Configuración de un sistema de compilación multiplataforma con distcc y aplicación de este en un entorno distribuido

Diego Martín Arroyo

21 de abril de 2015

Índice

Introducción

Uno de los graves inconvenientes a la hora de desarrollar software para el sistema distribuido es la pequeña capacidad de cómputo individual de cada nodo. Si bien esta circunstancia no supone un inconveniente significativo a la hora la ejecución, supone un grave obstáculo a la hora de realizar procesos de compilación, en particular, de paquetes de software voluminosos, tales como pueden ser bibliotecas de código, cuyo tiempo de compilación en una Raspberry Pi puede alcanzar varias horas.
Diversas soluciones han sido propuestas para solucionar este problema, siendo una de las más simples y efectivas la compilación cruzada. Un compilador cruzado es capaz de generar código ejecutable en el juego de instrucciones de otra máquina, permitiendo aprovechar las ventajas de otro equipo (en particular la mayor capacidad de cálculo) para agilizar el proceso de compilación. En el caso concreto del sistema, el equipo compilador cuenta con un procesador de arquitectura i686 con una capacidad de cómputo muy superior al del chip ARM presente en los nodos del sistema. Conseguir realizar compilaciones en este equipo facilitaría el trabajo de administración del sistema, la instalación de nuevo *software* e incluso el desarrollo de código.

crostool-ng

Crosstool-ng[1] es una utilidad para la generación de cadenas de herramientas (*toolchains*) tales como GCC *GNU Compiler Collection*, que permiten compilar y enlazar códigos fuente. El proyecto tiene como objetivo crear *toolchains* ajustadas a cada necesidad específica, ofreciendo un gran número de parámetros de configuración con el objetivo de optimizar al máximo el proceso de compilación y adaptarse a las necesidades de los usuarios. **Crosstool-ng** es un proyecto cuyo origen se remonta a la herramienta crosstool de Daniel Kegel[2].

Configuración de crosstool-ng

La configuración de crosstool-ng es sencilla una vez se conocen los diferentes parámetros a tener en cuenta. Para poder realizar compilaciones en nuestro sistema es necesario un *toolchain* con las siguientes características:

- Capaz de compilar en la arquitectura **ARMv7** de 32 bits con alineamiento de bits *little endian*. El procesador de la Raspberry implementa operaciones con números en coma flotante en el *hardware*, por lo que es posible aprovechar esta característica en los programas a compilar.
- Un compilador de C
- Compilador de C++
- Compilador de FORTRAN (Opcional)
- Bibliotecas estándar (librería estándar)

Para crear este compilador es necesario realizar los siguientes pasos:

1. Descarga de los archivos desde el sitio del proyecto [3]. El compilador se ha creado con la versión **1.20.0**, la más reciente durante la creación de este documento, publicada el 8 de septiembre de 2014.
2. Instalación de las dependencias de **crosstool-ng**:
   `gawk bison flex gperf cvs texinfo automake libtool ncurses-dev g++ subversion python-dev libexpat1-dev cmake`

[1] Existe una amplia cantidad de información sobre el proyecto en su sitio web, crosstool-ng.org
[2] http://kegel.com/crosstool/
[3] www.crosstool-ng.org/download/crosstool-ng

3. Descomprimir el fichero y realizar la configuración del mismo con `./configure -prefix=/opt/cross` (el directorio de compilación puede ser modificado). Ejecutar el comando `make` y `make install`.

4. Añadir a la variable `PATH` la ruta `/opt/cross/bin`. Una vez realizados estos pasos la herramienta queda instalada y se podrá crear *toolchains*.

5. Crear un directorio en el que se almacenarán los diferentes archivos auxiliares durante el proceso de creación del compilador.

6. Ejecutar `ct-ng menuconfig`, que presentará una serie de opciones en un panel interactivo:

   - En la opción *Paths and misc options*, activar la opción "Enable Try features marked as EXPERIMENTAL", y modificar el valor *Number of parallel jobs* a un valor menor o igual al doble de núcleos presentes en el procesador del servidor. También es posible modificar el directorio de despliegue.
   - En el menú *Target options*, seleccionar "arm" como *Target architecture bitness* como 32 bit y *Endianness* como *Little endian*
   - *Operating System*: Linux
   - Elegir la opción estable de *Binutils version* más reciente en el menú *Binutils*
   - Elegir la opción para *gcc linaro* cuya versión sea más reciente en el menú *C Compiler*. Activar además la compilación para los lenguajes **C++**, **FORTRAN** y **Java** si se desea.
   - Activar la opción *gdb* en *Debug Facilities*.

