Udev

Udev (https://www.freedesktop.org/software/systemd/man/udev.html) es un subsistema (actualmente incluido dentro de Systemd en forma de demonio *systemd-udevd* -permanentemente encendido-) que sirve para recibir del kernel y en tiempo real los eventos relacionados con la conexión/desconexión de los dispositivos hardware. Una vez recibidos, Udev es quien se encarga realmente de cargar los módulos involucrados y de generar al vuelo sus archivos "/dev" correspondientes (a partir de la información obtenida de "/sys", concretamente en "/sys/block" si se trata de dispositivos de bloques y "/sys/class" si son de caracter). De aquí se deduce que la carpeta "/dev", si es gestionada por Udev, en realidad es una partición montada en la RAM (concretamente con el sistema de ficheros "devtmpfs").

El uso de Udev tiene varias ventajas, como por ejemplo definir reglas que nos permitirán personalizar el comportamiento de nuestro sistema en el momento de detectar un dispositivo nuevo más o menos específico. Por ejemplo, podríamos cambiar el nombre de una partición de un pendrive USB para que fuera siempre "/dev/miusb" y así no tener que usar el nombre tan extraño asignado por el kernel como por ejemplo sería "/dev/sdc3" (el cual, además, podría cambiar entre conexión y conexión según el orden en el que el kernel detecte dicho dispositivo cada vez.).

Las reglas básicamente constan de dos partes: se ha de especificar de qué dispositivo estamos hablando mediante filtros (los cuales pueden ser más generales o más exhaustivos según si queremos que la regla afecte de forma más general -"todos los discos duros"- o tan solo a dispositivos específicos -"solo el disco duro *tal*"-) y qué acción se le quiere aplicar (por ejemplo, una acción sería efectuar el cambio de nombre respecto el nombre inicial que le asigne el kernel, pero habría muchas más posibilidades). Las reglas se pueden encontrar en dos sitios en forma de ficheros de texto con extensión ".rules":

*"/usr/lib/udev/rules.d" : reglas por defecto instaladas con los programas
*"/etc/udev/rules.d" : reglas que se editan/añaden localmente a “mano”

Tomará preferencia aquella regla que esté contenida en un fichero que tenga (de entre los que se llamen igual) un número mayor al principio de su nombre -ya que se leerá más tarde-. Si no tiene número inicial, se harán al final.

Reglas

-Las líneas que empiezan por # son comentarios. Cada regla va en una línea diferente

-Algunos de los valores que sirven para identificar de qué dispositivo estamos hablando son:

ACTION=={“add”|”remove”|"change"} (tipus d'event que "dispararà" la regla)
KERNEL (nom amb el que veu el kernel al dispositiu. Es poden usar comodins)
BUS (sistema de drivers que ese dispositivo usa: usb, scsi...)
SUBSYSTEM (subsistema en el que integra el dispositivo: usb, ata, net, tty...)
DRIVER (nombre del driver que gobierna el dispositivo,como p.ej usb-storage)
ATTRS{atributdebusqueda} (per exemple, “model”, “size”...)

Se pueden usar los símbolos “==” y “!=”

-Algunos de los valores que sirven para indicar qué deseamos hacer con ese dispositivo concreto son:

NAME (da directamente un nombre nuevo al dispositivo dentro de /dev)
SYMLINK (crea un link nuevo apuntando al nombre dado por el kernel en /dev a ese dispositivo)
OWNER (establece el propietario del dispositivo)
GROUP (establece el grupo propietario del dispositivo)
MODE (establece los permisos del dispositivo -en formato numérico: "0666"-)
RUN (indica la ruta del binario-y sus parámetros- que se ejecutará en detectarse el dispositivo)
GOTO (va a la etiqueta indicada, dentro del propio fichero de reglas)
LABEL (marca la etiqueta)

En este caso, para asignar el valor deseado se utilitzará en general el operador “=” . También existen los operadores “+=” (adición de un valor a los ya existentes) y “:=” (asignación de un valor y prohibición de que se cambie en otras reglas)

Ejemplos

Supondremos que creamos un fichero con un nombre cualquiera (por ejemplo, "lalala.rules") dentro de "/etc/udev/rules.d". Podríamos hacer, por ejemplo:

KERNEL==”ttyACM0”, SYMLINK+=”modem”
KERNEL==”sda”, SYMLINK+=”midisco”

