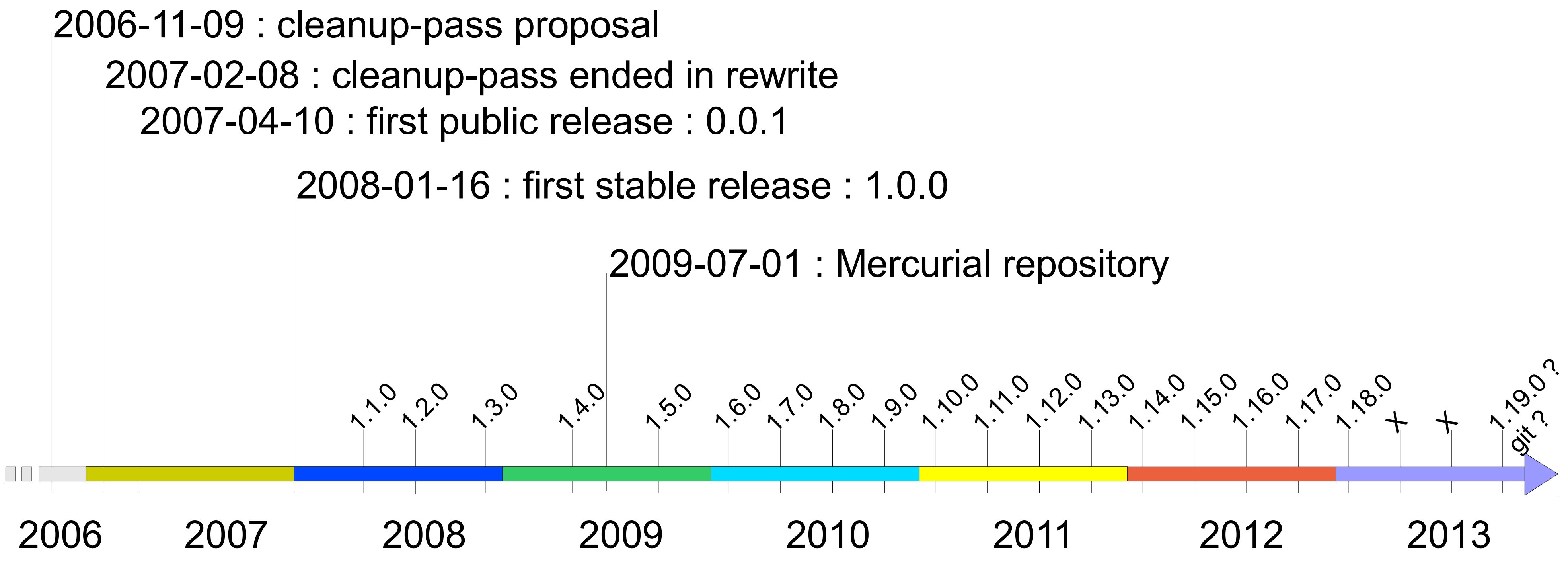
Crosstool-NG

A (cross-)toolchain generator

Yann E. MORIN
yann.morin.1998@free.fr
http://ymorin.is-a-geek.org/

History

- 2005..2006 : working with crosstool
- Needed newer versions
- Needed uClibc support


2006-11-09 : cleanup-pass proposal
2007-02-08 : cleanup-pass ended in rewrite
2007-04-10 : first public release : 0.0.1
2008-01-16 : first stable release : 1.0.0
2009-07-01 : Mercurial repository
1.1.0
1.2.0
1.3.0
1.4.0
1.5.0
1.6.0
1.7.0
1.8.0
1.9.0
1.10.0
1.11.0
1.12.0
1.13.0
1.14.0
1.15.0
1.16.0
1.17.0
1.18.0
X
X
1.19.0 ?
git ?
2006
2007
2008
2009
2010
2011
2012
2013

