Debian and Clang
Linux Kernel and Clang

Sylvestre Ledru – sylvestre@debian.org

Current status:
All C, C++, Objective-C sources are being built with gcc for all supported Debian arches (~13) and Kernel (3).

Clang ?

A C, C++ and Objective-C compiler

Rebuild of Debian using Clang

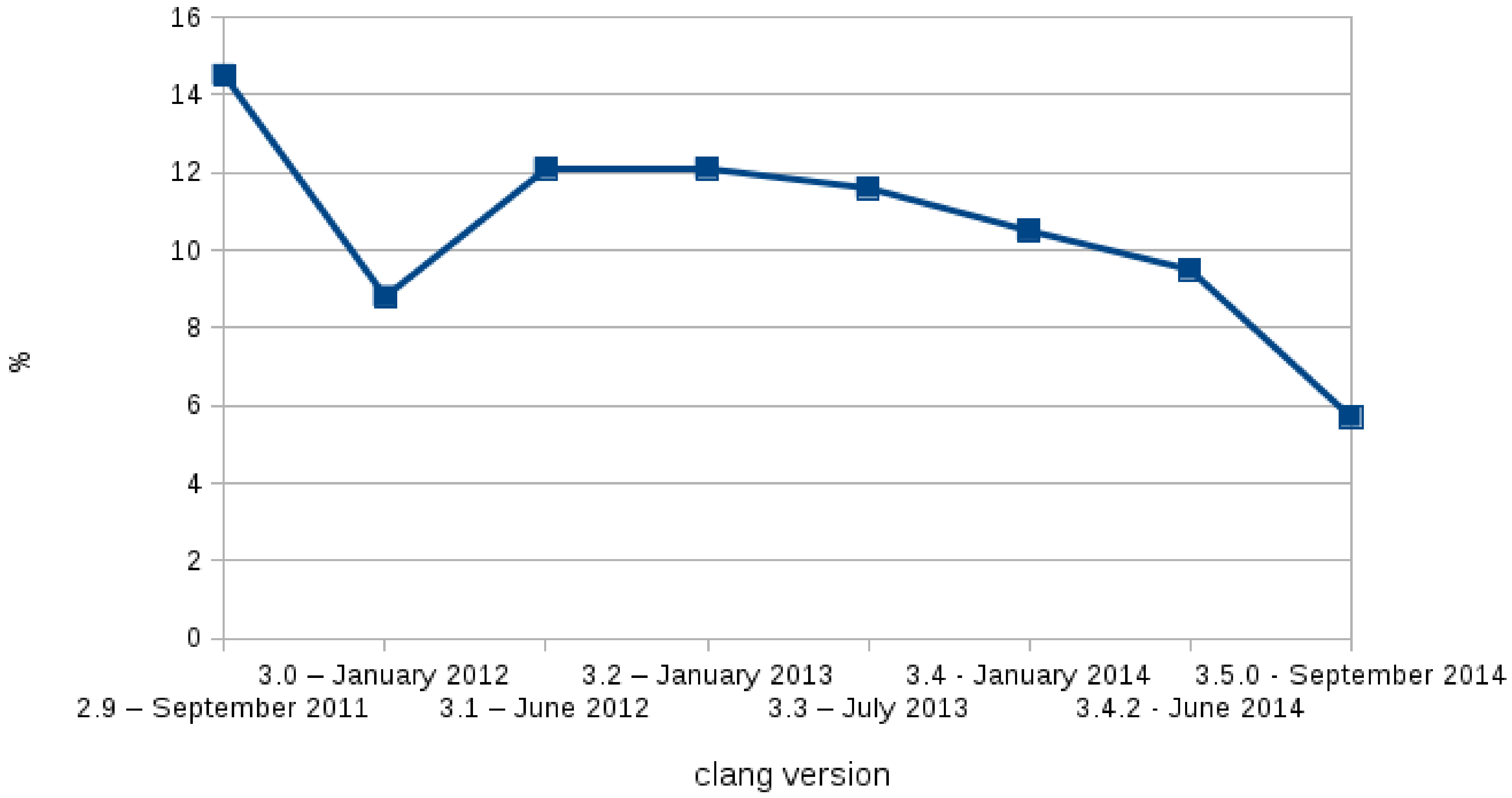
Crappy method:

```
VERSION=4.9
cd /usr/bin
rm g++-$VERSION gcc-$VERSION cpp-$VERSION
ln -s clang++ g++-$VERSION
ln -s clang gcc-$VERSION
ln -s clang cpp-$VERSION
cd -
```

