Development Tools

Akim Demaille Étienne Renault Roland Levillain
first.*last*@lrde.epita.fr

EPITA — École Pour l'Informatique et les Techniques Avancées

January 22, 2016

Development Tools

tc Tasks

The Tiger Compiler layout

- One module, one namespace
- One "library" per module, with a pure interface (`libfoo.*`)
- One task set per module, maybe impure (`tasks.*`)
- Tasks describe the command line interface
- Requirements over tasks order
- One class, one file base name:

  foo.hh Interface

  foo.hxx Inline implementation

  foo.cc Implementation

The Tiger Compiler layout

- One module, one namespace
- One "library" per module, with a pure interface (`libfoo.*`)
- One task set per module, maybe impure (`tasks.*`)
- Tasks describe the command line interface
- Requirements over tasks order
- One class, one file base name:

  foo.hh Interface

  foo.hxx Inline implementation

  foo.cc Implementation

The Tiger Compiler layout

- One module, one namespace
- One "library" per module, with a pure interface (`libfoo.*`)
- One task set per module, maybe impure (`tasks.*`)
- Tasks describe the command line interface
- Requirements over tasks order
- One class, one file base name:

  foo.hh Interface

  foo.hxx Inline implementation

  foo.cc Implementation

The Tiger Compiler layout

- One module, one namespace
- One "library" per module, with a pure interface (`libfoo.*`)
- One task set per module, maybe impure (`tasks.*`)
- Tasks describe the command line interface
- Requirements over tasks order
- One class, one file base name:

  foo.hh Interface

  foo.hxx Inline implementation

  foo.cc Implementation

The Tiger Compiler layout

- One module, one namespace
- One "library" per module, with a pure interface (`libfoo.*`)
- One task set per module, maybe impure (`tasks.*`)
- Tasks describe the command line interface
- Requirements over tasks order
- One class, one file base name:
  - foo.hh Interface
  - foo.hxx Inline implementation
  - foo.cc Implementation

The Tiger Compiler layout

- One module, one namespace
- One "library" per module, with a pure interface (`libfoo.*`)
- One task set per module, maybe impure (`tasks.*`)
- Tasks describe the command line interface
- Requirements over tasks order
- One class, one file base name:

  foo.hh Interface
  foo.hxx Inline implementation
  foo.cc Implementation

The Tiger Compiler layout

- One module, one namespace
- One "library" per module, with a pure interface (`libfoo.*`)
- One task set per module, maybe impure (`tasks.*`)
- Tasks describe the command line interface
- Requirements over tasks order
- One class, one file base name:

  foo.hh Interface
  foo.hxx Inline implementation
  foo.cc Implementation

The Tiger Compiler layout

- One module, one namespace
- One "library" per module, with a pure interface (`libfoo.*`)
- One task set per module, maybe impure (`tasks.*`)
- Tasks describe the command line interface
- Requirements over tasks order
- One class, one file base name:

  foo.hh Interface
  foo.hxx Inline implementation
  foo.cc Implementation

The Tiger Compiler layout

- One module, one namespace
- One "library" per module, with a pure interface (`libfoo.*`)
- One task set per module, maybe impure (`tasks.*`)
- Tasks describe the command line interface
- Requirements over tasks order
- One class, one file base name:

foo.hh	Interface
foo.hxx	Inline implementation
foo.cc	Implementation

Tasks: ast/tasks.hh

```
namespace ast
{
  namespace tasks
  {
    /// Global root node of abstract syntax tree.
    extern ast::DecsList* the_program;

    TASK_GROUP("2. Abstract Syntax Tree");

    /// Display the abstract syntax tree
    TASK_DECLARE("A|ast-display", "display the AST",
                 ast_display, "parse");

    /// Free the ast (if defined)
    TASK_DECLARE("D|ast-delete", "delete the AST",
                 ast_delete, "parse");
  } // namespace tasks
} // namespace ast
```

Tasks: ast/tasks.cc

```
namespace ast
{
  namespace tasks
  {
    ast::DecsList* the_program = nullptr;

    void ast_display()
    {
      precondition(the_program);
      std::cout << "/* Abstract Syntax Tree. */\n"
                << *the_program << '\n';
    }

    void ast_delete()
    {
      delete the_program;
      the_program = nullptr;
    }
  } // namespace tasks
```

