Was kann Grub?
Grub für Einsteiger

zusammengestellt von Thorsten van Lil

Das Geniale an Linux ist, dass die einzige Grenze durch das Wissen beziehungsweise Unwissen gezogen wird und es reizt mich immer, diese Grenze Stück für Stück nach hinten zu schieben. Ganz nach dem Motto „man wächst mit der Aufgabe“.

Vor einiger Zeit hatte ich Probleme mit GRUB. Ich versuchte, Mandriva, openSuSE und Windows parallel laufen zu lassen. Eigentlich keine große Sache, doch durch ständiges Erstellen, Verändern und Löschen von Partitionen geriet meinen Partitionstabelle so durcheinander, dass es nicht mehr möglich war, von den hinteren Partitionen zu booten. Von daher mag meine Einstieg nicht sonderlich gut gewählt sein, doch ich hab auch meinen Nutzen aus der Sache gezogen. Ich war gezwungen, mich mit GRUB auseinanderzusetzen und habe dabei viel gelernt.

In der letzten Ausgabe von MagDriva hat uns Karsten (aka Tuxdriver) einen kleinen Einstieg in GRUB ermöglicht. Ich gehe also davon aus, dass jeder den Artikel gelesen hat und will nicht weiter darauf eingehen.

GRUB ist der Standard-Bootloader der meisten aktuellen Linux-Distributionen, so auch bei Mandriva. Der Bootloader lädt in erster Linie den Kernel und startet somit den Bootvorgang. GRUB ist in 2 (eigentlich 3) Teile eingeteilt. Stage 1 ist der Teil von GRUB, der in den MBR geschrieben wird. Der MBR ist auf 512 Bytes beschränkt, was für GRUB nicht ausreichend ist. Aus diesem Grund lädt Stage 1 den Rest von GRUB, die Stage 2.

Karsten hat uns erzählt, dass GRUB die Datei /boot/grub/menu.lst ausliest, um die nötigen Einträge für einen Start zu bekommen. Schauen wir uns doch mal einen solchen Eintrag an und analysieren ihn:

```
title desktop 2.6.22.12-1
kernel (hd0,6)/boot/vmlinuz-2.6.22.12-desktop-1mdv root=/dev/sda7 splash=silent vga=791
initrd (hd0,6)/boot/initrd-2.6.22.12-desktop-1mdv.img
```

Dieser Eintrag wurde bei einem Kernel-Update automatisch erstellt. Jeder Eintrag beginnt mit der Zeile für den Titel, auf die ich nicht weiter eingehen muss.

Mit *kernel* beginnt die Zeile, in der wir GRUB sagen, wo der zu bootende Kernel liegt. Bei mir liegt also der Kernel *vmlinuz-2.6.22.12-desktop-1mdv* auf *sda7* (meine Systempartition) im Verzeichnis */boot/*.

Hier sei nochmal darauf hingewiesen, dass GRUB bei 0 anfängt zu zählen, deswegen ist (hd0,6) eben sda7. Hinter den Kernel können wir nun Bootparameter setzen, z.B. *acpi=off, noapic*, etc. In meinem Eintrag seht ihr noch *root=/dev/sda7*, damit zeige ich dem Kernel (den ich booten möchte) wo das Wurzelverzeichnis des zu startenden Linux liegt. *Splash=silent* sagt aus, dass der schöne Splash-Screen während des Bootvorgangs zu sehen ist, VGA=791 gibt die Auflösung an. Welcher VGA-Code für welche Auflösung steht, könnt ihr hier nachsehen:

http://kanotix.com/index.php?module=pnWikka&tag=GrubVgaCodes

In der zweiten Zeile wird das Initrd-Image angegeben. Dort sind im Grunde die nötigen Treiber für den Bootvorgang enthalten. Unter Umständen ist es aber so, dass wir eine eigene *Initrd* anlegen müssen, weil wir besondere Treibermodule benötigen. In diesem Fall können wir mit *mkinitrd* ein eigenes Image anlegen. Darauf näher einzugehen würde den Rahmen jedoch deutlich sprengen, von daher sei auf die man-Page verwiesen, '*man mkinitrd*'.

