**Simple example:**

Before describing the syntax in details, let's consider the simple "hello JNI" example, i.e. the files under:

apps/hello-jni/project

Here, we can see:

- The 'src' directory containing the Java sources for the sample Android project.

- The 'jni' directory containing the native source for the sample, i.e. 'jni/hello-jni.c'

This source file implements a simple shared library that implements a native method that returns a string to the

VM application.

- The 'jni/Android.mk' file that describes the shared library to the NDK build system. Its content is:

LOCAL\_PATH := $(call my-dir)

include $(CLEAR\_VARS)

LOCAL\_MODULE := hello-jni

LOCAL\_SRC\_FILES := hello-jni.c

include $(BUILD\_SHARED\_LIBRARY)

Now, let's explain these lines:

LOCAL\_PATH := $(call my-dir)

An Android.mk file must begin with the definition of the LOCAL\_PATH variable.

It is used to locate source files in the development tree. In this example, the macro function 'my-dir', provided by the build system, is used to return

the path of the current directory (i.e. the directory containing the Android.mk file itself).

include $(CLEAR\_VARS)

The CLEAR\_VARS variable is provided by the build system and points to a special GNU Makefile that will clear many LOCAL\_XXX variables for you

(e.g. LOCAL\_MODULE, LOCAL\_SRC\_FILES, LOCAL\_STATIC\_LIBRARIES, etc...), with the exception of LOCAL\_PATH. This is needed because all build

control files are parsed in a single GNU Make execution context where all variables are global.

LOCAL\_MODULE := hello-jni

The LOCAL\_MODULE variable must be defined to identify each module you describe in your Android.mk. The name must be \*unique\* and not contain

any spaces. Note that the build system will automatically add proper prefix and suffix to the corresponding generated file. In other words,

a shared library module named 'foo' will generate 'libfoo.so'.

IMPORTANT NOTE:

If you name your module 'libfoo', the build system will not add another 'lib' prefix and will generate libfoo.so as well.

This is to support Android.mk files that originate from the Android platform sources, would you need to use these.

LOCAL\_SRC\_FILES := hello-jni.c

The LOCAL\_SRC\_FILES variables must contain a list of C and/or C++ source files that will be built and assembled into a module. Note that you should

not list header and included files here, because the build system will compute dependencies automatically for you; just list the source files

that will be passed directly to a compiler, and you should be good.

Note that the default extension for C++ source files is '.cpp'. It is however possible to specify a different one by defining the variable

LOCAL\_CPP\_EXTENSION. Don't forget the initial dot (i.e. '.cxx' will work, but not 'cxx').

include $(BUILD\_SHARED\_LIBRARY)

The BUILD\_SHARED\_LIBRARY is a variable provided by the build system that points to a GNU Makefile script that is in charge of collecting all the

information you defined in LOCAL\_XXX variables since the latest 'include $(CLEAR\_VARS)' and determine what to build, and how to do it

exactly. There is also BUILD\_STATIC\_LIBRARY to generate a static library.

There are more complex examples in the samples directories, with commented Android.mk files that you can look at.