



Using Clang for fun and profit

Examples from the Chromium project

Hans Wennborg

hwennborg (at) google.com

GOTO Aarhus 2013

Outline

Chromium

- Background
- Numbers

Clang

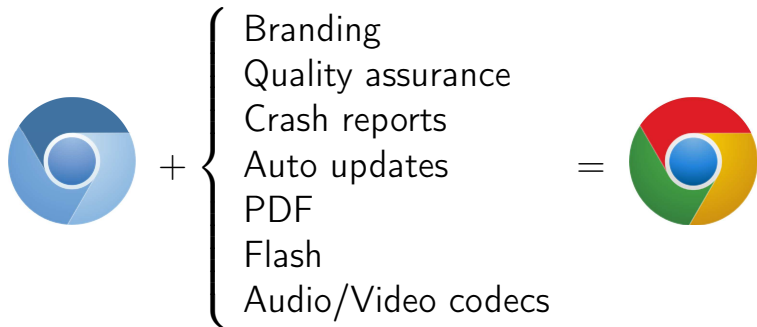
- Background
- Error Messages
- Warnings
- Tools

Conclusion

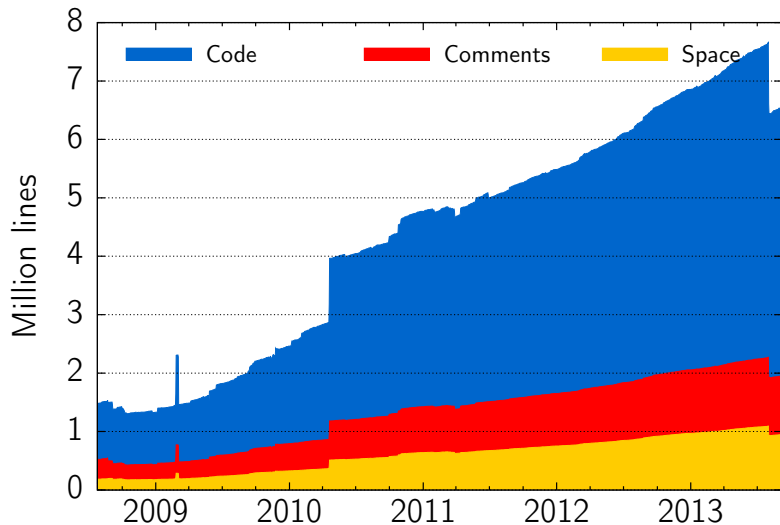
Introducing Chrome

- ▶ A web browser from Google
- ▶ First released 2008
- ▶ Pushing the web forward
- ▶ 750 M active users (May 2013)
- ▶ Windows, Mac, Linux, ChromeOS, Android, iOS
- ▶ Mostly open source.

Chromium vs. Chrome



The Code

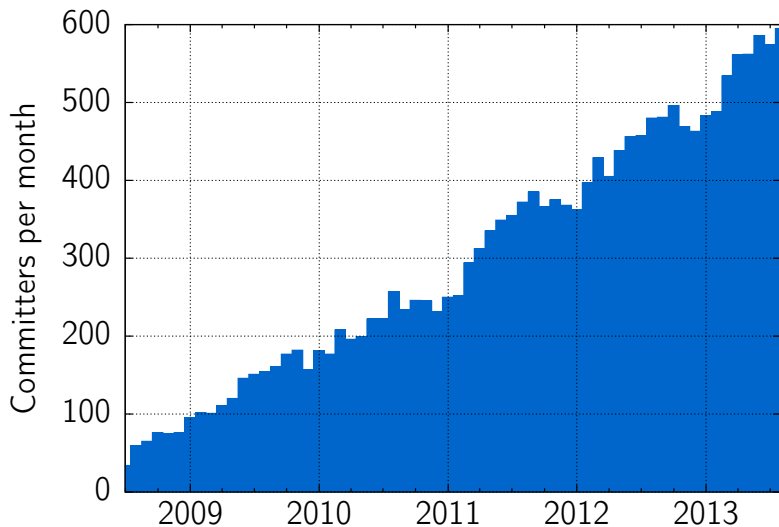


The Code (continued)

5 M lines more in third-party libraries:

- | | | |
|-------------|-------------|----------|
| ▶ Blink | ▶ gtest | ▶ lss |
| ▶ V8 | ▶ hunspell | ▶ NaCl |
| ▶ angle | ▶ icu | ▶ nss |
| ▶ breakpad | ▶ jsoncpp | ▶ sfntly |
| ▶ ffmpeg | ▶ leveldb | ▶ skia |
| ▶ flac | ▶ libjingle | ▶ webrtc |
| ▶ gmock | ▶ libsrtp | ▶ ... |
| ▶ googleurl | ▶ libvpx | |

The People



Clang

- ▶ Up-and-coming C++, C, Objective-C compiler
- ▶ Part of the LLVM project
- ▶ Announced 2007 by Apple
- ▶ Production quality compiler since ca 2010
- ▶ Open-source, BSD-style license
- ▶ Designed to be GCC compatible.

