

Linux für Einsteiger

Teil 3

Fortsetzung: Systeminstallation (Vertiefung)

Linux kann sehr gut mit anderen Systemen koexistieren, da es einfach auf freie (oder freigemachte) Partitionen installierbar ist. Dabei gibt es fast keine Beschränkungen, was die Lage der Partitionen angeht. Das fast keine ist keine Linux Einschränkung, sondern eine des PC BIOS. Für den Startvorgang müssen der Bootloader und der zu ladende Kernel vom BIOS erreichbar sein, hier schlagen die Größenbeschränkungen der Festplatten gnadenlos zu (z.B. 8 GB, 32 GB oder 128 GB bei aktuelleren PC's).

Wir haben hier, um Zeit zu sparen, bereits eine Linux Distribution auf die Rechner installiert, zur Demonstration der Integrationsfähigkeiten von Linux existiert auch noch eine kleine DOS Partition, auf der ein Free-Dos installiert ist (OpenSource, daher wie Linux keinerlei Lizenzprobleme). Zur Installation haben wir eine spezielle Abart der Knoppix-Distribution genommen, genannt Kanotix. Diese ist zwar ebenfalls als Live-CD zu benutzen, jedoch hat der Entwickler großen Wert darauf gelegt, von der CD aus eine funktionierende Festplatteninstallation herzustellen. Das Ergebnis ist eine Debian GNU/Linux Installation des unstable Zweiges der Distribution, ohne die mühsame Einrichtung aller benötigten Module und Treiber per Hand, weil es bisher für diese Distribution noch keinen Installationshelfer gibt, der dies automatisch kann. Kanotix ist eine Symbiose aus der excellenten Hardwareerkennung von Knoppix mit den bei Kanotix vorhandenen Scripten zur Festplatteninstallation, die als Ergebnis ein Linux hat, das mit den Standardmitteln von Debian weiter verwaltet werden kann. Die Paketverwaltung erlaubt es, Programme zu löschen, neue zu installieren und vorhandene upzudaten, ohne die bei Knoppix pur vorhandenen Versionsprobleme.

Wenn ihr später bei euch selber eine Installation durchführt, so werdet ihr, egal bei welcher Distribution, immer auf die gleichen Fragen stoßen, die bei der Installation abgefragt werden

Einstellungen im System

Nachdem unser Linux auf der Festplatte installiert ist und wir nun wissen, wie es über einen Bootmanager gestartet wird, ist es Zeit, den Startvorgang etwas genauer zu betrachten.

Bisher habt ihr euer Linux entweder von CD oder der Festplatte gestartet ohne weiter Einfluss auf das Ergebnis (graphischer Desktop mit seinem Aussehen) und die gestarteten Programme zu nehmen.

Linux ist sehr flexibel und es können viele Einstellungen verändert werden, ich persönlich ziehe die Bearbeitung der Einstelldateien im Textmodus einer graphischen Bedienoberfläche vor, weil das mich von den Vorgaben eines graphischen Einstellprogramms entbindet. Auch das graphische Programm ändert letztlich nur die eigentlichen Textdateien.

Sehen wir uns nacheinander die Dateien an, die im Verlauf des Linux Starts abgearbeitet werden. Die erste Datei ist die Einstellungsdatei des Bootmanagers, sie wurde bereits in der vorigen Unterrichtseinheit behandelt. Nachdem der Kernel in den Arbeitsspeicher geladen wurde, wird der erste Prozess mit Namen `init` gestartet, von ihm werden alle weiteren Prozesse gestartet. Beim späteren Beenden von Linux ist er auch der letzte Prozess, der beendet wird.

`init` beachtet als erstes die Datei `/etc/inittab`. Dort sind einige grundsätzliche Einstellungen vorgesehen, z.B. der sogenannte `default initlevel`. Linux kennt die `initlevel` 0..6. 0 und 6 haben die Sonderbedeutungen, 0=System anhalten, 6=System neu starten. Level 1 ist der Single-User-Modus für Wartungsarbeiten des Administrators, 2..5 sind im Prinzip individuell einrichtbar, es hat sich aber eingebürgert, Level 2 für den Textmodus ohne Graphikoberfläche zu nehmen, Level 5 ist der volle Betrieb mit allen Diensten incl. Graphik. Außerdem kann man der `inittab` Datei auch entnehmen, wie viele Textkonsolen ihr System zur Verfügung stellt und welche Reaktion auf das Drücken von CTRL-ALT-DEL (Computerauffengriff) erfolgen soll.