Purpose

- Build toolchains
- **Only** build toolchains

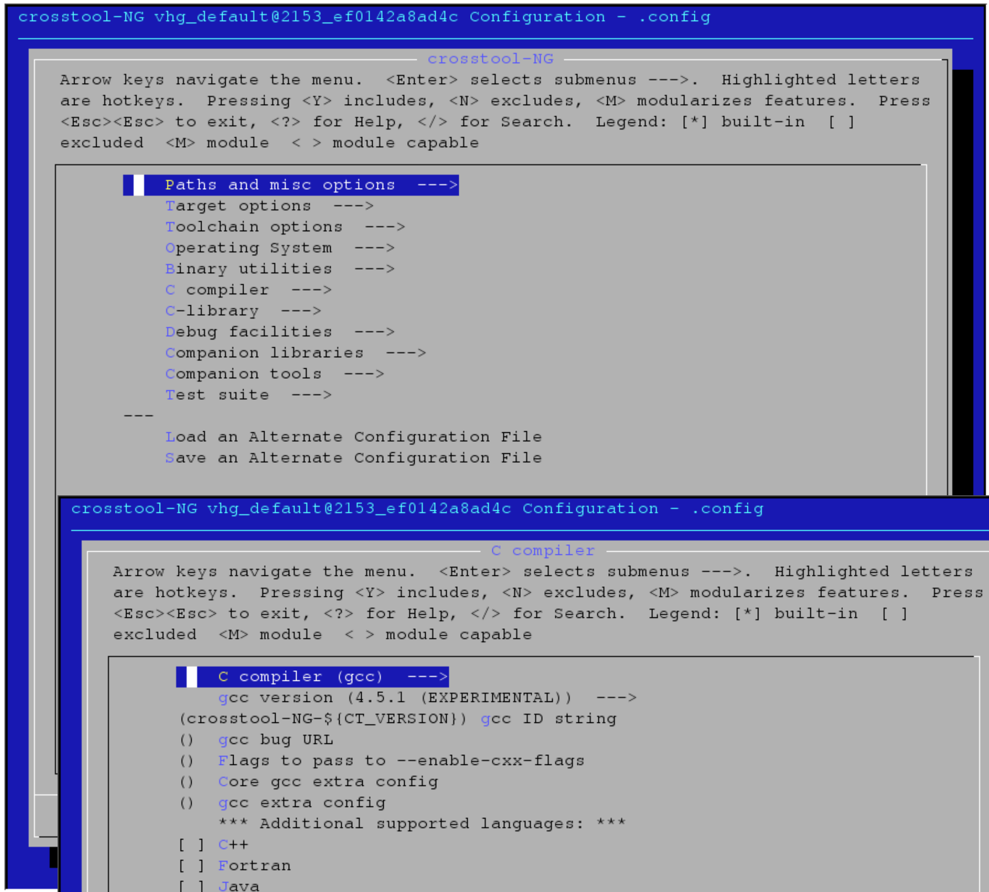
Goals

- Easy to use
- Easy to maintain
- Easy to enhance
- Act as a tutorial

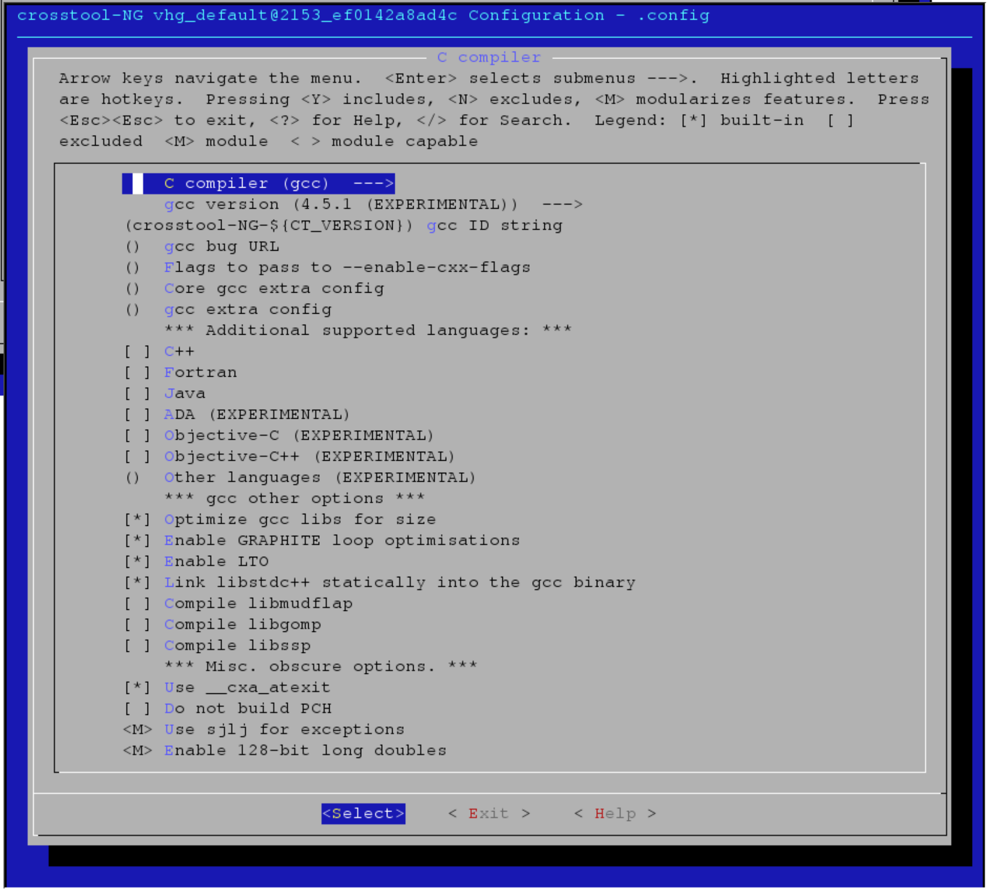
Make it easy to use : standard behaviour

