Rebuilding Debian with LLVM/Clang

Sylvestre Ledru – sylvestre@debian.org
Who am I?

- Involved in Debian for 7 years
- Maintainer of the LLVM toolchain in Debian/Ubuntu for the last 3 years
- LLVM, Clang and LLDB committer
- Co-organizer of Euro LLVM 2013
Current status in Debian & Ubuntu:

All C, C++, Objective-C sources are being built with gcc for all supported Debian arches and Kernel.

gcc is THE FLOSS compiler for the last 25 years
Used for (pretty much) everywhere or anything
Why a new compiler in Debian?
As we were able to do with decoupling Linux from Debian with kFreeBSD and the HURD, we are aiming to decouple gcc in Debian.
• Other compilers can find programming errors that gcc could not find

• Code built by many compilers is more likely to be more strictly correct and more portable then code only built with gcc

• Some compilers can have advantages on some archs (ex : clang on ARM)

• A big ecosystem is being built over LLVM/Clang
Make Debian compiler agnostic
Sylvestre Ledru
About performances:
Results presented by Google last April at the Euro LLVM conference

clang compared to gcc

Servers: 1.04
Image processing: 1.03
Video codecs: 1.00

Agregate of various internal benchmarks of Google

Source: http://www.irill.org/videos/euro-llvm-2013/carruth-hires
Libraries

OpenSSL: 1.00
Protocol Buffer: 1.12
Snappy: 1.05

Source: http://www.irill.org/videos/euro-llvm-2013/carruth-hires
Some advantages (bis)
More intelligent detections

```c
int main() {
    unsigned int i = 0;
    return i < 0;
}
```

$ gcc -Wall -Werror foo.c ; echo $?  
0

$ clang -Werror foo.c  
foo.c:3:17: error: comparison of unsigned expression < 0 is always false  
[-Werror,-Wtautological-compare]

    return i < 0;

      ~ ^ ~

1 error generated.
Rebuild of Debian using Clang
Crappy method:

```
VERSIONS="4.8 4.7 4.6"
cd /usr/bin
for VERSION in $VERSIONS; do
  rm g++-$VERSION gcc-$VERSION cpp-$VERSION
  ln -s clang++ g++-$VERSION
  ln -s clang gcc-$VERSION
  ln -s clang cpp-$VERSION
done

CC=clang CXX=clang++ dpkg-buildpackage fails to use clang in too many cases
```
Testing the rebuild of the package under amd64.

NOT the performances (build time or execution) nor the execution of the binaries
By Sylvestre Ledru (Debian, IRILL, Scilab Enterprises). February 28th 2012

Presentation

This document presents the result of the rebuild of the Debian archive with clang.

clang is now ready to build software for production (either for C, C++ or Objective-C) and integrating more than the core suite while not needing

Done on the cloud-qa - EC2 (Amazon cloud)

Thanks to Lucas Nussbaum and David Suarez
Percentage of failure using clang instead of gcc

<table>
<thead>
<tr>
<th>Version</th>
<th>Pkg/Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.9</td>
<td>16398</td>
</tr>
<tr>
<td></td>
<td>2372</td>
</tr>
<tr>
<td>3.0</td>
<td>15658</td>
</tr>
<tr>
<td></td>
<td>1381</td>
</tr>
<tr>
<td>3.1</td>
<td>17710</td>
</tr>
<tr>
<td></td>
<td>2137</td>
</tr>
<tr>
<td>3.2</td>
<td>18264</td>
</tr>
<tr>
<td></td>
<td>2204</td>
</tr>
<tr>
<td>3.3</td>
<td>18854</td>
</tr>
<tr>
<td></td>
<td>2188</td>
</tr>
</tbody>
</table>
Why these differences between 3.0 vs 3.1/3.2/3.3?
Some information about **-Wall & -Werror**:

**-Wall** enables many warnings (and different set from gcc)

**-Werror** transforms Warning to Error

```c
int main() {
    unsigned int i = 0;
    return i < 0;
}
```

```bash
$ gcc -Wall -Werror foo.c && echo $?  
0

$ clang -Wall -Werror foo.c && echo $?  
foo.c:3:14: error: comparison of unsigned expression < 0 is always false [-Werror,-Wtautological-compare]
    return i < 0;
       ^ ~
1 error generated.
```
Security check introduced in clang 3.1
36 occurrences

#include <stdio.h>

void foo(void) {
    char buffer[1024];
    sprintf(buffer, "%n", 2);
}

$ gcc -Werror -c foo.c && echo $?  
0
$ clang -Werror -c foo.c && echo $?  
foo.c:5:23: error: use of ' %n ' in format string discouraged  
(potentially insecure)  [-Werror,-Wformat-security]  
sprintf(buffer, " %n ", 2);  
^  
1 error generated.
Some of the most common errors
Unsupported options
49 occurrences