Maintaining Packages

GNU Tools

GNU Autotools

aclocal Create `aclocal.m4` from `configure.ac`'s requests

autoconf Create `configure` from `configure.ac` and `aclocal.m4`

autoheader Create `config.h.in` from `configure.ac` (and `aclocal.m4`)

automake Create `Makefile.in` from `Makefile.am` and `configure.ac`

autoreconf Run them as needed (`autoreconf -fivm`)

Read Alexandre Duret-Lutz's Tutorial [Duret-Lutz, 2006]

GNU Autotools

aclocal	Create `aclocal.m4` from `configure.ac`'s requests
autoconf	Create `configure` from `configure.ac` and `aclocal.m4`
autoheader	Create `config.h.in` from `configure.ac` (and `aclocal.m4`)
automake	Create `Makefile.in` from `Makefile.am` and `configure.ac`
autoreconf	Run them as needed (`autoreconf -fivm`)

Read Alexandre Duret-Lutz's Tutorial [Duret-Lutz, 2006]

GNU Autotools

aclocal	Create `aclocal.m4` from `configure.ac`'s requests
autoconf	Create `configure` from `configure.ac` and `aclocal.m4`
autoheader	Create `config.h.in` from `configure.ac` (and `aclocal.m4`)
automake	Create `Makefile.in` from `Makefile.am` and `configure.ac`
autoreconf	Run them as needed (`autoreconf -fivm`)

Read Alexandre Duret-Lutz's Tutorial [Duret-Lutz, 2006]

GNU Autotools

aclocal	Create `aclocal.m4` from `configure.ac`'s requests
autoconf	Create `configure` from `configure.ac` and `aclocal.m4`
autoheader	Create `config.h.in` from `configure.ac` (and `aclocal.m4`)
automake	Create `Makefile.in` from `Makefile.am` and `configure.ac`
autoreconf	Run them as needed (`autoreconf -fivm`)

Read Alexandre Duret-Lutz's Tutorial [Duret-Lutz, 2006]

GNU Autotools

aclocal	Create `aclocal.m4` from `configure.ac`'s requests
autoconf	Create `configure` from `configure.ac` and `aclocal.m4`
autoheader	Create `config.h.in` from `configure.ac` (and `aclocal.m4`)
automake	Create `Makefile.in` from `Makefile.am` and `configure.ac`
autoreconf	Run them as needed (`autoreconf -fivm`)

Read Alexandre Duret-Lutz's Tutorial [Duret-Lutz, 2006]

GNU Autotools

aclocal	Create `aclocal.m4` from `configure.ac`'s requests
autoconf	Create `configure` from `configure.ac` and `aclocal.m4`
autoheader	Create `config.h.in` from `configure.ac` (and `aclocal.m4`)
automake	Create `Makefile.in` from `Makefile.am` and `configure.ac`
autoreconf	Run them as needed (`autoreconf -fivm`)

Read Alexandre Duret-Lutz's Tutorial [Duret-Lutz, 2006]

Programming Tools: Packages

A set of packages to maintain packages:

- Autoconf package configuration [MacKenzie et al., 2003]
- Automake package build [Duret-Lutz and Tromey, 2003]
- Libtool portable build of shared libs
- Gettext package internationalization
- Flex scanner generation
- Bison parser generation
- Boost C++[2]

Programming Tools: Packages

A set of packages to maintain packages:

- **Autoconf** package configuration [MacKenzie et al., 2003]
- **Automake** package build [Duret-Lutz and Tromey, 2003]
- **Libtool** portable build of shared libs
- **Gettext** package internationalization
- **Flex** scanner generation
- **Bison** parser generation
- **Boost** C++[2]

Programming Tools: Packages

A set of packages to maintain packages:

- **Autoconf** package configuration [MacKenzie et al., 2003]
- **Automake** package build [Duret-Lutz and Tromey, 2003]
- **Libtool** portable build of shared libs
- **Gettext** package internationalization
- **Flex** scanner generation
- **Bison** parser generation
- **Boost** C++[2]

Programming Tools: Packages

A set of packages to maintain packages:

