Compiling Linux with LLVM

Presented by: Mark Charlebois
Presentation Date: 06/07/2012
Agenda

Why would I want to use Clang to compile Linux?
Status updates:
  - cross compiling for ARM with Clang
  - building Linux kernel with Clang
  - running Linux compiled with Clang

To do list
Why Would I Want to Use Clang to Compile Linux?
Better Diagnostics

```bash
$ gcc-4.2 -fsyntax-only t.c
t.c:7: error: invalid operands to binary + (have 'int' and 'struct A')
```

```bash
$ clang -fsyntax-only t.c
t.c:7:39: error: invalid operands to binary expression ('int' and 'struct A')
return y + func(y ? ((SomeA.X + 40) + SomeA) / 42 + SomeA.X : SomeA.X);
~~~~~~~~~~~~~~ ^ ~~~~~
```

See http://clang.llvm.org/diagnostics.html for more examples

GCC extensions: all extensions are explicitly recognized as such and marked with extension diagnostics, which can be mapped to warnings, errors, or just ignored.

Google builds their products also with Clang just for the better debug output

Rich diagnostic output enables auto-generation of patches
Fix-it Hints

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

```bash
$ 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 =
```
Macro Expansion

$ gcc-4.2 -fsyntax-only t.c
t.c: In function 'test':
t.c:80: error: invalid operands to binary < (have 'struct mystruct' and 'float')

$ clang -fsyntax-only t.c
t.c:80:3: error: invalid operands to binary expression ('typeof(P)' (aka 'struct mystruct') and 'typeof(F)' (aka 'float'))
   X = MYMAX(P, F);
      ^~~~~~~~~~~
t.c:76:94: note: instantiated from:
#define MYMAX(A,B) __extension__ ({ __typeof__(A) __a = (A); __typeof__(B) __b = (B); __a < __b ? __b : __a; })
               ^ ~~~
for_each_opt(opt, lecup_options, NULL) {
    if (optarg && strncasecmp("0x", optarg, 2) == 0)
        base = 16;
    else
        base = 10;
    switch (opt) {
        case 'H':
            handle = strtol(optarg, NULL, base);
            break;
        case 'm':
            min = strtol(optarg, NULL, base);
            break;
        case 'M':
            max = strtol(optarg, NULL, base);
            break;
        case 'l':
            latency = strtol(optarg, NULL, base);
            break;
        case 't':
            timeout = strtol(optarg, NULL, base);
            break;
    }
}
Clang/LLVM use in Open Source OSes

Minix moved to Clang as default compiler

FreeBSD is working on ClangBSD
  Using LLVM and KLEE for automatic test generation
  http://wiki.freebsd.org/BuildingFreeBSDWithClang

LLVM is the basis of the Renderscript compiler in Android
  Supported on ARM, MIPS and x86

LLVM a hard dependancy for Gallium3D
  llvm-pipe driver

  Clover – OpenCL state tracker

  May be used for GLSL shader optimizer
Clang and Debian

Building Debian with Clang:
“…most of the issues are either difference in C standard supported, difference of interpretation or corner cases.”

“My personal opinion is that clang is now stable and good enough to rebuild most of the packages in the Debian archive, even if many of them will need minor tweaks to compile properly.”

“In the next few years, coupled with better static analysis tools, clang might replace gcc/g++ as the C/C++ compiler used by default in Linux and BSD distributions.”

“The clang developers are progressing very fast: 14.5% of the packages were failing with version 2.9 against 8.8% with version 3.0.”

Sylvestre Ledru: http://sylvestre.ledru.info/blog/sylvestre/2012/02/29/rebuild_of_the_debian_archive_with_clang
Status of Cross Compiling for ARM with Clang
Clang Parameters for Building ARM Linux User Space

Getting much simpler now:

