Pendrive-ok automatikus mentése

Használjuk ki az udev képességeit és készítsünk biztonsági mentést Flash meghajtónk tartalmáról naponta, vagy akár minden egyes csatlakoztatáskor...

Tegye föl a kezét, aki még soha nem vesztette el az ***USB Pendrive***-ját – vagy legalább nem gyanított ilyesmit. Aztán amikor az ember már órákat töltött az elbitangolt jószág előkerítésével, akkor egyszer csak kinyitja a mosógép ajtaját, és hopp, az eszköz a lábai elé hullik. Nos, igen... Már megint elfelejtettük akkurátusan kiforgatni az összes nadrág zsebeit, mielőtt betettük volna ezt az adagot. De semmi gond, hiszen minden adatunkról van biztonsági másolatunk. Ugye van? Ugye bizonyos időközönként rutinszerűen bemásoljuk a Flash meghajtó teljes tartalmát a merevlemezre? Nem?! Hát, akkor gond van. A rendszeres adatmentés ilyetén kivitelezése természetesen számomra is borzalmasan unalmasnak tűnik. Mellesleg pont az ilyen feladatok automatizálására találták ki a számítógépeket. Meg aztán az sem mellékes, hogy az automatikus és főként fájdalommentes adatmentési módszer kidolgozása maga is élvezet egy vérbele felhasználó számára.
Na, de nem ártana tisztázni, mit is értek pontosan az alatt, hogy „fájdalommentes"? Mit szólna például a nyájas olvasó egy olyan megoldáshoz, amelynél az ember csak odasétál egy linuxos géphez, csatlakoztatja a pendrive-ját, megvárja „a mentés elkészült" hangot, aztán kiveszi az eszközt a csatlakozóból, és továbbsétál. Na és persze azt se felejtsük el, hogy néha nem árt, ha egy ilyen rendszernek van egy viszonylag rövid távú emlékezete is, vagyis mondjuk az utolsó hét mentés tartalma legyen bármikor hozzáférhető. Ja igen, és mondjuk kezelje gond nélkül a titkosított meghajtókat is.
Na és még valami: ha arra kerülne a sor, hogy tényleg vissza kelljen állítani valamit a mentéből, akkor legyen képes teljes és fájlonkénti visszaállításra is. Na, szóval ennyi lenne. Aki ***Linuxot*** használ, annak mindez nem gond. Valamennyi fent leírt funkciót megvalósíthatjuk az udev képességeire támaszkodva egy egyszer héjprogrammal. A szükséges eszközök már ott vannak valamennyiünk rendszerén. Ebben a cikkben és a ***CentOS 4.3*** rendszeren fogom bemutatni a probléma megoldását, amely 2.6-os kernelt használ, de minden bemutatott dolog természetesen más rendszereken is megvalósítható.

Az udev és az adatmentés

Az ***udev*** egy modern eszközkezelő, amely a 2.4-es ***Linux*** kernelben található ***devfs*** rendszert váltotta le. Az ***udev*** képes valamennyi eszközt a rendszerbe leképezni, beleértve természetesen a menet közben csatlakoztathatókat (hotplug) is. Az egyik legnagyszerűbb szolgáltatása az, hogy akár saját eseménykezelőket is írhatunk hozzá.
És ezzel el is érkeztünk a cikk témájához, hiszen a következőkben arról lesz szó, hogyan írhatunk meg egy olyan eseménykezelőt, amely az ***USB*** pendrive csatlakoztatásakor lementi annak teljes tartalmát a merevlemezre. A szabályok a */etc/udev/rules.d* könyvtárban találhatók. (Aki más terjesztést használ, az ellenőrizze a */etc/udev/udev.conf* fájlban az `udev_rules=` kezdetű sort. Elvileg ez tartalmazza a szabályokat hordozó könyvtár nevét.) Ebben a könyvtárban gyakorlatilag bármilyen ***udev*** szabályt elhelyezhetünk. A rendszer az új előírásokat ráadásul azonnal használni kezdi, anélkül, hogy a gépet újra kellene indítanunk.

