Einführung in Linux

Wolfgang Kinkeldei, 2000

Theodor-von-Zahn-Straße 17
91052 Erlangen
0 91 31-20 66 46
colour-press@01019freenet.de

1 Einleitung

1.1 Woher kommt Linux?

- Seit 1970: UNIX Systeme in der Mittleren Datentechnik erfolgreich eingesetzt und mehrfach weiterentwickelt; Heute existiert der Standard UNIX/System V.
- 1990: An Universitäten existierte ein experimentelles System „MINIX“, das viel von UNIX übernahm, aber nur zu Lehrzwecken gedacht war.
- 1992: Linus Torvalds: Erfand „Linux“ als Nachfolge des MINIX-Systems mit dem Zweck, ein Hochleistungssystem für 386er PCs zu entwickeln. Von Anfang an war jeder Entwicklungsstatus im Internet publiziert und Quellcodes von jedem lesbar.
- 199x: Eine Gruppe amerikanischer Programmierer (GNU) erstellte Unix-ähnliche Programmteile und ganze Programmpakete, die ebenfalls im Quellcode für jeden zugänglich waren. Ziel war es, UNIX zu ent-kommerzialisieren. ⇒ GNU Public License
- ab 1993: perfekte Harmonie zwischen Linux als Systemkern (ohne Anwendungsprogramme) und der GNU, die sich immer mehr aufeinander abzustimmen begannen. Der Siegeszug begann.
- Heute: Mehr als 10.000 Programmierer weltweit erstellen, testen und verbessern die Linux/GNU Programme und schaffen damit ein immer besseres und stabileres System. Alleine von SuSE werden jährlich ca. 1 Mio. Kopien von Linux verkauft...

1.2 Was erwartet und hier?

- Ein Überblick über den heutigen Stand von Linux.
- Die Durchführung einer exemplarischen Installation auf zeitgemäßen Rechnern mit friedlicher Co-Existenz zu Windows – wenn auch mit höherem Installationsaufwand.
- Bekanntmachung mit gängigen Begriffen, die den späteren Umgang mit diesem System erleichtern helfen sollen.
- Die bewußte Installation und Administration des Systems ohne graphische Hifsmittel, damit die eventuelle Problembeseitigung bei vielleicht zerstörter graphischer Oberfläche problemfrei durchführbar ist.

1.3 Auf welcher Hardware läuft Linux heute?

- Intel: 80386/486, Pentium, AMD, etc. ISA/EISA/PCI/VLB
- 680x0: Atari, Amiga, Mac
- PowerPC: Power Macintosh, G3, G4
- DEC Alpha PC
- Sun: Sparc
- weitere Systeme kommen ständig hinzu...

1.4 Was sind die Unterschiede zu DOS/Windows?

- Keine 100%ige graphische Installation – sie ist zwar im Kommen und zur Zeit große Mode, jedoch noch nicht ganz ausgereift.
- Keine korrekte und voll taugliche Hardware-Erkennung. Auch hier gibt es täglich neue Berichte, doch immer noch ist beim Installieren Hardware-Kenntnis gefragt und ein händischer Eingriff erforderlich.
- Die graphische Oberfläche ist ein Zusatzprodukt, das erst nach korrekter Installation lauffähig ist. Das Arbeiten in der Oberfläche ist zwar äußerst komfortabel, doch die Einrichtung von einigen Hindernissen begleitet.
- Die Administration des Systems nebst Fehlerbeseitigung kann z.T. nur an der Text-Console vorgenommen werden.
- Verschiedene Programme werden unterschiedlich konfiguriert und anders bedient, da jeweils andere Programmierer am Werk waren – es existiert keine für jeden Entwickler verbindliche Bedienungsrichtlinie.
- **Aber:** Linux arbeitet äußerst stabil (wenn die Hardware zuverlässig läuft) und ist durch fast nichts aus der Ruhe zu bringen.
- **Und:** Linux arbeitet um $10 - 50\%$ schneller als seine Konkurrenten, da die Hardware des PC besser ausgenutzt und zum Teil wesentlich effizienter programmiert wird.
- **Außerdem:** Linux ist extrem vielseitig einsetzbar.

1.5 Was kann Linux?

- Multi-Tasking/Multi-User: mehrere Programme laufen gleichzeitig auf dem System, ohne sich gegenseitig zu stören. Mehrere Benutzer können sich problemlos auf der gleichen Maschine anmelden (z.B. per Netzwerk), ohne gleich Engpässe zu verursachen.
- Multi-Prozessor: mehrere CPUs können gleichzeitig arbeiten, sofern das Motherboard mit mehreren CPUs bestückt ist. Damit erhöht sich im Idealfall die Leistung des Prozessors auf ein Vielfaches.
- Protected Mode: der Prozessor arbeitet permanent im Protected Mode. Linux ist damit ein wirkliches 32-Bit System, das vollständig auf 32-Bit Komponenten zurückgreift und nicht auf ein „aufgebohrtes" 8-Bit MS-DOS.
- Sicherheit: Programme sind gegenseitig voreinander geschützt. Sollte ein Programm abstürzen, bleiben alle anderen und vor allem das gesamte System stabil laufen – Der Wunschtraum aller PC Anwender!
- Speicher: sparsamer Umgang mit diesem kostbaren Medium. Wenn möglich wird nur bei Bedarf Speicher genutzt, viele Programme teilen sich Bibliotheken oder sonstige gemeinsame Resourcen.
- Puffer: Jedes im Vergleich zum Mikroprozessor langsame Gerät wird nicht direkt angesteuert, sondern per Puffer mit Daten versorgt, während der Rest des Systems weiterarbeitet.
- Daten: Linux kann alle gängigen Datenträger (DOS, Windows, NT, OS/2, Macintosh) lesen und schreiben – Datenaustausch ohne Progleme.
- Netzwerk: Linux ist als Server oder Client für alle gängigen Netzwerke wie TCP/IP, IPX, NetBEUI oder Appletalk eingesetzt werden – auf Wunsch alles gleichzeitig...

