Construcción de un sistema GNU para plataforma SPARC de 64 bits

Hugo J. Curti <hcurti@exa.unicen.edu.ar>
INCA/INTIA - Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Provincia de Buenos Aires
Campus Universitario - Pje. Arroyo Seco s/n (7000) Tandil - (02293)432466

Resumen

El presente artículo tiene como objetivo describir la experiencia obtenida en la construcción de un sistema GNU completo sobre plataforma SPARC. Dicho sistema incluye todas las utilidades necesarias para formar una estación de trabajo, una estación de desarrollo o un servidor. Incluye un kernel del sistema operativo LINUX con sus módulos, las librerías más importantes, todos los comandos básicos de UNIX, el ambiente de desarrollo completo, el sistema XFREE86 completo y el ambiente de escritorio GNOME.

1 Introducción

Un sistema GNU está formado por un conjunto de programas aportados por diferentes desarrolladores en todo el mundo. Estos programas tienen en común que se encuentran protegidos por la licencia GNU. Las características fundamentales de esta licencia son las siguientes:

- Protege los derechos de autor de los programas.
- Garantiza la distribución del código fuente de los programas.
- Permite el libre uso y distribución de los programas.

Estas condiciones de licencia facilitaron la orientación de las herramientas de la llamada *cadena de desarrollo* para diferentes plataformas y arquitecturas. A su vez, todo programa preparado para construirse con esta cadena de desarrollo puede construirse con poco o ningún esfuerzo extra en todas las plataformas hacia las cuales dicha cadena haya sido orientada. Esta operación de orientación de un programa a diferentes plataformas se denomina *portado*.

El kernel del sistema operativo, como cualquier otro programa, también puede ser portado. Sin embargo, las partes del mismo que interactúan directamente con el hardware, denominadas *drivers*, requieren código específico para cada plataforma, lo cual requiere un esfuerzo de desarrollo considerable. Algo similar ocurre con el servidor de terminal gráfica, que requiere código específico para cada controladora de video.

Como corolario de lo expuesto, se puede decir que para portar un sistema GNU a una plataforma es necesario que exista el código necesario para la orientación tanto en la cadena de desarrollo como en el kernel del sistema operativo. Si se desea portar la interfaz gráfica, se requiere además que el driver para la controladora de video esté presente en el servidor de terminal gráfica.

En la sección 2 se describen los componentes principales de un sistema GNU y su configuración. En la sección 3 se revisan los aspectos relevantes a este trabajo de la arquitectura SPARC y sus modalidades. En la sección 4 se describe la distribución ROCK LINUX, elegida como base para este trabajo. En la sección 5 se muestra el orden utilizado para desarrollar el trabajo. Por úl-

timo, en la sección 6 se presentan las conclusiones del trabajo.

2 Configuración de los programas GNU

Debido a la diversidad de arquitecturas, de plataformas y de sistemas que existen en plaza, para que un código pueda construirse en cualquiera de ellos se requiere una adaptación de dicho código a cada sistema particular. Este proceso de adaptación fue automatizado en la mayoría de los programas GNU por una herramienta denominada *autoconf*. A su vez, existe otra herramienta que automatiza el proceso de construcción de código denominada *make*.

2.1 La herramienta autoconf

La herramienta *autoconf* es utilizada por los desarrolladores para detectar y resolver problemas comunes de portabilidad en sus programas, a la vez que permite generar un script denominado *configure*.

El código de cada programa se distribuye con su propia versión de *configure*, con lo cual no es necesaria la presencia de la herramienta *autoconf* para realizar la configuración.

Para construir un programa GNU normalmente se debe ejecutar primero el script *configure* que generará los archivos de partida para la herramienta *make*. *Configure* puede recibir parámetros que permiten modificar su comportamiento o ajustar manualmente parámetros que no son correctamente detectados en forma automática.

2.2 La herramienta make

La herramienta *make* es un manejador de dependencias que permite automatizar la construcción y la instalación de un programa. *Make* decide, mediante sus archivos de configuración, la herramienta que se debe invocar para generar cada uno de los objetivos, que normalmente corresponden a los archivos binarios finales de los programas y sus archivos intermedios.