Az eszközök azonosítása

Ahhoz, hogy megírhassunk egy szabályt az ***udev*** rendszerhez, először is rendelkeznünk kell egy olyan módszerrel, amivel egyértelműen azonosítani tudjuk az egyes ***USB*** eszközöket.
A legtöbb pendrive-nak van sorozatszáma, bár ez nem mindegyikről mondható el. Szerencsére az ***udev***-nek arra az esetre is van megoldása, ha nincs az eszköznek ilyen egyedi azonosítója. Jómagam két pendrive-ot használok: egy ***JetFlash JF110***-et, amelynek a tartalma ***TrueCrypt*** segítségével van titkosítva, és egy ***Corsair Flash Voyager*** típusút. A ***JetFlash***-nek van egyedi azonosítószáma, a ***Corsairnak*** azonban nincs. Csatlakoztassuk a pendrive-ot, majd adjuk ki a

```
cat /proc/scsi/usb-storage/*
```

parancsot. Erre egy ehhez hasonló információhalmazt kapunk:

```
   Host scsi5: usb-storage
       Vendor: Unknown
      Product: USB Mass Storage
➥Device
Serial Number: 85a5b1f2c96492
     Protocol: Transparent SCSI
    Transport: Bulk
       Quirks:
```

Ha a kiírt adatok között van sorozatszám, akkor az ebbe a szakaszban leírtakat ki is hagyhatjuk, és nekiláthatunk magának a szabálynak a megírásához. Ha azonban a ***„Serial Number"*** rovatban a ***„None"*** kifejezés díszeleg, akkor sincs minden veszve, mert az `udevinfo` segítségével rendelhetünk az eszközhöz egyedi azonosítót. Ehhez a következőket kell tennünk.

1. Nézzük meg a `dmesg` parancs kimenetét. Egy tipikus részlet a következőképpen fest:

```
usb-storage: waiting for device
➥to settle before scanning
  Vendor: Corsair   Model:
➥Flash Voyager     Rev: 1.00
  Type:   Direct-Access
  ANSI SCSI SCSI device sde:
➥2031616 512-byte hdwr sectors
➥(1040 MB)
[...]
sde: assuming drive cache:
➥write through
 sde: sde1
Attached scsi removable disk
➥sde at scsi12, channel 0, id
➥0, lun 0
Attached scsi generic sg4 at
➥scsi12, channel 0, id 0, lun
➥0,  type 0
```

A fenti szöveg szerint eszközünk a `/dev/sde` ponton keresztül kapcsolódik a rendszerhez.

2. Futtassuk a következő parancsot:

```
udevinfo -a -p $(udevinfo -q
➥path -n /dev/sde)
```

majd vizsgáljuk meg a kimenetét. A következő sorokat kell keresnünk:

```
BUS=="scsi"
SYSFS{model}=="Flash Voyager  "
SYSFS{vendor}=="Corsair "
```

A kezelési szabály megírása

Ha kezünkben a sorozatszám, vagy a gyártó/modell páros, akkor nekiláthatunk a kezelési szabály megírásának. A szkript úgy fog működni, hogy létrehoz egy az eszközre mutató szimbolikus láncot a */dev* fában (esetünkben ez lehet például */dev/corsair_drive*), majd meghívja a */usr/local/bin/backup-thumb.sh* szkriptet, amit mindjárt részletesen is megvizsgálunk. Jelentkezzünk be rendszergazdaként (`su -`) majd a */etc/udev/rules.d* könyvtárban hozzunk létre egy *95.backup.rules* nevű szövegfájlt. Természetesen nem kötelező a 95-ös számot használni, de ne felejtsük el, hogy az *udev* ábécé sorrendben dolgozza fel a szabályokat, a helyi műveleteket pedig jobb a sorozat végére tenni.

1. Lista Az adatmentéshez használt szkript

```
#!/bin/bash
# Pendrive mentésére szolgáló
# szkript
##############################
# CONFIG szakasz
# Hol akarjuk tárolni a mentett
# adatokat
BACKUP_DIR=/backups/thumb
# Hány visszamenőleges
# változatot szeretnénk tárolni
GENERATIONS=7
# Naponta csak egy mentés
# készüljön
# Ha azt szeretnénk, hogy
# a pendrive minden egyes
# csatlakoztatásakor
# lefusson a folyamat, írjunk
# ide nullát
BACKUP_ONCE_DAY=1
# Ha a mentés kész, a következő
# nagállományt játssza le
# a rendszer
SOUND=/usr/share/sounds/KDE_Bee
➥p_ClockChime.wav
# END CONFIG
##############################
# Főprogram
# Megvárjuk, míg az eszköz
# üzemkész állapotba kerül
sleep 10
# Gondoskodunk róla, hogy
# a mentést senki más ne tudja
# lemásolni.
umask 077
# Ellenőrizzük, hogy létezik-e
# a könyvtár
DEVICE=$1
if [ ! -d ${BACKUP_DIR} ] ;
then
        mkdir -p ${BACKUP_DIR}
fi
# Naponta csak egyszer mentünk
if [ ${BACKUP_ONCE_DAY} -gt 0 ]
# ; then
       DIDTODAY=${BACKUP_DIR}/$
➥{DEVICE}.did_today
        find ${BACKUP_DIR} -
➥name ${DEVICE}.did_today -a
➥-mtime +1 -delete
        if [ -f ${DIDTODAY} ] ;
➥then
                exit
        else
                touch $
➥{DIDTODAY}
        fi
fi
# Mentések cseréje (rotációja)
cd ${BACKUP_DIR}
let GENERATIONS=${GENERATIONS}-1
while [ ${GENERATIONS} -ge 0 ]
➥; do
  let NEWFILE=${GENERATIONS}+1
        if [ -f ${DEVICE}.
➥backup.${GENERATIONS} ] ; then
          mv -f ${DEVICE}.
➥backup.${GENERATIONS}
${DEVICE}.backup.${NEWFILE}
        fi
  let GENERATIONS=$
➥{GENERATIONS}-1
done
# A mentés tényleges
# végrehajtása
dd if=/dev/${DEVICE} of=$
➥{BACKUP_DIR}/${DEVICE}.backup
➥.0 > /dev/null 2>&1
# Hangjelzés küldése a folyamat
# végén
aplay ${SOUND} > /dev/null 2>&1
```

