Compilación del kernel Linux

LIN - Curso 2020-2021

J.C. Sáez
High Performance Computing Division
ArTeCs Group
Universidad Complutense de Madrid

Compilación del Kernel Linux

Antes de empezar ...

1. Arrancar Máquina Virtual de Debian en vuestro equipo
2. Descargar este documento del Campus Virtual
3. Descargar el fichero de configuración del kernel del Campus Virtual
   - `config-4.9.111-mikernel`
   - Copiarlo en el directorio `~/Descargas` (de la máquina virtual)

- Todos los comandos que usaremos en la clase de hoy habrá que ejecutarlos en MV de Debian
  - El viernes veremos tutorial sobre compilación en puesto de laboratorio...

ArTeCS

Contenido

Contenido

ArTeCS

Compilación tradicional del kernel Linux (I)

Paso 1: Obtener las fuentes

- Vanilla: (The Linux Kernel Archives) www.kernel.org (repositorio git o tarball)
  - https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.9.111.tar.gz
- Las Distros facilitan las fuentes que utilizan (contienen parches).
- Por ejemplo Debian:
  - Instalación de paquetes con `apt-get` o `apt source`
    - Paquete de fuentes de linux-image*
    - Paquete .deb linux-source (instala tarball en /usr/src/)
  - Más información en Debian Linux Kernel Handbook

Compilación tradicional del kernel Linux (II)

Paso 2: Configurar el kernel

- ¿Qué componentes o módulos van a formar parte del kernel?
  - La compilación se extiende a todos aquellos módulos escogidos pero no todos quedarán enlazados en el binario final
- Sistema de configuración sofisticado
  - En Linux, un único árbol de fuentes para:
    1. Múltiples arquitecturas
    2. Sistemas monoprocesador/multiprocesador
    3. Características para servidores, portátiles, sistemas empotrados, ...
- Objetivos
  1. Ajustar Linux a nuestras necesidades y a las de la máquina donde se ejecutará
  2. Reducir el *footprint* del kernel
  3. Reducir el tiempo de compilación

Compilación tradicional del kernel Linux (III)

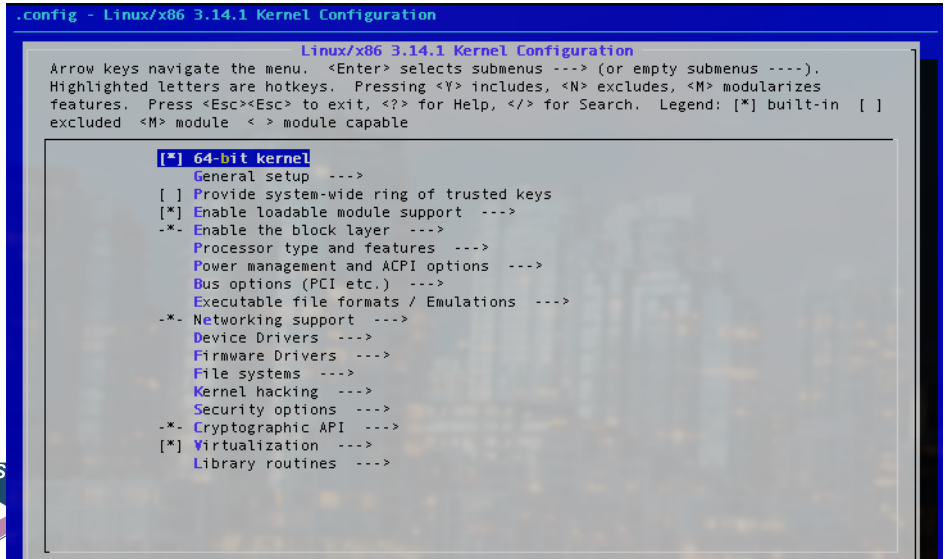
Paso 2: Configurar el kernel (Cont.)

- Gestionado por la herramienta make + scripts (scripts/kconfig)
- Tres alternativas:
  1. Modo interactivo:
     - `make config / menuconfig / xconfig / gconfig`
  2. Modo no interactivo (A) (`make oldconfig`)
     - Si disponemos de un fichero de configuración

```
$ cd ${LINUX_SOURCE}
$ cp /boot/config-4.9.111-lin .config
$ make oldconfig
```

  3. Modo no interactivo (B) - Opciones por defecto (`make defconfig`)
     - Busca fichero .config
     - Si no está presente se escoge la configuración del kernel que esté en ejecución /boot/config-4.x.y...

make menuconfig



Compilación tradicional del kernel Linux (IV)

