proposed by Jakob Stoklund Olesen (works with gcc and LLVM 3.3):

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

Unsupported GCC Language Extentions

- Variable Length Arrays In Structs (VLAIS) aren't supported in Clang (gcc extension)

```
struct foo_t {
    char a[n];/* Explicitly not allowed by C99/C11 */
    int b;
} foo;
```

- VLAs outside of structures are supported (gcc and llvm)

```
char foo[n];
```

- VLAIS is used in the Linux kernel in the netfilter code, the kernel hashing (HMAC) routines, gadget driver, and possibly other places

Nested Functions

- Thinkpad ACPI Driver still 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 (haven't heard back from the maintainer)

Incompatibilities with GCC

- __attribute ((alias)) is used for modules
- An alias doesn't copy over other attributes
- Since __section() isn't copied over, init and exit link sections need to be reapplied
- The various section mismatches reported during the build may be related to similar issues

Extern inline is different for gnu89 and gnu99

- GNU89
  - Function will be inlined where it is used
  - No function definition is emitted
  - A non-inlined function can also be provided
- GNU99 (C99)
  - Function will be inlined where it is used
  - An external function is emitted
  - No other function of the same name can be provided.
- Solution? Use “static inline” instead.

This code doesn't work in clang but does in gcc

```
--- a/crypto/shash.c
+++ b/crypto/shash.c
@@ -67,7 +67,8 @@ EXPORT_SYMBOL_GPL(crypto_shash_setkey);
 static inline unsigned int shash_align_buffer_size(unsigned len,
                                                    unsigned long mask)
 {
-       return len + (mask & ~(__alignof__(u8 __attribute__ ((aligned))) - 1));
+       typedef __attribute__ ((aligned)) u8 u8_aligned;
+       return len + (mask & ~(__alignof__(u8_aligned) - 1));
 }
```

- Clang has troubles with this statement as written
- Making it into 2 lines makes it more readable and works in both compilers

Challenges Using Clang for Cross Compilation

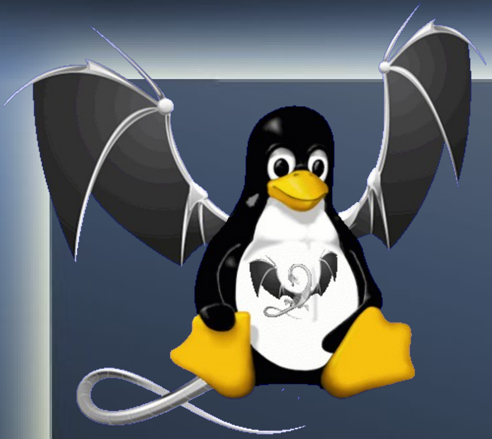
- The Integrated Assembler (IA) can't be used
  - Doesn't support .code16
  - ARM Kernel code isn't in Unified Format
- Dependence on GNU toolchain for assembly and linking (as and ld)
- Configuring GNU toolchain dependencies (-gcc-toolchain <path>)

Kernel Patches

- The patches that still need to make it upstream

Architecture	Number of patches
all	18
arm	11
aarch64	5
x86_64	8

What's Left to Do?



Todos

- Upstream patches
- Test and fix drivers/subsystems which haven't been tested yet or are known not to work

http://llvm.linuxfoundation.org/index.php/Broken_kernel_options

- Fix Segment mismatch and merged globals
- Enable Clang IA (i.e. rewriting ARM ASM in unified format)

How Can I Help?

- Make it known you want to be able to use Clang to compile the kernel
- Test LLVMLinux patches
- Report bugs to the mailing list
- Help get LLVMLinux patches upstream
- Work on unsupported features and Bugs
- Submit new targets and arch support
- Patches welcome

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/