7. Una vez realizados estos pasos es necesario ejecutar el comando `ct-ng build` que generará el conjunto de herramientas. El proceso de creación ha llegado a requerir una hora de procesado. Tras el proceso de compilación se contará con los ejecutables en el directorio `/opt/cross/x-tools/arm-unknown-linux-gnueabi/bin` o en aquella ruta especificada.

Alternativas

En los archivos adjuntos a la documentación se encuentra un *toolchain* generado para sistemas Linux en arquitecturas i386 y x86_64, así como los archivos de configuración para la creación de los mismos.
En el caso de utilizar Arch ARM y la arquitectura del equipo a utilizar como compilador sea x86_64 es posible descargar los parámetros de configuración desde la web del proyecto Arch Linux ARM o los propios ejecutables precompilados[4].

[4] Archivos .config:

- ARMv5: archlinuxarm.org/mirror/development/ct-ng/xtools-dotconfig-v5
- ARMv6: archlinuxarm.org/mirror/development/ct-ng/xtools-dotconfig-v6
- ARMv7: archlinuxarm.org/mirror/development/ct-ng/xtools-dotconfig-v7

Toolchains precompilados:

- ARMv5 *soft float*: http://archlinuxarm.org/builder/xtools/x-tools.tar.xz
- ARMv6 *hard float*: http://archlinuxarm.org/builder/xtools/x-tools6h.tar.xz
- ARMv7 *hard float*: http://archlinuxarm.org/builder/xtools/x-tools7h.tar.xz

También es posible utilizar el paquete **distccd-alarm**, que realiza todos los pasos necesarios para instalar **distccd** para todas las arquitecturas de forma simplificada. Únicamente válido para sistemas **Arch Linux** ejecutados en una arquitectura x86_64: github.com/WarheadsSE/PKGs/tree/master/distccd-alarm.

Uso

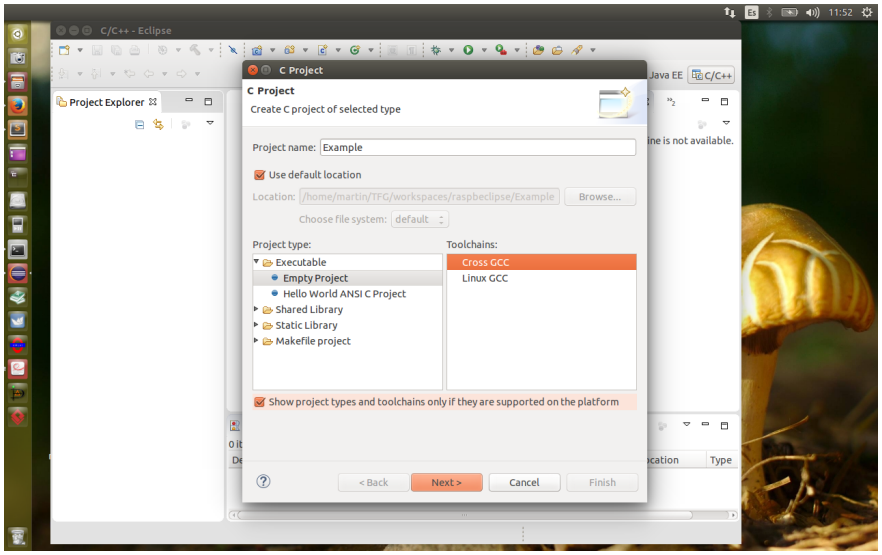
Ejemplo de ejecución:

```
$ arm-unknown-linux-gnueabi-gcc --version
arm-unknown-linux-gnueabihf-gcc (crosstool-NG 1.20.0) 4.9.2
arm-unknown-linux-gnueabihf-gcc main.c -o main
$
```

