# Compilación cruzada del kernel linux

## Introduccion

Admito libremente que hay otros HOWTOs sobre este tema, pero quería mostrarte una manera clara que funciona por detras del gran proyecto **Buildroot**, En lugar de hacer toda la configuración del compilador a mano. Como beneficio adicional, puedes apuntar a diferentes arquitecturas con bastante facilidad.

Para esta demostración, construiré un kernel Slackware de 32 bits en una máquina de 64 bits. Sí, sé que potencialmente puedes hacer esto con Multilib (aunque nunca lo he intentado) y otros métodos, Sin embargo, este método es bastante simple, permitirá apuntar a MIPS, m68k, Microblaze, PowerPC, SPARC y Dios sabe qué más, Todos con la misma técnica, y ni siquiera necesitas ser root para hacerlo..

## Acerca de Buildroot

El proyecto Buildroot ha estado funcionando durante algunos años y permite, literalmente, construir un sistema de archivos raíz (duh), Sin embargo, ignoraremos completamente la parte de construcción de rootfs de la ecuación y solo usaremos el compilador cruzado que construye primero. Como tengo una máquina rápida, ni siquiera me molesté en averiguar cómo omitir el paso de generación de rootfs. Solo tienes que construir tu compilador cruzado una vez después de todo.

## Instalando Slackware

Mi máquina de prueba de compilación cruzada es Slackware64 14.2 con conjuntos de discos A, AP, D, K, L, N. Estoy seguro de que puedes instalar mucho menos para construir Buildroot, de hecho, esperaría que solo se necesitaran A y D, sin embargo, el espíritu de la comunidad Slackware es que instalas todo, por lo que el riesgo es tuyo.

## Compilador Cruzado

Obtener la última versión estable de Buildroot desde https://buildroot.org, desempaquetar y configurarlo:

```
$ cd /usr/src
$ wget https://buildroot.org/downloads/buildroot-2018.02.2.tar.gz
$ tar xf buildroot-2018.02.2.tar.gz
$ cd buildroot-2018.02.2
$ make menuconfig
```

Verá un sistema de configuración muy parecido a la configuración del kernel. Para compilar núcleos de 32 bits con máxima compatibilidad, generalmente selecciono la opción de salida 486, (586 es la opción predeterminada):

```
Target options -> Target Architecture Variant (i486)
```

Para hacer que el compilador que estamos a punto de construir se comporte más como Slackware, queremos usar glibc en lugar del uClibc-ng que es el predeterminado (que es más adecuado para aplicaciones integradas):

```
Toolchain -> C library (glibc)
```

Si no hace esto, deberá deshabilitar la protección de la pila en la configuración del kernel lo cuando compilemos y queremos mantener una configuración estándar de Slackware, porque somos verdaderos slakers, ¿verdad, corazón y alma de Slackers? :).

Puedes jugar con muchas otras opciones, como versiones de encabezado de kernel, sin embargo, para la construcción del kernel en sí, nada de esto importa. La única opción que posiblemente pueda hacer una diferencia es la versión de GCC, especialmente si está compilando una versión antigua del kernel que no admite versiones posteriores de GCC. Sin embargo, para esta demostración podemos dejar los valores por defecto. Guarda la configuración y luego:

```
$ make
```

Mientras se ejecuta, configuraremos el kernel para que esté listo para compilar.

## Preparacion del Kernel

Despliegue una configuración del kernel apropiada para construir un kernel Slackware de 32 bits:

```
$ cd /usr/src/linux
$ wget
https://mirrors.slackware.com/slackware/slackware-14.2/kernels/hugesmp.s/con
fig -O .config
```

Ahora debería tener un archivo .config en /usr/src/linux que dice 'CONFIG_64BIT is not set' en la parte superior. Eso reemplazará a su antiguo .config del kernel de 64 bits que tenía antes (incluido en el conjunto de discos 'K'). ¡Obviamente copie el kernel en otro lugar si quiere mantener eso!

## Compilacion del Kernel

Cuando la construccion del Buildroot esta hecha, necesitas incluir el compilador cruzado generado en tu ruta.

```
export PATH=/usr/src/buildroot-2018.02.2/output/host/bin:$PATH
```

El ejecutable del compilador tiene el prefijo de arquitectura en su nombre para evitar la colisión con el sistema GCC, ahora puede ejecutarlo y probar si funciona:

```
$ i486-linux-gcc --version
```

Si está interesado, puede encontrar todas las otras herramientas de la cadena de herramientas como ld, ar y so con prefijos similares. Ahora configura tu kernel si quieres:

```
$ cd /usr/src/linux
$ make menuconfig CROSS_COMPILE=i486-linux- ARCH=i386
```

Dejo las opciones predeterminadas aquí, solo agregué un '-buildroot' al nombre del kernel. Finalmente crear el kernel.:

```
$ make bzImage CROSS_COMPILE=i486-linux- ARCH=i386
```

Copie el kernel construido en una máquina de 32 bits y debería arrancar. Si también desea compilar / instalar los módulos, asegúrese de no olvidar usar las mismas variables CROSS_COMPILE y ARCH cada vez que especifique los comandos make, todo lo debería usar el compilador cruzado:

```
$ make modules CROSS_COMPILE=i486-linux- ARCH=i386
$ make modules_install CROSS_COMPILE=i486-linux- ARCH=i386
```

y así. Probablemente saldrás sin agregar estos comandos, pero algunos comandos como 'make clean', es más seguro incluirlos siempre que realices algún trabajo en ese núcleo, sin duda no te harán daño.

# Sources

- Originally written by User bifferos

howtos, nfs, author bifferos

From:
https://docs.slackware.com/ - **SlackDocs**

Permanent link:
**https://docs.slackware.com/es:howtos:slackware_admin:cross_compiling_the_linux_kernel**

Last update: **2019/03/18 01:22 (UTC)**