Podríamos hacer lo mismo para dar nombres personalizados a nuestras llaves Usb, por ejemplo...pero ¿cómo lo hacemos si no sabemos de antemano qué nombre le dará el kernel a nuestra llave Usb? Recordemos que cuando se enchufa un lápiz Usb, no se tiene la seguridad de que cada vez sea el mismo fichero de dispositivo el que lo represente: unas veces puede ser por ejemplo /dev/sdb1, otras /dev/sdc1,...depende de los dispositivos Usb que ya tengamos en ese momento enchufados. Para saber cuál es el fichero de dispositivo que toca en cada momento, se podría hacer un dmesg en el momento de conectarlo y escribir la regla correspondiente que le diera el nombre siempre fijo que deseamos cada vez que ese dispositivo y ningún otro se enchufara (por ejemplo, el nombre /dev/pepito), pero ¿cómo poner en el valor KERNEL de esa regla de qué dispositivo estamos hablando si cada vez puede ser detectado de forma diferente? Para ello, debemos conocer el comando udevadm:

udevadm info -q all -n /dev/disp: mostra la informació d'Udev demanada sobre el dispositiu especificat (com l'UUID,etc). La “P” indica el path, es decir, la ruta en /sys a ese dispositivo, la “N” es el nombre del nodo en /dev, la “S” indica si hay algún enlace creado por udev a ese dispositivo y “E” muestra datos adicionales si los hay. Altres opcions pel paràmetre -q (que serveix per dir quin tipus d'informació es vol) a part de “all” justament són: “name”, “symlink”, “path”, “property”.

udevadm info -a -p elpathdeldispositiu : mostra els atributs del dispositiu especificat via el seu path (valor “P” obtingut amb la comanda anterior). També es podria fer *udevadm info -a -n /dev/disp*

Lo que tenemos que hacer, pues es ver los atributos del dispositivo usb en cuestión, enchufándolo y ejecutando por ejemplo: *udevadm info -a -p $(udevadm info -q path -n /dev/sdb)*. Saldrá una lista de ATTRS. Elegimos uno que tenga un valor no repetido (para identificar unívocamente el dispositivo, por ejemplo ATTRS{serial} -ó bien VendorID ó también ProductID...-) y procederemos a crear la regla udev correspondiente. Por ejemplo, si tenemos dos dispositivos Usb, podremos hacer:

KERNEL==”sd?”, ATTRS{size}==”4127760” , SYMLINK+=”pen2GB”
KERNEL==”sd?”, ATTRS{size}==”390719855” , SYMLINK+=”disco200GB”

Pero lo que nos interesará más siempre es acceder a las particiones (“sda1”,”sdb2”...) y no tanto a los discos duros en sí (“sda”,”sdb”), ¡que es lo que hemos hecho en las líneas anteriores! Para dar nombre a las distintas particiones podemos usar varios métodos pero el más habitual es

KERNEL==”sd”, ATTRS{product}=”loquesea” , SYMLINK+=”pepito%n”*

donde %n cogerá los valores 1,2,3...según las particiones detectadas en el dispositivo. También existe %k para indicar el nombre del dispositivo que le daría el kernel por defecto (ver man udev para más información). Tras reiniciar el servicio Udev (o bien desenchufando y volviendo a enchufar el dispositivo), la regla se activará y al conectar de nuevo la memoria usb se creará automáticamente un fichero de dispositivo con el nombre deseado, incluyendo los fichero de dispositivos de las particiones que éste incluya (/dev/pepito, /dev/pepito1, etc), los cuales no serán más que enlaces blandos a los ficheros de dispositivo “reales” /dev/sdbX (que cada vez podrán ser diferentes).

