Compiler Construction

Development Tools

Tools for the developer

How to manage a huge project?

How to avoid errors?

How to use an efficient tunable build system?

Tools for the developer

- Use warnings
- Use **assert**/**static_assert**(C++11)
- AddressSanitizer, ThreadSanitizer, Memory Sanitizer, Leaks Sanitizer
- lcov
- Gprof, OProfile
- clang tidy, clang format
- DTrace
- Clang Static Analyzer (LLVM)
- Cppcheck, clazy,
- lint
- callgrind, kcachegrind
- ...

Spot the bug (1/3)

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

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

Spot the bug (2/3)

```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;
```

Spot the bug (3/3)

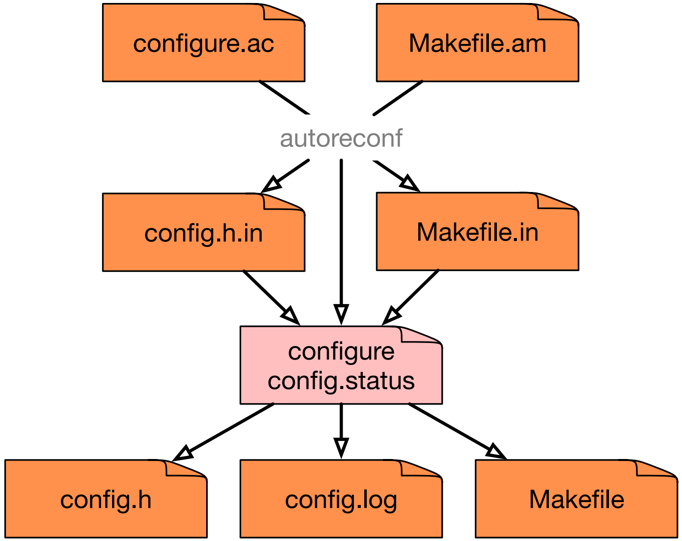
```c
#include <stdio.h>

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;
}
```

GNU Autotools

A set of packages to maintain packages

- **autoconf**: package configuration
- **automake**: package build
- **libtool**: a portable build of shared libs
- **gettext**: package internationalization

GNU Autotools overview



configure.ac (1/3): Initialization

```
AC_PREREQ([2.64])
AC_INIT([LRDE Tiger Compiler], [1.72a],
   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 (2/4): C++ Compiler

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

# Enable C++ 2020 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 (3/4): Auxiliary Programs

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

TC_PROG([bison], [>= 3.2], [BISON],
        [Bison parser generator])
AX_CONFIG_SCRIPTS([build-aux/bin/bison++])

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


BOOST_REQUIRE([1.63])
BOOST_CONVERSION # lexical_cast
BOOST_GRAPH
```

configure.ac (4/4): 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
```

Makefile.am & local.am

```
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)

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

Summary

configure

autoreconf

ASAN, Lint, cppcheck ...

automake