BERN - CH 2014
CONFERENCE

Compiler and Me

- Stephan Bergmann
- Red Hat, Inc.

LibreOffice

The compiler wants to be your friend, not your enemy

Compiler and Me

- C++ compilers have become way better over the last decade:
- Better C++ standards
- Better error messages
- Better static diagnostics
  - Compiler warnings
  - Plugin interfaces
  - Stand-alone tools
- Better dynamic diagnostics
  - -fsanitize
  - Stand-alone tools

Static feats

SAL_OVERRIDE

- A success story:
- Deep and broad class hierarchies, overloaded functions along the way
  - Nobody dares change any function signatures
- C++11 override feature
- Most compilers understand it (mostly)
- Clang plugin to add it in and enforce its use
- Much more confidence now when changing a function signature

Clang Plugins

- Clang has a plugin interface with a rather flat learning curve
  - Not 100% stable, but OK in practice (compat.hxx)
- Integrated into LO build system
  - Just drop a .cxx file into compilerplugins/clang/
- ~20 plugins:
  - ensure SAL_WARN(“area”, ...) consistency
  - sal_Bool ▶ bool; bad sal_Bool vs. int mixture
  - f(OUString) ▶ f(OUString const &)
  - ...
- Great work by Luboš and Noel
- Write a plugin yourself, today!

Clang Rewriting Plugins

- Instead of just generating a warning/error: automatically fix the code
- A bit tricky in the face of macros
- Different modes to only rewrite .cxx, or also .hxx
- Can even run multiple rewriters in parallel
- Was used to add SAL_OVERRIDE, convert sal_Bool to bool
- Plugins still useful after doing the mass rewrite, to warn about errors in new code

Coccinelle

- Coccinelle is a cool way to specify code rewrites as patches:

```
– return (E);
+ return E;
```

- Unfortunately more suitable for C than C++ (for now?)

Stand-Alone Static Analyzers

- Various tools with different approaches
  - Some overlap, but also differences in what they find
- Cppcheck (Julien)
- Clang Static Analyzer
- Coverity Scan (Caolán, Norbert)
  - No quick cycles, closed source

- Clean up also all the “harmless” warnings to make newly introduced ones stick out
  - Comparable to the original -Werror efforts

C++11/14

- C++11, C++14 (“bugfix release”)
  - GCC, Clang, (MSVC) catching up aggressively
- Bump requirements for LO 4.4 to make use of C++11:
  - CentOS devtools for TDF Linux baseline builds
  - MSVC support is still poor, though
    - No deleted functions
    - No variadic templates
      - wrongly claimed at wiki.apache.org/stdcxx/C++0xCompilerSupport
    - No virtual **inline** void f() **override** { ... }
  - Keep URE interface at C++03 for external clients?
- But make no mistake, C++ still a baroque pile of gotchas
  - “Effective Modern C++” by Scott Meyers to the rescue

Dynamic feats

Dynamic Sanitizers

- Recent Clang and GCC have -fsanitize=* feature
  - Instruments the code at compile time to find issues at runtime
  - More targeted and faster than valgrind
- -fsanitize=address:
  - out-of-bounds array access
  - heap use-after-free
  - stack use-after-return
  - leak detection
- “make check” clean (detect_leaks=0)

Dynamic Sanitizers

- -fsanitize=undefined: detect lots of different sorts of undefined behavior
  - signed integer overflow; negative double to unsigned
  - calling function pointers of wrong types
  - downcasts to wrong types
- Work in progress to clean all CppunitTests:
  - ~150 done, ~30 to go

- Issues with RTTI visibility (SAL_DLLPUBLIC_RTTI)
- Issues with Clang and DSOs having undef __asan/ubsan_* symbol references
  - JunitTest > stock java executable > libjpipe.so > libsal.so

BERN - CH 2014 CONFERENCE

“He is still on the go, his effort unceasing.

We must imagine him happy.”

–Jonathan Kandell, after Albert Camus

LibreOffice