📁 LPIC-2 / 🔁 Examen 202 - Arranque del Sistema

202.4 Gestionar drivers udev

Teoría

`udev` es el gestor de dispositivos dinámico de Linux. Se ejecuta como un demonio (`systemd-udevd.service`) y su función es reaccionar a los eventos del kernel cuando los dispositivos se añaden, eliminan o cambian de estado. Basándose en un conjunto de reglas, `udev` realiza acciones como crear o eliminar nodos de dispositivo en `/dev/`, cargar módulos del kernel, establecer permisos y propiedad de los dispositivos, y crear enlaces simbólicos persistentes.

Reglas de `udev`:

El comportamiento de `udev` se controla mediante archivos de reglas, que se procesan en un orden específico.

1. **Ubicación de Archivos de Reglas:**
    - `/usr/lib/udev/rules.d/`: Contiene las reglas por defecto proporcionadas por los paquetes de software de la distribución. Estos archivos no deben modificarse manualmente.
    - `/etc/udev/rules.d/`: Contiene las reglas personalizadas creadas por el administrador del sistema.
    - **Orden de Procesamiento:** Las reglas se procesan en orden numérico y alfabético. Las reglas en `/etc/udev/rules.d/` con el mismo nombre o un número menor que las reglas en `/usr/lib/udev/rules.d/` pueden anular las reglas por defecto. Es común usar números bajos (ej: `10-`) para reglas que deben tener precedencia o números altos (ej: `99-`) para reglas que deben ejecutarse después de las reglas por defecto. **Esta estructura y orden es estándar en ambas ramas (Debian/Red Hat).**
2. **Sintaxis de una Regla (`<numero>-<nombre>.rules`):** Cada regla es una línea de texto que consta de una serie de pares clave-valor separados por comas.
    - **Claves de Coincidencia (Matching Keys):** Se utilizan para identificar el dispositivo al que se aplica la regla. Si todas las claves de coincidencia en una regla son verdaderas para un dispositivo, se aplican las claves de asignación.
        - `KERNEL=="<patron>"`: Coincide con el nombre del dispositivo del kernel (ej: `sd*`, `ttyUSB*`).
        - `SUBSYSTEM=="<subsistema>"`: Coincide con el subsistema del kernel (ej: `block`, `net`, `usb`).
        - `ATTRS{<atributo>}=="<valor>"`: Coincide con un atributo específico del dispositivo leído de sysfs (ej: `ATTRS{idVendor}=="091e"` para el ID de fabricante USB). `udevadm info` es útil para encontrar estos atributos.

        - `ENV{<variable>}=="<valor>"`: Coincide con una variable de entorno de udev.
        - `PROGRAM=="<programa>"`, `RESULT=="<resultado>"`: Ejecuta un programa externo y coincide con su salida.
        - `MODE`, `OWNER`, `GROUP`: Coinciden con los permisos, propietario o grupo del nodo de dispositivo inicial.
    - **Claves de Asignación (Assignment Keys):** Definen acciones o propiedades para el dispositivo si la regla coincide.
        - `NAME="<nombre>"`: Establece el nombre del nodo de dispositivo en `/dev/`. **¡Cuidado! No es recomendable cambiar los nombres de dispositivos de bloque esenciales (`/dev/sda`) con esto.**
        - `SYMLINK+="<nombre_enlace>"`: Crea un enlace simbólico en `/dev/` que apunta al nodo del dispositivo. Es el método preferido para nombres persistentes personalizados (ej: `SYMLINK+="mi_pendrive"`). Puedes añadir múltiples enlaces con `SYMLINK+="enlace1 enlace2"`.
        - `MODE="<permisos>"`: Establece los permisos del nodo (ej: `MODE="0666"`).
        - `OWNER="<propietario>"`, `GROUP="<grupo>"`: Establece el propietario y grupo del nodo.
        - `ENV{<variable>}="<valor>"`: Establece una variable de entorno de udev para el dispositivo.
        - `RUN+="<programa_o_script>"`: Ejecuta un programa o script externo cuando la regla coincide. Útil para tareas como montar automáticamente dispositivos.

Uso de `udevadm` para Depuración y Gestión:

- `udevadm info /sys/class/<subsistema>/<nombre_dispositivo>` o `udevadm info /dev/<nombre_dispositivo>`: Muestra información sobre el dispositivo, incluyendo sus atributos y variables de entorno, útil para escribir reglas.
- `udevadm test /sys/class/<subsistema>/<nombre_dispositivo>`: Simula cómo `udev` procesaría el dispositivo especificado, mostrando qué reglas coinciden y qué propiedades resultan. Indispensable para depurar reglas personalizadas.
- `sudo udevadm control --reload-rules`: Recarga las reglas de udev en el demonio en ejecución sin necesidad de reiniciar el servicio. Es necesario después de añadir o modificar archivos de reglas.
- `sudo udevadm trigger`: Dispara eventos de udev para dispositivos, haciendo que las reglas se procesen para el hardware ya conectado (útil después de recargar reglas sin desconectar/reconectar el hardware).
- `sudo udevadm monitor`: (Visto en 201.4) Muestra eventos en tiempo real.

Proceso para Crear una Regla Personalizada:

1. Identifica el dispositivo y sus atributos usando `udevadm info` o `udevadm monitor` mientras lo conectas.
2. Escribe la regla en un nuevo archivo `.rules` en `/etc/udev/rules.d/`. Dale un nombre que empiece con un número (ej: `90-mi-regla.rules`, `10-mi-dispositivo.rules`).
3. Añade las claves de coincidencia necesarias para identificar *únicamente* tu dispositivo (una combinación de `SUBSYSTEM`, `KERNEL`, `ATTRS{idVendor}`, `ATTRS{idProduct}`, `ATTRS{serial}`, etc.).
4. Añade las claves de asignación para la acción deseada (`SYMLINK`, `MODE`, `OWNER`, `GROUP`, `RUN`).
5. Guarda el archivo.
6. Recarga las reglas de udev: `sudo udevadm control --reload-rules`.
7. Prueba la regla: Desconecta y vuelve a conectar el dispositivo, o ejecuta `sudo udevadm trigger` para el subsistema relevante (ej: `sudo udevadm trigger --subsystem-match=usb`).
8. Verifica si la regla funcionó (ej: busca el enlace simbólico en `/dev/`, verifica los permisos). Utiliza `udevadm test` para depurar si no funciona.