Kernel und Initrd-Image liegen in aller Regel im gleichen Ordner (in */boot/),* so auch bei mir. In den meisten Fällen findet ihr einen Link von *vmlinuz*. (bzw. initrd.img) auf den aktuellsten Kernel (bzw. Initrd-Image). Das hat zum einen den Vorteil, dass der Standardeintrag quasi immer aktuell bleibt und zudem lässt sich damit schnell ein funktionsfähiger Kernel starten, ohne zu wissen, welche Kernel überhaupt installiert sind.

Um zu prüfen, welcher Kernel mit vmlinuz verknüpft ist, müsst ihr in das Verzeichnis /boot wechseln und dort *ls -l* eintippen. Ihr seht nun eine detailierte Darstellung aller Unterverzeichnisse und Dateien.

Die Zeile für vmlinuz zum Beispiel, sieht bei mir wie folgt aus:

```
lrwxrwxrwx 1 root root 29 2007-12-08 21:30 vmlinuz -> vmlinuz-2.6.22.12-laptop-1mdv
```

Das erste Buchstaben Wirrwarr gibt die Dateirechte an (für mehr Informationen siehe '*man chmod*'). Danach steht der Besitzer der Datei, also der Benutzer root und die Gruppe root. Am Ende steht das eigentlich Interessante: Es existiert ein Link auf *vmlinuz,* der auf den Kernel *2.6.22.12-laptop-1mdv* zeigt. Das Gleiche sieht man auch bei dem Initrd-Image.

Stellen wir uns also vor, ich habe auf einer Partition (*/dev/sda5*) ein weiteres Linux (zum Beispiel SAM-Linux). Dafür möchte ich nun einen Bootеintrag in meinen GRUB einfügen.

Ich gucke also nach, wie die Dateien (*kernel* und *initrd*-Image) des zweiten Linux heißen, bzw., ob es einen gültigen Link gibt, z.B. *vmlinuz* und *initrd.img*. Dann sähe der Eintrag wie folgt aus:

```
title SAM-Linux
kernel (hd0,4)/boot/vmlinuz root=/dev/sda5
initrd (hd0,4)/boot/initrd.img
```

Alternativ kann man in einer ersten Zeile die Partition mit den Boot-Dateien angeben, dann muss man bei *kernel* und *initrd* nicht mehr die Partition, sondern nur noch das Verzeichnis nennen.

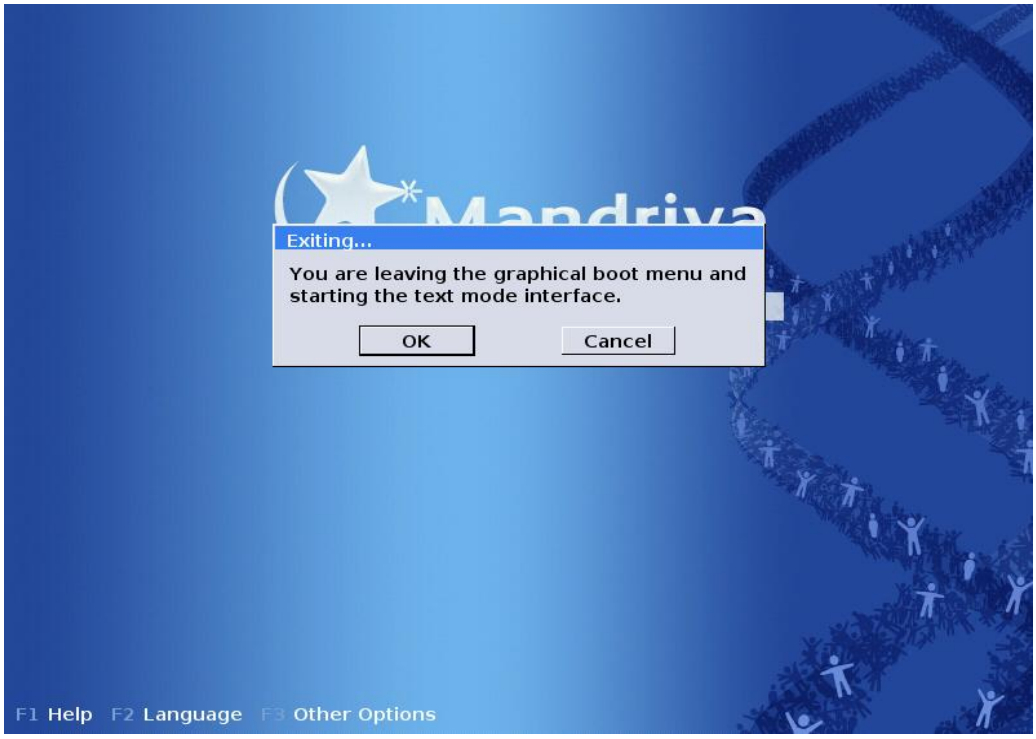
```
title SAM-Linux
root (hd0,4)kernel /boot/vmlinuz root=/dev/sda5
initrd /boot/initrd.img
```

```
grub> find /boot/grub/stage1
(hd0,4)
(hd0,6)
(hd0,7)
```