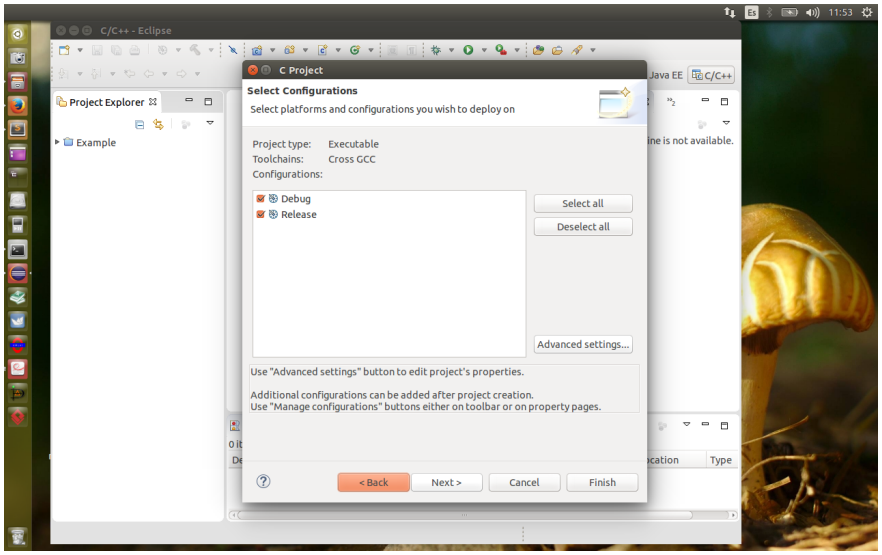
Integración con un entorno de desarrollo

Es posible integrar el compilador cruzado en un entorno de desarrollo integrado como **Eclipse**[5], siguiendo la siguiente secuencia de pasos. Es necesario contar con los *plugins* **C/C++ Development Tools**, **C/C++ Development Tools SDK** y **C/C++ GCC Cross Compiler Support**.

1. Creación de un nuevo proyecto para el lenguaje elegido (es importante que se haya creado el compilador para ese lenguaje), indicando que el tipo de compilador será cruzado (opción **Cross GCC**). Se deben seleccionar las opciones de configuración para el desarrollo en la siguiente ventana..

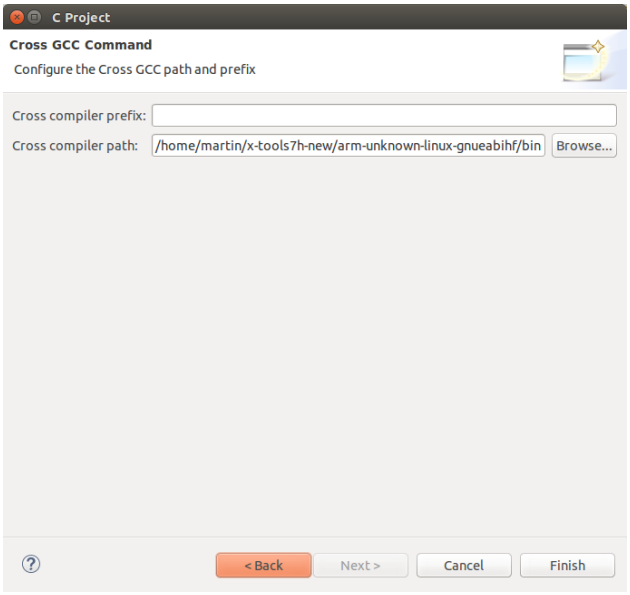


[5] http://eclipse.org/



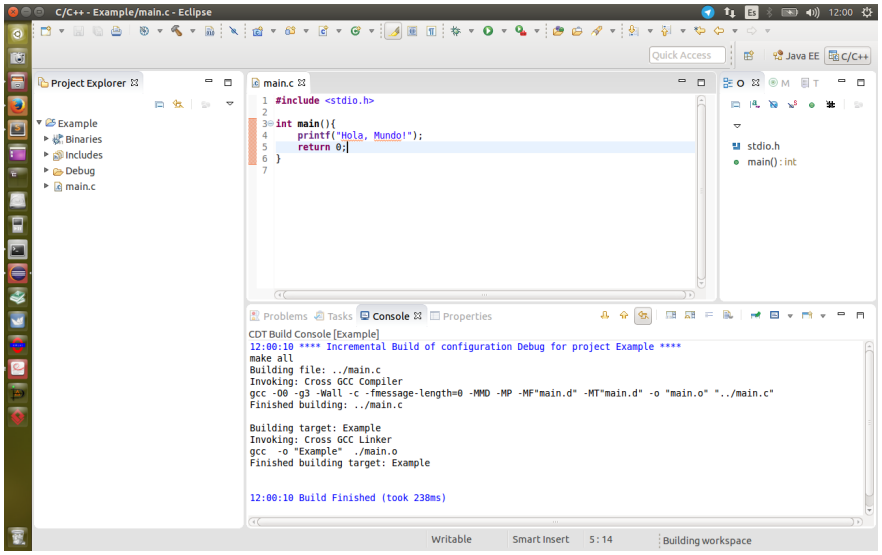
2. Definición de la ruta del compilador.

