Ein neuer Kernel für Slackware 9.1

Ausgangspunkt ist eine Standardinstallation vom Typ current.

Zur Zeit wird mit current ein Kernel in der Version 2.4.26 installiert. Dieser soll in diesem Beispiel mit Version 2.6.5 ergänzt werden.

UPDATE: Seit dem 10.05.2004 ist nun bereits Kernel 2.6.6 stable verfügbar, diese Anleitung sollte auch mit der neuen Kernelversion funktionieren.

Diese Anleitung läßt sich auch auf eine normale Slackware Standardinstallation anwenden, 2.4.26 muß dann in den jeweiligen Kommandos nur durch 2.4.22 ersetzt werden. Als Bootmanager wird hier Lilo verwendet.

Das von www.kernel.org herunter geladene Archiv wird nach /usr/src/ entpackt.

Zur Sicherheit passen wir die Lilo Konfiguration so an, das im Falle eines Fehlers bei der Kernelerstellung immer noch auf den Originalkernel zugegriffen werden kann. Somit kann man dann ins alte System booten und von dort aus auf Fehlersuche gehen:

```
ln -s /boot/vmlinuz-ide-2.4.26 /boot/vmlinuz.failsafe
ln -s /boot/System.map-ide-2.4.26 /boot/System.map.failsafe
```

In der lilo.conf fügen wir einen neuen Abschnitt hinzu:

```
image=/boot/vmlinuz.failsafe
label=Failsafe
read-only
vga=normal
```

Auf der Kommandozeile einmal lilo ausführen, damit die Änderungen auch übernommen werden.
Mit einem Neustart können wir testen, ob man auch wirklich ins failsafe System booten kann.

Geklappt? Gut. Wieder zurück im Arbeitssystem gehen wir nach /usr/src und löschen dort den Softlink linux -> linux-2.4.26.

Nun legen wir einen neuen Softlink an, der auf linux-2.6.5 zeigt:

```
ln -s linux-2.6.5 linux
```

Dann wechseln wir über den neu erstellten Softlink ins Quellverzeichnis des 2.6.5er Kernels:

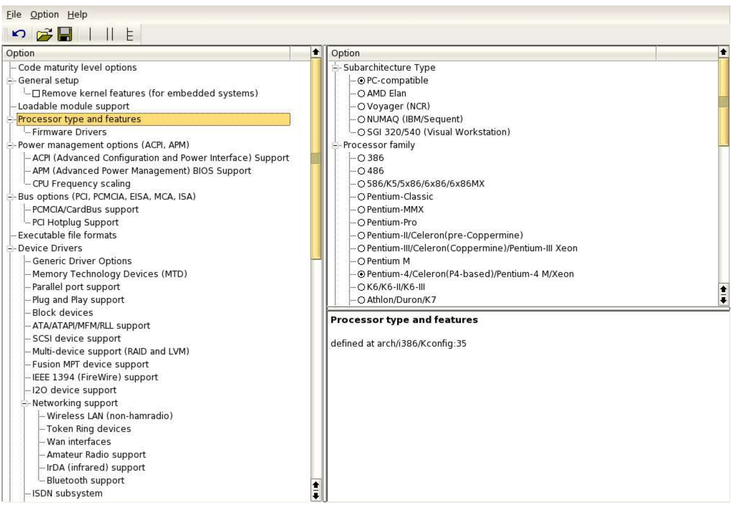
```
cd linux
```

Spätestens jetzt sollte man sich Gedanken darüber machen, was für ein System man überhaupt hat und mit welchen Gerätschaften es ausgestattet ist. Nicht nur, das wir mit falschen Angaben zur Konfiguration unter Umständen keinen lauffähigen neuen Kernel erzeugen können, je genauer die Angaben zur Kernel-konfiguration gemacht werden, umso besser wird der Kernel später an unser System angepasst sein. Unter Slackware helfen uns die Komandos lspci und dmesg. Mit lspci können wir alle im Moment erkannten PCI Geräte in unserem System auflisten, dmesg zeigt uns die Bootmeldungen des Kernels an, welchen man auch entnehmen kann, was so alles im System erkannt wurde. Handbücher zum Mainboard, zur Grafikkarte, zur Sound-karte sind natürlich auch sehr hilfreich :) Auch sollten wir Informationen zu installierten Netzwerkkarten beschaffen, externen USB Geräten usw.