Testing the rebuild of the package under amd64.

NOT the performances (build time or execution) nor the execution of the binaries

Percentage of failures using clang instead of gcc



Causes of failure ?

Various C/C++ issues:

- Variable length array (gcc extension)
- Inner functions
- Clang is C99 by default
- Gcc accept invalid C++ code (default arguments, scope, etc)
- ...

-*Wall* enables many warnings

-Werror transforms Warning to Error

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

```
$ 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.
```

How we have been able to fix bugs?

We used different paths

1. Fix upstream bugs

Different default behavior
133 occurrences

– noreturn.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.
```

Other kind of errors fixed:

- Wrong main declaration

```
int main(void);
int main(int argc, char *argv[]);
int main(int argc, char **argv);
int main(int argc, char **argv, char **envp);
```

- *Void function should not return a value*

```
void foo() {
        return 1;
}
```

- *Variable length array for a non POD (plain old data) element*

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

- Missing symbols at link time (inline in C99)

```
inline void xrealloc() { } // should be static inline void xrealloc()
int main(){
    xrealloc();
    return 1;
}
```

- ...

Results :

- 295 patches reported (Credits : Arthur Marble and Alexander Ovchinnikov, Debian GSoC)
- 90 bug fixed (ie uploaded in the Debian archive with the fix)

- Switch to Clang by FreeBSD & Mac OS X probably helped too

2. Hack into Clang

Unsupported options
50 occurrences

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

$ clang -O9 foo.c
error: invalid value '9' in '-O9'
```

Record by libdbi-drivers with -O20 \o/

=> We transformed that into a warning.

```
$ clang -O20 -c foo.c
warning: optimization level '-O20' is unsupported; using '-O3' instead
1 warning generated.
```

Unsupported args ~145 occurrences

```
$ clang-3.4 -c -fno-defer-pop  /tmp/foo.c
clang: error: unknown argument: '-fno-defer-pop'
$ clang-3.5 -c -fno-defer-pop  /tmp/foo.c
clang: warning: optimization flag '-fno-defer-pop' is not supported
clang: warning: argument unused during compilation: '-fno-defer-pop'
-------------------------------
$ clang-3.4 -c -finput-charset=UTF-8  /tmp/foo.c
clang: error: unknown argument: '-finput-charset=UTF-8'
$ clang-3.5 -c -finput-charset=UTF-8  /tmp/foo.c
```

Unsupported args ~145 occurrences

Gcc : unknown arguments are forwarded to the linker

Clang : unknown arguments trigger errors. Forward to the linker has to be explict

```
$ clang-3.4  -z lazy  /tmp/foo.c
clang: error: unknown argument: '-z'
clang: error: no such file or directory: 'lazy'

$ clang-3.5  -z lazy  /tmp/foo.c
```