- Standard package
  - ./configure
  - make
  - make install
- Toolchain configuration
  - menuconfig
  - samples
- Toolchain build
  - step by step
  - simplified log

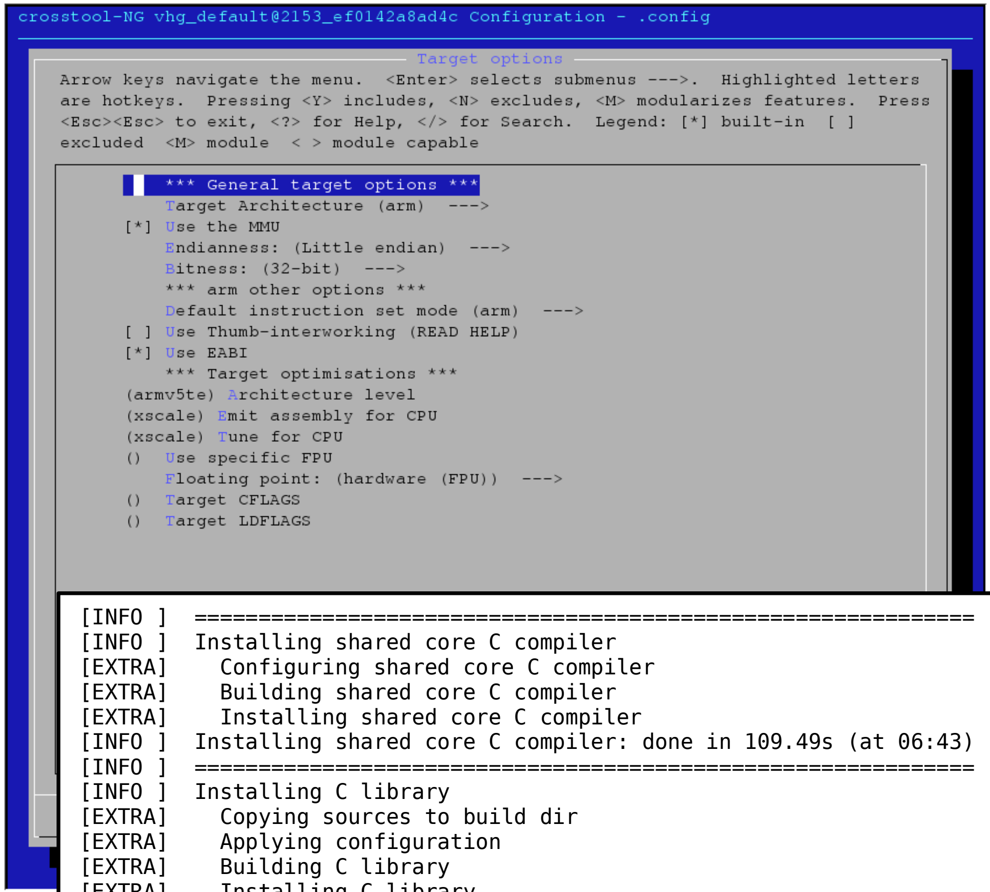
Kernel Recipes – September 2013 - crosstool-NG



