>> Device File System Tutorial

[Bitte Kapitel auswählen]

1. Was ist das devfs?

1.1 Die (guten?) alten Zeiten

Traditionelle Linux Distributionen bieten ihren Benutzern einen abstrakten Pfad zu den Geräten. Das ist /dev. In diesem Verzeichniss findet der Benutzer sogenannte **Device Nodes**. Das sind spezielle Dateien, die das entsprechende Gerät repräsentieren. Zum Beispiel repräsentiert /dev/hda das erste IDE Gerät im System. Durch das Bereitstellen von Gerätedateien kann der Benutzer Programme erstellen, die mit der Hardware wie mit einer regulären Datei interagieren, anstatt spezielle APIs zu benutzen.

Die Gerätedateien sind in 2 Gruppen unterteilt. Die erste Gruppe, **Character** Devices genannt, besteht aus Hardware, deren Schreib- und Lesevorgänge nicht gepuffert sind. Die zweite Gruppe nennt sich **Block** Devices und enthält natürlich die Hardware, deren Schreib- und Lesevorgänge gepuffert sind. Beide Gerätearten können zeichenweise oder in Blöcken ausgelesen werden. Die Namensgebung kann deswegen verwirrend klingen und ist in der Tat falsch.

Wenn Sie einen Blick auf eine Gerätedatei werfen, werden Sie etwas ähnliches wie das hier finden:

Befehlsauflistung 1: Die Informationen über eine Gerätedatei abfragen

```
# ls -l /dev/hda
brw-rw----    1 root     disk       3,   0 Jul  5  2000 /dev/hda
```

In dem vorangehenden Beispiel sehen wir, dass /dev/hda ein Block Device ist. Wichtiger sind jedoch die Zahlen, die ihm zugewiesen sind: 3, 0. Dieses Paar wird **Major-Minor** Paar genannt. Es wird vom Kernel dazu verwendet eine Gerätedatei dem echten Gerät zuzuweisen. Die erste Zahl verweißt auf ein bestimtes Gerät, die Zweite verweißt auf ein dem Ersten untergeordnetes Gerät. Das ist verwirrend? Ist es nicht.

Hier zwei Beispiele: /dev/hda4 und /dev/tty5. Die erste Gerätedatei verweißt auf die vierte Partition des ersten IDE Geräts. Ihr Major-Minor Paar ist 3, 4. Mit anderen Worten, die erste Zahl verweißt auf das Gerät und die zweite auf die entsprechende Partition. Die zweite Gerätedatei hat 4, 5 als Major-Minor Paar. In diesem Fall verweist die erste Zahl auf den Terminal Treiber und die Zweite auf die Terminal Nummer (hier das fünfte Terminal).

1.2 Probleme

Wenn Sie in solch einem /dev Verzeichniss einen schnellen Check machen werden Sie bemerken, dass nicht nur alle ihre Geräte, sondern **alle** möglichen denkbaren Geräte aufgeführt sind. Mit anderen Worten, Sie haben Gerätedateien für Geräte die Sie nicht haben. Solch eine Gruppe von Gerätedateien zu pflegen ist, gelinde gesagt, lästig. Stellen Sie sich vor, Sie müssen die Berechtigungen aller Gerätedateien ändern, welche auf ein Gerät in ihrem System verweisen, aber den Rest der Gerätedateien lassen wie er ist.

Wenn Sie neue Hardware in ihr System integrieren und diese vorher noch keine Gerätedatei hatte, werden Sie eine erstellen müssen. Erfahrene Benutzer wissen, dass diese Aufgabe mit *./MAKEDEV* im /dev Verzeichniss erledigt werden kann. Aber wissen Sie sofort welche Gerätedatei Sie erstellen müssen?

Wenn Sie Programme laufen haben die Hardware benutzen die Gerätedateien verwendet, können Sie ihre Root Partition nicht ohne Schreibzugriff mounten. Selbst wenn es keinen anderen Grund gibt sie mit Schreibzugriff gemountet zu haben. Sie können /dev nicht auf einer seperaten Partition haben, weil mount /dev benötigt, um Partitionen zu mounten.

1.3 Lösungen

Wie Sie sich sicher vorstellen können, haben die Kernel Hacker mehrere Wege gefunden die vorher genannten Probleme zu lösen. Jedoch hatten viele von ihnen andere Lösungen, die in diesem Dokument beschrieben sind. Wir werden diese Implementierungen nicht beachten, sondern uns auf die Eine konzentrieren, die es in die offiziellen Kernel Sourcen geschafft hat: devfs.

