LLVMLinux: The Linux Kernel with Dragon Wings

Presented by:

Behan Webster

(LLVMLinux project lead)

Presentation Date: 2013.04.16

What is Clang/LLVM?

LLVM is a Toolchain Toolkit

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

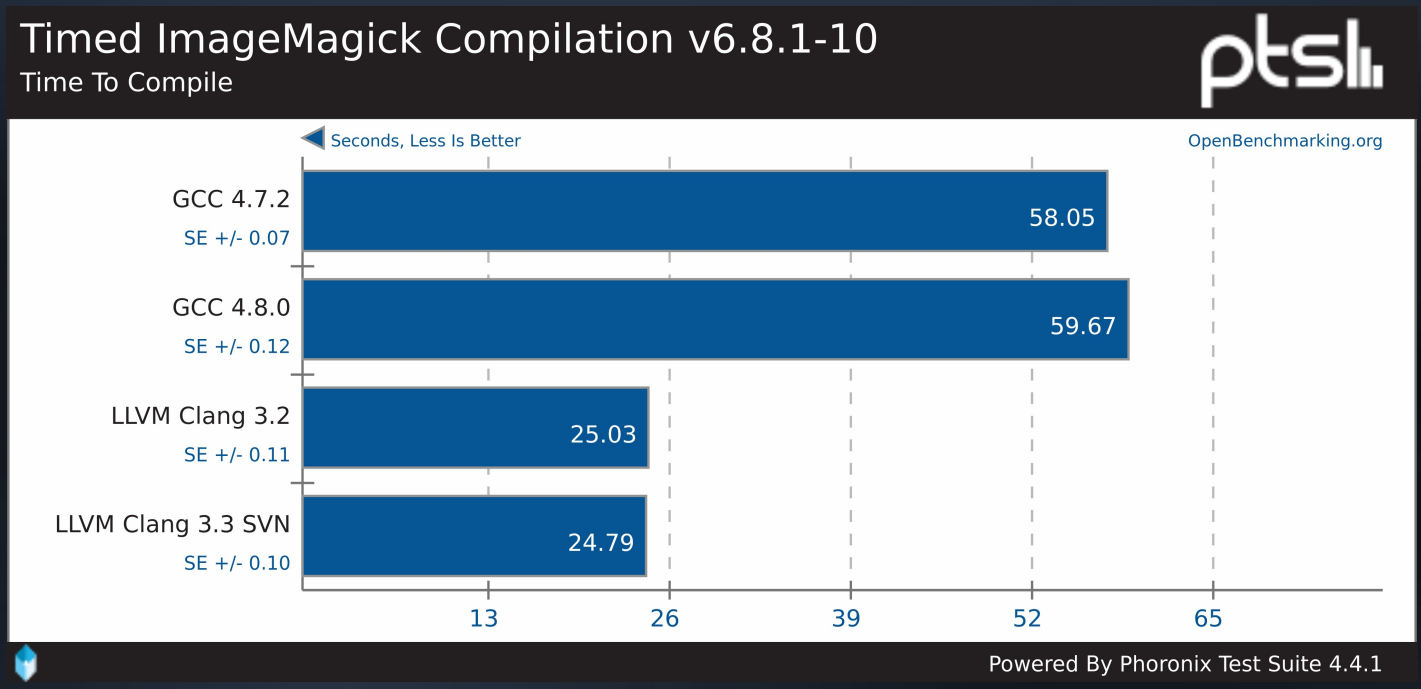
LLVM Toolchain Suite

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

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

Fast Compiles

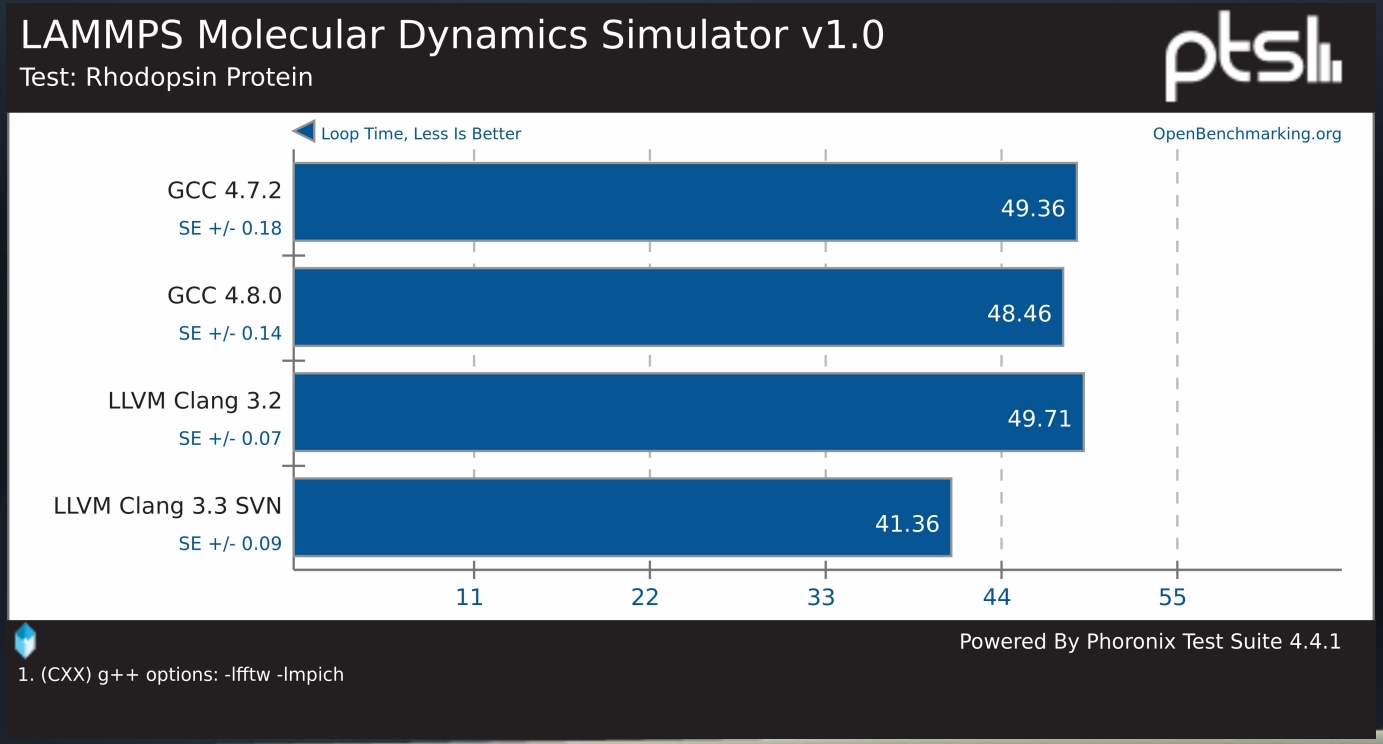
- Clang compiles code faster and use less memory than other toolchains



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

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

Other Kinds of Things

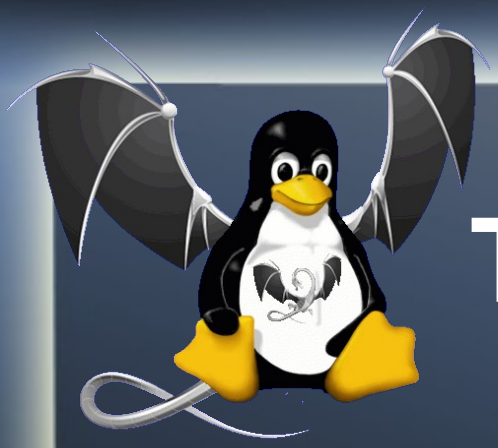
- Google is using LLVM to look for common bugs in their vast library of source code
- Once found bugs are found they are fixed automatically with minimal human involvement
  - http://youtu.be/mVbDzTM21BQ
- Checker could similarly be extended to look for common bugs in kernel code so that bugs could be 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