```bash
export COMPILER_PATH=/opt/arm-2011.03
CC=clang -ccc-host-triple arm-none-linux-gnueabi \ 
   -ccc-gcc-name arm-none-linux-gnueabi-gcc \ 
   --sysroot=${COMPILER_PATH}/arm-none-linux-gnueabi/libc \ 
   -march=armv7-a -mfpu=neon
```

The default for arm-none-linux-gnueabi is armv4t

Using triple armv7-none-linux-gnueabi will not find the codesourcery compiler and default to the native assembler: /usr/bin/as

User specifies just a “configuration”:
   clang --config=arm-cortex-a9-baremetal foo.c
   clang --config=cortex-m4-my-toaster morning-food.c

Under the hood this entry point (the universal driver) would have access to all the information that the driver, compiler, and other tools need to build applications for that target.

Status?
ELLCC - http://ellcc.org/

The primary emphasis of the ELLCC project is to create an easy to use multi-target cross compilation environment for embedded systems [based on Clang and LLVM]. Multi-target support: ARM, i386, Microblaze, Mips, Nios2[2], PowerPC, PowerPC64, Sparc[1] and X86_64
Challenges Using Clang to Build Linux Kernel
Challenges Using Clang for Cross Compilation

Cross compilation with Clang
  Not a supported configuration
  Dependence on GNU cross toolchain for assembly and linking
    Configuring GNU toolchain dependencies
  Finding the right triplet

Lots of warnings
  Must set –Wno-unused-value (otherwise slows compilation)

Clang/LLVM Bugs

GCC Dependencies:
  Clang C99 vs GCC GNU89
    Kernel expects some undocumented GCC behavior
    Unsupported GCC flags, built-in function behavior differences
Unsupported GCC Behavior Expected by Linux Kernel

scripts/Kbuild.include are gcc specific
  cc-option tests fail for gcc, pass erroneously for clang

Clang warning is for unused, not unsupported
  No way to check supported options in Clang

  http://clang.llvm.org/docs/DriverInternals.html#int_unused_warnings

  cc-option = $(call try-run,\
                 $(CC) $(KBUILD_CPPFLAGS) $(KBUILD_CFLAGS) $(1) -c -xc /dev/null -o "$$TMP",$(1),$2))

GCC returns false for unsupported flag and issues warning:
  cc1: error: unrecognized command line option "-fno-delete-pointer-checks"

Clang returns true for unused flag and issues warning:
  clang: warning: argument unused during compilation: '-fno-delete-pointer-checks'

  See LLVM/Clang bug 9701 – only helps with warnings, not flags
Unsupported GCC Flags

-fconserve-stack
   Attempt to minimize stack usage. The compiler will attempt to use less stack space, even if that makes the program slower. This option implies setting the large-stack-frame parameter to 100 and the large-stack-frame-growth parameter to 400.

-fdelete-null-pointer-checks (Bug 9251)
   Assume that programs cannot safely dereference null pointers, and that no code or data element resides there. This enables simple constant folding optimizations at all optimization levels. In addition, other optimization passes in GCC use this flag to control global dataflow analyses that eliminate useless checks for null pointers; these assume that if a pointer is checked after it has already been dereferenced, it cannot be null.

-fno-inline-functions-called-once
   Suppresses inlining of subprograms local to the unit and called once from within it, which is enabled if -O1 is used.

Unsupported GCC C Language Extensions

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

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

cannot be compiled in Clang, though declarations like:

```c
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.
Unsupported GCC C Language Extensions

Explicit register variables not supported
   register unsigned long current_sp asm ("sp");

Nested functions
   Only used in a thinkpad driver

Use of 'aligned' attribute in cast (Bug 11071)
   Crypto/shash.c
      Return len + (mask & ~(__alignof__(u8 __attribute__((aligned))) - 1));

GCC allows EXPORT_SYMBOL of inlined functions
   Linux kernel bugs
      Patches submitted upstream by Greg KH
Incompatibilities with GCC

Warnings for unused return values
   Thousands of instances in kernel, must use \texttt{\-Wno-unused-value}

   Re-enabled with \texttt{W=1}

Segment references
   More \texttt{\_\_refdata, \_\_initdata, \_\_exitdata} attributes required

   Investigate differences in linking and segments

Inline syntax handling
   GNU89

\texttt{\_\_builtin\_constant\_p()} fails for Clang (Bug 4898)
   \texttt{Include/linux/rcupdate.h}
ARM Specific Clang/LLVM Bugs or Missing Features

-mabi-linux not properly supported on ARM (Bug 11326)
  Causes incorrect structure member offsets

64 bit type parameter passing (Bug 11753)
  Must use register pairs

  Hack used at Linux Foundation site, no upstream fix

ARM paired register GNU inline assembly syntax
  Hack used at Linux Foundation site, no upstream fix

Clang Integrated Assembler (IA) not enabled for ARM (incomplete)
ARM Specific Clang Configuration Issues

Using triple arm-none-eabi, or triple arm-none-linux-gnueabi generates undefined reference to __aeabi_*

Must define __aeabi_memset, and __aeabi_memcpy

Note: be careful with args for __aeabi_memset!!!

Compiler-rt will not cross compile for ARM

Using triple “arm” or “armv7” will build kernel, but:
arch/arm/kernel/unwind.c:

  warning: Your compiler does not have EABI support.
  warning: ARM unwind is known to compile only with EABI compilers.
  warning: Change compiler or disable ARM_UNWIND option.