Autoconf	package configuration [MacKenzie et al., 2003]
Automake	package build [Duret-Lutz and Tromey, 2003]
Libtool	portable build of shared libs
Gettext	package internationalization
Flex	scanner generation
Bison	parser generation
Boost	C++[2]

Programming Tools: Packages

A set of packages to maintain packages:

- **Autoconf** package configuration [MacKenzie et al., 2003]
- **Automake** package build [Duret-Lutz and Tromey, 2003]
- **Libtool** portable build of shared libs
- **Gettext** package internationalization
- **Flex** scanner generation
- **Bison** parser generation
- **Boost** C++[2]

Programming Tools: Packages

A set of packages to maintain packages:

- **Autoconf** package configuration [MacKenzie et al., 2003]
- **Automake** package build [Duret-Lutz and Tromey, 2003]
- **Libtool** portable build of shared libs
- **Gettext** package internationalization
- **Flex** scanner generation
- **Bison** parser generation
- **Boost** C++[2]

Programming Tools: Packages

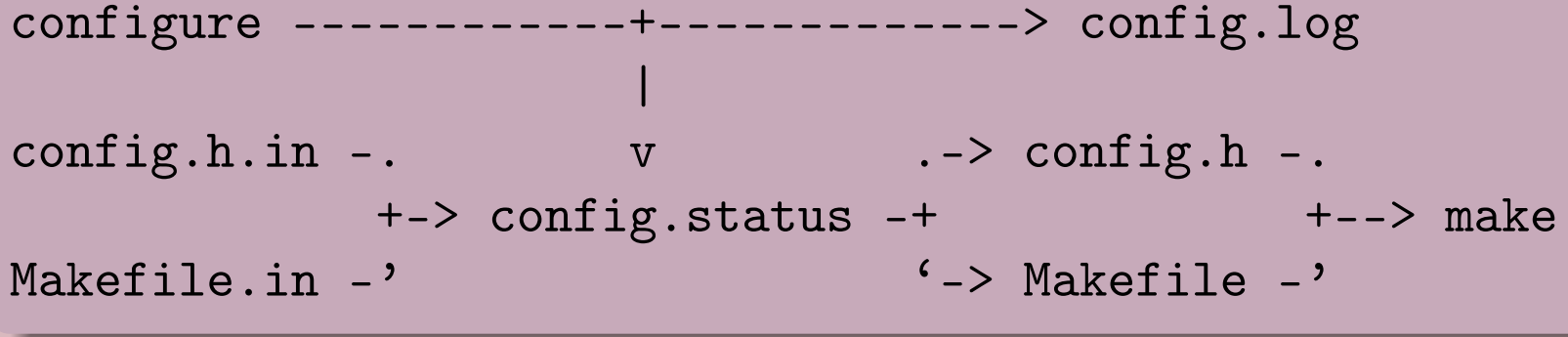
A set of packages to maintain packages:

Autoconf	package configuration [MacKenzie et al., 2003]
Automake	package build [Duret-Lutz and Tromey, 2003]
Libtool	portable build of shared libs
Gettext	package internationalization
Flex	scanner generation
Bison	parser generation
Boost	C++[2]