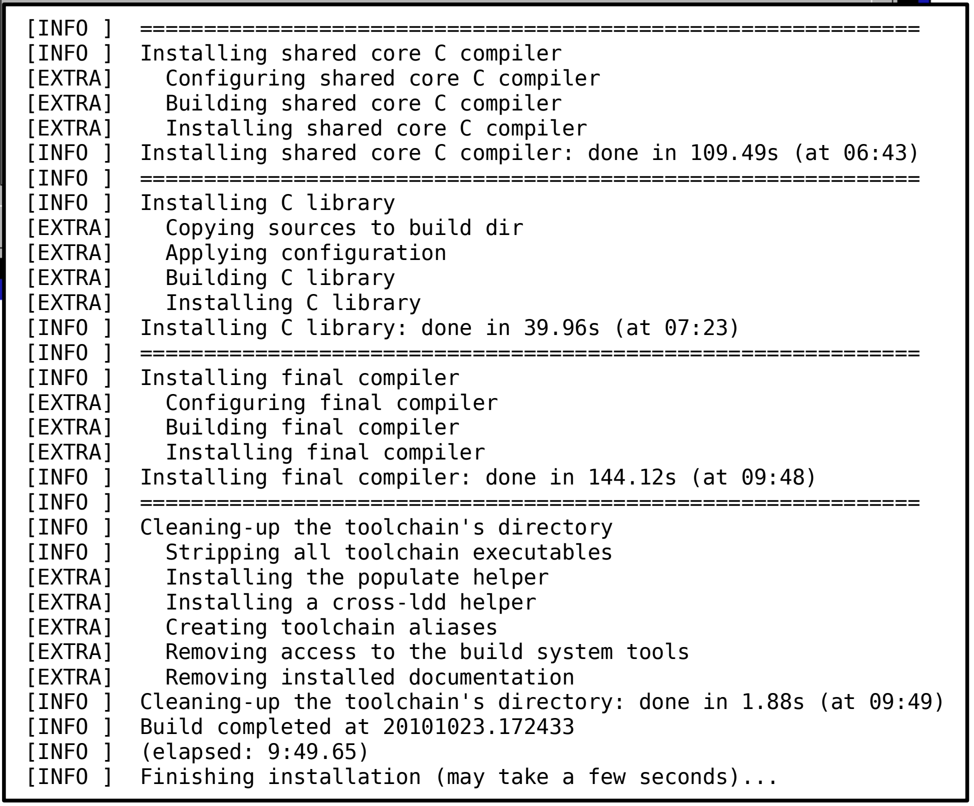
```
crosstool-NG vhg_default@2153_ef0142a8ad4c Configuration - .config
                              crosstool-NG
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters
 are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
 <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
 excluded  <M> module  < > module capable

        Paths and misc options  --->
        Target options  --->
        Toolchain options  --->
        Operating System  --->
        Binary utilities  --->
        C compiler  --->
        C-library  --->
        Debug facilities  --->
        Companion libraries  --->
        Companion tools  --->
        Test suite  --->
    ---
        Load an Alternate Configuration File
        Save an Alternate Configuration File
```



```
crosstool-NG vhg_default@2153_ef0142a8ad4c Configuration - .config
                              C compiler
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters
 are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
 <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
 excluded  <M> module  < > module capable

        C compiler (gcc)  --->
        gcc version (4.5.1 (EXPERIMENTAL))  --->
    (crosstool-NG-${CT_VERSION}) gcc ID string
    ()  gcc bug URL
    ()  Flags to pass to --enable-cxx-flags
    ()  Core gcc extra config
    ()  gcc extra config
        *** Additional supported languages: ***
    [ ] C++
    [ ] Fortran
    [ ] Java
    [ ] ADA (EXPERIMENTAL)
    [ ] Objective-C (EXPERIMENTAL)
    [ ] Objective-C++ (EXPERIMENTAL)
    ()  Other languages (EXPERIMENTAL)
        *** gcc other options ***
    [*] Optimize gcc libs for size
    [*] Enable GRAPHITE loop optimisations
    [*] Enable LTO
    [*] Link libstdc++ statically into the gcc binary
    [ ] Compile libmudflap
    [ ] Compile libgomp
    [ ] Compile libssp
        *** Misc. obscure options. ***
    [*] Use __cxa_atexit
    [ ] Do not build PCH
    <M> Use sjlj for exceptions
    <M> Enable 128-bit long doubles

              <Select>    < Exit >    < Help >
```



```
crosstool-NG vhg_default@2153_ef0142a8ad4c Configuration - .config
                              Target options
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters
 are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
 <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
 excluded  <M> module  < > module capable

        *** General target options ***
        Target Architecture (arm)  --->
    [*] Use the MMU
        Endianness: (Little endian)  --->
        Bitness: (32-bit)  --->
        *** arm other options ***
        Default instruction set mode (arm)  --->
    [ ] Use Thumb-interworking (READ HELP)
    [*] Use EABI
        *** Target optimisations ***
    (armv5te) Architecture level
    (xscale) Emit assembly for CPU
    (xscale) Tune for CPU
    ()  Use specific FPU
        Floating point: (hardware (FPU))  --->
    ()  Target CFLAGS
    ()  Target LDFLAGS
```



```
[INFO ]  =================================================================
[INFO ]  Installing shared core C compiler
[EXTRA]    Configuring shared core C compiler
[EXTRA]    Building shared core C compiler
[EXTRA]    Installing shared core C compiler
[INFO ]  Installing shared core C compiler: done in 109.49s (at 06:43)
[INFO ]  =================================================================
[INFO ]  Installing C library
[EXTRA]    Copying sources to build dir
[EXTRA]    Applying configuration
[EXTRA]    Building C library
[EXTRA]    Installing C library
[INFO ]  Installing C library: done in 39.96s (at 07:23)
[INFO ]  =================================================================
[INFO ]  Installing final compiler
[EXTRA]    Configuring final compiler
[EXTRA]    Building final compiler
[EXTRA]    Installing final compiler
[INFO ]  Installing final compiler: done in 144.12s (at 09:48)
[INFO ]  =================================================================
[INFO ]  Cleaning-up the toolchain's directory
[INFO ]    Stripping all toolchain executables
[EXTRA]    Installing the populate helper
[EXTRA]    Installing a cross-ldd helper
[EXTRA]    Creating toolchain aliases
[EXTRA]    Removing access to the build system tools
[EXTRA]    Removing installed documentation
[INFO ]  Cleaning-up the toolchain's directory: done in 1.88s (at 09:49)
[INFO ]  Build completed at 20101023.172433
[INFO ]  (elapsed: 9:49.65)
[INFO ]  Finishing installation (may take a few seconds)...
```