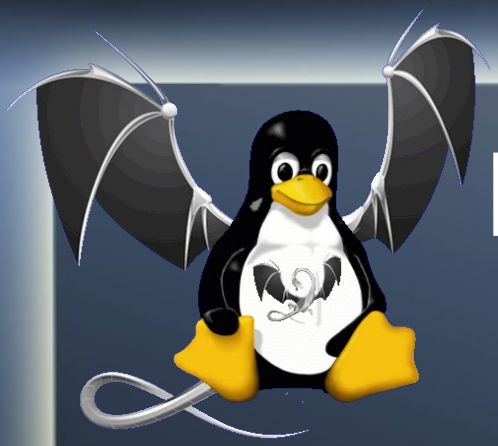
- Clang built Debian - Sylvestre Ledru

The LLVMLinux Project



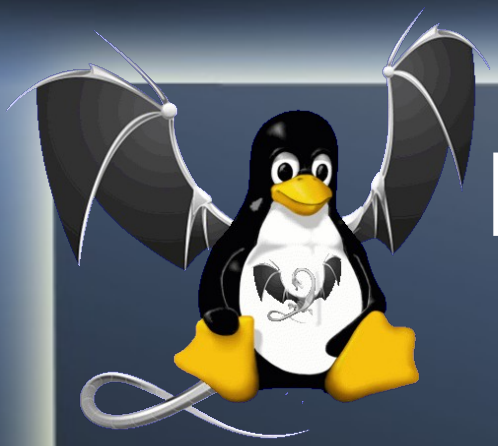
The LLVMProject Goals

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



LLVMLinux Automated Build Framework

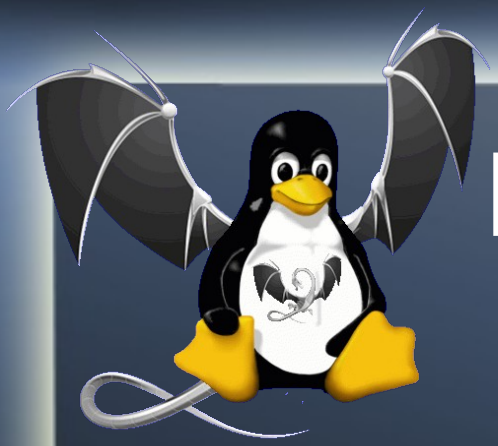
- git clone http://git.linuxfoundation.org/llvmlinux.git
- The framework consists of scripts and patches
- Automates fetching, patching, and building
  - LLVM, Clang,
  - Toolchains for cross assembler, linker
  - Linux Kernel
  - QEMU, and test images



LLVMLinux Automated Build Framework

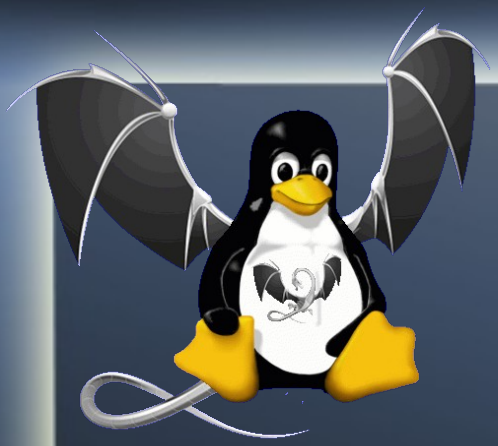
- Patch management using quilt
- Choice of cross- toolchain (as, ld)
  - Codesourcery (Default)
  - Linaro/Ubuntu
  - Android

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



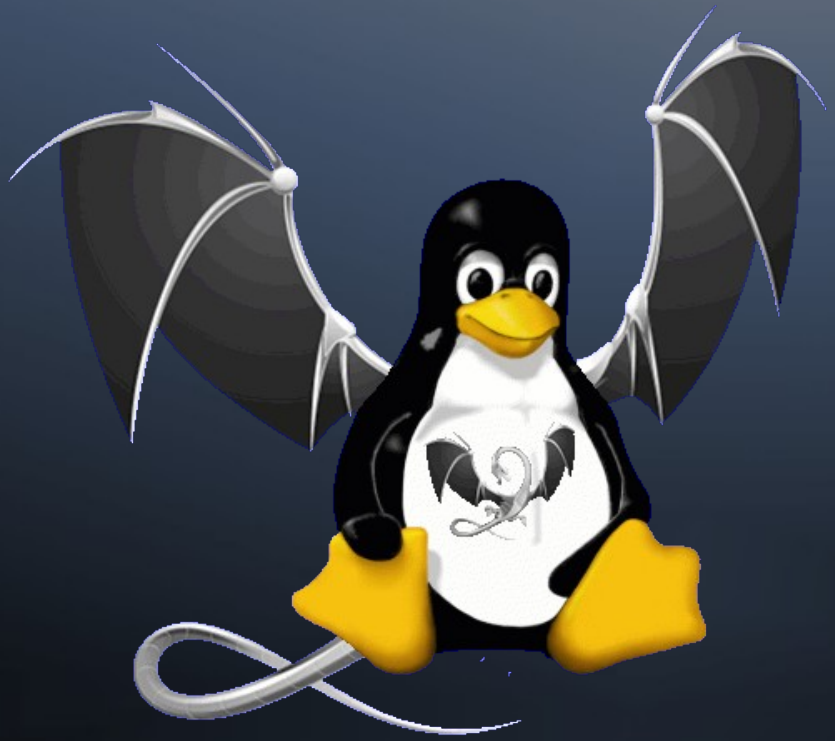
LLVMLinux Automated Build Framework

- Current support for various targets
  - Versatile Express (QEMU testing mainline)
  - X86_64 (mainline)
  - Qualcomm MSM (3.4)
  - Raspberry-pi (3.2 and 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