Autoconf for tc

Autoconf files

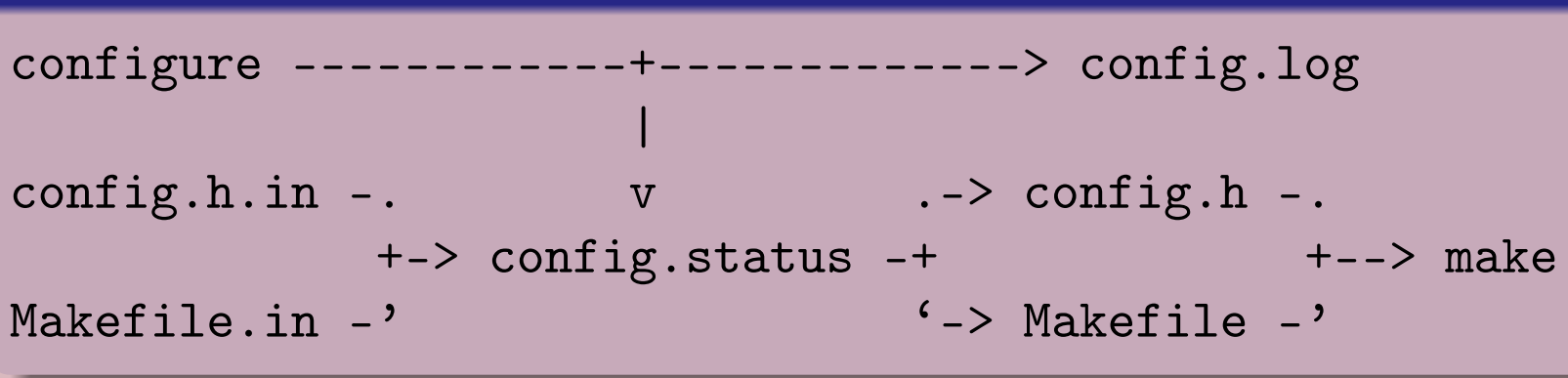
Configuring a package



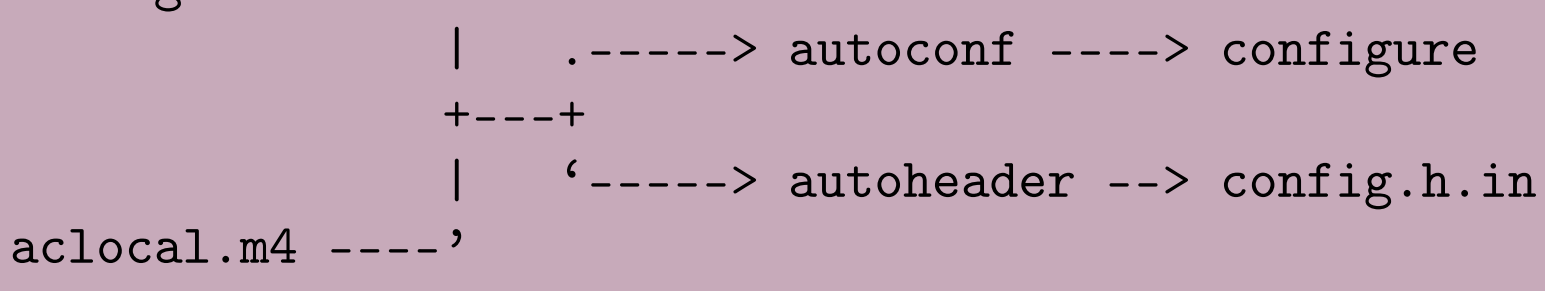
Preparing a package for distribution

Autoconf files

Configuring a package



Preparing a package for distribution



configure.ac 1: Initialization

```
AC_PREREQ([2.64])
AC_INIT([LRDE Tiger Compiler], [1.29a],
        [tiger@lrde.epita.fr], [tc])

# Auxiliary files.
AC_CONFIG_AUX_DIR([build-aux])
AC_CONFIG_MACRO_DIR([build-aux/m4])

# Automake.
AM_INIT_AUTOMAKE([1.14.1 check-news dist-bzip2 no-dist-gzip
                  foreign
                  color-tests parallel-tests
                  nostdinc silent-rules -Wall])
AM_SILENT_RULES([yes])
```

configure.ac 2: C++ Compiler

```
# Look for a C++ compiler.
AC_LANG([C++])
AC_PROG_CXX

# Enable C++ 2011 support.
...

# Using pipes between compiler stages is faster.
AX_CHECK_COMPILE_FLAG([-pipe], [CXXFLAGS="$CXXFLAGS -pipe"])

# Use good warnings.
TC_CXX_WARNINGS([[-Wall], [-W], [-Wcast-align], ...])
```

configure.ac 3: Auxiliary Programs

```
TC_PROG([flex], [>= 2.5.35], [FLEX],
        [Flex scanner generator])
AX_CONFIG_SCRIPTS([build-aux/bin/flex++])

TC_PROG([bison], [>= 3.0], [BISON],
        [Bison parser generator])
AX_CONFIG_SCRIPTS([build-aux/bin/bison++])

# We don't need static libraries, speed the compilation up.
LT_INIT([disable-shared])

TC_PROG([monoburg], [>= 1.0.6], [MONOBURG],
        [MonoBURG code generator generator])
AX_CONFIG_SCRIPTS([build-aux/bin/monoburg++])

TC_PROG([havm], [>=0.24], [HAVM],
        [The Tree Virtual Machine])
```

