Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Al

Agente hotplug para automontaje de dispositivos USB. (25993 lectures)

Per **Héctor Rivas Gándara**, ***keymon*** ()

Creado el 23/05/2004 15:12 modificado el 24/05/2004 19:09

En un articulo anterior publicado en Bulma por Josep Sort[1], se explicaba cómo automatizar el montaje de dispositivos u
Inspirado en ese articulo y en otros de internet, propongo un script para hotplug (agent) que realizará el montaje automá
dispositivos permitiendo configurar el funcionamiento para dispositivos conocidos.
***Actualizado**: Soporte para varias particiones y unidades sin particionar*

Presentación y requisitos

El script nos permitirá:

- Solucionar el problema de la gestión de los dispositivos de almacenamiento usb, pues cada vez que los enchufamos se les *device* distinto (/dev/sda, /dev/sdb....).
- Usar supermount, con las ventajas asociadas. Adaptarlo a otros sistemas es sencillo.
- Montar (o crear un enlace simbólico) de los dispositivos usb conectados.
- Definir un directorio por defecto en el que se montarán los dispositivos *usb-storage* no configurados (añadiendo un sufijo ocupado dicho punto de montaje)
- Configurar determinados dispositivos, pudiendo definir un punto de montaje alternativo, para cada dispositivo, LUN, o pa
- Configurar un enlace simbólico que se creará apuntando al dispositivo usb, en vez de montarlo. Esto es muy interesante co podemos configurar supermount en nuestro /etc/fstab para que apunte a este enlace en vez del dispositivo.
- Soporte de dispositivos con varios *luns*, como los lectores de tarjetas *n in 1*.

Para su funcionamiento correcto necesitaremos lo siguiente:

- *hotplug*
- *usb-utils*
- *supermount*, aunque es sencillo adaptarlo a otros sistemas.
- *kernel 2.6.X*, con soporte del sistema de ficheros /sys.
- Soporte *usb-storage* en el kernel
- Activar la opción CONFIG_SCSI_MULTI_LUN si queremos usar lectores de tarjetas *n in 1*

Nota: El script sólo ha sido probado en Debian, y es posible que en otras distribuciones no existan algunos directorios y ficheros de utilizados, o que el funcionamiento del software empleado se distinto. Se agradece cualquier aclaración o comentario al respecto :)
Además daré por supuesto que el sistema está correctamente configurado para el uso de este tipo de sistemas.
No entraré en detalles de la programación del script, sólo de su configuración.

Instalación / configuración

1. Copiaremos el script de abajo en /etc/hotplug/usb/usb-storage.
   No os olvidéis de los permisos de ejecución!!!.
   Opcionalmente, podemos editarlo para cambiar las opciones por defecto, como DEFAULT_MOUNT_POINT DEFAULT_MOUNT_OPTIONS. También podemos modificar las funciones mountUsbDevice y umountRemoveScript, p sistema en vez de supermount, o añadir funcionalidades extra, como crear enlaces en escritorios.
2. Si deseamos configurar determinados dispositivos, debemos crear el fichero /etc/hotplug/usb-storage.map con la siguiente

```
#
# Configuration file for usb-storage agent
#
```

```
#    mount
# or
#    symbolic
#
# if partition number = 0, it will use whole unit (/dev/sd?)
#
05dc:0080 0 0 mount    /mnt/disco_usb defaults,user,umask=0002,rw
0424:223a 0 1 symbolic /dev/usbcard0
0424:223a 1 1 symbolic /dev/usbcard1
0424:223a 2 1 symbolic /dev/usbcard2
0424:223a 2 2 mount    /mnt/usbcard_particion_2 defaults,user,umask=0002,rw
```

Donde:

- **idVendor:idProduct**: son los identificadores de vendedor y producto que se pueden obtener con el comando lsus incluir el "0x"
- **lun**: número de LUN (Logical Unit Number) para este dispositivo. Para usar cuando tenemos un dispositivo con como los lectores de tarjetas.
  Normalmente pondremos 0.
- **partition**: Partición dentro de esta unidad. Actualmente el script no soporta esto, y siempre emplea la partición 1
- **mount/symbolic**: Si deseamos montar esta partición o crear un enlace simbólico del *device*.
- **mount point**: Directorio en el que se montará
- **mount options**: Opciones que se pasarán a mount
- **symbolic device**: Nombre del enlace simbólico que se creará apuntando al dispositivo.