Paso 3: Compilar el kernel

- `make`
  - Opcionalmente se puede usar con `-j<N>`
    - Resultado: kernel (`vmlinux`) + módulos cargables

Paso 4: Instalación (como root)

1. Instalar módulos en /lib
   - `make modules_install`
2. Instalar imagen del kernel en /boot
   - `make install`
3. Incluir entrada en el gestor de arranque (ej.: GRUB, uboot, UEFI)
   - Dependiente de la versión del gestor de arranque y de la distribución de GNU/Linux

Contenido

Compilación a la Debian (I)

Debian style

- En las fuentes se incluyen scripts para construcción de paquetes del kernel
  - Reemplazan antigua utilidad `make-kpkg`
- No sólo se genera la imagen del kernel y los módulos. También se generan los paquetes correspondientes
  - "linux-image*"
    - La instalación de uno de estos paquetes supone la instalación efectiva del kernel (copia /boot, actualizar entradas gestor arranque, ...)
  - "linux-headers*"
    - Instala en /usr/src/linux-headers* los ficheros de cabecera utilizados en la compilación del kernel
    - Con la macros/defines utilizadas en la compilación
    - Útil para compilar drivers (módulos) adicionales sin necesidad de utilizar todas las fuentes

Compilación a la Debian (II)

Targets específicos de make (después de configurar el kernel)

1. deb-pkg
   - image (kernel y módulos)
   - image-dbg (símbolos de depuración)
     - Si CONFIG_DEBUG_INFO activado
   - headers (cabeceras kernel)
   - firmware-image (firmware para drivers)
   - libc-dev (cabeceras para libc, etc.)
   - source (tar.gz con fuentes)
2. bindeb-pkg
   - Se omiten el paquete source
   - Usaremos este target para compilar el kernel en la asignatura

Compilación a la Debian (III)

bindeb-pkg

- Después de configurar el kernel

```
$ make bindeb-pkg LOCALVERSION=-custom
```

- Puede explotarse paralelismo utilizando `-j<NHILOS>`
  - Se suele escoger número de hilos = #cores +1

Paquetes en Debian

- Dependencias
  - Filosofía Unix: proporcionar herramientas específicas
    - Muchas de las aplicaciones se construyen de forma modular combinando/ampliando herramientas específicas (dependen de)
  - Las herramientas de gestión de paquetes son una parte fundamental de las distros Linux

Paquetes en Debian

- Dependencias
  - Filosofía Unix: proporcionar herramientas específicas
    - Muchas de las aplicaciones se construyen de forma modular combinando/ampliando herramientas específicas (dependen de)
  - Las herramientas de gestión de paquetes son una parte fundamental de las distros Linux

- Paquetes en Debian
  - Binarios .deb
    - Aplicaciones, comandos, librerías, documentación,... pueden estar vacíos (virtuales)
  - Fuentes src
    - Código fuente + scripts (compilación) para generar .deb

Anatomía de un Paquete Binario (I)

- Nombre de un paquete binario
  - Tres campos separados por "_" (no pueden contener "_")
    - Nombre del paquete
    - Versión + revisión Debian/Ubuntu
    - Arquitectura: i386, amd64, ...
- Ejemplos:
  - `sudo_1.7.4p4-2.squeeze.4_amd64.deb`
  - `gcc-4.4-multilib_4.4.5-8_amd64.deb`
- Restaurar nombre: `dpkg-name`

Anatomía de un Paquete Binario (II)

Formato .deb:

- archivo BSD ar
- `ar t foo.deb`
  - Debian-binary
    - Identifica archivo ar como paquete .deb
  - data.tar.gz
    - Tarball con el contenido del paquete
- control.tar.gz * Tarball con la información de control necesaria por el gestor de paquetes
- Ejemplo: extraer linux-image*

```
$ cd /tmp; mkdir decomp; cd decomp
$ ar x linux-image-4.9.111-lin_4.9.111-1_amd64.deb
$ tar xzvf control.tar.gz
```

Anatomía de un Paquete Binario (III)

Ficheros de control (control.tar.gz)

- control: metainformación (único fichero obligatorio)
- conffiles: listado de ficheros de "configuración". Gestión especial para preservar ficheros de configuraciones previas.
- Scripts para la instalación / desinstalación
  - preinst
  - postinst
  - prerm
  - postrm
- md5sums
  - md5 de los ficheros instalados por el paquete
- Interacción con administrador
  - config: obtiene información del admin (debconf db)
  - templates: cuestiones que se presentarán al admin

Gestión de Paquetes dpkg

- dpkg, dpkg-deb, dpkg-query
  - dpkg: instalación y eliminación de paquetes
    - Instalar: `-i` ó `--install`: `--unpack` + `--configure` (postinst)