NOTA: La diferencia entre ATTR y ATTRS es que el primero se refiere a los atributos del dispositivo (p.ej, una memoria) y el segundo se refiere a los atributos del dispositivo y además de todos sus dispositivos padres (p.ej, la memoria y su controladora). Esto se puede ver con *udevadm info*...además, se puede observar que el path de un dispositivo hijo incluye el

de su padre. En este sentido, hay que tener en cuenta que, efectivamente, en la salida del comando *udevadm* nos sale tanto la controladora de la memoria/ratonusb/... como lo que es la memoria/ratonusb/... en sí, la segunda como hija de la primera. Se puede ver que el path de la memoria/ratonusb/... está formado por el path de la controladora seguido de su propio path como cola, y que es el path al cual apuntan los ficheros en presentes /sys/class (para dispositivos carácter) ó /sys/block (para dispositivos bloque). Para diferenciar un disco de su controladora, nos podemos fijar en los drivers usados: “sd” para los discos, “usb” para las controladoras, normalmente, además de algún atributo típico de disco (“max_sectors”) o de controladora (“bMaxPower”). Ojo con esto para usar Udev con lectores de múltiples tarjetas: en este caso mejor poner BUS==”scsi” en vez de BUS==”usb”, y establecer una regla diferente para cada ranura de tarjeta diferente,

También se podría añadir una entrada a "/etc/fstab" de manera que cada vez que el sistema arrancara con la memoria usb encendida se montara directamente en el directorio deseado sin más inconvenientes, así: */dev/pepito1 /mnt/mipunto vfat defaults,umask=000 0 0*

NOTA: Por otro lado, también es interesante destacar que Udev crea automáticamente (a partir de los archivos /usr/lib/udev/rules.d/*persistent*.rules) los siguientes ficheros para disponer de nombres fijos predefinidos de dispositivos de bloques:

/dev/disk/by-id : contiene enlaces blandos ya predefinidos a los ficheros de disp de disco/part “reales”
/dev/disk/by-path : lo mismo, pero los links están agrupados por el path a sysfs
/dev/disk/by-uuid : lo mismo, pero los links están agrupados por el identificador único
/dev/disk/by-label : lo mismo, según las etiquetas de las particiones (éstas han de tenerlas: se pueden poner con el comando *tune2fs/jfs_tune/xfs_admin/ntfslabel -L etiqueta /dev/dispositiu*)

Otro ejemplo habitual es el de querer ajustar los permisos del dispositivo...

KERNEL==”sda1”, OWNER=”root”, GROUP=”root”,MODE=”0700”

...o disparar la ejecución de programas/scripts al detectarlo (un ejemplo de uso puede ser la realización automática de backups en detectar un dispositivo de disco externo enchufado, o la descarga automática de fotografías de la cámara digital, etc).

KERNEL==”sda1”, RUN+=”/bin/mount /dev/sda1 /mnt/disco”

NOTA: Además de RUN existe también la directiva PROGRAM que hace algo similar con la ventaja de que el resultado del programa, además de enviarlo a stdout, se puede recoger mediante RESULT para poder hacer comprovaciones de cadenas en la propia regla si RESULT resulta que vale lo que se le indica...(ver man udev y buscar %c).

Otros parámetros de *udevadm* son *-V* para saber la versión de Udev que tenemos, y *info -e* para ver todos los dispositivos detectados hasta el momento. También está *udevadm trigger* , que vuelve a escanear los dispositivos y ejecutar las reglas en /etc/udev/rules.d como se haría en un arranque normal, ó *udevadm control --reload-rules* para recargar las reglas sin tener que reiniciar el servidor ó desenchufar los dispositivos. Para modificar el nivel de logeo en syslog de Udev, se puede hacer *udevadm control –log-priority={err|info|debug}*. Para saber más lo que ocurre en el interior del sistema udev cuando se produce un evento, ejecutar *udevadm monitor* .També està *udevadm test /sys/class...* para comprobar que las reglas estén correctas.

Además de todo lo dicho, Udev tiene un archivo de configuración de sí mismo como programa que es: "/etc/udev/udev.conf" (ver *man udev*).

Protecciones contra "bad USBs"

Para protejer nuestro sistema de "bad USBs" como https://shop.hak5.org/products/usb-rubber-ducky-deluxe , https://ducktoolkit.com o https://maltronics.com/collections/malduinos se pueden utilizar o bien dispositivos hardware como https://github.com/robertfisk/USG o bien reglas Udev. No obstante, para facilitar el manejo de estas últimas, se puede emplear el software USBGuard (https://usbguard.github.io), el cual se basa en Udev para construir listas negras o blancas y sus reglas asociadas pero permite hacerlo de una forma mucho más cómoda y sencilla (un tutorial está aquí:https://wiki.archlinux.org/index.php/USBGuard)

NOTA: Desgraciadamente, no existe protección software contra el "USB Killer": https://usbkill.com/products/usb-killer-v3