redhat.

Cursing Compilers

Stephan Bergmann

October 2017

- <sberg> btw, I think I'll do a “pros and cons of fascist toolchains” talk at LOCon
- <thorsten> sberg: sounds fun indeed - special attire required for attendance? or just bring supply of soft vegetables? :)
- <sberg> thorsten, I'll concentrate on the pros, so there'll be no use for rotten tomatoes
- <thorsten> damn

- <buovjaga> a fascist toolchain is characterized by dictatorial power and forcible suppression of opposition

What the Fuck

- “Why does Gerrit complain about my change?”
  - “It compiled just fine here!”

- “Who removed that code I was still working on?”
  - “It *isn’t* supposed to be unused!”

loplugin

- 61 plugins that are enabled by default:

badstatics	cstylecast	externvar	loopvartoo small	ptrvector	salbool	static methods	unnecessary override
blockblock	datamember shadow	faileddyn cast	nullptr	rangedfor copy	sallogareas	stringconcat	unnecessary paren
casttovoid	derefnullptr	finalprotect	oslendian	redundant cast	salunicode literal	string constant	unoany
charright shift	dllprivate	flatten	override	redundant copy	sfxpoolitem	stringstatic	unreffun
checkconfig macros	dyncast visibility	fpcompari son	override param	redundant inline	simplifybool	subtlezero init	unusedva riablecheck
commaoper ator	dynexcspec	implicitbool conversion	passparams byref	redundant pointerops	staticaccess	unicodeto char	vclwidgets
conststring var	expression alwayszero	inlinevisible	passstuff byref	refcounting	static anonymous	unnecessary catchthrow	weakobject
cppunit assertequals	externand notdefined	literaltobool conversion	privatebase	reservedid			

loplugin

- 61 plugins that are enabled by default:

badstatics	cstylecast	externvar	loopvartoo small	ptrvector	salbool	static methods	unnecessary override
blockblock	datamember shadow	faileddyn cast	nullptr	rangedfor copy	sallogareas	stringconcat	unnecessary paren
casttovoid	derefnullptr	finalprotect	oslendian	redundant cast	salunicode literal	string constant	unoany
charright shift	dllprivate	flatten	override	redundant copy	sfxpoolitem	stringstatic	unreffun
checkconfig macros	dyncast visibility	fpcompari son	override param	redundant inline	simplifybool	subtlezero init	unusedva riablecheck
commaoper ator	dynexcspec	implicitbool conversion	passparams byref	redundant pointerops	staticaccess	unicodeto char	vclwidgets
conststring var	expression alwayszero	inlinevisible	passstuff byref	refcounting	static anonymous	unnecessary catchthrow	weakobject
cppunit assertequals	externand notdefined	literaltobool conversion	privatebase	reservedid			

staticmethods + staticaccess

- struct S {
  int f() { return 0; } // loplugin:staticmethods
  };
  int g(S * s) { return s->f(); }

- struct S {
  static int f() { return 0; }
  };
  int g(S * s) { return s->f(); } // loplugin:staticaccess

- struct S {
  static int f() { return 0; }
  };
  int g() { return S::f(); }

staticmethods + staticaccess

- struct S {
  int f() {
  (void) this; // loplugin:staticmethods
  return 0;
  }
  };
  int g(S * s) { return s->f(); }

- ...not unlike

  int e = someFn();
  assert(e != 0);
  (void) e;

unreffun

- class Test {
  void test1() { ... }
  void test2() { ... }
  void test3() { ... }

  CPPUNIT_TEST_SUITE(Test);
  CPPUNIT_TEST(test1);
  CPPUNIT_TEST(test2);
  ~~CPPUNIT_TEST(test3);~~
  CPPUNIT_TEST_SUITE_END();
  };

unreffun

- CPPUNIT_TEST_SUITE(Test);
  ~~CPPUNIT_TEST(test1);~~
  ~~CPPUNIT_TEST(test2);~~
  CPPUNIT_TEST(test3);
  ~~CPPUNIT_TEST(test4);~~
  ~~CPPUNIT_TEST(test5);~~
  ~~CPPUNIT_TEST(test6);~~
  ~~CPPUNIT_TEST(test7);~~
  ~~CPPUNIT_TEST(test8);~~
  ~~CPPUNIT_TEST(test9);~~
  ~~CPPUNIT_TEST(test10);~~
  CPPUNIT_TEST_SUITE_END();
  - temporarily --disable-werror

The code is with you

- If it doesn't suite your needs, change it
  - [loplugin:*foo*] → compilerplugins/clang/*foo*.cxx
- Many plugins have whitelists in the code
  - Either based on specific identifiers, or on source files/hierarchies

We’re not the only pigheads

- struct S { char b[3]; };
  char buf[2];
  void f(struct S const * s) {
  assert(strlen(s->b) == 1);
  snprintf(buf, sizeof (buf), "%s", s->b);
  }
- gcc -Wall -c test.c
- warning: ‘snprintf’ output may be truncated before the last format character [-Wformat-truncation=]

We’re not the only pigheads

- struct S { char b[3]; };
  char buf[2];
  void f(struct S const * s) {
  assert(strlen(s->b) == 1);
  (void) snprintf(buf, sizeof (buf), "%s", s->b);
  }
- gcc -Wall -c test.c
- warning: ‘snprintf’ output may be truncated before the last format character [-Wformat-truncation=]

We’re not the only pigheads

- struct S { char b[3]; };
  char buf[2];
  void f(struct S const * s) {
  assert(strlen(s->b) == 1);
  int dummy = snprintf(buf, sizeof (buf), "%s", s->b);
  (void) dummy;
  }
- gcc -Wall -c test.c

We’re not the only pigheads

- struct S { char b[3]; };
  char buf[2];
  void f(struct S const * s) {
  assert(strlen(s->b) == 1);
  int dummy = snprintf(buf, sizeof (buf), "%s", s->b);
  (void) dummy;
  }
- gcc -Wall **-O** -c test.c
- warning: ‘snprintf’ output may be truncated before the last format character [-Wformat-truncation=]
- GCC 7 bug 80354

Sometimes I feel so happy
Sometimes I feel so sad
But mostly you just make me mad

Lou Reed