montage des disques externes

Sommaire

dans un environnement minimal, le montage des disques externes se fait à la ligne de commande.

pour une clé usb partitionnée en fat32:

```
# mkdir /mnt/usb
# mount -t vfat /dev/sda2 /mnt/usb
```

la manipulation est simple mais utilise les droits 'root'. pour automatiser le montage et passer par le compte utilisateur, nous pouvons utiliser plusieurs outils: fstab, udisks-glue ou devmon.

montage semi-automatique avec fstab

/etc/fstab est le fichier qui regroupe les points de montage du système.

montage de clé lambda

le principe : créer des points de montage “universel” pour les clés usb en vfat:

- création des points de montage: autant que de ports usb disponibles...

```
# mkdir /media/usbdisk-1
# mkdir /media/usbdisk-2
# ....
```

- déclaration des points de montages dans fstab: ouvrir fstab

```
# vim /etc/fstab
```

et ajouter les lignes correspondantes aux usbdisk (une par point de montage)

```
/dev/sdb1    /media/usbdisk-1    vfat    rw,users,noauto    0    0
/dev/sdc1    /media/usbdisk-2    vfat    rw,users,noauto    0    0
....
```

de cette façon, toute clé usb formatée en vfat (la majorité) sera accessible par le dossier /media/usbdisk-1. si vous insérez une seconde clé , elle sera accessible depuis /media/usbdisk-2

... et ainsi de suite pour tous vos ports usb disponibles.

notez que cette méthode ne prend pas en compte les clé avec 2 partitions ou plus, ni celle formatées en ext3/4.

montage de clé identifiée

Si vous utilisez toujours les mêmes clés , vous pouvez définir des points de montage spécifiques:

- création du point de montage spécifique:

```
# mkdir /media/le_nom_qui_claque
```

- récupérer l'uuid du disque usb à monter: brancher votre clé usb puis lister les disques disponibles:

```
23:56 arp > ls -l /dev/disks/by-uuid
total 0
lrwxrwxrwx 1 root root 10 Mar  3 23:29 7a2980d8-593c-4fcf-a145-b448f4dee02c
-> ../../sda2
lrwxrwxrwx 1 root root 10 Mar  3 23:56 8A43-6E99 -> ../../sdb1
lrwxrwxrwx 1 root root 10 Mar  3 23:29 e171837c-895c-444d-8699-0f4760cca2ca
-> ../../sda1
 23:56 arp >
```

vous remarquez l'uuid associée à /dev/sdb1 , c'est celui de votre clé usb.

- déclarer votre clé dans fstab

```
# vim /etc/fstab
```

et ajouter la ligne suivante (à adapter selon votre uuid) :

```
UUID=8A43-6E99    /media/le_nom_qui_claque    vfat    rw,users,noauto    0    0
```

et voilà , votre clé préférée se montera dans votre dossier spécifique , quel que soit le port utilisé.

montage automatique avec udisks-glue

installation

udisks-glue est un démon qui reçoit et envoi des instructions depuis/vers **udisks**.

le paquet est disponible sur Debian Wheezy, il suffit de l'installer:

```
# apt-get install udisks-glue
```

configuration

udisks-glue se configure depuis un simple fichier texte: on le récupère depuis /etc

```
$ cp /etc/udisks-glue.conf ~/.udisks-glue.conf
```

mon fichier pour l'exemple avec notifications (dépend de dzen2)

```
default {
        post_insertion_command = "echo -e '%device_file insert' | dzen2 -x 0 -y 0
-ta l -p 4"
}
filter disks {
        optical = false
        partition_table = false
        usage = filesystem
}
match disks {
        automount = true
        post_mount_command = "echo -e '^fg(cyan)usb mounted: ^fg(white)
%device_file on %mount_point' | dzen2 -x 0 -y 0 -ta l -p 5 && rox %mount_point"
        post_unmount_command = "echo -e '^fg(red)usb unmounted: ^fg(white)
%device_file on %mount_point' | dzen2 -x 0 -y 0 -ta l -p 5"
}
filter optical {
        optical = true
}
match optical {
        automount = true
        automount_options = ro
        post_mount_command = "echo -e '^fg(yellow)cd-rom mounted: ^fg(white)
%device_file on %mount_point' | dzen2 -x 0 -y 0 -ta l -p 7 && rox %mount_point"
        post_unmount_command = "echo -e '^fg(red)cd-rom unmounted: ^fg(white)
%device_file on %mount_point' | dzen2 -x 0 -y 0 -ta l -p 7"
}
```

ce fichier affiche une notification avec dzen2 lors de l'insertion d'un média externe, annonce le montage du périphérique, puis ouvre le point de montage avec rox-filer (à adapter à vos préférences).

lancement

udisks-glue se lance simplement depuis votre script de démarrage, ~/.xinitrc ou autre (dbus doit être lancé et actif)

```
sleep 5s && udisks-glue --session &
```

montage automatique avec devmon

devmon est un script bash communiquant avec **udisks** et permettant le montage automatique des disques externes en tant que simple utilisateur. il dépend de udisks/udev et recommande **zenity** pour les notifications (optionnel) et **eject**.

installation

devmon fait partie de udevil mais peut être installé indépendamment.

```
$ wget https://raw.github.com/IgnorantGuru/udevil/master/src/devmon
# apt-get install eject zenity
# install devmon /usr/local/bin/devmon
```

la configuration de devmon se fait depuis la ligne de commande, au lancement de l'application.

lancement

- **mode démon** : devmon se lance depuis votre script de démarrage, ~/.xinitrc ou autre (dbus doit être lancé et actif):

```
$ devmon [AUTOMOUNT-OPTIONS]  # lancement en mode démon pour un (dé)montage automatique
```