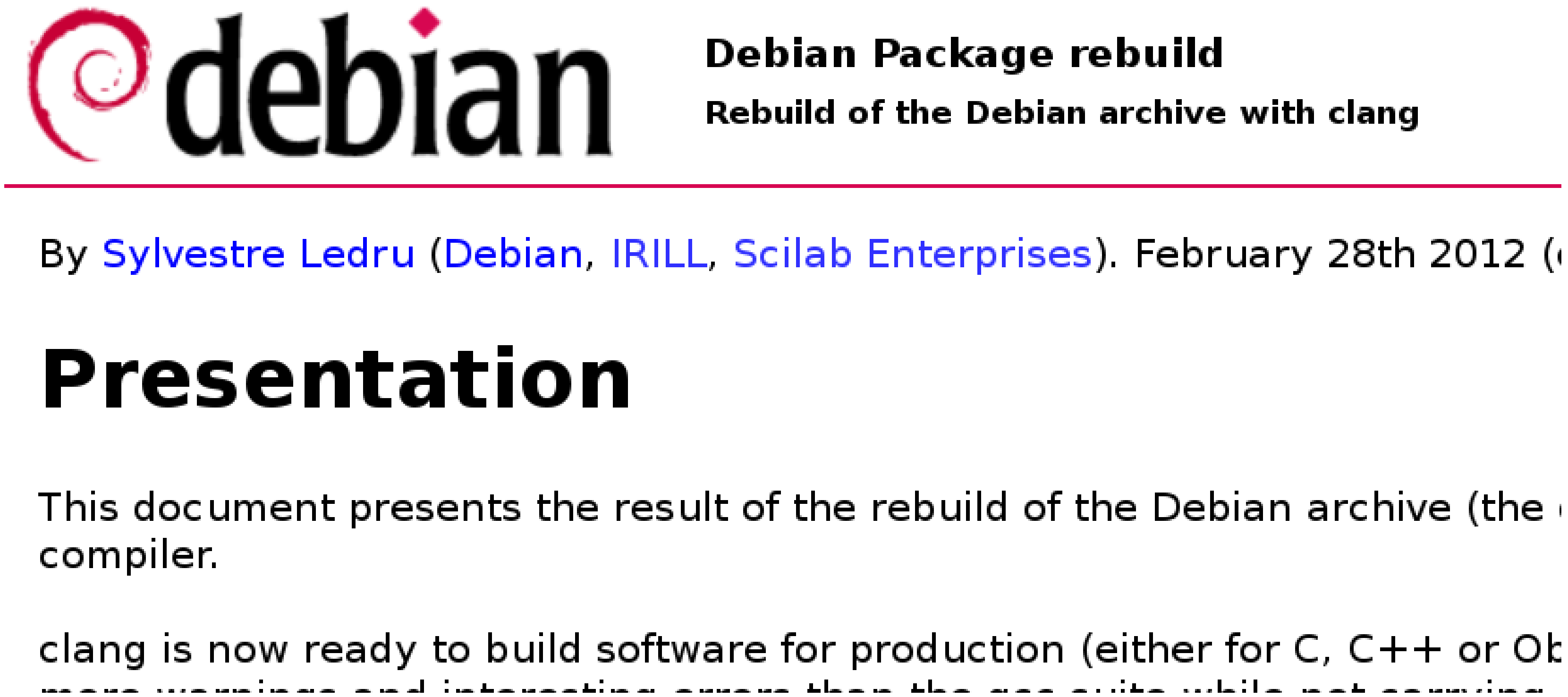
3. Hack into gcc

Well, trying to...

Last rebuild proved that clang is now ready

Remaining problems are upstream

Full results published:
http://clang.debian.net/



Debian Package rebuild

Rebuild of the Debian archive with clang

By Sylvestre Ledru (Debian, IRILL, Scilab Enterprises). February 28th 2012 (

Presentation

This document presents the result of the rebuild of the Debian archive (the
compiler.

clang is now ready to build software for production (either for C, C++ or Ob
more warnings and interesting errors than the gcc suite while not carrying

Done on the cloud-qa - EC2 (Amazon cloud)
Thanks to Lucas Nussbaum & David Suarez

Now, what about the Linux Kernel?

The LLVMLinux project

Same approach :

- Hack the kernel sources code
- Update clang to support some black magic

69 patches maintained on top of the kernel tree
Example : build system, Variable Length Arrays in Structs (vlais), nested functions, etc

http://git.linuxfoundation.org/llvmlinux.git/
arch/*/patches

Status:

- Working kernel on some archs (arm, arm64, amd64, x86, etc)
- Still need to upstream a bunch of patches

Thanks !