En ese ejemplo podemos ver que el dispositivo **05dc:0080** (un disco usb Lexar) es montado en el directorio /mnt/disco_us tiene particiones, se monta directamente toda la unidad, por lo que tiene el número de particion 0
En cambio, el dispositivo **0424:223a** (un lector de tarjetas) tiene varios LUNs, y se creará un enlace para cada uno en /mn LUN 2, tenemos configuradas dos particiones, la 1 y la 2, y la segunda partición, en vez crear un enlace, se montará.
Para estos dispositivos tengo lo siguiente en /etc/fstab.

```
none /mnt/tarxetas/CompactFlash supermount dev=/dev/usbcard0,--,defaults,user,umask=0002,rw 0 0
none /mnt/tarxetas/MemoryStick supermount dev=/dev/usbcard1,--,defaults,user,umask=0002,rw 0 0
none /mnt/tarxetas/SmartMedia supermount dev=/dev/usbcard2,--,defaults,user,umask=0002,rw 0 0
none /mnt/tarxetas/SecureDigital supermount dev=/dev/usbcard3,--,defaults,user,umask=0002,rw 0 0
```

Ahora sólo tenemos que enchufar y desenchufar nuestros dispositivos

Notas

- No hagais cosas raras con las particiones y el fichero de configuración. Respetad la sintaxis, y no pongais cosas como otra poneis la partición especial "0", etc...
- Cuando el dispositivo no está configurado tarda en montar, porque tiene que comprobar si la partición (/dev/sd?1 o /dev/s

Trabajos futuros

Se pueden mejorar varias cosas, pero destacan:

- Pasarlo a perl, con lo que sería muuucho más sencillo %-?
- Traducir este articulo :)