      ```
      $ dpkg -i <ruta_fichero_deb>
      ```

    - Desinstalar: `-r` ó `--remove` (prerm)

      ```
      $ dpkg -r <nombre_paquete_deb>
      ```

    - `--purge` (postrm)
  - dpkg-deb: manipulación ficheros .deb
    - `--info`, `--contents`, `--field`, `--control`, `--extract`, `--build`
  - dpkg-query: acceso (lectura) bases de datos utilizadas por dpkg (/var/lib/dpkg)
    - `--show`, `--status`, `--list`, `--search`

Gestión de Paquetes: apt-get

apt-get y apt: complementan dpkg

- Resolución automática de conflictos
- Interacción con repositorios para adquisición de paquetes
  - Los repositorios se especifican en `/etc/apt/sources.list`
    - deb URI distribución componentes
    - deb-src URI distribución componentes
- Comandos útiles
  - `apt-get install (--reinstall / --download-only), apt-get remove (--purge)`
  - `apt-get update, apt-get upgrade, apt-get dist-upgrade`
  - `apt-get builddep`
  - `apt-cache search, apt-cache show`
  - `apt-file search, apt-file list`
  - `apt-get autoclean (/var/cache/apt/archives)`

Gestión de Paquetes: aptitude

aptitude: complementa dpkg

- Características similares a las de `apt-get` pero:
  - Gestión más robusta de las dependencias
    - Modo de actualización seguro (`safe-upgrade`)
  - Interfaz de configuración ncurses (opciones para el administrador)
- Comandos básicos
  - `aptitude install, aptitude remove`
  - `aptitude update, aptitude safe-upgrade, aptitude full-upgrade`
- Ejemplo

```
$ sudo -i
$ aptitude update
$ aptitude install cowsay
$ [Ctrl+D]
$ cowsay "Hola"
```

Compilar el kernel desde la MV (I)

Preparación de las fuentes para la compilación (en $HOME)

- Descomprimir las fuentes en nuevo directorio ~/build:

```
$ cd; mkdir build; cd build
$ tar xzvf ~/linux-4.9.111.tar.gz
```

- Situarse dentro del directorio donde están las fuentes del kernel:

```
$ cd linux-4.9.111
```

- Copiar el archivo de configuración (previamente descargado del Campus):

```
$ cp ~/Descargas/config-4.9.111-mikernel  .config
```

- Configurar el kernel usando configuración existente:

```
$ make oldconfig
```

Compilar el kernel desde la MV (II)

Compilación con `buildeb-pkg`

- Compilación con múltiples threads

```
$ make -j3 bindeb-pkg LOCALVERSION=-mikernel
```

Compilar el kernel desde la MV (III)

Instalación

- ... Y por último se instalan con `dpkg` (Hay que ir al directorio superior)

```
$ cd ..
$ sudo dpkg -i linux-image-4.9.111-mikernel_4.9.111-1_amd64.deb \
               linux-headers-4.9.111-mikernel_4.9.111-1_amd64.deb
```

- Una vez instalado reiniciar el equipo y seleccionar el kernel en el gestor de arranque

```
$ sudo reboot
```

Compilar el kernel desde la MV (IV)

Compilar el kernel desde la MV (V)

Compilación desde la máquina virtual

- La compilación del kernel puede ser **bastante lenta** en la MV

- 2 Alternativas para acelerar compilación:
  1. Añadir más "hierro" a la MV
     - Incrementar el número de cores de la MV y la memoria
  2. Compilación desde otro sistema Debian 9.x con más recursos
     - Copiar los paquetes generados a la MV e instalarlos
     - Si sistema no es Debian 9.x, usar *debootstrap*

Compilar el kernel desde el host (I)

Antes de empezar ...

1. Arrancar equipo en Ubuntu e iniciar sesión con "Usuario Local"
2. Descargar en el directorio `~/Descargas` lo siguiente del campus virtual
   - Este documento (para seguir las instrucciones)
   - `config-4.9.111-mikernel` (fichero de configuración del kernel)
   - `linux-4.9.111.tar.gz` (fuentes del kernel)
3. Abrir una terminal en Ubuntu (host) con 3 pestañas:
   - *Tab (a):* Arrancar la Máquina Virtual
     - `$ VM_LIN-debian.sh`
     - Averiguar su dirección IP (`ip a`)
   - *Tab (b):* Ejecutar comandos en el *host* (Compilación)
     - Accederemos al entorno Debian9 con Debootstrap
   - *Tab (c):* Sesión SSH (shell) con la máquina virtual
     - `$ ssh kernel@192.168.201.129 -X`
     - opción -X: Forwarding de las X

