Fürstenberg Systementwicklung

Clang Compiler Warnings

Lars-Christian
Fürstenberg
2014-03-18

What are warnings?

- gcc warning definition:
  `Warnings are diagnostic messages that report constructions that are not inherently erroneous but that are risky or suggest there may have been an error.´
- erroneous syntactic constructions are flagged as errors by the compiler
- syntactically correct, but inherently problematic constructions can be found by setting appropriate compiler warnings
- warnings indicate, that there is a problem with your code and you should understand the cause of the problem and fix it
- down side: false-positives are annoying
- fact is: a lot of developers actually ignore warnings

Typical warning settings

- typically developers set the following warning flag groupings:
  -Wall -Wextra -Wpedantic
- this sounds very ample
- unfortunately it isn’t
- you get a false feeling of safety
- finding good warning flags isn't an easy task
- different compilers and even different versions of the same compiler have different warning flag groupings

Example code

- Apples `goto fail´ disaster

```
...
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
==>     goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

      err = sslRawVerify(ctx, ...);
      if(err) {
            sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                      "returned %d\n", (int)err);
            goto fail;
      }

fail:
    return err;
}
```

- the only way to find this problem is using the ***-Wunreachable-code*** flag (not in ***-Wall*** nor ***-Wextra***)
- if Apple had used **better** warning flags, they would have **avoided** the 'goto fail' **disaster**!

Clang warning flags

- clang warning flags are not well documented
- flag groupings (like ***-Wextra***) change very often
- clang warnings do not equal exactly gcc warnings (there are even a couple of NOP gcc warnings in clang)
- some warning control options
  - ***-Werror=foo*** => set foo as error
  - ***-Wno-error*** => disable foo as error
  - ***-Wfoo*** => enable warning foo
  - ***-Wno-foo*** => disable warning foo
  - ***-Weverything*** => enable really all warnings (do it just for fun ;-)
- use ***#pragma clang diagnostic*** to manipulate warnings from code

Recommended warning flags (1)

- ***-Werror*** => treat all warnings as errors
- ***-Wall*** => warning group that contains warnings, that are easy to avoid
- ***-Wextra*** => warning group that contains warnings, that are harder to avoid
- ***-Wpedantic*** => tests for strict ISO conformity
- ***-Wshadow*** => local construct that shadows a previous one
- ***-Wheader-hygiene*** => warning group for things like ***using namespace*** declarations in a header
- ***-Wcast-align*** => warns when pointer casts increase the alignment of the target
- ***-Wconversion*** => warns if implicit conversions may alter a value (signedness, smaller)

Recommended warning flags (2)

- ***-Wfloat-equal*** => warns if floats are compared using equality
- ***-Wformat=2*** => checks printf and scanf formats
- ***-Wmissing-declarations*** => finds global functions that are not declared in header files
- ***-Wmissing-prototypes*** => the same as above , but for c
- ***-Woverlength-strings*** => warns for over length string constants
- ***-Wunreachable-code*** => warns for dead code, that is not reachable by any code path
- ***-Wno-unused-parameter*** => ***-Wunused-parameter*** is part of the ***-Wextra*** grouping, but too annoying for me, so I disabled it

Summary

- start your projects with a good warning set, as setting a tighter set in a running project is a nightmare
- treat warnings as errors
- eliminate every warning and understand the cause of the problem
- the more warning flags you set, the better
- check your warning flags when you update your compiler
- **and again: don’t ignore warnings, dumb!**

Resources

- http://clang.llvm.org/docs/UsersManual.html
- http://blog.llvm.org/2013/09/clang-warnings.html
- http://programmers.stackexchange.com/questions/122608/clang-warning-flags-for-objective-c-development (see the answer from Chandler Carruth, clang developer)
- http://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html