1.4 devfs als Sieger in allen Klassen

devfs bewältigt alle angeführten Probleme. Es stellt dem Benutzer nur vorhandene Geräte zur Verfügung, erstellt neue Gerätedateien wenn neue Geräte gefunden werden und ermöglicht es, die Root Partition ohne Schreibzugriff zu mounten. Zusätzlich beseitigt devfs noch mehr Probleme, die vorher nicht berücksichtigt wurden, weil sie für den Benutzer weniger interessant sind...

Zum Beispiel müssen Sie sich mit devfs keine Gedanken über Major-Minor Paare machen. Major-Minor Paare werden noch unterstützt (der Abwärtskompatibilität halber), aber sie werden nicht mehr gebraucht. Dies ermöglicht es Linux noch mehr Geräte zu unterstützen, seitdem es keine Grenzen mehr gibt (Zahlen haben immer Grenzen :)).

2. Navigieren in der /dev/ Verzeichnisstruktur

2.1 Verzeichnisse

Als erstes werden Sie bemerken, dass devfs Verzeichnisse benutzt um die Geräte zu gruppieren. Diese Vorgehensweise verbessert die Lesbarkeit, weil sich alle verwandten Geräte in einem Unterverzeichniss befinden.

Zum Beispiel befinden sich alle IDE Geräte im Unterverzeichniss /dev/ide/ und alle SCSI Geräte in /dev/scsi/. SCSI und IDE Festplatten werden gleich behandelt, d. h. sie haben die selbe Verzeichnisstruktur im jeweiligen Unterverzeichnis.

IDE und SCSI sind an einem IDE/SCSI Controller (Onboard oder eine extra PCI Karte) angeschlossen, den man **host** nennt. Jeder Controller kann mehrere Kanäle haben, die **Bus** gennant werden. In jedem Kanal können Sie mehrere IDs haben. Die ID, welche **Target** gennant wird, kennzeichnet eine Festplatte. Manche SCSI Geräte haben mehrere luns (**Logical Unit Numbers**). Zum Beispiel Geräte wie Hi-End Streamer, die mehrere Bänder auf einmal verwalten. Meistens gibt es aber nur eine lun, nämlich lun0/.

Wo wir vorher /dev/hda4 benutzt haben, haben wir nun /dev/ide/host0/bus0/target0/lun0/part4. Das ist viel einfacher... Nein, streiten Sie nicht mit mir... es **ist** einfacher... ach, wie auch immer! :)

Notiz
Sie können für Festplatten auch Unix ähnliche Namen für die Gerätedateien verwenden, wie zum Beispiel c0b0t0u0p2. Sie finden sich in /dev/ide/hd, /dev/scsi/hd, usw.

Um Ihnen eine Vorstellung zu verschaffen, hier die Liste der Verzeichnisse in /dev/ auf meinem Laptop:

Befehlsauflistung 2: Verzeichnisse in /dev

```
cdroms/     cpu/        discs/          floppy/
ide/        input/      loop/           misc/
netlink/    printers/   pts/            pty/
scsi/       sg/         shm/            sound/
sr/         usb/        vc/             vcc/
```

2.2 Abwärtskompatibilität bei der Verwendung von devfs

Es hört sich spassig an, mit diesem Schema zu arbeiten, aber einige Programme nutzen das alte Schema. Um sicherzustellen, das alles im System intakt bleibt, wurde *devfsd* geschrieben. Dieser Daemon erstellt Symlinks mit den alten Namen, die auf die neuen Gerätedateien zeigen.

Befehlsauflistung 3: Erstellte Symlinks

```
$ ls -l /dev/hda4
lr-xr-xr-x    1 root     root           33 Aug 25 12:08 /dev/hda4 -> ide/host0/bus0/target0/lun0
```

Mit *devfsd* können Sie auch Berechtigungen setzen, neue Gerätedateien erstellen, Aktionen definieren, usw. Das alles wird im nächsten Kapitel erklärt.

3. Administrieren von /dev/

3.1 devfsd neustarten

Wenn Sie etwas in /etc/devfsd.conf ändern, müssen Sie nicht neustarten um die Änderungen zu übernehmen. Je nachdem was Sie wollen, können Sie eines der folgenden Signale nutzen:

SIGHUP bringt den *devfsd* dazu, die Konfigurationsdatei neu zu lesen, die mehrfach genutzten Objektdateien neu zu laden und die Aktionen zum Registrieren der Gerätedateien auszuführen.

