Compilación del kernel Linux

LIN - Curso 2015-2016

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

Compilación del Kernel Linux

Antes de empezar ...

1. Arrancar equipo en Debian 7 e iniciar sesión con "Usuario Local"
2. Descargar este documento del Campus Virtual
3. Abrir una terminal en Debian (host) con 3 pestañas:
   - *Tab (a):* Arrancar la Máquina Virtual
     - `$ VM_LINyAISOySO-debian.sh`
     - Las carpetas compartidas han de estar habilitadas
     - `/tmp` (host) → `/mnt/hgfs/shared` (guest)
   - *Tab (b):* Ejecutar comandos en el *host*
   - *Tab (c):* Sesión SSH (shell) con la máquina virtual
     - `$ ssh kernel@192.168.206.133 -X`
     - Introducir contraseña del usuario kernel
     - opción -X: Forwarding de las X

Contenido

Contenido

Compilación del Kernel Linux Tradicional (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/v3.x/linux-3.14.1.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 del Kernel Linux Tradicional (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, equipos de escritorio, portátiles, sistemas de tiempo real,...
- 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 del Kernel Linux Tradicional (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-3.14.1.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-2.6.x.y-...

Compilación del Kernel Linux Tradicional (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

- Utilidad para construcción de paquetes del kernel: `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)

make-kpkg

- Después de configurar el kernel (*aconsejable*)
- Al igual que make funciona con *targets*:
  - `kernel_image` (imagen)

```
$ make-kpkg --initrd --rootcmd fakeroot \
  --append-to-version ".mikernel" \
  --revision=1.0 kernel_image
```

  - `kernel_headers` (cabeceras específicas)
  - `kernel_source` (código fuente excepto doc)
  - `binary`: image + headers + source + docs
  - `kernel_debug` (símbolos de depuración)
  - `clean`
- Puede explotarse paralelismo utilizando variables de entorno
  - `export CONCURRENCY_LEVEL=6`
    - 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
  - 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
  - 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*

```
$ sudo -i
$ cd /tmp; mkdir decomp; cd decomp
$ ar x /root/Kernel/linux-image-3.14.1.lin_120814_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 coq 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: complementa 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 build-dep`
  - `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 el host (I)

Obtener las fuentes (3 opciones)

- **(Opción A)** Descargar las fuentes directamente en el host

```
$ cd /tmp
$ wget --no-check-certificate <URL>
```

- **(Opción B)** Copiar el *tarball* de las fuentes de la máquina virtual al host usando la carpeta compartida
  - Ejecutar los siguientes comandos en una shell de la MV

```
$ cp ~/linux-3.14.1.tar.gz /mnt/hgfs/shared
$ chmod a+r /mnt/hgfs/shared/linux-3.14.1.tar.gz
```

    - El fichero estará disponible en /tmp del host

- **(Opción C)** Copiar el *tarball* de las fuentes de la máquina virtual al host usando SCP desde el host

```
$ scp kernel@192.168.206.133:./linux-3.14.1.tar.gz /tmp
```

Compilar el kernel desde el host (II)

Obtener fichero de configuración

- Usaremos el fichero de configuración del kernel en ejecución de la máquina virtual (`/boot/config-3.14.1.lin`)
  - Ejecutar los siguientes comandos en una shell de la MV

```
$ cp /boot/config-3.14.1.lin /mnt/hgfs/shared
$ chmod a+r /mnt/hgfs/shared/config-3.14.1.lin
```

    - El fichero estará disponible en /tmp del host
  - Alternativamente podemos obternerlo vía SCP:

```
$ scp kernel@192.168.206.133:/boot/config-3.14.1.lin /tmp
```

Compilar el kernel desde el host (III)

Preparación de las fuentes para la compilación (en `/var/tmp`)

- Descomprimir las fuentes en /var/tmp