Clang in Chromium

- ▶ Used since 2010
- ▶ Work on 20% time by three Googlers
- ▶ Used for Mac binaries, and more.

What makes Clang interesting?

- ▶ Fast
- ▶ Good output
- ▶ Clear error messages
- ▶ High quality warnings
- ▶ Hackable and extendable.

The missing semicolon (GCC 4.6)

```
int f(int x) {  
    int s = 0  
    for (int i = 0; i < x; ++i)  
        s += i;  
    return s;  
}
```

```
a.cc: In function 'int f(int)':  
a.cc:3:9: error: expected ',' or ';' before 'for'  
a.cc:3:25: error: 'i' was not declared in this scope  
a.cc:3:35: error: expected ';' before ')' token
```

The missing semicolon (Clang)

```
int f(int x) {  
    int s = 0  
    for (int i = 0; i < x; ++i)  
        s += i;  
    return s;  
}
```

a.cc:2:18: **error:** expected ';' at end of declaration

```
    int s = 0  
           ^  
           ;
```

1 error generated.

The missing semicolon (GCC 4.9 trunk)

a.cc: In function 'int f(int)':

a.cc:3:9: **error:** expected ',', or ';' before 'for'
for (int i = 0; i < x; ++i)

^

a.cc:3:25: **error:** 'i' was not declared in this scope
for (int i = 0; i < x; ++i)

^

Typo correction

```
a.cc:5:9: error: use of undeclared identifier 'dout';  
      did you mean 'cout'?  
      dout << "Hello, world!" << endl;  
      ~~~~  
      cout
```

The Conditional Operator

```
int f(bool b, int x, int y) {  
    return 7 + b ? x : y;  
}
```

The Conditional Operator

```
a.cc:2:16: warning: operator '?:' has lower precedence
        than '+'; '+' will be evaluated first
    return 7 + b ? x : y;
           ~~~~~ ^
```

```
a.cc:2:16: note: place parentheses around the '?:'
        expression to evaluate it first
    return 7 + b ? x : y;
           ^
```

```
        (      )
```

```
1 warning generated.
```


The Boolean Accident

```
void f(bool *delete_user_data) {  
    if (!migrate_data_to_new_location()) {  
        delete_user_data = false;  
        return;  
    }  
  
    ...  
}
```

The Boolean Accident

```
a.cc:5:36: warning: initialization of pointer of type  
      'bool *' to null from a constant boolean expression  
      delete_user_data = false;  
                        ~~~~~
```

Hackability

- ▶ Clang is extendable
- ▶ It is a set of libraries
- ▶ We can use it to build tools.

Chromium Style Checker

- ▶ Mark functions `virtual` explicitly
- ▶ Use `override` for overriding functions
- ▶ ...

Chromium Style Checker

```
a.cc:8:5: warning: [chromium-style] Overriding  
        method must have "virtual" keyword.
```

```
void f();
```

```
^
```

```
virtual
```

```
1 warnings generated.
```

clang-format

- ▶ Formatting is important
- ▶ Formatting is boring
- ▶ Automatic formatting of C++ is hard.

AddressSanitizer (ASan)

- ▶ Fast memory error detector for C, C++, etc.
- ▶ Compile-time instrumentation
- ▶ Typically ca 2x slow-down
- ▶ Catches many kinds of errors.

ThreadSanitizer (TSan)

- ▶ Data race detector for C, C++, etc.
- ▶ Uses Clang/LLVM to insert compile-time instrumentation
- ▶ Overhead is large but not crazy
- ▶ Points out racy situations.

Undefined Behavior Sanitizer (UBSan)

- ▶ C and C++ have a thing called undefined behaviour
- ▶ Division by zero, array overflow, NULL ptr deref, etc.
- ▶ Helps efficient language implementation
- ▶ Also source of subtle bugs
- ▶ UBSan tries to catch those bugs for you.



Fun and Profit.

References

- ▶ www.chromium.org
- ▶ clang.llvm.org
- ▶ blog.llvm.org/2013/09/clang-warnings.html
- ▶ code.google.com/p/chromium/wiki/WritingClangPlugins
- ▶ clang.llvm.org/docs/ClangFormat.html
- ▶ code.google.com/p/address-sanitizer/