configure.ac 4: Libraries

```
AC_CONFIG_SUBDIRS([lib/argp])

BOOST_REQUIRE([1.53])
BOOST_CONVERSION # lexical_cast
BOOST_GRAPH
BOOST_PREPROCESSOR
BOOST_STRING_ALGO
BOOST_TRIBOOL
BOOST_VARIANT
```

configure.ac 5: SWIG & tcsh

```
TC_WITH_TCSH([with_tcsh=yes], [with_tcsh=no])
AM_CONDITIONAL([ENABLE_TCSH], [test x$with_tcsh = xyes])

AC_CONFIG_FILES([tcsh/Makefile
                 tcsh/python/Makefile
                 tcsh/ruby/Makefile])
AX_CONFIG_SCRIPTS([tcsh/run])
```

configure.ac 6: File Creation

```
# Ask for the creation of config.h.
AC_CONFIG_HEADERS([config.h])

# Ask for the creation of the Makefiles.
AC_CONFIG_FILES([Makefile])

# Instantiate the output files.
AC_OUTPUT
```

Automake for tc

src/local.am 1: Variables

```
AUTOMAKE_OPTIONS = subdir-objects

AM_DEFAULT_SOURCE_EXT = .cc

BUILT_SOURCES =
CLEANFILES =
EXTRA_DIST =
MAINTAINERCLEANFILES =
TESTS =
EXTRA_PROGRAMS = $(TESTS)
dist_noinst_DATA =
noinst_LTLIBRARIES =

RECHECK_LOGS =
```

src/local.am 2: Common Options

```
# Most headers are to be shipped and to be found in src/, e.g.
# tasks/tasks.hh is shipped in $(top_srcdir)/src/task/tasks.hh.
# Some are *built* in src, e.g., $(top_builddir)/src/modules.hh.
AM_CPPFLAGS = -I$(top_srcdir)/lib
AM_CPPFLAGS += -I$(top_srcdir)/src -I$(top_builddir)/src
AM_CPPFLAGS += $(BOOST_CPPFLAGS)
# Find the prelude.
AM_CPPFLAGS += -DPKGDATADIR="\"$(pkgdatadir)\""

AM_CXXFLAGS = $(WARNING_CXXFLAGS)
```

src/local.am 3: Tasks

```
TASKS =
include task/local.am
include ast/local.am
[...]
include regalloc/local.am

EXTRA_DIST += tiger_common.i
```

src/local.am 3: Building libtc

```
lib_LTLIBRARIES = src/libtc.la
src_libtc_la_SOURCES = src/version.hh
nodist_src_libtc_la_SOURCES = src/version.cc
src_libtc_la_LDFLAGS = $(BOOST_PROGRAM_OPTIONS_LDFLAGS)

src_libtc_la_LIBADD =                                   \
  $(top_builddir)/lib/misc/libmisc.la                   \
  $(BOOST_PROGRAM_OPTIONS_LIBS)
```

src/local.am 4: Building tc

```
bin_PROGRAMS = src/tc
dist_src_tc_SOURCES =                                   \
  src/doc.hh                                            \
  $(TASKS)                                              \
  src/common.cc src/common.hh                           \
  src/tc.cc

src_tc_LDADD = src/libtc.la
```

src/bind/local.am: Binding Names

```
## bind module.
EXTRA_DIST += %D%/tiger_bind.i

src_libtc_la_SOURCES +=                                    \
  %D%/binder.hh %D%/binder.hxx %D%/binder.cc               \
  %D%/libbind.hh %D%/libbind.cc

TASKS += %D%/tasks.hh %D%/tasks.cc

# Tests.
check_PROGRAMS += %D%/test-bind
%C%_test_bind_LDADD = src/libtc.la
```

Tools for the Developer

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Other developing tools

- Use warnings
- Use the `assert` macro
- Use `static_assert` (C++11)
- Electric Fence, DUMA, PageHeap (Windows), Guard Malloc (Mac OS X)
- Dmalloc
- AddressSanitizer, ThreadSanitizer (built in GCC 4.8+)
- Mudflap (built in GCC 3.x–4.8)
- GDB (GUIs : DDD, KDbg), LLDB
- Purify, Insure++, BoundsChecker (proprietary)
- Valgrind
- Pin
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, flint
- Gprof, OProfile