```
$ cd /var/tmp
$ tar xzvf /tmp/linux-3.14.1.tar.gz
```

- Situarse dentro del directorio donde están las fuentes del kernel

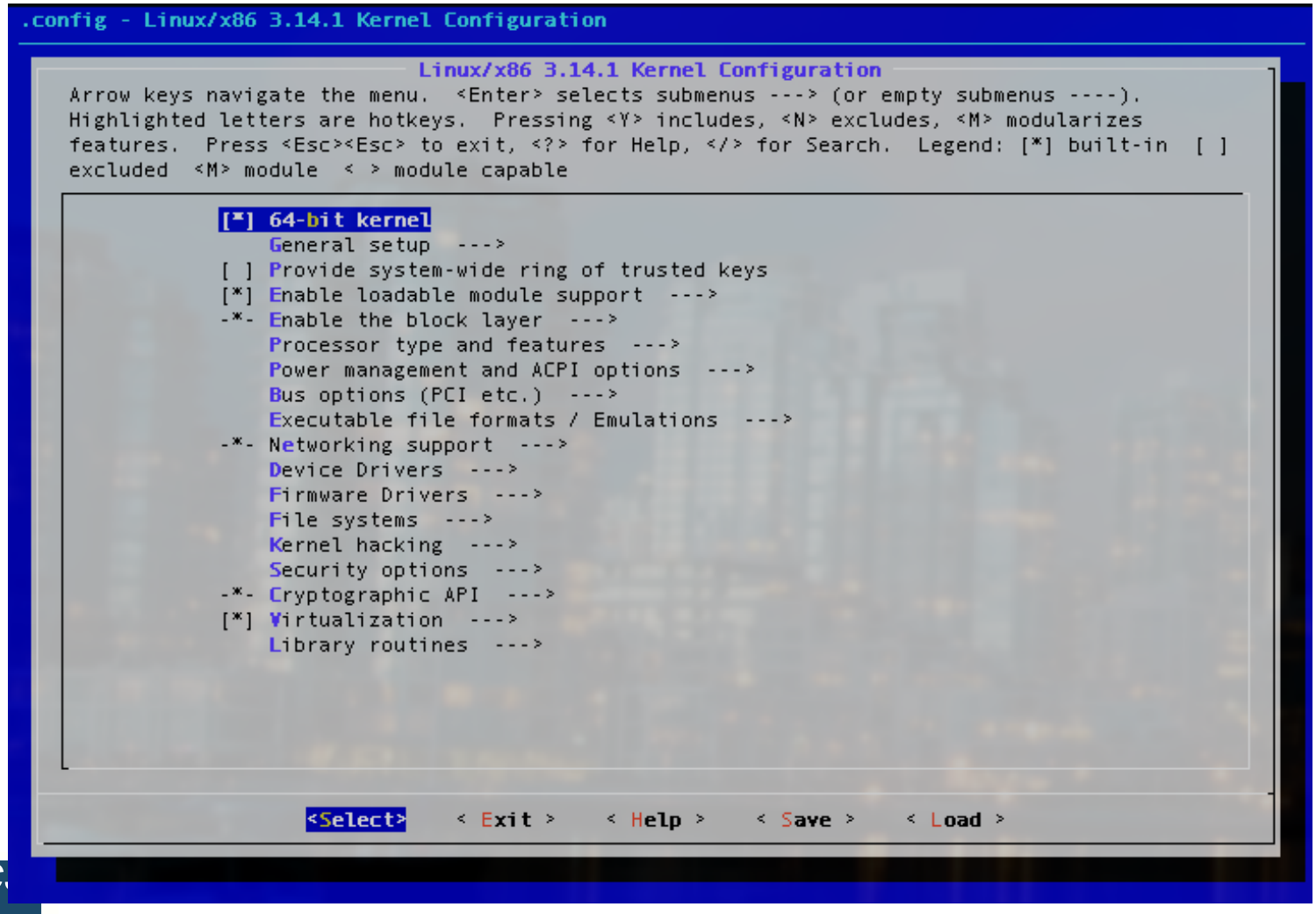
```
$ cd linux-3.14.1
```

- Copiar el archivo de configuración:

```
$ cp /tmp/config-3.14.1.lin  .config
```

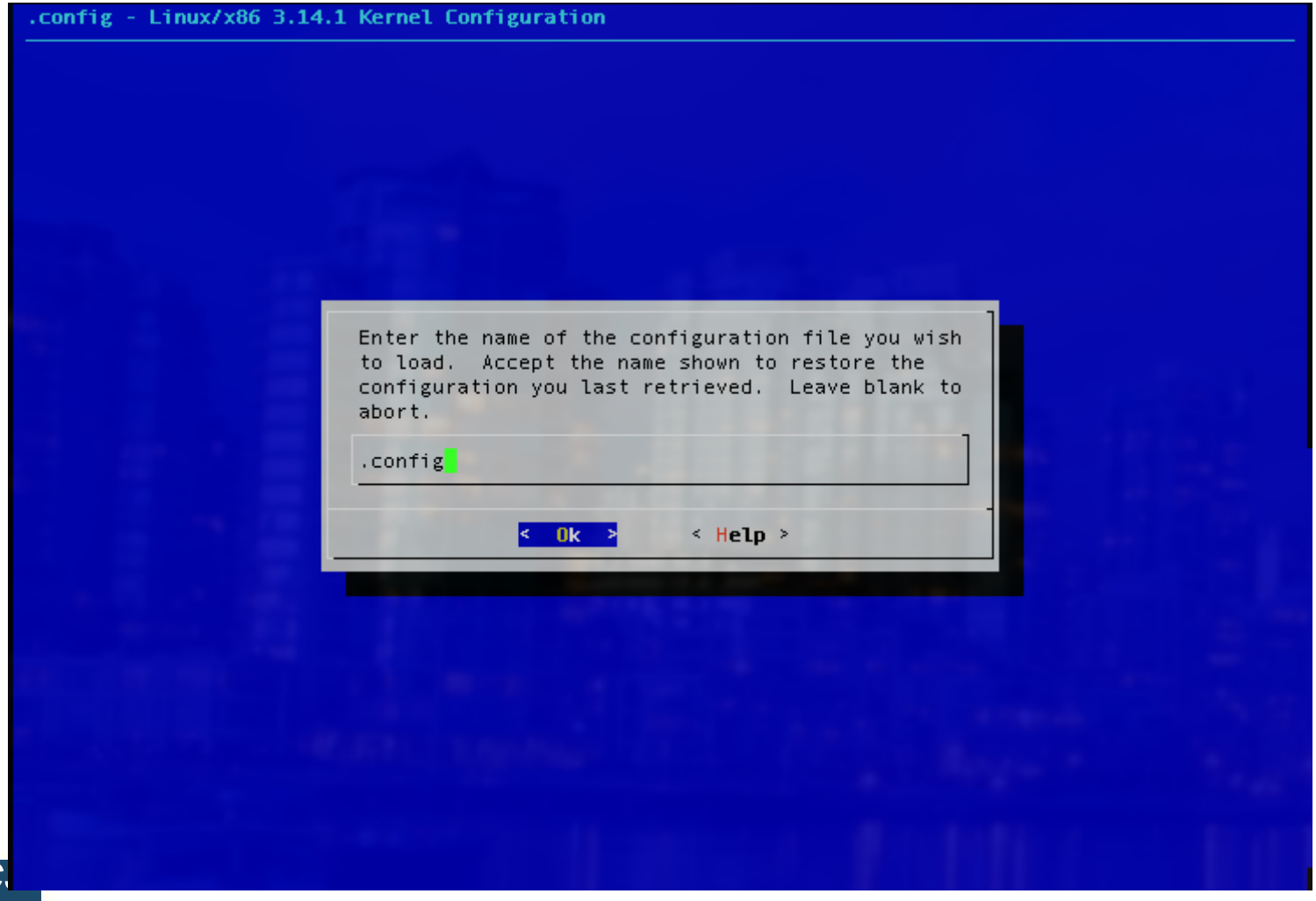
- Aplicar los cambios del .config. Dos opciones:
  1. `make menuconfig` + Load .config + Save
  2. `make oldconfig`
     - Más apropiado si no tenemos que modificar `.config`

make menuconfig (I)



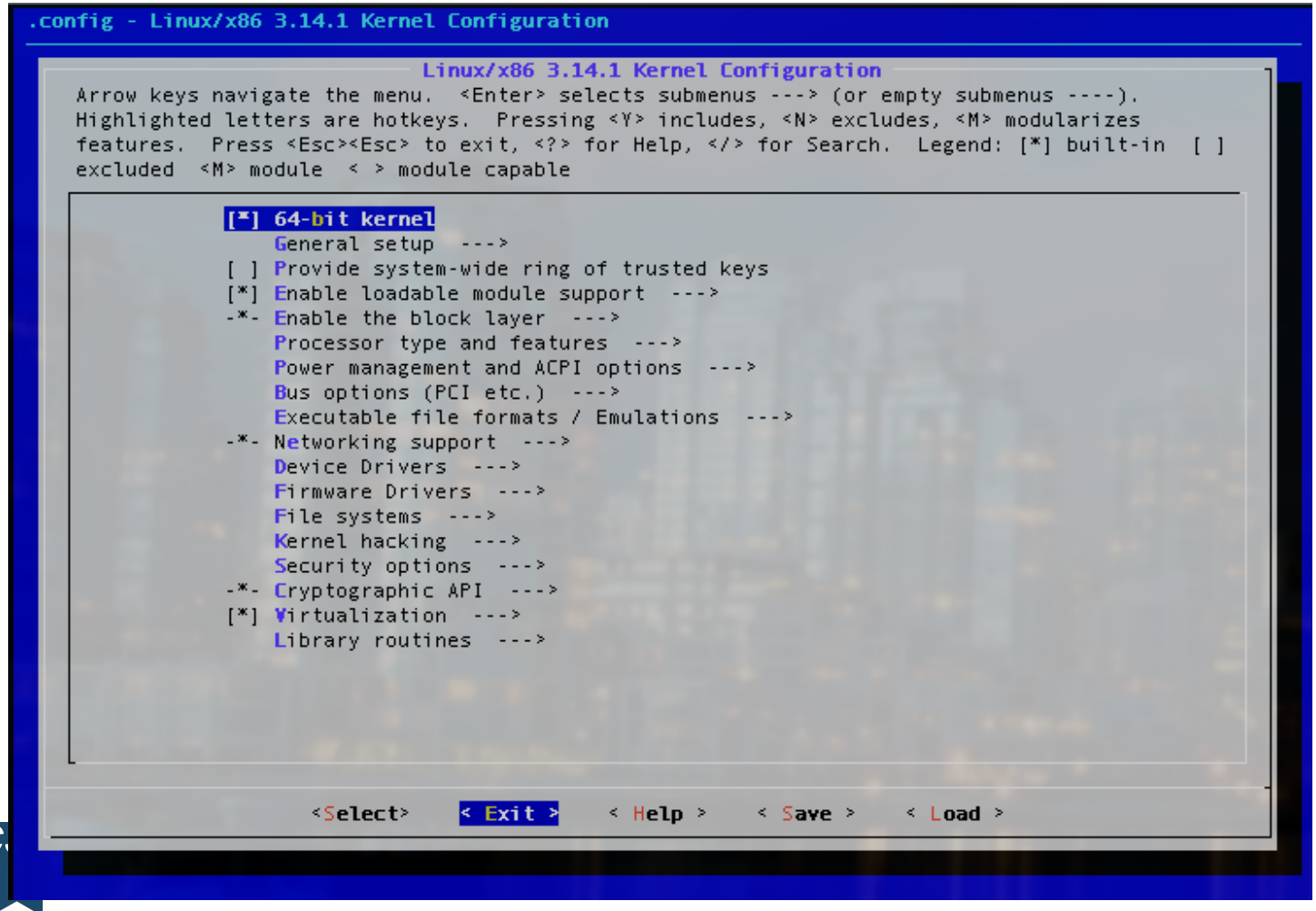