SIGUSR1 bewirkt das Gleiche, aber es werden keine Registrierungen vorgenommen.

Um solch ein Signal zu senden, benutzen Sie *kill* oder *killall*:

Befehlsauflistung 4: Das SIGHUP Signal an den devfsd senden

```
# kill -s SIGHUP `pid von devfsd`
oder
# killall -s SIGHUP devfsd
```

3.2 Symlinks für die Abwärtskompatibilität entfernen

Warnung
Zur Zeit ist Gentoo ohne diese Symlinks nicht funktionsfähig.

Wenn Sie die Symlinks, die /dev/ vollstopfen, aus ihrem Gentoo System entfernen wollen (sie sind standardmäßig aktiviert), editieren Sie /etc/devfsd.conf und entfernen Sie die folgenden zwei Zeilen:

Befehlsauflistung 5: Einträge zur Abwärtskompatibilität in /etc/devfsd.conf

```
# Kommentieren Sie die folgenden zwei Zeilen aus um die Symlinks zu entfernen
REGISTER        .*  MKOLDCOMPAT
UNREGISTER      .*  RMOLDCOMPAT
```

Um diese Änderungen zu übernehmen, müssen Sie ihr System neustarten.

3.3 Deaktivieren der Autoload Funktionalität

Wenn Sie ein Modul laden erstellt devfs automatisch die zugehörigen Gerätedateien. Wenn Sie das nicht wollen, entfernen sie die folgende Zeile aus ihrer /etc/devfsd.conf:

Befehlsauflistung 6: /etc/devfsd.conf, Autoload Funktionalität

```
LOOKUP          .*  MODLOAD
```

4. Berechtigungen

4.1 Setzen/Ändern von Berechtigungen mit PAM

Obwohl sie Berechtigungen in /etc/devfsd.conf setzen können, raten wir Ihnen PAM (**Pluggable Authentification Modules**) zu nutzen. Und zwar weil PAM das letzte Wort hat, was Berechtigungen betrifft und möglicherweise ihre Änderungen in /etc/devfsd.conf ignoriert.

Die Berichtigungsinformationen für PAM stehen in /etc/security/console.perms. Die Datei besteht aus 2 Abschnitten: Im ersten Abschnitt werden die Gruppen, und im Zweiten die Berechtigungen festgelegt.

Zuerst werfen wir einen Blick auf den Abschnitt mit den Gruppen. Hier als Beispiel die Sound Gruppe:

Befehlsauflistung 7: Die Sound Gruppe in /etc/security/console.perms

```
<sound>=/dev/dsp* /dev/audio* /dev/midi* \
    /dev/mixer* /dev/sequencer* \
    /dev/sound/* /dev/snd/* /dev/beep \
    /dev/admm* \
    /dev/adsp* /dev/aload* /dev/amidi* /dev/dmfm* \
    /dev/dmmidi* /dev/sndstat
```

Die Syntax ist recht einfach: Sie beginnt mit dem Namen der Gruppe und führt dahinter alle Geräte auf, die zur betreffenden Gruppe gehören.

Nun, Gruppen sind nicht sehr nützlich wenn man nichts mit ihnen machen kann. Der Nächste Abschnitt beschreibt, wie Berichtigungen gesetzt werden.

Befehlsauflistung 8: Berechtigungen für die Sound Gruppe in /etc/security/console.perms

```
<console>  0600 <sound>      0600 root.audio
```

Im ersten Feld wird das Terminal überprüft. Bei den meisten Systemen ist das die Console Gruppe. PAM überprüft dieses Feld bei jedem Login. Falls der Login auf einem Gerät erfolgt, welches sich in der Console Gruppe befindet, wird PAM die Berechtigungen einiger Gerätedateien überprüfen und möglicherweise ändern.

Das zweite Feld enthält die Berechtigungen, die nach einem erfolgreichen Login für eine Gerätedatei gesetzt werden. Wenn sich ein Benutzer am System einloggt und die Gerätedateien eine(n) Standard Owner/Group haben, ändert PAM die Berechtigungen und gibt dem Benutzer Owner Rechte. Auch werden die Zugriffsrechte entstprechend gesetzt, hier 0600 (Nur der aktuell angemeldete Benutzer hat Lese/Schreibzugriff).