Mudflap

```
int
main()
{
  int tab[10];
  int i;

  for (i = 0; i <= 10; ++i)
    tab[i] = 0;
  return 0;
}
gcc -fmudflap -lmudflap bounds-violation.c
```

Mudflap

```
env MUDFLAP_OPTIONS=-viol-abort ./a.out
*******
mudflap violation 1 (check/write): time=1292501299.526454 ptr=0xbfc35d34 size=44
pc=0xb77d13bd location='bounds-violation.c:8:5 (main)'
      /usr/lib/libmudflap.so.0(__mf_check+0x3d) [0xb77d13bd]
      ./a.out(main+0xb7) [0x804883b]
      /usr/lib/libmudflap.so.0(__wrap_main+0x49) [0xb77d0b89]
Nearby object 1: checked region begins 0B into and ends 4B after
mudflap object 0x8c7a080: name='bounds-violation.c:4:7 (main) tab'
bounds=[0xbfc35d34,0xbfc35d5b] size=40 area=stack check=0r/4w liveness=4
alloc time=1292501299.526444 pc=0xb77d0b2d
number of nearby objects: 1
zsh: abort      env MUDFLAP_OPTIONS=-viol-abort ./a.out
```

Address Sanitizer (GCC 4.8, Clang 3.3)

```
int
main()
{
  int tab[10];
  int i;

  for (i = 0; i <= 10; ++i)
    tab[i] = 0;
  return 0;
}
```

```
gcc -fsanitize=address bounds-violation.c
```

```
=================================================================
==8359== ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7fff23839248 at pc 0x400730 bp 0x7fff23839
WRITE of size 4 at 0x7fff23839248 thread T0
    #0 0x40072f (/work/roland/src/cours-ccmp/ccmp/dev-tools/a.out+0x40072f)
    #1 0x7f5c1b107eac (/lib/x86_64-linux-gnu/libc-2.13.so+0x1eeac)
    #2 0x4005b8 (/work/roland/src/cours-ccmp/ccmp/dev-tools/a.out+0x4005b8)
Address 0x7fff23839248 is located at offset 72 in frame <main> of T0's stack:
  This frame has 1 object(s):
    [32, 72) 'tab'
HINT: this may be a false positive if your program uses some custom stack unwind mechanism or swapcontext
      (longjmp and C++ exceptions *are* supported)
Shadow bytes around the buggy address:
  0x1000646ff1f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000646ff200: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000646ff210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000646ff220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000646ff230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x1000646ff240: f1 f1 f1 f1 00 00 00 00 00[f4]f4 f4 f3 f3 f3 f3
  0x1000646ff250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000646ff260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000646ff270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000646ff280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x1000646ff290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:     fa
  Heap righ redzone:     fb
  Freed Heap region:     fd
  Stack left redzone:    f1
  Stack mid redzone:     f2
  Stack right redzone:   f3
  Stack partial redzone: f4
  Stack after return:    f5
  Stack use after scope: f8
  Global redzone:        f9
  Global init order:     f6
  Poisoned by user:      f7
```

Valgrind and Memory Violation

```c
#include <stdio.h>


typedef struct list_s { int val; struct list_s *next; } list_t;

list_t *list_new(int val, list_t *next) {
  list_t res = { val, next };
  return &res;
}


void list_print(const list_t *const list) {
  if (list)
    printf("%d\n", list->val), list_print(list->next);
}

int main(void) {
  list_print(list_new(2, list_new(1, list_new(0, NULL))));
  return 0;
```

Valgrind and Memory Leaks

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct list_s { int val; struct list_s *next; } list_t;

list_t *list_new(int val, list_t *next) {
  list_t *res = (list_t *) malloc(sizeof(list_t));
  res->val  = val; res->next = next;
  return res;
}

void list_print(const list_t *const list) {
  if (list)
    printf("%d\n", list->val), list_print(list->next);
}

int main(void) {
  list_print(list_new(2, list_new(1, list_new(0, NULL))));
  return 0;
```

Valgrind and Memory Leaks