Challenges Using Clang/LLVM to Build the Linux Kernel

Challenges Using Clang for Cross Compilation

- The Integrated Assembler (IA) can't be used, 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 conforms to gnu89, clang to gnu99
  - Kernel currently expects some undocumented GCC behavior
  - Unsupported GCC 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

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 Language Extentions

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

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

- VLAs outside of structures however are supported

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

- VLAIS is used in the Linux kernel in the iptables 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

Unsupported GCC Language Extentions

- Explicit register variables are not supported
  - X86

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

  - ARM

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

Incompatibilities with GCC

- Segment reference mismatches
  - It has just been determined that certain attributes used for __init and __exit are being mishandled by cpp 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)
  - mm/slab.c

Status of Building Linux Kernel With Clang/LLVM

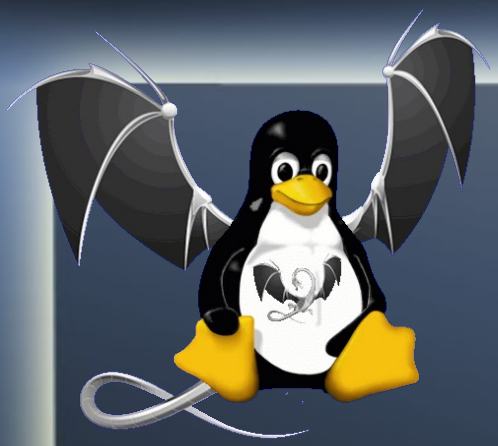
LLVM for Linux Status

- All required patches now upstream
- Clang IA disabled due to inline asm syntax
- Clang 3.3 will likely work mostly out-of-the-box for the Linux Kernel (with our kernel patches)
- Outstanding issue:
  - Stripped attribute issue affecting linking needs to be tracked down
  - SegFault compiling arch/arm/mm/context.c

Linux Kernel Patches

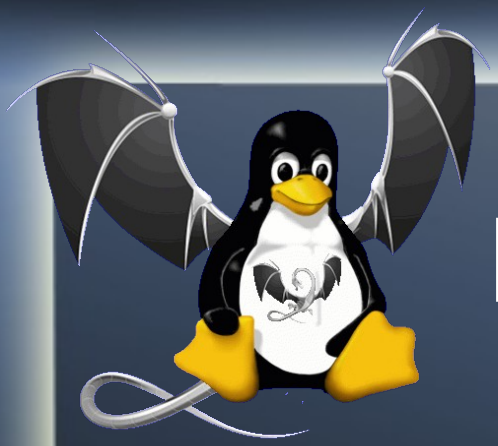
- Kbuild support
- Explicit ASM to handle register variables
- Remove the use of VLAIS
- Segment linkage mismatches related to attributes
- “extern inline” in ARM ftrace.h (GNU89 vs GNU99)
- __builtin_constant_p() workaround
- GCC specific use of aligned attribute in cast

What's Left to Do?



Todos

- Upstream VLAIS patches
- Segment linkage/merged globals fix
- Find solution for lack of __builtin_constant_p()
- Enabling Clang IA (Integrated Assembler)
- Getting Checker to work with the Kernel



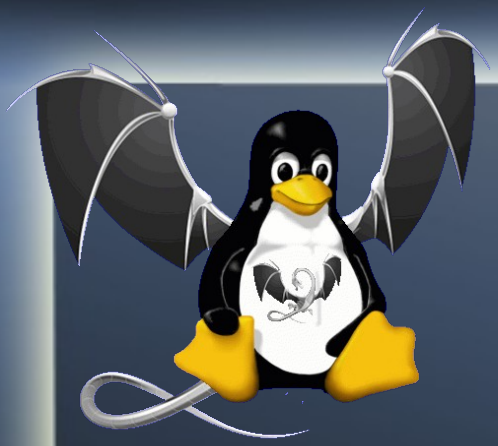
How Can I Help?

- Get involved
- Help get patches upstream
- Work on unsupported features and Bugs
- Submit new targets and arch support
- Test LLVMLinux patches
- Report bugs to the mailing list
- 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/