Im dritten Feld steht die Gerätegruppe, deren Berechtigungen geändert werden. Hier ist das die Sound Gruppe (Alle Gerätedateien die etwas mit Sound zu tun haben).

Im vierten Feld stehen die Berechtigungen, welche gesetzt werden, wenn die Gerätedatei wieder in den Standardzustand zurückversetzt wird. Mit andern Worten, wenn sich der Owner der Gerätedateien ausloggt, setzt PAM die Berechtigungen auf den im vierten Feld beschriebenen Stand zurück.

Im fünften Feld steht der Owner (mit Gruppe wenn nötig) der Gerätedatei für den Standardzustand. Mit anderen Worten, wenn sich der Owner der Gerätedateien ausloggt, werden die Owner Rechte auf den im fünften Feld beschriebenen Stand zurückgesetzt.

4.2 Setzen/Ändern von Berechtigungen mit devfsd

Wenn Sie die Berechtigungen wirklich mittels /etc/devfsd.conf setzen wollen, nutzen Sie bitte die Syntax aus dem folgendem Beispiel:

Befehlsauflistung 9: Berechtigungen in /etc/devfsd.conf

```
REGISTER    ^cdroms/.*  PERMISSIONS root.cdrom 0660
```

Im zweiten Feld wird die Gerätegruppe angegeben, beginnend in /dev. Hier werden reguläre Ausdrücke verwendet, Sie können also mehrere Gerätedateien auf einmal auswählen.

Im vierten Feld wird der Owner (mit Gruppe) angegeben. Im Gegensatz zu PAM wird das nicht automatisch geändert (bis es in console.perms angegeben wird), weil PAM immer gewinnt.

Im fünften Feld stehen die Berechtigungen für die Gerätedatei.

4.3 Berechtigungen von Hand setzen und mit devfsd speichern

Hier die normale Vorgehensweise bei Gentoo Installationen: Wenn sie mit *chown* (CHangeOWNer) und *chmod* (CHange MODe) einige Gerätedateien verändern, speichert *devfsd* die Informationen wenn Sie das System herunterfahren. Das passiert weil /etc/devfsd.conf folgendes enthält:

Befehlsauflistung 10: /etc/devfsd.conf zum Speichern von Berechtigungen

```
REGISTER        ^pt[sy]/.*   IGNORE
CHANGE          ^pt[sy]/.*   IGNORE
CREATE          ^pt[sy]/.*   IGNORE
DELETE          ^pt[sy]      IGNORE
REGISTER        ^log         IGNORE
CHANGE          ^log         IGNORE
CREATE          ^log         IGNORE
DELETE          ^log         IGNORE
REGISTER        .*           COPY    /lib/dev-state/$devname $devpath
CHANGE          .*           COPY    $devpath /lib/dev-state/$devname
CREATE          .*           COPY    $devpath /lib/dev-state/$devname
DELETE          .*           CFUNCTION GLOBAL unlink
/lib/dev-state/$devname
RESTORE         /lib/dev-state
```

Genauer erklärt, geänderte Gerätedateien werden beim Herunterfahren nach /lib/dev-state kopiert und beim Hochfahren wieder nach /dev zurückkopiert.

Eine andere Möglichkeit besteht darin während des Bootvorgangs /lib/dev-state nach /dev zu mounten. Allerdings müssen Sie sicherstellen, dass devfs nicht automatisch beim Starten gemountet wird (Sie müssen dazu ihren Kernel neu kompilieren) und das /dev/console existiert. Dann platzieren Sie irgendwo am Anfang ihrer Startscripte folgendes:

Befehlsauflistung 11: Mounten von /lib/dev-state nach /dev

```
mount --bind /dev /lib/dev-state
mount -t devfs none /dev
devfsd /dev
```

5. Quellen

Für mehr Informationen über devfs, hier noch ein paar Quellen:

Die devfsd.conf Manpage erläutert die Syntax von /etc/devfsd.conf. Um sie anzusehen führen Sie *man devfsd.conf* aus.

Die devfs FAQ erklärt alles rund um devfs. Sie enthält auch Informationen über die interne devfs Struktur und wie Treiber devfs unterstützen können.

Im LinuxJournal ist ein informativer Artikel über devfs zum Managen und Administrieren.

Daniel Robbins schrieb einige Artikel für IBMs DeveloperWorks über erweiterte Dateisysteme. Drei behandeln devfs:

- Introduction to devfs
- Setting up devfs
- Implementing devfs