📁 LPIC-2 / 🧠 Examen 201 - Kernel de Linux - Ejercicios

Nota: Estos ejercicios se centran en los pasos de preparación y configuración del kernel. Realizar la compilación completa lleva mucho tiempo y debería hacerse solo en un entorno de prueba dedicado. No intentaremos una instalación completa aquí.

Ejercicio 1.2.1: Obteniendo e Instalando el Código Fuente del Kernel

- **Objetivo:** Conseguir el código fuente del kernel para tu distribución.
- **Requisitos:** Acceso a la línea de comandos. Privilegios de superusuario (`sudo`). Conocer tu gestor de paquetes.
- **Desarrollo Paso a Paso:**
    1. Abre una terminal.
    2. **Identifica el nombre del paquete de código fuente de tu distribución:**
        - En Debian/Ubuntu: Suele ser `linux-source-<versión>`, ej: `linux-source-5.15`. Puedes buscar: `apt search linux-source`.
        - En Red Hat/CentOS/Fedora: El paquete es `kernel-devel`. Busca: `dnf search kernel-devel`.
    3. **Instala el paquete de código fuente (requiere sudo):** Ejecuta `sudo apt install linux-source-$(uname -r | cut -d- -f1)` (para Debian/Ubuntu, usando la versión principal del kernel) o `sudo dnf install kernel-devel` (para Red Hat/Fedora).
    4. **Verifica la ubicación del código fuente:** Suele instalarse en `/usr/src/`. Ejecuta `ls -l /usr/src/`.
    5. **Ve al directorio del código fuente:** El paquete Debian/Ubuntu instala un archivo `.tar.xz`. Necesitas extraerlo: `sudo tar xvfJ /usr/src/linux-source-<version>.tar.xz -C /usr/src/`. Luego ve al directorio extraído: `cd /usr/src/linux-source-<version>` (o similar). En Red Hat/Fedora, el código fuente se extrae automáticamente a un directorio como `/usr/src/kernels/<versión_completa>`. Ve a ese directorio: `cd /usr/src/kernels/$(uname -r)/`.

Ejercicio 1.2.2: Preparando y Configurando el Kernel

- **Objetivo:** Empezar con la configuración del kernel en ejecución y usar `make menuconfig`.
- **Requisitos:** Estar en el directorio del código fuente del kernel (del Ej. 1.2.1). Tener los paquetes de desarrollo necesarios (`build-essential`, `libncurses-dev` o `ncurses-devel`, `flex`, `bison`, `libssl-dev` o `openssl-devel`). Instálalos si no los tienes. Privilegios de superusuario (`sudo`) pueden ser necesarios para algunos pasos de preparación.
- **Desarrollo Paso a Paso:**
    1. Abre una terminal y ve al directorio del código fuente del kernel.

2. **Instala dependencias de compilación:**
    - En Debian/Ubuntu: `sudo apt install build-essential libncurses-dev flex bison libssl-dev`
    - En Red Hat/CentOS/Fedora: `sudo dnf group install "Development Tools"` y `sudo dnf install ncurses-devel flex bison openssl-devel`
3. **Copia la configuración del kernel en ejecución:** Ejecuta `sudo cp /boot/config-$(uname -r) .config`. Asegúrate de estar en el directorio del código fuente para que el archivo se copie aquí. (Puede que necesites permisos para copiarlo; usa `sudo cp` si es necesario). Luego cambia el propietario para que tu usuario pueda modificarlo si no estás operando como root: `sudo chown -R tu_usuario:tu_grupo .` (reemplaza `tu_usuario` y `tu_grupo`).
4. **Asegúrate de que el archivo `.config` existe en el directorio actual:** Ejecuta `ls -a .config`.
5. **Ejecuta `make oldconfig` (opcional pero recomendado al actualizar código fuente):** Si hubieras descargado un código fuente de versión diferente a la que usaste para copiar `.config`, ejecutar `make oldconfig` te preguntaría sobre las nuevas opciones. Ejecútalo (te preguntará sobre opciones que no estaban en el `.config` copiado, si las hay, o simplemente dirá que está actualizado).
6. **Ejecuta `make menuconfig`:** Esto abrirá la interfaz de texto. Explora los menús (Device Drivers, File systems, Networking support, General setup). Usa las flechas, Enter para entrar en submenús, Espacio para cambiar el estado de una opción (`*` = compilado en, `M` = módulo, vacío = excluido).
7. **Haz un pequeño cambio (solo para practicar la interfaz):** Por ejemplo, ve a "General setup" -> "Timers subsystem" y mira las opciones (no cambies nada importante a menos que sepas lo que haces). Ve a "Device Drivers" -> "Network device support" y explora los drivers disponibles.
8. **Sal de `make menuconfig` SIN GUARDAR cambios (a menos que quieras un `.config` modificado para una compilación real):** Selecciona "Exit", confirma que no quieres guardar los cambios. Si guardas, se creará un nuevo archivo `.config`.
9. **(Concepto):** Si hubieras guardado cambios, el archivo `.config` en el directorio actual reflejaría tu configuración personalizada.