Non-EABI kernel hangs at boot

__kernel_size_t vs size_t and posix functions
Status of Building Linux Kernel With Clang
Wiki and git Repository at llvm.linuxfoundation.org

Links to known LLVM bugs, organized by architecture, place to aggregate information about building Linux with Clang

Automated Build Framework
   Documented in Wiki

   Git repository

   Current support for:
      » ARM Cortex A9 (Versatile Express)
      » Qualcomm MSM
   Easy to add new arch/platforms

Anyone welcome to participate, would especially like to see x86 and MIPS support

Patches organized by common, arch, subarch/board
   Easy to add new architectures and platforms
Automated Build Framework

Automated build to simplify fetching and building Clang, QEmu, and initrd
Automates fetching, patching and building the Linux kernel

Git repository of build scripts and patches
http://git.linuxfoundation.org/llvm-setup.git

Proper build dependencies for Clang, kernel, QEMU and initramfs

Patches organized as:
- General
- Arch specific
- SoC family or board specific

Tracks which patches apply and which do not

Python tools for managing and maintaining patches

All build targets can be listed
Make list-targets
Common/Arch Independent Status

Only 2 required Clang/LLVM patches
Required for missing ARM functionality

One optional patch:
Error on unsupported warnings (GCC compatibility)

Linux Kernel patches for
Explicit register variables
VLAIS (not for IP tables yet)
Segment linkage differences
  Additional __refdata needed for some drivers
return_address and extern inline in ftrace.h
__builtin_constant_p() workaround
GCC specific use of aligned attribute in cast
IP Tables use of VLAIS

net/ipv4/netfilter/ip_tables.c
net/ipv4/netfilter/../../netfilter/xt_repldata.h

#define xt_alloc_initial_table(type, typ2) ({
    unsigned int hook_mask = info->valid_hooks;
    unsigned int nhooks = hweight32(hook_mask);
    unsigned int bytes = 0, hooknum = 0, i = 0;
    struct {
        struct type##_replace repl;
        struct type##_standard entries[nhooks];
        struct type##_error term;
    } *tbl = kzalloc(sizeof(*tbl), GFP_KERNEL);

    /* Today's hack: quantum tunneling in structs 'entries' and 'term' are never anywhere referenced by word in code. In fact, they serve as the hanging-off data accessed through repl.data[]. */

    /* */
Status of Building the ARM Linux Kernel With Clang

ARM Versatile Express
  Compiles (3.4 kernel)
  Supports crypto, ext4, SD card, initramfs
  Boots and runs multiple initramfs images, or SD card image under QEMU
  Builtbot status at http://88.198.35.80:8765/waterfall

Qualcomm MSM
  Compiles with Clang/LLVM patches
  More __refdata fixes
  Have not yet tested on HW

Clang IA not yet enabled for ARM by default

Tracing is not yet enabled
Specific ARM Issues

Unsupported flags
-mlittle-endian

Everything assumes little-endian byte order

-mno-thumb-interwork

-mshort-load-bytes

Broken flags
-mabi=aapcs-linux

Creates struct member offset issues

Replace

register unsigned long current_sp asm ("sp");

asm ("mov %0, r13" : "=r" (current_sp));
Unsupported ARM Flags

-mno-thumb-interwork
Generate code that supports calling between the ARM and Thumb instruction sets. Without this option, on pre-v5 architectures, the two instruction sets cannot be reliably used inside one program. The default is -mno-thumb-interwork, since slightly larger code is generated when -mthumb-interwork is specified. In AAPCS configurations this option is meaningless.

-mlittle-endian
Generate code for a processor running in little-endian mode. This is the default for all standard configurations

-mshort-load-bytes
deprecated alias for -malignment-traps.

-malignment-traps
This option is ignored when compiling for ARM architecture 4 or later, since these processors have instructions to directly access half-word objects in memory.

http://gcc.gnu.org/onlinedocs/gcc/ARM-Options.html
TODO

Check status of other issues reported by Bryce Lelbach
- mregparm
  - fc-all-saved-reg
  - pg and mcount
  - fno-optimize-sibling-calls

Status of Clang IA (Integrated Assembler)
Test gadget driver and crypto VLAIS patches
Segment linkage differences
Inline differences
Try building LTP with LLVM and create virtual SD card FS
Fix cc-option issues and other GCC specific dependencies
Call for Participation

Others are welcome to participate at the LLVM work at Linux Foundation
Wishlist:
  Integrating the patches from lll-linux tree for x86/x86_64
  X86 QEMU test target
  MIPS support and QEMU test target
  LTP integration
  Unit tests for known LLVM Linux bugs
Thank You
Disclaimer

Nothing in these materials is an offer to sell any of the components or devices referenced herein. Certain components for use in the U.S. are available only through licensed suppliers. Some components are not available for use in the U.S.