make menuconfig (II)

make menuconfig (III)

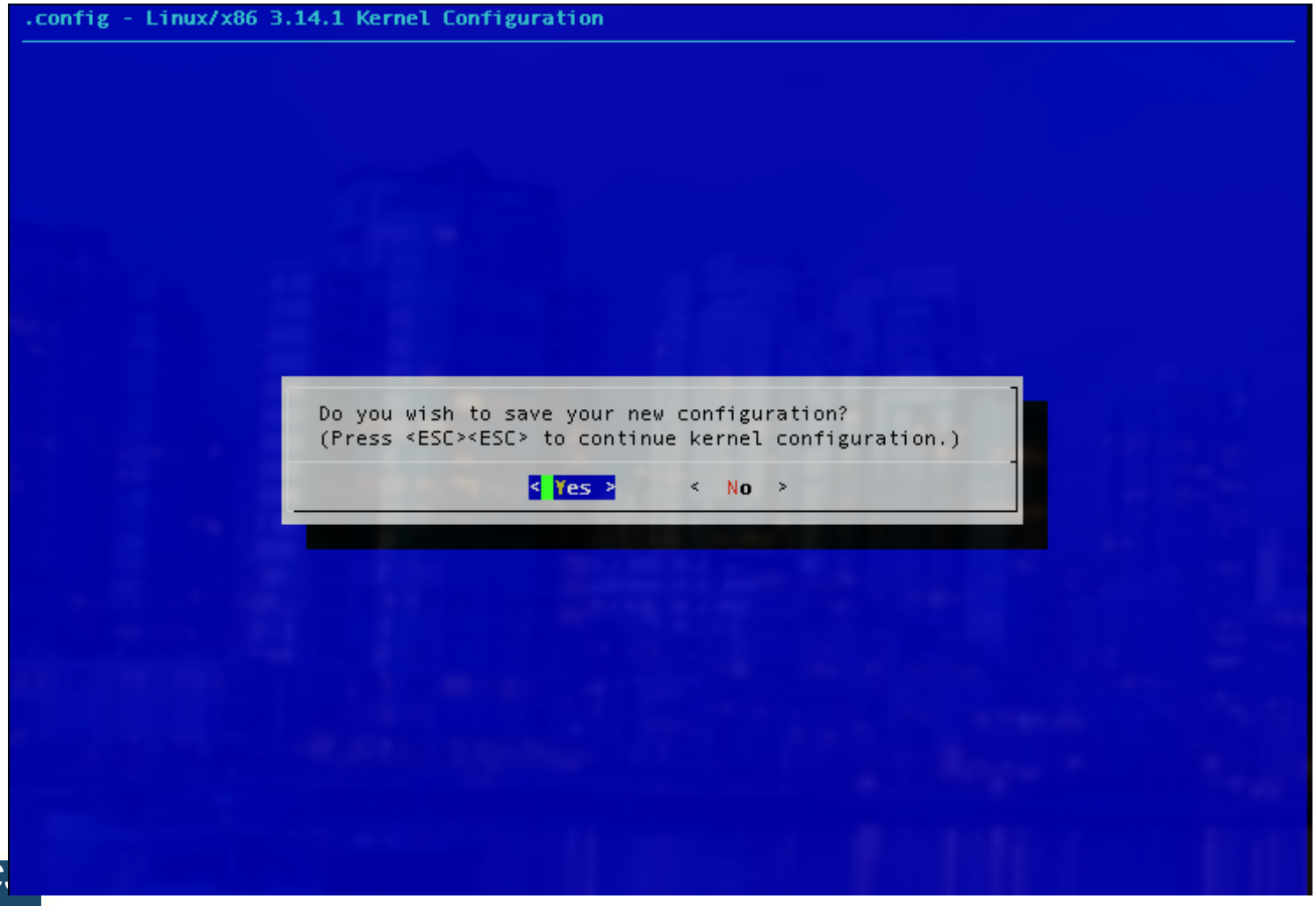


make menuconfig (IV)

- Modificar la configuración si se desea



make menuconfig (V)



Compilar el kernel desde el host (IV)

Compilación con `make-kpkg`

- Habilitar compilación con múltiples threads