```
gcc -g memory-leaks.c
valgrind --leak-check=full ./a.out
==9702== Memcheck, a memory error detector
==9702== Copyright (C) 2002-2010, and GNU GPL'd, by Julian Seward et al.
==9702== Using Valgrind-3.6.0.SVN-Debian and LibVEX; rerun with -h for copyright info
==9702== Command: ./a.out
==9702==
2
1
0
==9702==
==9702== HEAP SUMMARY:
==9702==     in use at exit: 24 bytes in 3 blocks
==9702==   total heap usage: 3 allocs, 0 frees, 24 bytes allocated
==9702==
==9702== 24 (8 direct, 16 indirect) bytes in 1 blocks are definitely lost in loss record 3 of 3
==9702==    at 0x4023F50: malloc (vg_replace_malloc.c:236)
==9702==    by 0x8048405: list_new (memory-leaks.c:7)
==9702==    by 0x804848D: main (memory-leaks.c:18)
==9702==
==9702== LEAK SUMMARY:
==9702==    definitely lost: 8 bytes in 1 blocks
==9702==    indirectly lost: 16 bytes in 2 blocks
==9702==      possibly lost: 0 bytes in 0 blocks
==9702==    still reachable: 0 bytes in 0 blocks
==9702==         suppressed: 0 bytes in 0 blocks
==9702==
==9702== For counts of detected and suppressed errors, rerun with: -v
==9702== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 11 from 6)
```

A Clear Winner?

- Valgrind doesn't catch the previous Mudflap example.
  - Padding
  - Overrun into neighbor regions
- Mudflap and ASan do not know about uninitialized regions.

A Clear Winner?

- Valgrind doesn't catch the previous Mudflap example.
  - Padding
  - Overrun into neighbor regions
- Mudflap and ASan do not know about uninitialized regions.

A Clear Winner?

- Valgrind doesn't catch the previous Mudflap example.
  - Padding
  - Overrun into neighbor regions
- Mudflap and ASan do not know about uninitialized regions.

A Clear Winner?

- Valgrind doesn't catch the previous Mudflap example.
  - Padding
  - Overrun into neighbor regions
- Mudflap and ASan do not know about uninitialized regions.

A Clear Winner? Kind of!

Actually Valgrind *does* catch the previous Mudflap example now, thanks to SGCheck, a stack and global array overrun detector leveraging debugging information.