Der Start in den `default initlevel` erfolgt nun in zwei Schritten, zunächst wird das Verzeichnis `/etc/rcS.d` durch Aufruf von `/etc/init.d/rcS` abgearbeitet. Dort werden alle Dienste und Programme gestartet, die nur beim Booten (=starten des Rechners) benötigt werden. Ich greife hier nur ein Beispiel heraus: Die Unversehrtheit des oder der Dateisysteme wird überprüft und gegebenenfalls wiederhergestellt, bevor wieder damit gearbeitet werden kann. Der nächste Schritt ist der Aufruf von `„/etc/init.d/rc“` mit dem Argument des `defaultlevels` (1..5), was zu der Abarbeitung des entsprechenden Verzeichnisses `/etc/rcX.d` ($X=1..5$) führt. In diesen Verzeichnissen stehen nicht die Programme selber, sondern nur Verweise, sogenannte „symbolische Links“ auf die Dateien unterhalb `/etc/init.d` mit vorangestelltem S für Starten des Dienstes oder K (=kill) für Beenden. Außerdem ist noch eine zweistellige Nummer vorangestellt, die die Reihenfolge der Abarbeitung festlegt. Es werden zunächst die Programme in aufsteigender Reihenfolge gestoppt und dann wird aufsteigend die Startliste abgearbeitet.

Wir haben es hier mit einem der Autostartmechanismen von Linux zu tun. Hier ist die erste Möglichkeit, sich mit eigenen Programmen einzuklinken, jedoch müssen diese über ein Start-Stopp-Script verfügen. An dieser Stelle begnügen wir uns mit dieser Aussage, weil die Erstellung von Scripten (Bash, Perl u.a.) einen eigenen Lehrgang ausfüllen würde, im Vergleich dazu wirken DOS-Batchdateien wie vorsintflutlich.

Einrichtung von Komponenten und Geräten

Im Regelfall werden heutzutage die meisten Komponenten eines Computers direkt beim Start von Linux erkannt und die entsprechenden Module geladen und die Geräte eingerichtet. Was ist zu tun, wenn das nicht der Fall sein sollte ? Bei eingebauten Komponenten (Graphikkarte, Sound, SCSI-Controller, RAID-Controller, Modem, Netzwerk) gibt es den Befehl „lspci“, der die Komponenten auflistet und als was sie erkannt wurden. Fehlermöglichkeiten sind:

1. Die Hardware wird überhaupt nicht erkannt (Herstellercode und Geräteerkennung sind unbekannt)
2. Die Hardware wird erkannt, jedoch ist kein Kernelmodul verfügbar.
3. Die Hardware wird falsch erkannt. Es wird ein falsches Modul geladen.

Welche Module geladen worden sind, kann mit dem Befehl „lsmod“ festgestellt werden. Neue Module werden mit dem Befehl „modprobe (*modulname*)“ geladen.

Woher soll ich wissen, welches Modul geladen werden muss ?

Als erstes ist möglichst viel über die nicht funktionierende Komponente herauszufinden (Kartentyp, Hersteller, genaue Bezeichnung, verwendeter Chipsatz). Für etliche, gerade ältere Komponenten gibt es generische Module, die für eine kompatible Kartengruppe funktionieren (Beispiel: Das Modul „ne“ für „NE2000-kompatible“ Netzwerkkarten, bzw. für einen bestimmten Chipsatz eingesetzt werden können (vielleicht nur eingeschränkt, da herstellerspezifische Erweiterungen nicht berücksichtigt werden können).

Ansonsten ist die Suche im Internet, die Frage an Experten oder auch der Besuch einer LUG hilfreich.

Es muss aber auch klar gesagt werden, dass es Geräte und Karten gibt und auch zukünftig geben wird, für die keine geeigneten Module existieren und damit unter Linux nicht lauffähig sind. Die Schwierigkeit besteht sehr oft darin, die Interessen des Hardwareherstellers in Bezug auf Geheimhaltung bestimmter Eigenschaften ihrer Produkte mit der Offenheit von Linux und der OpenSource Welt in Einklang zu bringen. Zur Zeit wird in der Linux-Kernel-Community darüber diskutiert, ob es eine definierte Kernelschnittstelle geben soll, die es Herstellern erlaubt, für ihre Produkte binäre Module ohne Veröffentlichung des Quelltextes zu erstellen, die auf allen mit dieser Schnittstelle versehenen Kernen lauffähig sind. Bislang müssen binäre Module für jede Kernelversion einzeln angepasst werden, ein Beispiel sind Nvidia Graphikkartentreiber (für jede Kernelversion ein anderes Modul !).