```
$ export CONCURRENCY_LEVEL=6
```

- Compilación

```
$ make-kpkg --rootcmd fakeroot --initrd \
    --revision=1.0 --append-to-version=".mikernel" \
    kernel_image kernel_headers
```

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-3.14.1.mikernel_1.0_amd64.deb \
          linux-headers-3.14.1.mikernel_1.0_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}-3.14.1.mikernel_1.0_amd64.deb \
      kernel@192.168.206.133:/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-3.14.1.mikernel_1.0_amd64.deb \
        linux-headers-3.14.1.mikernel_1.0_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 (I)

Preparación de las fuentes para la compilación

- Descomprimir las fuentes en /var/tmp:

```
$ cd /var/tmp
$ tar xzvf ~/linux-3.14.1.tar.gz
```

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

```
$ cd /var/tmp/linux-3.14.1
```

- Copiar el archivo de configuración:

```
$ cp /boot/config-3.14.1.lin  .config
```

- Configurar el kernel:

```
$ make oldconfig
```

Compilar el kernel desde la MV (II)

Compilación con `make-kpkg`

- Habilitar compilación con múltiples threads

```
$ export CONCURRENCY_LEVEL=3
```

- Compilación

```
$ make-kpkg --rootcmd fakeroot --initrd \
    --revision=1.0 --append-to-version=".mikernel" \
    kernel_image kernel_headers
```

Compilar el kernel desde la MV (III)

Instalación

- ... Y por último se instalan con dpkg

```
$ cd ..
$ sudo dpkg -i \
        linux-image-3.14.1.mikernel_1.0_amd64.deb \
        linux-headers-3.14.1.mikernel_1.0_amd64.deb
```

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

