Compilación del Kernel Linux

AISO - Introducción

Manuel Prieto-Matías
High Performance Computing Division
ArTeCS Group
Universidad Complutense de Madrid

Compilación del Kernel Linux Tradicional (I)

- Paso 1: Obtener las fuentes
  - Vanilla: kernel.org (repositorio git o tarball)
  - Las Distros facilitan las fuentes que utilizan (contienen parches). Por ejemplo Ubuntu:
    - Paquete .deb linux-source (instala tarball en /usr/src/)
    - Git (http://kernel.ubuntu.com/git-repos/)
    - Paquete de fuentes de linux-image-*

Compilación del Kernel Linux Tradicional (II)

- Paso 2: Configurar el kernel
  - Qué controladores 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 (no sobredimensionar el kernel con código esporádico)
  - Gestionado por la herramienta make + scripts (scripts/kconfig)
    - make config / menuconfig / xconfig / gconfig
    - Opciones por defecto: fichero .config
      - Si no esta presente se escoge la configuración del kernel que este en ejecución /boot/config-2.6.x.y-...

Compilación del Kernel Linux Tradicional (III)

- Paso 3: Compilar el kernel
  - make
    - Resultado: kernel (vmlinux) + módulos cargables
- Paso 4: Instalación
  - Instalar módulos en /lib
    - make modules_install
  - Instalar módulos en /boot
    - make install

Compilación a la Debian (I)

- 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 Cabeceras utilizadas en la compilación del kernel
      - Con la macros/defines utilizadas en la compilación
    - Util para compilar drivers (modulos) adicionales sin necesidad de utilizar todas las fuentes
- Utilidad para construcción paquetes kernel: make-kpkg

Compilación a la Debian (II)

- make-kpkg
  - Al igual que make funciona con targets:
    - kernel_image (imagen)
      - make-kpkg --initrd --rootcmd fakeroot --append-to-version aiso --revision r-$(date +%H-%d-%y) –config menuconfig kernel_image
    - kernel_headers (cabeceras especificas)
    - kernel_source (codigo fuente excepto doc)
    - binary: image + headers + source + documentacion
    - modules_image (deb para modulos adicionales)
    - clean
  - Puede explotarse paralelismo utilizando variables de entorno
    - export CONCURRENCY_LEVEL=#Proc+1

Paquetes en Debian (I)

- 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
    - exe, documentación,... pueden estar vacíos (virtuales)
  - Fuentes src
    - Codigo fuente + scripts (compilación) para generar .deb

Paquetes en Debian (II)

- Secciones: existen categorías definidas para clasificar los paquetes en función de su contenido
  - Viene determinado por la Debian/Policy
    - Por citar algunas: admin, gnome, KDE, python...
- Prioridades: cada paquete tiene un nivel de prioridad
  - No pueden existir dependencias con paquete de menor prioridad
  - Util en el proceso de liberación de una nueva release
    - required: Un sistemas solo con paquetes required probablemente no será usable pero tiene suficiente funcionalidad para que un administrador arranque e instale más software
    - important: utilidades comunes
    - standard: una instalación mínima sólo contiene paquetes required, important y standard
    - optional (defecto): la mayoría del software
    - extra: tienen conflictos con otros en required, important, standard u optional

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...
  - Ejemplo: foo-3.1_3.1+5ubuntu1.1_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
    - control.tar.gz
      - Tarball con la información de control necesaria por el gestor de paquetes
    - data.tar.gz
      - Tarball con el contenido del paquete
  - Ejemplo: extraer linux-image*

Anatomía de un Paquete Binario (III)

- Ficheros de control (control.tar.gz)
  - control: metainformación (unico fichero obligatorio)
  - conffiles: listado de ficheros de “configuración” . Gestión especial para preservar ficheros de configuraciones previas.
  - Scripts para la instalación / desistalación
    - 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
    - --install: --unpack + --configure (postinst)
    - --remove (prerm) y --purge (postrm)
  - dpkg-deb: manipulación ficheros .deb
    - --info, --contents, --field * version/suggests, --control, --extract, --build
  - dpkg-query: acceso (lectura) bases de datos utilizadas por dpkg (/var/lib/dpkg)
    - --show, --status, --list, --search

Gestión de Paquetes apt

- Complementa dpkg con
  - Resolución automática de conflictos
  - Interacción con repositorios para aquisición de paquetes
    - Los repositorios se especifican en /etc/apt/sources.list
      - deb URI distribucion componetes
      - deb-src URI distribucion componentes
  - Comandos utiles
    - apt-get update, apt-get upgrade, apt-get dist-upgrade
    - apt-get install (--reinstall / --download-only), apt-get remove (--purge)
    - apt-get build-dep
    - apt-cache search, apt-cache show
    - apt-file search, apt-file list
    - apt-get autoclean (/var/cache/apt/archives)

Anatomía Paquetes Fuentes

- Los paquetes fuentes constan de 2 ó 3 ficheros
  - Fichero DSC (Debian Source Control)
    - Describe el paquete y da información de los ficheros que forman parte.
    - Habitualmente firmado con GPG por el developper
  - Fuentes originales *.orig.tar.gz
  - Parche *.diff.gz
    - Los paquetes nativos solo tienen las fuentes originales
- Adquisición:
  - apt-get source --download-only nombre-paquete (ex: linux-image-$(uname -r))
  - dpkg-source -x nombre-paquete-deb.dsc

Compilación en Karmic (new way)

- Es posible utilizar directamente los scripts del arbol de las fuentes (./debian.master y ./debian)
  - La configuración en debian.master/config/ARCH/
    - debian/rules updateconfigs
    - debian/scripts/misc/oldconfig ARCH
  - Compilación
    - fakeroot debian/rules clean
    - AUTOBUILD=1 fakeroot debian/rules binary-debs
      - DEB_BUILD_OPTIONS=parallel=2
      - NOEXTRAS=1
- Una descripción detallada en
  - https://help.ubuntu.com/community/Kernel/Compile

Enunciado AISO-1

- Trabajo AISO-1
  - Compilar el kernel de ubuntu (2.6.31)
- Challenge AISO-1
  - Minimal-ubuntu (8 Abril 2010)

AISO Introducción

Versión 0.1

© **Manuel Prieto Matias**

Este documento (o uno muy similar) esta disponible en
https://cv2.sim.ucm.es/moodle/course/view.php?id=3235