Auf den Partitionen sda5, sda7 und sda8 liegt also ein GRUB und damit wohl auch ein Kernel. Mit der Autovervollständigung kann ich jetzt vorhandene Kernel finden.

Mit '*kernel (hd0,4)/boot/vmlinuz*' und der TAB-Taste sehe ich alle bootbaren Kernel des Systems.

```
grub> kernel (hd0,4)/boot/vmlinuz
 Possible files are: vmlinuz-laptop vmlinuz-2.6.22.9-
laptop-1mdv vmlinuz vmlinuz-2.6.22.12-
laptop-1mdv
```



Hin und wieder mag es jedoch vorkommen, dass die vorhanden Booteinträge nicht mehr funktionieren (weil der Kernel kaputt oder nicht mehr vorhanden ist). Man kommt also nur noch in GRUB und muss von da aus sehen, wie es weiter geht. Aber nicht immer weiß man, wo sich *kernel* und *initrd* befinden bzw. wie sie genau heißen.

Auch hier lässt uns GRUB nicht im Stich. Wir beenden einfach die grafische Oberfläche von GRUB mit ESC und wechseln dann mit 'c' in die GRUB-Konsole, in der uns einige Kommandos zur Verfügung stehen.

Mit *find* kann man nach einzelnen Dateien suchen. Das ist hilfreich, um überhaupt erst einmal festzustellen wo GRUB installiert ist. Mit

```
find /boot/grub/stage1
```

zeigt mir GRUB alle Partitionen, in der sich ein Ordner */boot*/grub/ mit der Datei stage1 finden lässt. Bei mir sieht das so aus:

Um ein Linux aus der GRUB-Konsole zu starten, muss ich die Zeilen eingeben, die normalerweise in der /boot/grub/menu.lst stünden und mit boot starten:

```
grub> kernel (hd0,4)/boot/vmlinuz root=/dev/sda5
grub> initrd (hd0,4)/boot/initrd.img
grub>boot
```

In den letzten Tagen habe ich mir eine USB-Festplatte gekauft, um endlich auch mal Platz zu haben, andere Distributionen zu testen. Da ich aber von Grund auf faul bin, wollte ich die neuen ISO-Dateien nicht auf CD oder DVD brennen um sie dann zu installieren. Im Internet bin ich auf einen Weg gestoßen wie man es schafft mittels GRUB von einer ISO-Datei zu booten. Es ist nicht unbedingt elegant aber es funktioniert.

Nachdem wir uns nun also ein ISO-Image runtergeladen haben, z.B. *Archlinux-i686-2007.08.1-Don't-Panic.current.iso* können wir es unter Linux mounten, ähnlich wie eine CD.

```
mount -o loop Archlinux-i686-2007.08.1-Don\'t-Panic.current.iso /mnt/
```

Jetzt haben wir also das ArchLinux-Image in das Verzeichnis /mnt eingebunden. Danach müssen wir gucken, wo in dem Image sich *kernel* und *initrd* befinden, meisten ist das auch im */boot/*-Verzeichnis (also */mnt/boot/*), in meinem Beispiel ist es allerdings */mnt/isolinux*. Das komplette Verzeichnis kopieren wir nun auf eine beliebige leere Partition.

```
mount /dev/sda8 /media/sda8
cp -R /mnt/isolinux /media/sda8/
```