```
gcc -g bounds-violation.c
valgrind --tool=exp-sgcheck ./a.out
```

```
==22757== exp-sgcheck, a stack and global array overrun detector
==22757== NOTE: This is an Experimental-Class Valgrind Tool
==22757== Copyright (C) 2003-2013, and GNU GPL'd, by OpenWorks Ltd et al.
==22757== Using Valgrind-3.9.0 and LibVEX; rerun with -h for copyright info
==22757== Command: ./a.out
==22757==
==22757== Invalid write of size 4
==22757==    at 0x4004BE: main (bounds-violation.c:8)
==22757==  Address 0xfff000408 expected vs actual:
==22757==  Expected: stack array "tab" of size 40 in this frame
==22757==  Actual:   unknown
==22757==  Actual:   is 0 after Expected
==22757==
==22757==
==22757== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 4 from 4)
```

Intermission: goto Still Considered Harmful

Old (bad!) habits die hard: Apple's Feb. 2014 SSL/TLS Bug [Langley, 2014]

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
    OSStatus        err;
    ...

    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;
    ...

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

Intermission: goto Still Considered Harmful

Old (bad!) habits die hard: Apple's Feb. 2014 SSL/TLS Bug [Langley, 2014]

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
    OSStatus        err;
    ...

    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) // Unreachable code.
        goto fail;
    ...

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```

Caught by Clang's `-Wunreachable-code` flag (but not `-Wall`)

Version Control

- Makes working in group *a lot* easier.
- Gives the possibility to travel back in time (e.g, to hunt bugs).
- Allows several, non-linear developing models (branches).
- Add some semantics to the development itself.
- Provides a kind of backup
  But cannot make up for the lack of a real backup solution!
- EPITA provides Git repositories for the Tiger project.

Version Control

- Makes working in group *a lot* easier.
- Gives the possibility to travel back in time (e.g, to hunt bugs).
- Allows several, non-linear developing models (branches).
- Add some semantics to the development itself.
- Provides a kind of backup
  But cannot make up for the lack of a real backup solution!
- EPITA provides Git repositories for the Tiger project.

Version Control

- Makes working in group *a lot* easier.
- Gives the possibility to travel back in time (e.g, to hunt bugs).
- Allows several, non-linear developing models (branches).
- Add some semantics to the development itself.
- Provides a kind of backup
  But cannot make up for the lack of a real backup solution!
- EPITA provides Git repositories for the Tiger project.

Version Control

- Makes working in group *a lot* easier.
- Gives the possibility to travel back in time (e.g, to hunt bugs).
- Allows several, non-linear developing models (branches).
- Add some semantics to the development itself.
- Provides a kind of backup
  But cannot make up for the lack of a real backup solution!
- EPITA provides Git repositories for the Tiger project.

Version Control

- Makes working in group *a lot* easier.
- Gives the possibility to travel back in time (e.g, to hunt bugs).
- Allows several, non-linear developing models (branches).
- Add some semantics to the development itself.
- Provides a kind of backup
  But cannot make up for the lack of a real backup solution!
- EPITA provides Git repositories for the Tiger project.

Version Control

- Makes working in group *a lot* easier.
- Gives the possibility to travel back in time (e.g, to hunt bugs).
- Allows several, non-linear developing models (branches).
- Add some semantics to the development itself.
- Provides a kind of backup
  But cannot make up for the lack of a real backup solution!
- EPITA provides Git repositories for the Tiger project.

Document with Doxygen

- Use comments to annotate code entities (namespaces, files, functions, classes, type aliases, etc.).
- Generate a hyper-text reference documentation.
- Several back-ends: HTML, PDF, RTF, etc.
- Use `make doc` in the Tiger Project.
- See `http://doxygen.org/manual.html`.

Document with Doxygen

- Use comments to annotate code entities (namespaces, files, functions, classes, type aliases, etc.).
- Generate a hyper-text reference documentation.
- Several back-ends: HTML, PDF, RTF, etc.
- Use `make doc` in the Tiger Project.
- See `http://doxygen.org/manual.html`.

Document with Doxygen

- Use comments to annotate code entities (namespaces, files, functions, classes, type aliases, etc.).
- Generate a hyper-text reference documentation.
- Several back-ends: HTML, PDF, RTF, etc.
- Use `make doc` in the Tiger Project.
- See `http://doxygen.org/manual.html`.

Document with Doxygen

- Use comments to annotate code entities (namespaces, files, functions, classes, type aliases, etc.).
- Generate a hyper-text reference documentation.
- Several back-ends: HTML, PDF, RTF, etc.
- Use `make doc` in the Tiger Project.
- See `http://doxygen.org/manual.html`.

Document with Doxygen

- Use comments to annotate code entities (namespaces, files, functions, classes, type aliases, etc.).
- Generate a hyper-text reference documentation.
- Several back-ends: HTML, PDF, RTF, etc.
- Use `make doc` in the Tiger Project.
- See `http://doxygen.org/manual.html`.

Document with Doxygen: type/libtype.hh

```
/// \file type/libtype.hh
/// \brief Declare the function exported by type module.
#ifndef TYPE_LIBTYPE_HH
# define TYPE_LIBTYPE_HH

# include "misc/error.hh"
# include "ast/fwd.hh"

/// Type-checking an ast::Ast.
namespace type
{

  /** \brief Check types in a (bound) AST.
   ** \param tree   abstract syntax tree's root.
   ** \return       synthesis of the errors possibly found. */
  misc::error types_check(ast::Ast& tree);

} // namespace type
```

Bibliography I


Duret-Lutz, A. (2006).
The Autotools Tutorial.
http://www-src.lip6.fr/homepages/Alexandre.Duret-Lutz/dl/autotools.pdf/.


Duret-Lutz, A. and Tromey, T. (2003).
GNU Automake.
http://www.gnu.org/software/automake/.


Langley, A. (2014).
ImperialViolet — Apple SSL/TLS bug.
https://www.imperialviolet.org/2014/02/22/applebug.html.


MacKenzie, D. J., Elliston, B., and Demaille, A. (2003).
GNU Autoconf.
http://www.gnu.org/software/autoconf/.