Zur Kernelkonfiguration bedienen wir uns des grafischen Tools xconfig:

```
make xconfig
```

Damit wird nun xconfig kompiliert und gestartet:



Im linken Bereich sind die Hauptkategorien der Kernel-konfiguration aufgelistet, die im oberen rechten Bereich bei Bedarf weiter verzweigen. Unten rechts sehen wir eine kurze Beschreibung der gerade angewählten Option.

Die genauen Einstellungen hängen nun vom jeweiligen System ab, auf welchem wir konfigurieren und unter-scheiden sich hinsichtlich der Prozessorarchitektur, der verbauten Komponenten und der Chipsätze zum Teil erheblich voneinander. Wir ändern hier Einstellungen, die entscheidend sind für das Funktionieren des neuen Kernels und sollten uns deswegen exakt an die für unser System zutreffende Konfiguration halten. Das setzt natürlich eine möglichst genaue Kenntnis unserer Systemkonfiguration voraus. Wenn man sich bei einigen Optionen nicht sicher ist, sollte man auf die Hilfstexte im unteren rechten Bereich achten und gegebenenfalls die dort empfohlenen Hinweise zur Einstellung übernehmen. Wer am Anfang überhaupt nicht durchsieht, der bediene sich bitte der Hilfe einer kundigen Person seines Vetrauens.

Wenn alle Einstellungen erfolgt sind, speichern wir über File->Save ab und verlassen das Programm.

Jetzt kann die Kompilierung gestartet werden mit:

```
make all install modules_install
```

Während der Kompilierung können wir in der Konsole den Vorgang verfolgen und auf einen erfolgreichen Durchlauf hoffen. Die Kompilierung dauert je nach Rechner-leistung einige Augenblicke.

Nach erfolgter Kompilierung verschieben wir den neu erstellten Kernel an seinen Bestimmungsort und kopieren die von xconfig angelegte Kernelkonfigurationsdatei:

```
mv /vmlinuz /boot/vmlinuz-2.6.5 && mv /System.map /boot/System.map-2.6.5
cp /usr/src/linux/.config /boot/config-2.6.5
```

Zur erfolgreichen Einrichtung unseres neuen Kernels erfolgen jetzt noch drei weitere Kommandos, die wir im Verzeichnis /boot ausführen:

```
ln -s vmlinuz-2.6.5 vmlinuz
ln -s System.map-2.6.5 System.map
ln -s config-2.6.5 config
```

Um mit unserem alten Kernel 2.4.26 in Zukunft auch noch booten zu können, machen wir folgendes in /boot:

```
ln -s vmlinuz-ide-2.4.26 vmlinuz.old
ln -s System.map-ide-2.4.26 System.map.old
ln -s config-ide-2.4.26 config.old
```

Damit wir den alten Kernel beim booten auch auswählen können, sind noch kleine Ergänzungen an der lilo.conf notwendig. Zwischen vmlinuz und vmlinuz.failsafe fügen wir noch einen weiteren Abschnitt hinzu:

```
image=/boot/vmlinuz.old
label=Old
read-only
vga=normal
```

Danach nochmal lilo ausführen. Jetzt kann der Rechner neu gestartet und im lilo Bootmanager der neue Kernel zum booten ausgewählt werden.
Der Eintrag 'Linux' startet jetzt den neuen Kernel 2.6.5, 'Old' startet unseren alten Kernel 2.4.26 und mit 'Failsafe' fahren wir ein Rettungssystem mit unserem alten Kernel hoch.

Für Rechner mit Nvidia Grafikkarte muß jetzt noch das Installationsscript für den Linux Nvidia Treiber von der Nvidia Webseite gestartet und gegebenenfalls die Datei /etc/X11/XF86Config angepasst werden.