```
$ sudo reboot
```

Compilación desde la máquina virtual

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

- 2 Alternativas:
  1. Compilación desde el *host*
  2. Añadir más "hierro" a la MV
     - Incrementar el número de cores de la MV y la memoria
     - Posible desde `usuario_vms` del laboratorio o desde vuestro portátil

Contenido

Gestión de parches

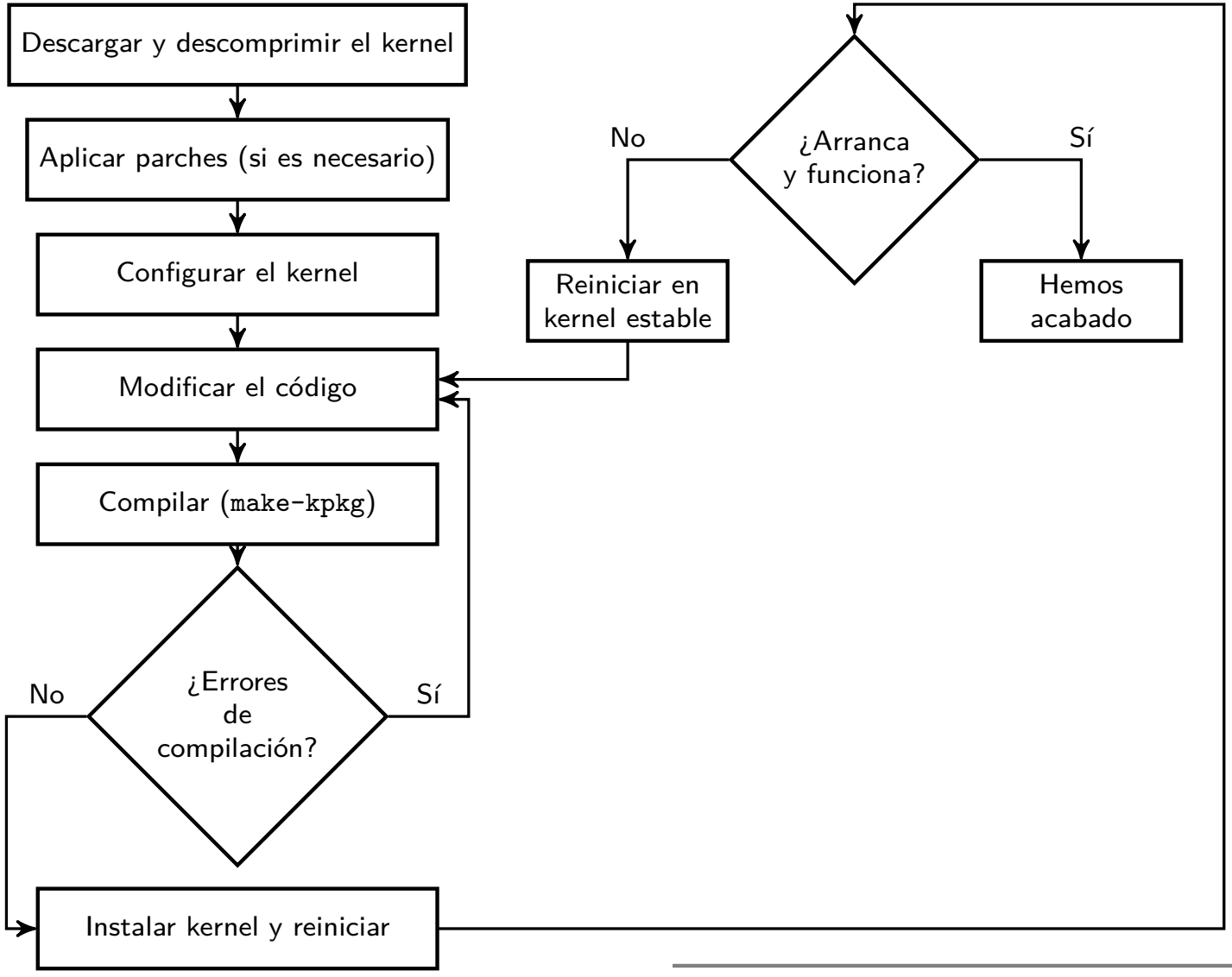
Creación parche

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

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

Aplicar el parche

```
## Descomprimir las fuentes originales
$ tar xzvf linux-3.14.1.tar.gz
$ cd linux-3.14.1

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

Flujo de trabajo con el kernel



Licencia

LIN - Compilación del kernel Linux
Versión 0.2

Este documento (o uno muy similar) está disponible en
`https://cv4.ucm.es/moodle/course/view.php?id=62472`