   Se especifican la ruta donde se encuentran los diferentes ejecutables del compilador, indicando el prefijo que les acompaña (Eclipse realizará una llamada convencional al compilador como si se tratase de un proyecto típico, por lo que para identificar cada uno de los ejecutables, se debe indicar dicho prefijo, si lo hay). En este caso, dado que se han creado enlaces simbolicos que no incluyen el prefijo, no es necesario indicarlo.



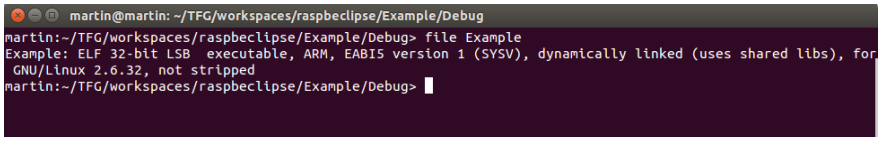
3. Creación del código y compilación.

   Se añaden o crean los archivos a compilar y se ejecuta la acción **Build**. La consola mostrará el resultado de la operación.



4. Evaluación del resultado.

   Utilizando la herramienta **file** es posible verificar la arquitectura del ejecutable.



distcc

La compilación distribuida consiste en una serie de máquinas dedicadas a realizar procesos de compilación comandadas por una serie de nodos denominados maestros, que envían una serie de instrucciones de compilado a la máquina remota para que sean ejecutadas, retornando el resultado de dicho proceso de compilación. Distcc es una herramienta que actúa como demonio del sistema, escuchando peticiones de compilación de forma continua.

Problemas conocidos

Muchas de las versiones de **crosstool-ng** presentes en su repositorio presentan problemas debido a que los archivos que necesita descargar ya no se encuentran disponibles. La versión utilizada en el sistema es la 1.20.0, liberada el 8 de septiembre de 2014.
En ocasiones el repositorio subversion de EGlibC presenta un error en la descarga, debido a la sobrecarga de su servidor. En este caso es necesario utilizar Glibc como biblioteca estándar para C. Además, el desarrollo de Eglibc ha sido interrumpido, debido a que los desarrolladores del proyecto consideran que los objetivos del mismo ya han sido cumplidos y se han integrado en Glibc, por lo que es recomendable utilizar esta versión en futuras versiones del compilador.

Referencias

[1] C. Boot, "How to build a cross compiler for your Raspberry Pi." http://www.bootc.net/archives/2012/05/26/how-to-build-a-cross-compiler-for-your-raspberry-pi/, July 2013.

[2] pointclouds.org, "Using DistCC to speed up compilation." http://pointclouds.org/documentation/advanced/distcc.php.

[3] J. Hall, "Distributed Compiling with distcc." http://www.linuxjournal.com/article/9814, oct 2007.

[4] M. Pool, W. Davison, D. Papadopoulos, F. Raabe, *et al.*, "distcc manual page." http://distcc.googlecode.com/svn/trunk/doc/web/man/distcc_1.html#TOC_20, may 2008.

[5] M. Pool, "distcc frequently asked questions." http://distcc.googlecode.com/svn/trunk/doc/web/faq.html, may 2008.

[6] J. Nicola, "Cross-compilation & Distributed compilation for the Raspberry Pi." , dec 2012.

[7] M. Borgerson, "Cross-Compiling for the Raspberry Pi." https://mborgerson.com/cross-compiling-for-the-raspberry-pi/, jul 2013.