Easy to maintain & enhance : modular design

- Isolate components
  - One config file
  - One build script
  - One patchset
- Define an API
  - Generic: download, extract and patch
  - Specific (component categories):
    - C library: headers & start-files, full
    - Compiler: bootstrap 1, bootstrap 2, final
- Add alternatives
  - Architectures
  - C libraries
  - Kernels
  - ...

Goodies

- *Companion* libraries
  - gmp, mpfr, ppl, CLooG/ppl, mpc, libelf, isl
- *Companion* tools
  - auto-stuff et al.
- Debug tools
  - gdb, gdbserver
  - ltrace, strace
  - dmalloc, D.U.M.A
- Pre-configured sample toolchains

Toolchain types

- Different systems involved

build	system that builds the toolchain
host	system that runs the toolchain
target	system the toolchain generates code for

Native	build == host == target	✗
Cross	build == host != target	✓
Cross-native	build != host == target	✗
Canadian	build != host != target	✓

Status : what's already in?

- 12 Archs
- 5 C libraries
- 2 binary utilities
- 2 kernels
- 1 compiler

- patchset
  - required by many components to build
  - controversy

Pros

- Your choice of components versions
- Optimised for your processor
- Known patchset (if any)
- Upstream fixes easy to apply
- Same sources for all targets
- Reproducible
- Fits your build-system

crosstool-NG for kernel developpers

- Compile-test for others architectures
  - endiannes, bitness...
  - drivers, filesystems, core changes
- Test newer tools
  - Latest optimisations
  - New diagnostics
- Quickly bring up a minimal target system

Future : short- and long-term plans

- Add latest component versions
  - gcc, Linux, C libraries...
- Consolidate or drop backend-mode
  - currently only used by buildroot, dropping
- Consolidate canadian-crosses
  - needed before cross-native, and native
- Look at LLVM / Clang
  - see how it all fits together
- ...

Thank you!

Questions?

Yann E. MORIN
yann.morin.1998@free.fr
http://ymorin.is-a-geek.org/