Für USB-Geräte gilt, die Linux Hotplug-Funktionalität ist schon weit fortgeschritten, d.h. die Erkennung für im laufenden Betrieb angestöpselte Geräte führt meist zu deren einwandfreier Funktion. Aber auch hier gilt,

manchmal sind die Geräte noch nicht in der Linux-eigenen Datenbank vorhanden, der Neuling steht vor einer großen Hürde und braucht Hilfe.

Die Einrichtung eines Druckers braucht die Mitwirkung des Benutzers, da es für viele Drucker mehr als einen möglichen Treiber gibt. Das heutige Standarddrucksystem ist CUPS (Common Unix Printing System), welches die Druckereinrichtung wesentlich vereinfacht hat. Die Netzwerkfähigkeiten von CUPS sorgen dafür, dass ein Druckertreiber zu einem Drucker nur an einem Rechner installiert sein muss. Dieser Rechner sorgt auch für das Spooling (Zwischenspeichern der Druckjobs und Ausgabe an den Drucker in der richtigen Reihenfolge). Bei richtiger Konfiguration steht dieser Drucker dann allen Rechnern im Netzwerk zur Verfügung.

Welche Drucker werden von Linux unterstützt ?

Linux verwendet wie andere Un*x-Betriebssysteme Postscript als Standardausgabeformat. Postscript ist eine Druckerbeschreibungssprache, die Ausgabeanweisungen in beschreibender Form enthält (Beispiele: Setze den oberen Rand auf x, den linken Rand auf y, den rechten Rand auf z, den unteren Rand auf a, die Seitengröße auf b. Gebe jetzt das Wort „Hallo“ in der Größe c mit der Schriftart d und den Attributen e und f aus. Zeichne ein Quadrat mit der Seitenlänge g, die linke obere Ecke hat die Koordinaten h,i und fülle sie mit 50% grau aus). Postscript ist druckerunabhängig und wurde von der Firma Adobe entwickelt, um überall gleichförmige Ausdrücke zu bekommen. Die Drucker, die direkt Postscript verarbeiten können, haben eingebaute Intelligenz (eigener Prozessor und Speicher) um die Befehle für das eigene Drucksystem (Laser oder Inkjet u.a.) umzusetzen und den Ausdruck zu erzeugen. Linux kann mit allen solchen Druckern problemlos arbeiten, sie sind jedoch im oberen Preissegment angesiedelt. Auch für preiswertere Drucker gibt es Möglichkeiten, sie unter Linux anzusprechen. Postscript wird dabei in ein Rasterformat umgewandelt, und dieses Zwischenformat ist der Ausgangspunkt für die druckerspezifische Umwandlung in die eigentlichen Druckbefehle. Wieder einmal ist ein Unterschied zu machen zwischen Druckerherstellern, die Linux unterstützen, und denen, die nur für W*****s- und evtl. MAC-Rechner Treiber erstellen. Besonders schwierig wird es, wenn es sich um einen sogenannten GDI-Drucker handelt, der keine standardisierte Druckerbefehlssprache (Postscript, PCL 3,4.. usw.) beherrscht, dort muss der Druckertreiber mit Hilfe des Rechnerprozessors die komplette Berechnung der Druckdaten übernehmen. Die Linux Gemeinde hat zunächst generische Treiber entwickelt, die für bestimmte Druckerfamilien (HP, Epson, Canon) einfache Druckfunktionen bereitstellen. Erst später sind Verbesserungen in Druckqualität und z.B. Einstellungen zum Duplexdruck hinzugekommen.

Fazit daraus: Es gibt inzwischen sehr viele Drucker, die unter Linux sehr gut bis ausreichend arbeiten, bei ganz neuen Modellen und bei „Low Cost“ Geräten (GDI) kann es zu Problemen bis hin zum völligen Versagen kommen.

Ob bestimmte Geräte unter Linux arbeiten, kann man im Internet an vielen Stellen nachlesen (z.B. www.sane-project.org für Stand-Alone Scanner).

Einstellungen des Desktops und von Programmen

Bisher haben wir die „Stellschrauben“ des Systems betrachtet, bevor die graphische Bedienoberfläche startet, aber auch auf dem Desktop gibt es zahlreiche Einstellmöglichkeiten (Hintergrundbild, Menüstruktur, Bildschirmschoner, Auflösung, Maus- und Tastatureinstellungen, Login-Fenster).

Diese Spielwiese ist für ihre eigenen Einstellungen wie geschaffen, erreichbar ist sie nach Aufruf des Kontrollzentrums. Wir wollen uns zwei spezielle Einstellungen einmal genauer betrachten.

