Brown Deer Technology

Application Note: Building COPRTHR SDK for Android

Copyright (c) 2013 Brown Deer Technology, LLC

Verbatim copying and distribution of this entire document is permitted in any medium, provided this notice is preserved.

1 Introduction

The COPRTHR SDK has been successfully built using the Android Native Development Kit (NDK) targeting ARM processors.
This build has been done using the dev-android branch found in COPRTHR SDK git repository.

Warning: Using the Android NDK is complicated, and Linux or BSD developers will find that Android support for compiling native code is relatively immature. Executing your code is equally complicated.

For developers who wish to push ahead despite this warning, the following instructions describe how to successfully build the COPRTHR SDK for an Android platform. These instructions are subject to change and the present support should be considered experimental at this time.

1.1 Preparing the Android NDK Environment

First, download the android-ndk-r8e package from http://developer/android.com/tools/sdk/ndk/index.html and unpack it in a work directory which we refer to as $WORK. The package should unpack to a root directory called $WORK/android-ndk-r8d/ which we will refer to as $NDK.

Next create a directory $WORK/android-install/ to use as a target for the installation of the packages that will be built. Since this is a cross-compile you do not want any packages installing on your host system.

Next we must create symlinks for *all* of the cross-compilation tools we will need so that they will have the proper names. By default these tools have names that a normal build system will not recognize. (If you like to make things more difficult than they have to be, skip this step and try to tell the packages to use the cross-compilation tools directly When that fails, rename them as shown in this step.) Go to

```
$NDK/toolchains/arm-linux-androideabi-4.7/prebuilt/linux-x86/bin
```

and create links for the following programs so they may be recognized by their conventional commands: cc, cpp, gcc, g++, ld, nm, objcopy, objdump, readelf. For example,

```
] cd $NDK
] ln -s arm-linux-androideabi-gcc cc
```

The cross-compilation will be accomplished by adding $NDK/toolchains/arm-linux-androideabi-4.7/prebuilt/linux-x86/bin to your PATH and using –sys-root flag to the compiler and preprocessor. The most convenient way to accomplish the former is to create a small shell script to modify your PATH and change your command line prompt to let you know you are in your “cross-compilation environment” to avoid confusion if you are easily confused.

```
export PATH=$NDK/toolchains/arm-linux-androideabi-4.7/prebuilt/linux-x86/bin:$PATH
PS1=NDK]
```

Assuming you are using bash as your shell, and that the above shell script is called ndk.sh, source this file and you should now find the command prompt to be

```
NDK]
```

If you echo $PATH you should find that it is modified.

There are three prerequisites that must be installed prior to building COPRTHR SDK, namely, libelf-0.8.13, libconfig-1.4, and libevent-2.0. You may *not* substitute libelf-1.x since the required functionality will not work properly. Specific information regarding the exact packages required is provided in the COPRTHR SDK release notes.

1.2 Building libelf-0.8.13 for Android.

Unpack the package in the work directory $WORK and cd into the top directory. Building libelf for Android requires a small patch and adding a header that is missing in the Android NDK. First, apply the patch below to modify the files libelf.h and hash.c.

hash.c.patch:

```
/hash.c libelf-0.8.13/lib/hash.c
--- orig/libelf-0.8.13/lib/hash.c   2008-05-23 04:15:35.000000000 -0400
+++ libelf-0.8.13/lib/hash.c    2013-05-02 15:38:59.000000000 -0400
@@ -23,7 +23,11 @@
 static const char rcsid[] = "@(#) $Id: hash.c,v 1.10 2008/05/23 08:15:35 michael Exp $";
 #endif /* lint */

+#if defined (__ANDROID__)
+unsigned int
+#else
 unsigned long
+#endif
 elf_hash(const unsigned char *name) {
     unsigned long hash = 0;
     unsigned long tmp;
```

libelf.h.patch:

```
/libelf.h libelf-0.8.13/lib/libelf.h
--- orig/libelf-0.8.13/lib/libelf.h 2009-07-07 13:57:43.000000000 -0400
+++ libelf-0.8.13/lib/libelf.h  2013-05-02 15:37:17.000000000 -0400
@@ -199,7 +199,11 @@
 extern Elf32_Phdr *elf32_getphdr __P((Elf *__elf));
 extern Elf_Scn *elf_getscn __P((Elf *__elf, size_t __index));
 extern Elf32_Shdr *elf32_getshdr __P((Elf_Scn *__scn));
+#if defined(__ANDROID__)
+extern unsigned int elf_hash __P((const unsigned char *__name));
+#else
 extern unsigned long elf_hash __P((const unsigned char *__name));
+#endif
 extern Elf_Kind elf_kind __P((Elf *__elf));
 extern size_t elf_ndxscn __P((Elf_Scn *__scn));
 extern Elf_Data *elf_newdata __P((Elf_Scn *__scn));
```

Next, "borrow" the file ar.h from a real Linux operating system using /usr/include/ar.h from Ubuntu 12.04.1 LTD is known to work. Place this header in lib/ Now you are ready to build the package using the following three steps:

```
NDK] CFLAGS="--sys-root=$NDK/platforms/android-14/arch-arm" ./configure --host=arm --prefix=\$WORK/andr
NDK] make
NDK] make install
```

The library and headers will be installed in $WORK/android-install/lib/ and $WORK/android-install/include/ respectively.
Note that only the static library is built and this will be used when building the COPRTHR SDK.

1.3 Building libconfig for Android

Unpack the package in the work directory $WORK and cd into the top directory. No package modifications are required and the build may be done in three steps:

```
NDK] CFLAGS="--sys-root=$NDK/platforms/android-14/arch-arm" \
    CXXFLAGS="--sys-root=$NDK/platforms/android-14/arch-arm" \
    CPPFLAGS="--sys-root=$NDK/platforms/android-14/arch-arm" \
    ./configure --host=arm --prefix=$WORK/android-install \
    --disable-cxx --disable-examples
NDK] make
NDK] make install
```

The library and headers will be installed relative to the work directory in $WORK/lib/ and $WORK/include/ respectively. Note that only the static library is built and this will be used when building the COPRTHR SDK.

1.4 Building libevent for Android

Unpack the package in the work directory $WORK and cd into the top directory. No package modifications are required and the build may be done in three steps:

```
NDK] CFLAGS="--sys-root=$NDK/platforms/android-14/arch-arm" \
    ./configure --host=arm --prefix=\$WORK/android-install
NDK] make
NDK] make install
```

The library and headers will be installed in $WORK/android-install/lib/ and $WORK/android-install/include/ respectively.
Note that only the static library is built and this will be used when building the COPRTHR SDK.

1.5 Building COPRTHR

Unpack the source distribution form the COPRTHR SDK in the work directory $WORK and cd into the top directory. No package modifications are required and the build may be done in three steps:

```
NDK] CFLAGS="--sys-root=$NDK/platforms/android-14/arch-arm" \
    ./configure --host=arm --prefix=$WORK/android-install \
    --with-libelf=$WORK/android-install \
    --with-libconfig=$WORK/android-install \
    --with-libevent=$WORK/android-install \
    --disable-cltrace --enable-android-cross-compile
NDK] make
NDK] make install
```

The library and headers will be installed relative to the directory $WORK/android-install .