Script *usb-storage*

Copia y pega el siguiente script en /etc/hotplug/usb/usb-storage:

```
#!/bin/bash
#
# This code is free software covered by GNU GPL license version 2 or above.
# Please refer to http://www.gnu.org/ for the full license text.
#
# Some code lifted from Konstantin Riabitsev's diskonkey (GPL)
```

```
#
# INSTALL
# -------
# Put this in /etc/hotplug/usb/usb-storage
# Be sure that in /etc/hotplug/usb.distmap are the usb-storage entries
# Create file etc/hotplug/usb-storage.map with this sintax:
#      mount
#   or
#      symbolic
#
# For a device with no partitions (the filesystem is in /dev/sd?),
# use partition number 0
#
# For Example:
#   05dc:0080 0 0 mount    /mnt/disco_usb defaults,user,umask=0002,rw
#   05dc:0081 0 0 mount    /mnt/otro_disco_usb defaults,user,umask=0002,rw
#   0424:223a 0 1 symbolic /dev/usbcard0
#   0424:223a 1 1 symbolic /dev/usbcard1
#   0424:223a 2 1 symbolic /dev/usbcard2_1
#   0424:223a 2 2 symbolic /dev/usbcard2_2
#
# TODO
# ----
# Refactorize this script in perl :)
#
# AUTHOR and SUPPORT
# ------------------
# Hector Rivas Gandara (keymon), keymon(aT)wanadoo.es .
#
DEFAULT_MOUNT_POINT=/mnt/chave
DEFAULT_MOUNT_OPTIONS=defaults,user,umask=0002,rw
CONFIG_FILE=/etc/hotplug/usb-storage.map

# Get's the usb device serial number for a given usb device.
# Return the 20 first digits
function getUsbDeviceSerialNumber {
    lsusb -v -D $1 | grep -m 1 iSerial | sed 's/  */ /g' | cut -f 4 -d " " | sed 's/\(.\{20\}\).*/\1/g'
}

# Get's the usb device vendor:product for a given usb device.
function getUsbDeviceVendorProduct {
    local vendor=`lsusb -v -D $1 | grep -m 1 idVendor | sed 's/  */ /g;s/^ //' | cut -f 2 -d " " | sed 's
    local product=`lsusb -v -D $1 | grep -m 1 idProduct | sed 's/  */ /g;s/^ //' | cut -f 2 -d " " | sed
    echo $vendor:$product
}

# Get's the scsi host number associated to a usb-storage device
# with the given serial number
function getScsiDeviceFromSerialNumber {
    for i in /proc/scsi/usb-storage/*; do
        if grep -q $1 $i ; then
            local scsiDev=`echo $i | cut -f 5 -d /`;
        fi
    done
    echo $scsiDev
}


# Gets device name for the given scsi address
function getScsiDeviceNameFromAddress {
    local scsiAddress=$1
    # get the device minor number and calculate the leter
    local minor=`cat "/sys/class/scsi_device/${scsiAddress}/device/block/"dev | cut -f 2 -d :`
    local letter=`expr substr abcdefghijkmnopqrs \( $minor / 16 + 1 \) 1`
    echo /dev/sd$letter
}

# Gets partition devices in the given scsi address
function getScsiDevicePartitions {
    local scsiAddress=$1
    local devs=""
```

```
    for i in "/sys/class/scsi_device/${scsiAddress}/device/block/"sd*; do
        dev=/dev/`echo $tmp | cut -f 8 -d /`
        devs="$devs $dev"
    done
    echo devs
}

# checks for a valid scsi address
function isValidScsiAddress {
    local scsiAddress=$1
    test -e /sys/class/scsi_device/${scsiAddress}
    return $?
}

# check if a device is mountable
# FIXME: I don't like too much this way. Have anybody a better idea???
function isMountable {
    local device=$1;
    local tmpMountPoint=/tmp/.usb-storage.$$
    mkdir -p $tmpMountPoint
    if mount -t auto $device $tmpMountPoint &> /dev/null; then
        umount $tmpMountPoint
        rmdir $tmpMountPoint
        return 0
    else
        rmdir $tmpMountPoint
        return 1
    fi
}

# Adds a umount line in the remove script. creates the remove script
# if necesary
# Param: mount point of mounted device
# Param: removeMountPoint = 1 if have to remove or 0 if not
function addMountPointForRemover {
    local mountPoint=$1
    local removeMountPoint=$2

    # create file if necesary
    if [ ! -e $REMOVER ]; then
        echo "#!/bin/sh"> $REMOVER
        chmod +x $REMOVER
    fi

    # lazy umount
    echo "umount -l \"$mountPoint\"" >> $REMOVER

    if [ "$removeMountPoint" = "1" ]; then
        echo "rmdir \"$mountPoint\"" >> $REMOVER
    fi

}

# Mounts a device in the given point, with the given options
# If mount point is already mounted, create other with a suffix
# Returns the effective mount point
function mountDevice {
    local device=$1;
    local mountPoint=$2;
    local mountOptions=$3;
    local removeMountPoint=0;

    # check that mount point is not busy
    local -i counter=2;
    local realMountPoint=$mountPoint;
    while mount | grep -q $realMountPoint; do
        # check that the device is not mounted already in the same mountpoint
        if mount | grep $realMountPoint | grep -q $device; then
            return
        fi
        realMountPoint=$mountPoint$counter