$ gcc -O9 foo.c && echo $?
0

$ clang -O9 foo.c

**error**: invalid value '9' in '-O9'

Record by libdbi-drivers with -O20 \o/
Different default behavior
120 occurrences

```c
int foo(void) {
    return;
}
```

$ gcc -c noreturn.c; echo $? 0

# -Wall shows it as warning

$ clang -c noreturn.c

`noreturn.c:2:2: error: non-void function 'foo' should return a value [-Wreturn-type]
return;
^`

1 error generated.
Different default behavior (bis)
16 occurrences

$ gcc -c returninvoid.c; echo $?  
returninvoid.c: In function ‘foo’:
returninvoid.c:2:2: warning: ‘return’ with a value, in function returning void [enabled by default]
0  
$ clang -c returninvoid.c  
returninvoid.c:2:2: error: void function 'foo' should not return a value  
[-Wreturn-type]
    return 42;
    ^    ~~

1 error generated.
void foo() {
  void bar() {
    
  }
}

$ gcc -Wall -c foo.c; echo $?
0

$ clang -c foo.c

foo.c:3:17: error: function definition is not allowed here
  
  void bar() {
    ^

foo.c:5:7: error: expected ';' at end of declaration
    }
    ^

...
gcc extensions which won't be supported
21 occurrences

– foo.cpp –

```cpp
#include <vector>

void foo() {
    int N=2;
    std::vector<int> best[2][N];
}
```

$ g++ -c foo.cpp; echo $?
0

$ clang++ -c foo.cpp
foo.cpp:4:29: error: variable length array of non-POD element type
    std::vector<int> best[2][N];
    ^
1 error generated.
Last rebuilds proved that clang is now ready

Remaining problems are upstream
How to make about 2000 upstreams fix their issues?
Provide them with an interface which shows clang and other build

Working on debile aka Debuild.me (by Paul Tagliamonte) as part of Léo Cavaillé's Google Summer of Code
This experimental infrastructure aims to provide a generic rebuild platform. Normal build, custom builds (clang based) or static analyzers (coccinelle, scan-build, etc) are managed through this infrastructure.

By package

By maintainer

Perhaps you're looking for the [last uploads](#)

**Active Jobs**

<table>
<thead>
<tr>
<th>Type</th>
<th>Arch</th>
<th>Suite</th>
<th>Assigned</th>
<th>Builder</th>
<th>Source Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>build</td>
<td>amd64</td>
<td>unstable</td>
<td>12 minutes ago</td>
<td>irill4-builder1</td>
<td>openmpi</td>
</tr>
</tbody>
</table>

**Builder Status**

<table>
<thead>
<tr>
<th>Name</th>
<th>Last ping</th>
<th>Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>irill4-builder1</td>
<td>12 minutes ago</td>
<td>build</td>
</tr>
<tr>
<td>irill4-builder2</td>
<td>a day ago</td>
<td></td>
</tr>
</tbody>
</table>
Package info
Name: eztrace
Version: 0.9.1-2
Maintainer: Samuel Thibault <sthibault@debian.org>
Uploaded by: Fred the autobuilder

source jobs
Launched on the source package provided

<table>
<thead>
<tr>
<th>Type</th>
<th>Version</th>
<th>Machine</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>build/clang-3.3 (amd64)</td>
<td>0.9.1-2</td>
<td>irill4-builder1</td>
<td>Failed</td>
</tr>
<tr>
<td>build/gcc-4.8 (amd64)</td>
<td>0.9.1-2</td>
<td>irill4-builder1</td>
<td>Uploaded</td>
</tr>
<tr>
<td>clanganalyzer (all)</td>
<td>0.9.1-2</td>
<td>irill4-builder1</td>
<td>X Errors found</td>
</tr>
<tr>
<td>coccinelle (all)</td>
<td>0.9.1-2</td>
<td>irill4-builder1</td>
<td>✓ Nothing found</td>
</tr>
<tr>
<td>cppcheck (all)</td>
<td>0.9.1-2</td>
<td>irill4-builder1</td>
<td>X Errors found</td>
</tr>
<tr>
<td>lintian (all)</td>
<td>0.9.1-2</td>
<td>irill4-builder1</td>
<td>✓ Nothing found</td>
</tr>
</tbody>
</table>
Provides various workers:
• Normal builds
• scan-build
• Lintian
• Coccinelle
• Cppcheck
• Clang
• ...

With repository of clang-built packages
Ask gcc people to add more default warnings to fix the `-Wall -Werror` issue?
Other ideas? (besides writing and forwarding upstream 2000 patches)
Any questions? Remarks?