Anschließend kopieren wir das ganze Image ebenfalls auf die Partition.

```
cp Archlinux-i686-2007.08.1-Don\'t-Panic.current.iso /media/sda8
```

Was jetzt noch fehlt, ist ein passender Eintrag in der */boot/grub/menu.lst,* um das Iso zu starten. Das könnte dann so aussehen:

```
title ISO-Boot
kernel (hd0,7)/isolinux/vmlinuz fromiso=/*.iso
initrd (hd0,7)/isolinux/initrd.img
```

Liegt das Image auf einer USB-Platte bzw. Stick hilft eventuell der Kernelparameter *bootusb2* ergänzen.

Wenn ihr ein Image erwischen solltet, das im Wurzelverzeichnis des Images außer Verzeichnissen auch noch Dateien enthält, solltet ihr die Dateien ebenfalls auf die Partition mit dem Image kopieren. Wenn ihr nun beim Booten den neuen Eintrag auswählt, bootet er von dem Image wie von einer CD beziehungsweise DVD.

Nun wissen wir, wie sich ein Boot-Eintrag zusammensetzt, wie man GRUB bedient und wissen sogar wie man ein Iso-Image booten kann. Aber wie schreibt man GRUB überhaupt in den MBR und woher weiß es, wo die *menu.lst* sitzt? Der einfache Weg ist natürlich über das MCC, aber den muss ich hier nicht weiter erklären.

Wir können unter Linux die GRUB-Konsole starten, die wir weiter oben schon kennen lernen durften. Hier können wir zum Einen festlegen, in welchen MBR GRUB geschrieben werden soll (meist der MBR der ersten Platte) und wo es die *menu.lst* und alles Weitere zum Booten findet, also welche die root-Partition ist.

```
[root@localhost /]# grub
```

Dadurch starten wir GRUB und bekommen die GRUB-Konsole zu sehen. Jetzt müssen wir die root-Partition (bzw. Boot-Partition) nennen, also die Partition, auf der der Kernel und die *menu.lst* liegt:

```
grub> root (hd0,6)
```

und danach installieren wir GRUB in den MBR der ersten Platte

```
grub> setup (hd0)
```

Zum Schluß wird die GRUB-Konsole geschlossen:

```
grub> quit
```

Die Möglichkeiten von GRUB sind schier unendlich und das ist auch gut so, denn gerade Windows macht bei einem Dual-Boot-System gerne mal Probleme. Windows will am liebsten auf der ersten Partition der ersten Platte liegen und meckert eventuell, wenn es nicht so ist.

GRUB kann Windows aber in der Hinsicht ein bischen anschwindeln, nach dem Motto „was er nicht weiß, macht ihn nicht heiß“.

Mit der Option *map* lassen sich Platten vertauschen.

```
map (hd0) (hd1)
map (hd1) (hd0)
```

Jetzt gaukelt GRUB vor, das sda jetzt sdb ist und umgekehrt. Zusätzlich kann man einzelne Partitionen verstecken (hide).

```
hide (hd1,0)
```

versteckt die erste Partition der 2. Platte. Windows (oder was auch immer gebootet werden soll) kann die Partition nicht sehen. Die zweite Partition (hd1,1) ist dann also die „neue“ erste Partition.

Und was, wenn....

Allerdings kann ich euch mit eurem neuen Wissen noch nicht auf GRUB loslassen, denn es klappt leider nicht immer alles wie erhofft. Hat GRUB Probleme beim Booten, dann erscheint in der Regel eine Error-Nummer und eine stichwortartige Fehlerbeschreibung. Oftmals reichen die Meldungen aber nicht aus und für den Fall ist es gut zu wissen, was die Error-Nummer genauer aussagt.

Hier hilft folgender Link, wo alle Meldungen mit genauer Beschreibung zu finden sind:

http://www.gnu.org/software/grub/manual/html_node/Stage2-errors.html#Stage2-errors

Außerdem findet ihr wesentlich mehr Infos in den man-Pages (*man grub*) oder eben im Internet, wie hier zum Beispiel (auf Englisch):

http://www.gnu.org/software/grub/manual/grub.html

denn das hier ist ...

...auch nur ein kurzer Einstieg.