```
cmake_minimum_required(VERSION 3.13.0)
set(ETH_CMAKE_DIR   "${CMAKE_CURRENT_LIST_DIR}/cmake"   CACHE PATH "The
the path to the cmake directory")
list(APPEND CMAKE_MODULE_PATH ${ETH_CMAKE_DIR})
# Set the build type, if none was specified.
if(NOT CMAKE_BUILD_TYPE AND NOT CMAKE_CONFIGURATION_TYPES)
      if(EXISTS "${PROJECT_SOURCE_DIR}/.git")
            set(DEFAULT_BUILD_TYPE "RelWithDebInfo")
      else()
            set(DEFAULT_BUILD_TYPE "Release")
      endif()
      set(CMAKE_BUILD_TYPE "${DEFAULT_BUILD_TYPE}" CACHE STRING "Choose
the type of build, options are: Debug Release RelWithDebInfo MinSizeRel"
FORCE)
      set_property(CACHE CMAKE_BUILD_TYPE PROPERTY STRINGS "Debug"
"Release" "RelWithDebInfo" "MinSizeRel")
endif()
include(EthToolchains)
# Set cmake_policies
include(EthPolicy)
eth_policy()
# project name and version should be set after cmake_policy CMP0048
set(PROJECT_VERSION "0.8.21")
# OSX target needed in order to support std::visit
set(CMAKE_OSX_DEPLOYMENT_TARGET "10.14")
project(solidity VERSION ${PROJECT_VERSION} LANGUAGES C CXX)
include(TestBigEndian)
TEST_BIG_ENDIAN(IS_BIG_ENDIAN)
if (IS_BIG_ENDIAN)
      message(FATAL_ERROR "${PROJECT_NAME} currently does not support big
endian systems.")
endif()
option(SOLC_LINK_STATIC "Link solc executable statically on supported
platforms" OFF)
option(SOLC_STATIC_STDLIBS "Link solc against static versions of libgcc
and libstdc++ on supported platforms" OFF)
option(STRICT_Z3_VERSION "Use the latest version of Z3" ON)
option(PEDANTIC "Enable extra warnings and pedantic build flags. Treat
all warnings as errors." ON)
option(PROFILE_OPTIMIZER_STEPS "Output performance metrics for the
optimiser steps." OFF)
# Setup cccache.
include(EthCcache)
# Let's find our dependencies
include(EthDependencies)
include(fmtlib)
include(jsoncpp)
include(range-v3)
include_directories(SYSTEM ${JSONCPP_INCLUDE_DIR})
find_package(Threads)
if(NOT PEDANTIC)
  message(WARNING "-- Pedantic build flags turned off. Warnings will not
make compilation fail. This is NOT recommended in development builds.")
endif()
```

```
if (PROFILE_OPTIMIZER_STEPS)
    add_definitions(-DPROFILE_OPTIMIZER_STEPS)
endif()
# Figure out what compiler and system are we using
include(EthCompilerSettings)
# Include utils
include(EthUtils)
# Create license.h from LICENSE.txt and template
# Converting to char array is required due to MSVC's string size limit.
file(READ ${PROJECT_SOURCE_DIR}/LICENSE.txt LICENSE_TEXT HEX)
string(REGEX MATCHALL ".." LICENSE_TEXT "${LICENSE_TEXT}")
string(REGEX REPLACE ";" ",\n\t0x" LICENSE_TEXT "${LICENSE_TEXT}")
set(LICENSE_TEXT "0x${LICENSE_TEXT}")
configure_file("${PROJECT_SOURCE_DIR}/cmake/templates/license.h.in"
include/license.h)
include(EthOptions)
configure_project(TESTS)
set(LATEST_Z3_VERSION "4.12.1")
set(MINIMUM_Z3_VERSION "4.8.16")
find_package(Z3)
if (${Z3_FOUND})
  if (${STRICT_Z3_VERSION})
    if (NOT ("${Z3_VERSION_STRING}" VERSION_EQUAL ${LATEST_Z3_VERSION}))
      message(
        FATAL_ERROR
        "SMTChecker tests require Z3 ${LATEST_Z3_VERSION} for all tests
to pass.\n\
Build with -DSTRICT_Z3_VERSION=OFF if you want to use a different
version. \
You can also use -DUSE_Z3=OFF to build without Z3. In both cases use --
no-smt when running tests."
      )
    endif()
  else()
    if ("${Z3_VERSION_STRING}" VERSION_LESS ${MINIMUM_Z3_VERSION})
      message(
        FATAL_ERROR
        "Solidity requires Z3 ${MINIMUM_Z3_VERSION} or newer. You can
also use -DUSE_Z3=OFF to build without Z3."
      )
    endif()
  endif()
endif()
if(${USE_Z3_DLOPEN})
  add_definitions(-DHAVE_Z3)
  add_definitions(-DHAVE_Z3_DLOPEN)
  find_package(Python3 COMPONENTS Interpreter)
  if(${Z3_FOUND})
    get_target_property(Z3_HEADER_HINTS z3::libz3
INTERFACE_INCLUDE_DIRECTORIES)
  endif()
  find_path(Z3_HEADER_PATH z3.h HINTS ${Z3_HEADER_HINTS})
  if(Z3_HEADER_PATH)
    set(Z3_FOUND TRUE)
```

```
  else()
    message(SEND_ERROR "Dynamic loading of Z3 requires Z3 headers to be
present at build time.")
  endif()
  if(NOT ${Python3_FOUND})
    message(SEND_ERROR "Dynamic loading of Z3 requires Python 3 to be
present at build time.")
  endif()
  if(${SOLC_LINK_STATIC})
    message(SEND_ERROR "solc cannot be linked statically when dynamically
loading Z3.")
  endif()
elseif (${Z3_FOUND})
  add_definitions(-DHAVE_Z3)
  message("Z3 SMT solver found. This enables optional SMT checking with
Z3.")
endif()
find_package(CVC4 QUIET)
if (${CVC4_FOUND})
  add_definitions(-DHAVE_CVC4)
  message("CVC4 SMT solver found. This enables optional SMT checking with
CVC4.")
endif()
if (NOT (${Z3_FOUND} OR ${CVC4_FOUND}))
  message("No SMT solver found (or it has been forcefully disabled).
Optional SMT checking will not be available.\
  \nPlease install Z3 or CVC4 or remove the option disabling them
(USE_Z3, USE_CVC4).")
endif()
add_subdirectory(libsolutil)
add_subdirectory(liblangutil)
add_subdirectory(libsmtutil)
add_subdirectory(libevmasm)
add_subdirectory(libyul)
add_subdirectory(libsolidity)
add_subdirectory(libsolc)
add_subdirectory(libstdlib)
add_subdirectory(tools)
if (NOT EMSCRIPTEN)
      add_subdirectory(solc)
endif()
if (TESTS AND NOT EMSCRIPTEN)
      add_subdirectory(test)
endif()
```