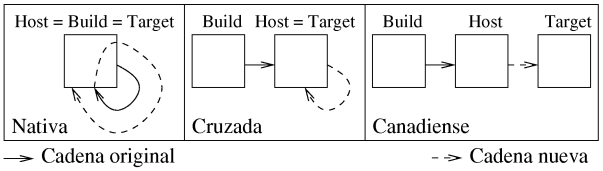

Figura 1: Modalidades de construcción.

2.3 La cadena de desarrollo de GNU

La cadena de desarrollo es la columna vertebral de todo el sistema [Frea]. Está formada por el conjunto de herramientas que permite crear código ejecutable a partir del código fuente de un programa: las utilidades binarias, el compilador, un conjunto de librerías estándar y un depurador.

2.4 Modalidades de construcción de la cadena de desarrollo

Existen tres parámetros que permiten definir la modalidad de construcción de la cadena de desarrollo [Freb]. Estos parámetros se pueden especificar en el script *configure* de las utilidades binarias y del compilador. Las librerías standard y el depurador se construyen utilizando los dos primeros elementos una vez que estén creados.

- El sistema de construcción (*build*) es aquel en el que se va a hacer la construcción de la cadena de desarrollo. Este sistema debe poseer ya una cadena de construcción correctamente configurada e instalada para crear programas en el sistema anfitrión.
- El sistema anfitrión (*host*) es aquel en el cual funcionará la nueva cadena de desarrollo.
- El sistema de destino (*target*) es aquel para el cual se generará código ejecutable.

Se denomina *construcción nativa* (*native build*) a la configuración en la cual los parámetros *build, host* y *target* son iguales. Se denomina *construcción cruzada* (*cross build*) a la configuración en la cual *build* y *host* son iguales y *target* es diferente y *construcción canadiense* (*canadian build*) a la configuración en la cual *build* y *host* son diferentes entre sí independientemente del valor de *target* [How]. La figura 1 muestra gráficamente las tres configuraciones.

3 La arquitectura *sparc*

3.1 Descripción

Para este trabajo se utilizó una estación de trabajo *Sun Ultra Sparc 10* con un procesador *ultrasparc II* que puede trabajar en un modo nativo con un ancho de palabra de 64 bits y en un modo de compatibilidad con procesadores anteriores de la misma familia cuyo ancho de palabra es de 32 bits. Además utiliza para el almacenamiento en memoria de los enteros el formato denominado *big endian*, en el cual se almacenan primero los bytes de mayor peso.

3.2 La cadena de desarrollo para *sparc* y *sparc64*

Los dos modos de trabajo del procesador conforman dos arquitecturas diferentes que requieren cadenas de desarrollo diferentes. La arquitectura del modo de 64 bits se denomina *sparc64,* mientras que del modo de 32 bits se denomina *sparc.* Al momento de la realización de este trabajo la cadena de desarrollo GNU para *sparc* se encontraba en un estado de madurez elevado, con lo que se podía construir perfectamente la cadena completa. No ocurría lo mismo con la versión para *sparc64*, para la cual las utilidades binarias funcionaban, el compilador requería ajustes y las librerías estándar aún no estaban desarrolladas.

3.3 El kernel de Linux para *sparc* y *sparc64*

Para el kernel de Linux tanto la versión *sparc* como la versión *sparc64* estaban desarrolladas al momento de la realización de este trabajo. Sin embargo no es posible utilizar un kernel para sparc en un procesador *ultrasparc II* debido a que este procesador debe ser inicializado con código de 64 bits [Roc02].