Ha eszközünk rendelkezik sorozatszámmal, akkor a szabálynak valahogy így kell kinéznie (az egésznek egyetlen sorban kell lennie):

```
BUS="usb", SYSFS{serial}=
➥"85a5b1f2c96492", SYMLINK=
➥"jet_drive",
RUN+="/usr/local/bin/
➥backup-thumb.sh jet_drive "
```

Ha nem tartozik az eszközhöz egyedi numerikus azonosító, akkor a gyártó/modell párost kell használnunk erre a célra. Ebben az esetben az új szabály a következőképpen fest:

```
BUS="scsi", SYSFS{vendor}==
➥"Corsair ", SYSFS{model}==
➥"Flash Voyager  ",
SYMLINK="corsair_drive", RUN+=
➥"/usr/local/bin/
➥backup-thumb.sh
corsair_drive"
```

Érdemes talán megjegyezni, hogy szükség esetén tetszőleges számú `SYSFS{}` bejegyzés fűzhető össze, a lényeg, hogy a végeredmény valóban egyedi módon azonosítsa az eszközt. A most létrehozott új szabály mostantól minden alkalommal lefut, valahányszor csatlakoztatjuk a kérdé-

ses pendrive-ot. Szintén érdemes mindehhez hozzátenni, hogy egy eszközhöz több szabály is rendelhető. Ilyenkor az *udev* azokat a fájlban felülről lefelé haladva hajtja végre.

Az adatmentő szkript megírása

Esetünkben az egész automatikus rendszer lelke a ***backup.thumb.sh*** szkript lesz, ami a tényleges adatmentést elvégzi. Az imént létrehozott szabály tulajdonképpen semmi egyebet nem tesz, csak meghívja ezt a programot, miközben átadja neki annak az eszközfájlnak a nevét (esetünkben az imént említett szimbolikus láncot), amelyen keresztül a kérdéses hardverelem elérhető.
Minden más beállítás a `CONFIG` szakaszban található. A mentőszkript kódját az *1. Lista* tartalmazza.
Mentsük ezt a szkriptet */usr/local/bin/backup-thumb.sh* néven, és persze ne felejtsük el rá kiadni a `chmod +x` parancsot. A következő lépésben saját igényeinknek megfelelően szerkesszük át a beállításokat. A beállítható paraméterek a következők:

`BACKUP_DIR` : Hova akarjuk menteni a pendrive tartalmát.
`GENERATION` : Hány napra visszamenőleg szeretnénk megőrizni a lementett változatokat. A mentések 0-tól kezdődően számozódnak (0 a mindenkori legfrissebb változat) egészen a megadott határig. Persze ne feledkezzünk meg arról sem, hogy ha ezt az értéket magasra állítjuk, akkor rendelkeznünk kell az ehhez szükséges mennyiségű szabad kapacitással a háttértáron. Ha egy közönséges 1 GB-os USB eszközt használunk, a `GENERATIONS` értékét pedig 7-re állítottuk, akkor 7 GB helyre lesz szükségünk a merevlemezen.
`BACKUP_ONCE_DAY` : Ha naponta többször is csatlakoztatjuk és leválasztjuk a pendrive-unkat, akkor bizonyára nem szeretnénk, hogy minden egyes alkalommal mentés készüljön róla, megelégszünk egyetlen változattal is.
A ***backup-thumb.sh*** szkript egy jelzőfájlt használ, amivel megoldható, hogy ilyenkor naponta csak egyszer fusson le a kód. Ha meg szeretnénk változtatni ezt a viselkedésformát, vagyis kifejezetten azt akarjuk, hogy minden egyes csatlakoztatáskor készüljön másolat az adatokról, akkor sincs más dolgunk, csak nullára állítani ennek a változónak az értékét.
`SOUND` : Ebben a példában egy a ***KDE*** rendszerrel érkező hangeffektust használtam a mentés végének jelzésére, de természetesen bármilyen más ***WAV*** fájl megteszi. Ha pedig nem ***WAV***, hanem ***MP3*** kódolású hangállományt szeretnénk lejátszatni, akkor a kódban jelenleg használt `aplay-t` cseréljük le `madplay`-re.

