**DIEGO**

**1.- Que es udev**

Udev es el gestor de dispositivos que usa el kernel Linux en su versión 2.6. Su función es controlar los ficheros de dispositivo en /dev. Es el sucesor de devfs y de hotplug, lo que significa que maneja el directorio /dev y todas las acciones del espacio de usuario al agregar o quitar dispositivos, incluyendo la carga de firmwares.

Fue desarrollado por Greg Kroah-Hartman, con ayuda de Dan Stekloff, Kay Sievers, y muchos otros.

Fue lanzado en noviembre de 2003 y su última versión estable es la 217, que fue lanzada en 2014.

Udev es ahora parte de systemd y es instalado de forma predeterminada en los sistemas Arch Linux.

**2.- Objetivo**

En un sistema Linux tradicional (sin *udev* ni *devfs*), en el directorio /dev hay nodos de dispositivo creados para cada dispositivo conocido, esté o no en el sistema.

Se dice que es un conjunto de ficheros *estático*, ya que los nodos no cambian.

Además, la forma de acceder a un periférico concreto no es siempre la misma, ya que depende de qué otros aparatos hay conectados. Ejemplo: si se conectan los discos A y B, se llamarán disco1 y disco2 respectivamente. Pero si está sólo un disco (el B, por ejemplo), se llamará disco1, porque sólo hay uno. El B ha cambiado de nombre.

**3.- Problemas antes**

El directorio /dev es enorme y difícil de manejar, ya que incluye todos los dispositivos posibles.

Los números mayor y menor que se asocian a cada dispositivo se estaban acabando.

Los usuarios necesitan que cada dispositivo sea accesible de la misma manera.  
Ejemplo: no aceptarán que por conectar un disco USB al sistema tengan que reconfigurar la cámara de vídeo.Los programas necesitan poder detectar cuándo se ha conectado o desconectado un dispositivo, y cuál es la entrada que se le ha asociado en /dev

udev soluciona estos problemas, sobre todo como hemos dicho antes el de poder acceder a un dispositivo con un nombre siempre fijo. Ésta fue la razón por la que se hizo udev, ya que antes estaba devfs, que solucionaba alguno de estos problemas, pero no todos.

**4.- devfs**

Device Filesystem (devfs) es un sistema de archivos virtual, utilizado por el sistema operativo Unix y los sistemas operativos derivados de este, cuyo propósito es controlar los archivos de dispositivos, que están almacenados en el directorio /dev de la estructura de archivos convencional.

Se introdujo como solución a los problemas de límite de números de dispositivos en los kernel de versiones anteriores y en la nomenclatura. Ha dejado de usarse en favor de udev, que hace la misma función, pero soluciona varios problemas que devfs no trata.

Devfs permite crear los archivos de dispositivos cuando se carga el módulo correspondiente. Además, el autor de módulo puede controlar el nombre del archivo y los derechos de acceso a éste.

**5.- Diferencias y similitudes udev/devfs**

devfs sólo muestra en /dev los dispositivos conectados, al igual que udev.

devfs, al igual que udev, permite que los programas sepan cuándo se conecta o desconecta un dispositivo, mediante consultas a un proceso demonio.

devfs no soluciona el problema de los números mayor/menor que se acaban, ya que no permite que sean dinámicos.

devfs no permite dar un nombre fijo a cada dispositivo (ésta es la razón por la que se hizo udev)

En devfs la política de nombres de dispositivo sigue dentro del kernel, no en los programas.

devfs no sigue el estándar de nombres definido en LSB (Linux Standard Base)

devfs es pequeño, pero se encuentra en espacio de kernel, que es memoria que no se puede llevar a disco.

**FRAN**

**6.- Características udev**

udev resuelve los problemas anteriores de la siguiente forma:  
  
Sólo dispositivos conectados: udev mantiene en /dev sólo las entradas correspondientes a los dispositivos que hay conectados al sistema. Así se soluciona el problema del /dev superpoblado.  
  
No se usa mayor y menor: No se usa el número mayor y menor para reconocer a cada dispositivo. Puede funcionar incluso aunque estén elegidos al azar. Por tanto, no le afecta el que se acaben las combinaciones mayor/menor asignables.  
  
Permite dar nombre fijo: Permite dar un nombre fijo para cada dispositivo, por ejemplo cámara, sin que éste dependa de qué otros dispositivos hay conectados ni del orden en que se han conectado.  
  
Un disco duro, por ejemplo, se reconoce por el identificador de su sistema de ficheros, el nombre del disco, y el conector físico en el que está.  
  