La versión 2.95 del compilador (última versión considerada estable al momento del desarrollo de este trabajo) no estaba preparada para construir el kernel en la arquitectura *sparc64*. Por lo tanto no se podía utilizar la vía convencional para construir el kernel. Existían sin embargo dos formas de hacerlo:

- Utilizar una versión especial del compilador denominada *egcs64*, que contiene una serie de *parches* que permiten construir el kernel, aunque no sirve para construir aplicaciones.
- Utilizar la versión 3.0.4 del compilador (considerada inestable para uso general), que fue anunciada mientras este trabajo estaba en progreso y que sirve tanto para construir el kernel como para construir aplicaciones.

4 La distribución ROCK LINUX

Se eligió para este trabajo una distribución denominada ROCK LINUX [SSC00, W$^+$]. Dicha distribución presenta las siguientes ventajas que facilitaron este trabajo [Wol99]:

- Se distribuyen códigos fuentes.
- El proceso de descarga, construcción e instalación se encuentra gobernado exclusivamente por *scripts* de shell que pueden ser seguidos y modificados con facilidad.
- Existe un grupo de trabajo activo que tiene como fin portar la distribución a la plataforma *sparc*.
- Posee una política de manejo de *parches*[1] mínima y eficiente.

4.1 Descarga

Los *scripts* y los archivos de configuración que conforman ROCK LINUX se distribuyen en forma separada de los paquetes. Una vez instalados los *scripts* se puede ejecutar *Download*, que buscará cada paquete por la Internet y lo descargará desde el sitio de su desarrollador y mediante

[1] Un *parche* (*patch*) es un archivo que codifica diferencias entre dos árboles de código fuente. Cuando se hace necesario modificar el código fuente original se genera este archivo parche que puede luego ser aplicado al código original para obtener el código modificado. Además puede ser transferido en forma semiautomática a nuevas versiones de código original.

una herramienta de espejado de web. Cualquier cambio o error que pueda producirse en los sitios puede reflejarse en forma sencilla en los archivos de configuración de la distribución y reintentar el *Download* [Pri02, WB+].

Los paquetes se dividen en dos conjuntos: el conjunto *base* formado por todos los paquetes considerados esenciales y el conjunto *extensiones*, conformado por los paquetes de menos relevancia. Para este trabajo se utilizó el conjunto *base* exclusivamente.

4.2 Construcción

La construcción es completamente automática mientras no haya errores y se realiza en cinco pasos [Pri02]:

- En el primer paso se construye un árbol de directorios básico de UNIX, se construye y se instala la cadena de desarrollo completa en dicho árbol y se configura el kernel.
- En el segundo paso se realiza un *chroot*[2] al árbol recién creado, se reconstruye la cadena de desarrollo, se construye el kernel, el shell y los comandos básicos de UNIX.
- En el tercer paso se reconstruye todo lo anterior y luego se construye el grueso de los paquetes.
- En el cuarto paso se construye la documentación que suele depender de los paquetes construidos en el paso anterior.
- En el quinto paso se reconstruyen todos los paquetes una vez más y se crean las versiones binarias para distribuir.

4.2.1 Errores de compilación

Cuando se producen errores de compilación todo el *script* de construcción se detiene. Entonces se debe revisar y corregir el error en forma manual. Luego se debe generar el *parche* y colocarlo junto con los archivos de configuración de la distribución. El proceso de construcción se reanuda en el punto donde se detuvo.

[2]La llamada *chroot* permite cambiar el directorio raíz de un proceso. Al hacer un cambio de raíz sobre el proceso de construcción, la misma se realizará sobre el mismo árbol donde residirá una vez instalada.

4.2.2 Imagen ISO

Una vez construidos todos los paquetes de la distribución, siempre mediante *scripts*, se puede generar una imagen ISO que se puede grabar en un CD. El CD puede ser utilizado para iniciar un kernel que cargará en un disco emulado en memoria un sistema mínimo y permitirá iniciar la instalación.

Para este trabajo no fue posible crear una imagen ISO de inicio porque la herramienta no funcionaba correctamente para *sparc* en ese momento [Roc02].

4.3 Instalación y configuración

Para instalar se deben crear las particiones y los sistemas de archivos en forma manual. Luego se puede iniciar el *script* de instalación donde se pueden elegir los paquetes a instalar. Los paquetes generarán el mismo árbol de directorios donde fueron construidos. Por último se podrán generar en forma semiautomática o manual las configuraciones de los paquetes principales, quedando de esta forma el sistema instalado.