- **mode client** : devmon peut aussi être lancé en cours de session afin d'obtenir des informations sur les disques présents et un (dé)montage manuel:

```
$ devmon [MOUNT-OPTIONS]      # lancement en mode client pour un (dé)montage manuel
```

- les **logs devmon** : si vous désirez garder une trace des actions de devmon, il suffit de le lancer comme ceci:

```
$ devmon &>/tmp/devmon.log
```

options

devmon accepte les options suivantes selon le mode de lancement démon-auto/manuel (extrait de devmon –help):

options (démon)	arguments	description
--exec-on-device	DEVICE “COMMAND”	exécute COMMAND après avoir monté DEVICE
--exec-on-label	“LABEL” “COMMAND”	exécute COMMAND après avoir monté LABEL
--exec-on-video	“COMMAND”	exécute COMMAND après le montage d'un DVD
--exec-on-audio	“COMMAND”	exécute COMMAND après le montage d'un CD audio
--exec-on-disc	“COMMAND”	exécute COMMAND après avoir monté un CD/DVD de données
--exec-on-drive	“COMMAND”	exécute COMMAND après avoir monté un disque dur externe (hdd/usb)
--exec-on-unmount	“COMMAND”	exécute COMMAND après avoir démonté un volume
--exec-on-remove	“COMMAND”	exécute COMMAND après avoir retiré un média. macros: %d: point de montage (ex:/media/cd), %f: nom du périphérique (ex:/dev/sdd1) et %l: label du volume monté.
		vous pouvez utiliser plusieurs “--exec-on-XXX” afin d'exécuter plusieurs commandes. les autres “exec-on-XXX” sont ignorées si vous utilisez “exec-on-device” ou “-label”.
--mount-options	“OPTIONS”	défaut: $defaultmountoptions
--info-on-mount		affiche les informations sur le volume monté dans une fenêtre pop-up.
--no-mount		ne fait rien, désactive --exec-on-video.
--no-unmount		ne démonte pas les médias amovibles à la fermeture du démon.
options (client)	**arguments**	**description**
--unmount-removable, -r		synchronise et démonte tous les médias amovibles et affiche une fenêtre de dialogue.
--unmount-recent, -c		démonte le média monté en dernier.
--unmount-optical, -o		démonte les cd/dvd et affiche une fenêtre pop-up en cas d'erreur.

--unmount-all, -u		idem '-unmount-removable -unmount-optical'.
--unmount	DIR, DEVICE	démonte DEVICE ou le point de montage DIR.
--eject	DIR, DEVICE	démonte et éjecte DEVICE ou le point de montage DIR.
--mount-all, -a		monte tous les médias amovibles et cd/dvd.
--mount	DEVICE	monte DEVICE.
--mount-options	OPTIONS	les OPTIONS seront passées au montage et à l'éjection d'un média.
--mount-fstype		
--eject-options		
options (communes)	**arguments**	**description**
--ignore-device	DEVICE	ignorer DEVICE (ex:/dev/sdd1).
--ignore-label	“LABEL”	ignorer le volume “LABEL”.
--sync, -s		ajoute la synchronisation lors du montage pour ext2-4 ntfs ufs, ou 'flush' pour fat & vfat (écriture ralentie mais sécurisée).
--internal		tente de gérer les périphériques internes (pour fixer le bug des Esata).
--no-gui, -g		pas de fenêtres pop-up.

dbus actif

udisks (utilisé par udisks-glue et devmon) communique grâce à dbus. votre session doit donc être active. pour cela, dans un environnement minimal, on utilise la commande “ck-launch-session”.ex dans mon ~/.xinitrc:

```
#!/bin/bash
# ~/.xinitrc by arpinux 2013
############################
## D-Bus ##-----------------------------------------------------------
if which dbus-launch >/dev/null && test -z "$DBUS_SESSION_BUS_ADDRESS"; then
        eval "$(dbus-launch --sh-syntax --exit-with-session)"
fi
(...)vos application(...)
## launch dwm-session script ##############################################
exec ck-launch-session bash -c "devmon [OPTIONS] & $HOME/bin/dwm-session
```

pour tester l'activité de votre session:

```
$ ck-list-sessions
```

vous devez avoir une sortie de ce style (notez le 'active = TRUE'):

```
     unix-user = '1000'
     realname = 'arpinux'
     seat = 'Seat1'
     session-type = ''
     active = TRUE
     x11-display = ':0'
     x11-display-device = '/dev/tty7'
     display-device = '/dev/tty1'
     remote-host-name = ''
     is-local = TRUE
     on-since = '2013-03-03T22:30:00.423066Z'
     login-session-id = '1'
```

cas particulier: l'autorisation ne vous est pas accordé malgré la session active: éditez le fichier /etc/pam.d/common-session et assurez-vous d'avoir ceci (éditez en conséquences):

```
#
# /etc/pam.d/common-session - session-related modules common to all services
#
# This file is included from other service-specific PAM config files,
# and should contain a list of modules that define tasks to be performed
# at the start and end of sessions of *any* kind (both interactive and
# non-interactive).
#
# As of pam 1.0.1-6, this file is managed by pam-auth-update by default.
# To take advantage of this, it is recommended that you configure any
# local modules either before or after the default block, and use
# pam-auth-update to manage selection of other modules.  See
# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
session    [default=1]                  pam_permit.so
# here's the fallback if no module succeeds
session    requisite            pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
session    required             pam_permit.so
# and here are more per-package modules (the "Additional" block)
session    required           pam_unix.so
session optional                  pam_loginuid.so
session    optional             pam_ck_connector.so nox11
# end of pam-auth-update config
```