1. Das Login-Fenster bei der Anmeldung
2. Die Menüstruktur des „K“-Menüs und seine speziellen Programmeinstellungen

Für die Einstellungen zum Login klicken wir zunächst auf Systemverwaltung – Anmeldungsmanager. Nachdem wir in den Systemverwaltungsmodus gegangen sind (Root-Rechte erforderlich), können wir das Aussehen und die sichtbaren Benutzereinträge verändern, sogar ein automatisches Login eines Benutzers ist einstellbar.

Über Arbeitsflächen – Kontrollleiste – Menüs – K-Menü bearbeiten gelangen wir an eine Stelle, an der wir Einträge im K-Menü bearbeiten, löschen oder hinzufügen können. Interessant ist an dieser Stelle die Möglichkeit, den Programmen Parameter zu übergeben oder sie mit einer anderen Benutzerkennung aufzurufen.

Benutzer und Gruppen anlegen, ändern, löschen

Wir haben sowohl graphisch (Kuser) als auch über die Textkonsole die Möglichkeit, die Benutzer- und Gruppenverwaltung vorzunehmen.

Zuerst Kuser: Schon bei der Installation habt ihr ein Root-Passwort vergeben und einen normalen Benutzer angelegt. Linux verwendet intern noch weitere Nutzer und Gruppen, sie sind systemimmanent, d.h. für das Funktionieren erforderlich, werden von Linux selbst angelegt und sollten nicht verändert werden. Linux verwendet zur Unterscheidung nicht die Namen, sondern die Kennnummern Benutzernummer (UID= User ID) und Gruppennummer (GID= Group ID). Diese Tatsache sollte jeder beachten, wenn er mehr als einen Rechner im Netzwerk hat und mit seinen Nutzerrechten auf Dateien anderer Rechner zugreifen will. Gleiche Nutzernamen müssen nicht auf jedem Rechner auch die gleiche UID haben (Hier gibt es auch Unterschiede in der UID Vergabe bei verschiedenen Distributionen). Nur Root als Superuser hat immer die UID 0 und die Gruppe root die GID 0.

Auch im Textmodus lassen sich die Verwaltungsaufgaben lösen, die Befehle `adduser`, `deluser`, `addgroup`, `delgroup`, `useradd`, `userdel`, `usermod`, `groupadd`, `groupdel`, `groupmod`, `passwd` sind in der Lage, die gewünschten Einstellungen vorzunehmen. Der genaue Gebrauch ist auch hier durch den Aufruf von „`man (befehl)`“ nachzulesen.

Ergänzung der Dateiattribute (besondere Rechtevergabe)

Die bisher kennengelernten Dateiattribute (lesen-schreiben-ausführen) reichen nicht aus, um bestimmte Einstellungen vorzunehmen. Daher gibt es noch Erweiterungen der folgenden Art:

1. Das SUID-Bit. Ein Programm mit gesetztem SUID-Bit wird mit Root-Rechten ausgeführt, auch wenn es von normalen Benutzern aufgerufen wird. Ein Beispiel ist der Befehl „`passwd`“. Ein normaler Nutzer soll sein Passwort ändern können, jedoch sind die Passwortdateien nur für Root les- und schreibbar, „`passwd`“ mit gesetztem SUID-Bit kann auf die nötigen Dateien zugreifen und sie auch ändern. Gekennzeichnet wird das gesetzte Bit durch Ersetzen des „`x`“-ausführbar Zeichens durch „`s`“ bzw. „`S`“(bei nichtgesetztem „`x`“) beim Eigentümer der Datei.
2. Das GUID-Bit. Die Kennzeichnung erfolgt analog der SUID-Kennzeichnung, aber bei der Gruppenzugehörigkeit. Mit gesetztem GUID-Bit wird ein Programm immer mit der GID der gesetzten Gruppe ausgeführt, egal welcher Gruppe der Nutzer angehört. Wird bei einem Verzeichnis das GUID-Bit gesetzt, so bekommen alle neu im Verzeichnis angelegten Dateien automatisch die Gruppenzugehörigkeit des Verzeichnisses.
3. Das „Sticky“-Bit. Kennzeichnung: das „`x`“-ausführbar für übrige Nutzer wird durch „`t`“ bzw. „`T`“(bei nichtgesetztem „`x`“) ersetzt. Sinnvoll ist das nur bei Verzeichnissen. Dort bewirkt das gesetzte Bit, dass nur der Eigentümer einer Datei im Verzeichnis diese wieder verändern bzw. löschen kann. Ein wichtiges Verzeichnis mit dieser Eigenschaft ist das `/tmp` Verzeichnis. Dort können alle Benutzer ihre Dateien ablegen, für die einwandfreie Funktion ist es erforderlich, dass andere Nutzer diese nicht einfach löschen dürfen.