5 Desarrollo del trabajo

Debido al estado de inmadurez de la cadena de desarrollo en la arquitectura *sparc64* y a la necesaria disposición de un kernel *sparc64* se optó por una configuración mixta, donde el kernel es *sparc64* pero las librerías y todos los programas de usuario son *sparc*.

Al comenzar el trabajo se disponía de la estación de trabajo *Sun Ultrasparc 10* ya descripta con un sistema MANDRAKE LINUX antiguo y sin mantenimiento. El trabajo se desarrolló en los siguientes pasos:

- Se realizó una construcción nativa de la última versión de la cadena completa de desarrollo para *sparc*, utilizando la versión estable del compilador (gcc 2.95).

- Utilizando la cadena nueva se realizó una construcción cruzada de las utilidades binarias y el compilador (versión inestable gcc 3.0.4) hacia *sparc64*. Este paso requirió ajustes en el proceso de construcción del compilador debido a errores en la configuración de la herramienta *make*.

- Utilizando la cadena cruzada se construyó un kernel de Linux *sparc64* con los *drivers* necesarios para manejar la estación de trabajo, reemplazando con éste al viejo kernel del sistema MANDRAKE. La versión del kernel compilado correspondía al de la distribución.

- Utilizando la cadena de desarrollo para *sparc* se construyó la distribución ROCK LINUX completa. Los errores de compilación que fueron presentándose durante la construcción se debieron a problemas de portabilidad, por lo cual se generaron *parches* necesarios para que algunos paquetes puedan funcionar correctamente en *sparc*. Con la excepción de aquellos paquetes cuyos autores indicaban explícitamente que *no funcionaban en sparc*, toda la distribución se pudo compilar exitosamente.

- Ante la imposibilidad de crear la imagen ISO, se preparó una partición nueva en el disco y se corrió el *script* de instalación sobre ella.

- Se reemplazó manualmente el kernel de la distribución (*sparc*) por el kernel en uso y se instaló el cargador manualmente, porque la distribución solamente contemplaba el cargador para las arquitecturas x86.

- Se reinició el nuevo sistema. Se eliminó por completo el viejo sistema y se ordenaron las particiones.

- Se configuraron las aplicaciones y los servidores.

- Por último se instaló el servidor de terminal gráfica, que en un principio no se construía correctamente. Requirió la búsqueda de una serie de *parches* y la adaptación de los mismos a la distribución en uso.

6 Conclusiones

Como resultado de este trabajo se obtuvo un sistema GNU completo funcionando sobre una plataforma *sparc*, basado exclusivamente en software de libre distribución y con una interface que permite que un usuario de un sistema similar funcionando en una arquitectura x86 pueda utilizar la plataforma *sparc* con un costo de migración prácticamente nulo.

Los *parches* obtenidos a partir de este trabajo fueron enviados a [Roc02] para su inclusión en la distribución oficial.

Referencias

[Frea] **Free Software Fundation**. "GCC installation instructions." http://gcc.gnu.org/install.

[Freb] **Free Software Fundation**. "GCC manual." http://gcc.gnu.org/onlinedocs/gcc.

[How] **S. Howard**. "CrossGCC FAQ."

[Pri02] **P. Prins**. "ROCK linux guide." http://www.rocklinux.org, 2002.

[Roc02] "ROCK ports mailing list." rockports@rocklinux.org, 2002.

[SSC00] **SSC**. "Rock linux." *Linux Journal*, página 12, Oct 2000.

[W$^+$] **C. Wolf et al.** "ROCK linux distribution." http://www.rocklinux.org.

[WB$^+$] **C. Wolf, D. Bakke et al.** "ROCK linux FAQ." http://www.rocklinux.org.

[Wol99] **C. Wolf**. "Rock linux philosophy." http://e-zine.nluug.nz/cid=1, Nov 1999.