Ejercicio 1.2.3: Comprendiendo los Pasos de Compilación e Instalación

- **Objetivo:** Entender los comandos para compilar e instalar, sin ejecutarlos completamente.
- **Requisitos:** Estar en el directorio del código fuente del kernel con un archivo `.config` válido.
- **Desarrollo Paso a Paso:**
    1. Abre una terminal y ve al directorio del código fuente del kernel.

2. **Comprende el comando para compilar la imagen principal del kernel:** `make bzImage`. Este comando lee el archivo `.config` y compila el kernel. Puedes añadir `-j $(nproc)` para paralelizar. **NO LO EJECUTES COMPLETO; si lo haces, cancela con Ctrl+C después de que empiece.**
3. **Comprende el comando para compilar los módulos:** `make modules`. También lee `.config` y compila los componentes marcados como `=m`. **NO LO EJECUTES COMPLETO; si lo haces, cancela con Ctrl+C.**
4. **Comprende el comando para instalar los módulos (requiere sudo):** `sudo make modules_install`. Copia los archivos `.ko` (objetos de módulo del kernel) al directorio `/lib/modules/` bajo un subdirectorio con el nombre del nuevo kernel. **NO LO EJECUTES COMPLETO; si lo haces, cancela con Ctrl+C.**
5. **Comprende el comando para instalar el kernel y actualizar el gestor de arranque (requiere sudo):** `sudo make install`. Copia la imagen `bzImage`, `System.map`, `config` a `/boot/`, genera un initramfs y modifica el archivo de configuración de GRUB/LILO. **ESTE ES EL PASO MÁS RIESGOSO EN UN SISTEMA REAL. NO LO EJECUTES EN UN SISTEMA DE PRODUCCIÓN NI SIQUIERA EN UNA VM DE PRUEBA A MENOS QUE ESTÉS PREPARADO PARA RECUPERAR EL SISTEMA.**

Ejercicio 1.2.4: (Conceptual) Creando Paquetes Instalables

- **Objetivo:** Entender que construir paquetes es el método de instalación recomendado en la práctica.
- **Requisitos:** Estar en el directorio del código fuente del kernel con un archivo `.config` válido.
- **Desarrollo Paso a Paso:**
    1. Abre una terminal y ve al directorio del código fuente del kernel.
    2. **Comprende el comando para construir paquetes .deb (Rama Debian/Ubuntu):** Si tuvieras `make-kpkg` instalado, usarías algo como `make-kpkg --initrd kernel_image kernel_headers`. Esto crearía archivos `.deb` en el directorio padre, que luego instalarías con `sudo dpkg -i <archivos.deb>`. **NO EJECUTES ESTE COMANDO.**
    3. **Comprende el comando para construir paquetes .rpm (Rama Red Hat/CentOS/Fedora):** Esto es más complejo e implica usar `rpmbuild` y archivos `.spec` (a menudo ya proporcionados en el código fuente de distribución). El comando general sería `rpmbuild -ba kernel.spec` desde el lugar correcto. **NO EJECUTES ESTE COMANDO.**
    4. **(Concepto):** La ventaja de este método es que la instalación y desinstalación son manejadas por el gestor de paquetes, lo que es más limpio y menos propenso a errores que `make install` directo.

Ejercicio 1.2.5: Limpieza (del Código Fuente)

- **Objetivo:** Limpiar los archivos generados durante la configuración/compilación.
- **Requisitos:** Estar en el directorio del código fuente del kernel.
- **Desarrollo Paso a Paso:**
    1. Abre una terminal y ve al directorio del código fuente del kernel.
    2. **Limpia los archivos de configuración/compilación:** Ejecuta `make clean` o `make mrproper`. `make mrproper` es una limpieza más profunda que elimina también los archivos de configuración generados. Esto es útil antes de empezar una nueva configuración o si quieres liberar espacio.