```

```
        counter=counter+1
    done

    # Supermount mount
    # Create dir if necesary
    if [ ! -e $realMountPoint ]; then
        mkdir -p $realMountPoint
        removeMountPoint=1
    fi

    mount -t supermount none $realMountPoint -o dev=$device,fs=auto,--,$mountOptions

    # Add it to remove script
    addMountPointForRemover $realMountPoint $removeMountPoint

}


# Reads the configuration file, striping comments and removing blanks
function getConfiguration {
    cat $CONFIG_FILE | sed 's/\(.*\)#.*/\1/;s/  */ /g;s/^ //'
}

# Process the action for a concrete device and partition
# params:
#
# or
#
function processPartition {
    local device=$1;
    local action=$2;

    if [ "$action" = "mount" ]; then
        local mountPoint=$3;
        local mountOptions=$4;
    else
        local symbolicDevice=$3;
    fi

    if [ "$action" = "mount" ]; then
        # Mount it to the mount point
        mountDevice $device $mountPoint $mountOptions
    else
        # Set symbolic device
        ln -sf $device $symbolicDevice
    fi

}

# Process a device. This function gets the configuration for the defined
# device and calls processScsiLunPartition for each partition
# param: usbVendorProduct String with the key "idProduct:idVendor"
# param: usbHostNumber asigned host number
# param: lun Scsi lun number
function processScsiLun {

    local usbVendorProduct=$1
    local usbHostNumber=$2
    local lun=$3


    local confType;
    local mountPoint;
    local mountOptions;
    local symbolicDevice;
    local partition;
    local configurationLine;
    local partitionDevice;

    # Get the base  device name (/dev/sd?)
    local deviceBase=$(getScsiDeviceNameFromAddress $scsiHostNumber:0:0:$scsiLun)
```

```
    # get all partitions defined in configuration file for this product and lun
    local configuredPartitions=`getConfiguration | grep "$usbVendorProduct $lun" | cut -f 3 -d " "`

    # if there is configuration, process each partition
    if [ ! "$configuredPartitions" == "" ]; then
        for partition in $configuredPartitions; do
            configurationLine=`getConfiguration | grep -m 1 "$usbVendorProduct $lun $partition"`

            # If there is a configuration line, use it. It should be there :)
            if [ ! "$configurationLine" == "" ]; then
                # check for special partition number 0
                if [ "$partition" = "0" ]; then
                    partitionDevice=${deviceBase}
                else
                    partitionDevice=${deviceBase}${partition}
                fi
                confType=`echo $configurationLine | cut -d " " -f 4`
                if [ "$confType" = "mount" ]; then
                    mountPoint=`echo $configurationLine | cut -d " " -f 5`
                    mountOptions=`echo $configurationLine | cut -d " " -f 6`
                    processPartition $partitionDevice $confType $mountPoint $mountOptions
                else
                    symbolicDevice=`echo $configurationLine | cut -d " " -f 5`
                    processPartition $partitionDevice $confType $symbolicDevice
                fi
            fi
        done
    # if there is no configuration, search for filesystem in sd? or sd?1
    else
        # check if there is a filesystem in sd?1, using fdisk (is faster)
        if fdisk -l ${deviceBase} | grep -q ${deviceBase}1; then
            processPartition ${deviceBase}1 mount $DEFAULT_MOUNT_POINT $DEFAULT_MOUNT_OPTIONS
        else
            # check if there is a filesystem in sd?1
            if isMountable ${deviceBase}; then
                processPartition ${deviceBase} mount $DEFAULT_MOUNT_POINT $DEFAULT_MOUNT_OPTIONS
            fi
        fi
    fi

}

if [ "${ACTION}" = "add" ]; then

    #####
    # Set the remover to this script
    # ln -s $0 $REMOVER

    #####
    # Wait a little, for the device to be created
    sleep 1


    #####
    # Get device info
    usbSerial=$(getUsbDeviceSerialNumber $DEVICE);
    usbVendorProduct=$(getUsbDeviceVendorProduct $DEVICE)

    # Get the scsi host
    scsiHostNumber=$(getScsiDeviceFromSerialNumber $usbSerial);

    ####
    # Search for all luns
    declare -i scsiLun=0
    while isValidScsiAddress $scsiHostNumber:0:0:$scsiLun; do
        processScsiLun $usbVendorProduct $scsiHostNumber $scsiLun
        scsiLun=$scsiLun+1
    done

fi
```

```
if [ "${ACTION}" = "remove" ]; then
    true
fi

exit;
```

Lista de enlaces de este artículo:

1. http://bulma.net/body.phtml?nIdNoticia=2024

E-mail del autor: keymon _ARROBA_ wanadoo.es
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=2034