Build a Native Developement Kit (NDK) with Replicant 4.2

See feature #837 for the discussion about a NDK for Replicant and this page.

Principle of operation

The Android 4.2 source code ships a ndk directory with NDK build scripts.
These tools download various upstream imports from https://git.androidproject.xxx/toolchain/XXX/ with optional commit date (no branches or tags), apply patches, and build them.

Note: as of 2015-07, the upcoming Android version moved to a different build system, and will populate $topdir/toolchain/ through repo instead of invoking git in a temporary directory (i.e. ditched download-toolchain-sources.sh).

Recreate matching build environment

Replicant 4.2 is based on CyanogenMod 10.1 which is based on AOSP 4.2.2, released 2013-02.
The build environment used by NDK release managers should be Ubuntu LTS 12.04 (or maybe Ubuntu LTS 10.04).

To recreate it easily with LXC, follow:

- Ubuntu 12.04 Precise with LXC (Trisquel support in progress)
- Non-privileged user setup

Note: compiling from Debian 8 will yield errors from texinfo (doc in binutils) and Perl (POD in LLVM) due to more recent, stricter versions of these.

Build dependencies

```
# Base Android build dependencies (from https://source.android.com/source/initializing.html, secti
on "Installing required packages (Ubuntu 12.04)")
# Dropping mingw to skip windows builds for now.
# Installing g++-multilib first otherwise apt-get complains.
apt-get install build-essential g++-multilib
apt-get install wget git gnupg flex bison gperf \
  zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \
  libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 \
  libgl1-mesa-dev \
  python-markdown libxml2-utils xsltproc zlib1g-dev:i386

# Additional NDK build dependencies (completed from https://android.googlesource.com/platform/ndk/
+/master/README.md)
apt-get install curl texinfo bison flex libtool pbzip2 groff

# Make 'bash' the default shell otherwise 'build-gabi++.sh' will fail
dpkg-reconfigure dash
```

Preparing the sources

Login as user replicant and prepare the Replicant 4.2 source code as described in ReplicantSourceCode.

Optionally configure ccache:

```
cat <<EOF >> ~/.bashrc
export USE_CCACHE=1
EOF
exec bash

prebuilts/misc/linux-x86/ccache/ccache -M 10G  # unused in Trisquel 4.1?
```

Determining the checkout date

Since there's no NDK release tag for the toolchain/ Git checkouts, we need to find the right date.

cd ndk/ && git log says the last commit is 2012-10-05, so it's around NDK r8b (2012-07) and NDK r8c (2012-11).
Checking build/tools/toolchain-patches/ in the NDKs shows r8c has matching patches.

Also NDKs from google contain several SOURCES files listing Git repos and commit IDs.
With r8c's release date (ls --full-time RELEASE.TXT => 2012-10-22 05:59), build/tools/download-toolchain-sources.sh checkouts the same Git commit IDs, and the patches apply.
Only, r8c references a LLVM Git repo, while our scripts download a tarball.
So our scripts match a NDK version between r8b and r8c, and can almost exactly reproduce r8c. Let's call it r8b2.

Building the NDK

Based on ndk/docs/DEVELOPMENT.html, with much improvement :)

```
export NDK=~/replicant-4.2/ndk
cd $NDK

# Allow checking build errors
export NDK_LOGFILE=$HOME/ndk.log

# Follow Python download redirection
sed -i -e 's/curl -S/curl -L -S/' build/tools/download-toolchain-sources.sh

# Get the sources from toolchain/ repos, with r8c release date:
bash $NDK/build/tools/download-toolchain-sources.sh --git-date='2012-10-22T05:59:40Z' ~/ndk-dl
# ~5mn with good network, ndk-dl: 2.5GB

# Define NDK_TMPDIR and and use --incremental so the build can be resumed on error
export NDK_TMPDIR=~/ndk-tmp
mkdir -p $NDK_TMPDIR/release-r8b2/

# Build the release
bash $NDK/build/tools/make-release.sh --toolchain-src-dir=$HOME/ndk-dl --release=r8b2 --incrementa
l
# 30mn with 4 cores, 28GB (inc. 17GB .repo), 2.5GB ndk-dl, ?GB ndk-tmp, 1GB /tmp
```

The GNU/Linux NDK release is in /tmp/ndk-replicant/release/android-ndk-r8b2-linux-x86.tar.bz2 :)

TODOs

Mirror toolchain/ repos

download-toolchain-sources.sh's default is --git-base=https://android.googlesource.com/toolchain.
Should we use a git.replicant.us repo? (note: there's no 'toolchain' there yet)

Windows build

Build deps:

```
# windows build is triggered by the presence of mingw32
apt-get install mingw32 tofrodos
# and requires running maketab.exe
apt-get install wine
# I had troubles in LXC registering direct .exe execution:
# $ /tmp/ndk-replicant/tmp/build-3038/maketab.exe
# run-detectors: unable to find an interpreter for /tmp/ndk-replicant/tmp/build-3038/maketab.exe
# Work-around:
update-binfmts --disable
dpkg-reconfigure wine1.4 --pri=high  # or wine1.2 for Trisquel 4.1
```

Code fixes:

```
# build-ndk-stack/elff/ produces a warning with mingw32, and uses -Werror
# cf. https://android.googlesource.com/platform/ndk/+/32e74f3f1b969ff65f037e1ee89e21a5cbc0ecf0
sed -i -e 's/-Werror//' sources/host-tools/ndk-stack/GNUMakefile
# TODO: maybe 'export EXTRA_CFLAGS=-Wall' would be enough?
```

...

```
bash $NDK/build/tools/make-release.sh --toolchain-src-dir=$HOME/ndk-dl --release=r8b2 --incrementa
l
# /home/replicant/ndk-dl/build/../binutils/binutils-2.21/gold/arm.cc:2171: internal compiler error
: in make_rtl_for_nonlocal_decl, at cp/decl.c:4971
```

Exact same errors with Trisquel 4.1 (Ubuntu 10.04). For reference, the build dependencies:

```
# Based on https://web.archive.org/web/20121201011547/http://source.android.com/source/initializin
g.html
apt-get install gnupg flex bison gperf build-essential \
  zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs \
  x11proto-core-dev libx11-dev lib32readline5-dev lib32z-dev \
  libgl1-mesa-dev g++-multilib python-markdown \
  libxml2-utils xsltproc
```

No idea how google made the windows build for r8c. Maybe a later ndk.git has fixes?

Use of prebuilts

Check how much of $topdir/prebuilts/ is used to compile the NDK. We shouldn't rely on untrusted binaries.

Other build method : directly use ndk.git and development.git

Attempt to rebuild r8e with the same build environment, but checkout-ing precise .git revisions from ndk.git and development.git: less patches + local LLVM repo + precise control over the ndk scripts revision

According to ndk/docs/DEVELOPMENT.html one only needs:
git clone https://git.androidproject.xxx/platform/ndk.git ndk
git clone https://git.androidproject.xxx/platform/development.git development

We wouldn't rely on exactly the build scripts shipped in Replicant 4.2, but official NDK releases are not sync'd with an Android release either.
This would allow us to reproduce an existing NDK release precisely.

Other build method : rely on previously built NDK

ndk/docs/DEVELOPMENT.html references using an existing (trusted) NDK to build the next one.
Maybe this is how the NDK release team worked around the windows build errors (by not recompiling everything)?