Hogyan működik

A ***backup-thumb.sh*** szkript először is tíz másodpercig vár a rendszer indulásakor, mivel meg kell várnia, amíg a kernel befejezi a pendrive pásztázását. Ha csatlakoztatunk egy ilyen eszközt, majd azonnal kiadjuk a `dmesg` parancsot, akkor a „`waiting fo device to settle`” üzenetet fogjuk látni. Nos, erről van itt szó. Tíz másodperc ugyanakkor még az „idősebb” rendszereken is elegendő kell legyen arra, hogy a kernel befejezze ezt a műveletet.
A következő lépésben a ***backup-thumb.sh*** meglehetősen szigorú jogosultsági korlátokat léptet életbe, mégpedig úgy, hogy a mentéseket csak a root legyen képes olvasni. Ellenkező esetben bármelyik kíváncsi embertársunk átmásolhatná a lemezképeket egy másik gépre és ott szabadon turkálhatna bennük.
A szkript lényege tulajdonképpen a `dd` parancs megfelelő paraméterekkel való végrehajtása. A pendrive tartalmának mentése összességében csak ennyiből áll, ráadásul ez a módszer akkor is működik, ha az eszköz tartalma titkosított. Amikor a folyamat befejeződött, hangjelzést foguk kapni. Ha gépünk ***USB 2.0***-ás portokkal rendelkezik, akkor 1 GB adat lementése körülbelül egy percig fog tartani.

Adatvisszaállítás

Ha bekövetkezett a baj, vagyis elvesztettük vagy kimostuk jó öreg pendrive-unkat, akkor nincs más dolgunk, mint a legutóbbi lementett képet a `dd` paranccsal visszaírni egy másik eszközre, valahogy így:

```
dd if=corsair_drive.backup.0
➥of=/dev/corsair_drive
```

Ha pedig csak egyes fájlokat szeretnénk visszaállítani a mentés alapján, a következőt kell tennünk:

```
mkdir /mnt/thumb
mount -o loop corsair_drive.
➥backup.0 /mnt/thumb
```

Ezek után a */mnt/thumb* könyvtárban található bármilyen fájlról másolatot készíthetünk.
Ha a pendrive tartalmát a ***TrueCrypt*** segítségével titkosítottuk, akkor a lemezkép becsatolását a következőképpen kell végeznünk:

```
truecrypt corsair_drive.
➥backup.0 /mnt/thumb/
```

Látható tehát, hogy a visszaállítás gyakorlatilag ugyanolyan egyszerű, mint a rendszeres mentések elkészítése.
Aki pedig még arra is lusta, hogy odaballagjon az asztali gépéhez, bedugja a pendrive-ot és megvárja, amíg csippant a gép, hogy kész a mentés, nos az ... maradjon távol a mosógépektől.
Ez a cikk tulajdonképpen csak a felületet karcolta. Az ***udev*** szabályok segítségével az itt bemutatottaknál sokkal több, és sokkal bonyolultabb dolgot is meg lehet oldani. A menet közben csatlakoztatható (hot plug) eszközökhöz tartozó bármilyen eseményhez rendelhetünk udev szabályokat, amelyekkel gyakorlatilag bármilyen művelet elvégzését előírhatjuk. Automatizálhatjuk például az eszközön található fájlrendszer becsatolását, digitális fényképezőgépről a képek átmásolását, vagy egy hálózati kapcsolat fölállítását. Az ***udev***-szabályok megírására használatos nyelv ráadásul igen hajlékony. Használhatunk többek között a ***printf*** vezérléséhez használatosakhoz hasonló helyettesítőkaraktereket, illetve a kezelt objektumokhoz a legkülönbözőbb jogosultságokat adhatjuk meg.
Az ***udev*** szabályok írásának rejtelmeiről talán a legjobb összefoglalót ***Daniel Drake*** készítette el. Ez az anyag ***„Writing udev Rules”*** címen a ➲ ***www.reactivated.net/writing_udev_rules.html*** címen érhető el.

Linux Journal 2006., 154. szám

Andrew Fabbro ugyan Oracle DBA fokozattal rendelkezik, de azért otthon még nem mindig ő viseli a nadrágot. Aki íri szeretne neki a témával kapcsolatban, az az andrew@fabbro.org címen érheti el.