Avisa a los programas: Avisa mediante mensajes D-BUS, D-Bus (Desktop Bus) es un sistema de comunicación entre procesos (IPC), para aplicaciones de software con el fin de comunicarse entre sí; para que cualquier programa del espacio de usuario pueda enterarse cuando un dispositivo se conecta o desconecta (esto es útil para HAL). También permite a los programas consultar la lista de dispositivos conectados y la forma de acceder a cada uno de ellos.

**7.- Características udev (2)**  
  
Todo en espacio de usuario: udev hace que toda la política de nombres de dispositivo esté en espacio de usuario, y no en el kernel. Esto hace posible que un programa cualquiera pueda decidir el nombre de un dispositivo, por ejemplo basándose en sus características. No es necesario modificar el kernel si no se está conforme con el nombre que se le da a un dispositivo. Esto supone una diferencia con respecto a su antecesor, devfs, pues éste se ejecutaba en modo kernel  
  
Otra ventaja de tener el demonio de udev (udevd) en espacio de usuario es que esta memoria se puede llevar a disco (al espacio de intercambio), a diferencia de la memoria de kernel. Esto lo hace apropiado para sistemas embebidos con poca memoria física.  
  
Otras características: udev respeta la forma de nombrar dispositivos definida en el LSB, aunque permite que los usuarios usen otros nombres.  
  
El proceso (udevd) ocupa poca memoria y no necesita ejecutarse siempre. Esto también favorece a los sistemas embebidos y equipos poco potentes.

**7.- Implementación**

udev se ejecuta mediante un demonio: el proceso udevd, que detecta cuándo se ha conectado o desconectado un dispositivo del sistema.  
  
Cuando pasa uno de estos eventos, udev obtiene información de contexto (subsistema del kernel, conector físico usado, nombre dado por el kernel, ...) y también del propio dispositivo (número de serie, fabricante, ...). Esta información la puede encontrar mediante los datos que hay en /sys, en donde está montado el sistema de ficheros sysfs, es un sistema de archivos virtual que proporciona el núcleo Linux v2.6. Sysfs exporta información sobre los dispositivos y sus controladores desde el modelo de dispositivos del núcleo hacia el espacio del usuario, también permite configurar parámetros.  
Un conjunto de reglas (en /etc/udev/rules.d/) deciden qué acción hay que hacer a partir de los datos obtenidos. Lo que la regla hará será -probablemente- dar un nombre al dispositivo, crear el fichero de dispositivo apropiado, y ejecutar el programa que se haya configurado para que acabe de hacer funcionar el dispositivo.

En el caso de que existan dos reglas con el mismo nombre en /usr/lib y en /etc, la norma que se encuentra en la carpeta /etc tendrá prioridad.  
  
Las reglas pueden asociar un nombre fijo a un dispositivo, pero también pueden llamar a un programa externo que dé más información sobre el dispositivo; así se puede conseguir un nombre más específico.

Udev detecta automáticamente cambios en los archivos de reglas, por lo que los cambios surtan efecto inmediatamente sin necesidad de reiniciar udev. Sin embargo, las reglas no se recargan automáticamente en aquellos dispositivos que están en funcionamiento. Los dispositivos que se ponen en funcionamiento al conectarse, como los dispositivos USB, si ya estaban conectados, probablemente tendrán que volver a conectarse para que la nueva regla entre en vigor, o, al menos, recargar los módulos del kernel ohci-hcd y ehci-hcd y, por lo tanto, volver a cargar todos los controladores USB.

Algunas reglas para las reglas:

1-Las reglas están todas en una línea (las líneas se pueden romper con \ justo antes de la nueva línea).  
2.-Las reglas consisten en "matches" y "acciones".  
3.-Los matches y las acciones son trios de "valor" "operador" "valor".  
4.-Los matches tienen == ó != para el operador.  
5.-Las acciones tienen = (asignación) para el operador.  
6.-Los matches verifican uno o más atributos del evento para ver si se aplicará la acción.  
7.-Las acciones especifican lo que sucederá.  
  
Ejemplo de reglas:  
  
Dos posibles reglas que dan un nombre fijo a dos impresoras distintas; las dos conectadas por USB pero con un número de serie distinto (obtenido por sysfs):  
  
BUS=="usb", \ SYSFS\_serial=="W09090207101241330", \  
NAME="lp\_color"  
BUS=="usb", SYSFS\_serial=="HXOLL0012202323480", NAME="lp\_plain"