Compilar el kernel desde el host (II)

- La compilación se realizará desde un entorno chroot de Debian accesible desde Ubuntu
  - Entorno creado con herramienta `debootstrap`

Acceso a Debian chroot

1. Abrir sesión Debian 9

```
usuario_local@pto0301:~$ LIN_debian-login.sh
```

2. Verificar que la sesión se ha iniciado correctamente

```
usuario_local@pto0301:~$ cat /etc/debian_version
9.13
```

3. Modificar el prompt del shell para mayor claridad

```
usuario_local@pto0301:~$ debian_chroot=DEBIAN9
(DEBIAN9)usuario_local@pto0301:~$
```

Compilar el kernel desde el host (III)

Preparación de las fuentes para la compilación (en $HOME)

- Descomprimir las fuentes en nuevo directorio ~/build

```
$  cd; mkdir build; cd build
$ tar xzvf ~/Descargas/linux-4.9.111.tar.gz
```

- Situarse dentro del directorio donde están las fuentes del kernel

```
$ cd linux-4.9.111
```

- Copiar el archivo de configuración:

```
$ cp ~/Descargas/config-4.9.111-mikernel  .config
```

- Configurar el kernel (usando .config existente)

```
$ make oldconfig
```

Compilar el kernel desde el host (IV)

Compilación con `buildeb-pkg`

- Compilación con múltiples threads

```
$ make -j12 bindeb-pkg LOCALVERSION=-mikernel
```

Compilar el kernel desde el host (V)

Copiar los paquetes a la MV

- **(Opción 1)** Usar la carpeta compartida
  - Desde el host, hacer lo siguiente...

```
$ cd ..
$ cp linux-image-4.9.111-mikernel_4.9.111-1_amd64.deb \
    linux-headers-4.9.111-mikernel_4.9.111-1_amd64.deb \
          /tmp
```

  - Desde un shell de la MV, copiar al HOME

```
$ cp /mnt/hgfs/shared/*.deb ${HOME}
```

- **(Opción 2)** Copiar paquetes por SCP (Red) a `/home/kernel`

```
$ cd ..
$ scp linux-{image,headers}-4.9.111-mikernel_4.9.111-1_amd64.deb \
        kernel@192.168.201.129:/home/kernel
```

Compilar el kernel desde el host (VI)

Instalación

- ... Y por último se instalan con dpkg desde la máquina virtual

```
$ sudo dpkg -i linux-image-4.9.111-mikernel_4.9.111-1_amd64.deb \
               linux-headers-4.9.111-mikernel_4.9.111-1_amd64.deb
```

- Una vez instalado reiniciar el equipo y seleccionar el kernel en el gestor de arranque

```
$ sudo reboot
```

Compilar el kernel desde el host (VII)

Compilar el kernel desde el host (VIII)

Contenido

Gestión de parches

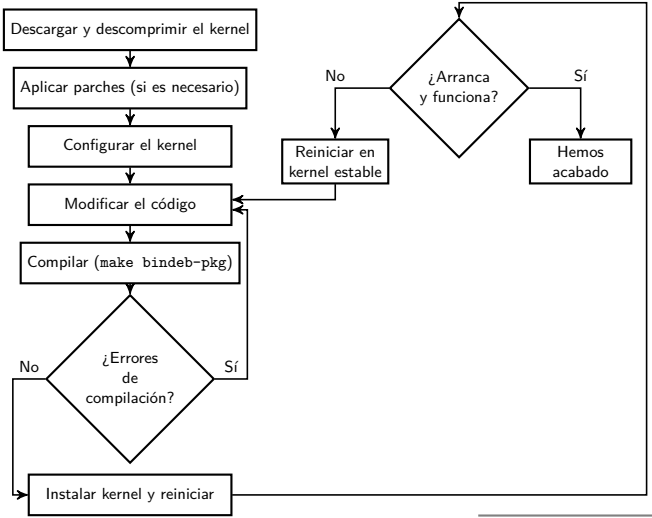
Creación parche

```
## Borrar restos de compilaciones anteriores
$ cd ${DIR_KERNEL_MODIFICADO}
$ make mrproper

## Crear parche
$ cd ..
$ diff -urpN ${DIR_KERNEL_ORIGINAL} ${DIR_KERNEL_MODIFICADO} > mi_parche
```

Aplicar el parche

```
## Descomprimir las fuentes originales
$ tar xzvf linux-4.9.111.tar.gz
$ cd linux-4.9.111

## Aplicar parche
$ patch -p1 < ${RUTA_MI_PARCHE}
```

Flujo de trabajo con el kernel



Licencia

LIN - Compilación del kernel Linux
Versión 2.0