1.6 Was ist der minimale Rechner?

- Prozessor ab 386 – Pentium 166 sinnvoll
- Speicher ab 4MB – 32MB für graphische Oberfläche sinnvoll
- Festplatte ab 60 MB – 1GB vernünftig (SCSI oder IDE)
- CD-ROM (viele Typen) unbedingt sinnvoll
- Video-Karte VGA oder besser
- evtl. Netzwerk-Karte (viele Typen...)
- evtl. Modem oder ISDN-Karte
- evtl. Soundblaster o.ä.
- evtl. Maus (viele Typen...)

1.7 Welche Programme gibt es für Linux?

- Standard Unix-Kommandos
- Zum Teil extrem hochwertige Programm-Entwicklungswerkzeuge und praktisch alle gängigen Programmiersprachen (außer Visual Basic) verfügbar. Alle Sprachen halten sich an die heutigen Standards.
- Zahlreiche graphische Oberflächen, die sich durch Verwandlungsfähigkeit, Anpassbarkeit, Geschwindigkeit und vor allem durch Programmierbarbeit auszeichnen.
- Office-Anwendungen wie Corel Wordperfect, StarOffice, Applixware, die für private Nutzung kostenlos sind.
- Grafikprogramme, DTP, Video-Bearbeitung mit zum Teil sehr hohem Leistungsspektrum (Corel Draw ist als Beta verfügbar)
- Telekommunikationsprogramme, eMail, News-Anwendungen, Internet-Browser, Internet-Server
- Spiele, die zum Teil auch von kommerziellen Herstellern kommen (z.B. Myth), aber auch von vielen privaten Anbietern
- Datenbank-Systeme wie SQL-Server etc.
- Bis heute kein bekannter Virus

1.8 Woher bekomme ich Linux?

- aus dem Internet direkt von den Entwicklern oder per Download von diversen Mirror-Sites – bis auf Telefonkosten absolut kosten- und lizenzfrei
- von einem Distributor (z.B. SuSE, Red-Hat, Debian, Slackware, Corel, Caldera,...). Distributoren unterscheiden sich meist durch die Installations-Prozedur, die Hardware-Erkennung und den Umfang an Programmen

⇒ Distribution

1.9 Was sind die Probleme der Installation?

- Keine vollwertige Hardware-Erkennung – eingesetzte Hardware muß vorher bekannt sein. Am bestem Rechner aufschrauben, jede Karte analysieren oder Daten von laufendem Windows-System „abspicken."
- Viele neue Begriffe tauchen auf, was sagen sie aus? Was passiert genau bei der Installation? Wie kann man was beeinflussen?
- Angst vor Datenverlusten beim Einrichten der Festplatte(n). Vor allem wenn andere Systeme beibehalten werden sollen, können sich Probleme ergeben.
- Auswahl der Systemkonfiguration – bei mehr als 1000 verschiedenen Einzelkomponenten keine leichte Aufgabe. Zumahl zwischen den einzelnen „Paketen" viele Abhängigkeiten existieren.
- Korrektes Einrichten der Berechtigungen am System. Zwar darf der Administrator alles, doch sollte der Administrator nie ins Internet gehen oder Spiele laufen lassen...
- Gerade wenn etwas nicht funktioniert – Woher Hilfe holen???

1.10 Worauf läßt man sich ein?

- Jede Software ist möglicherweise fehlerhaft – Freie Software kommt ohne Garantieansprüche und es gibt keine Haftung.
- Da die Entwicklung sehr schnell geht, stehen evtl viele Updates an, die teils aus dem Internet kommen. Ist der aktuelle Stand OK oder brauche ich immer das neueste?
- Nicht jede Hardware funktioniert unter Linux. Zwar gehen die Anpassungen immer schneller vonstatten, doch meist quittieren brandneue Boards, Grafikkarten, ISDN-Modems und vor allem Sound-Karten teilweise den Dienst. Nach $5-6$ Monaten läßt sich dieses Problem meist lösen.
- Zu Linux gibt es nicht *das Handbuch*, sondern viele verschiedene kaufbare oder ladbare Dokumentationen, die zum Teil in englisch verfaßt sind. Zur korrekten Nutzung ist solche Lektüre teilweise erforderlich.
- Viel Geduld mit dem neuen System.
- Sammeln von Erfahrungen oder Erfahrungsaustausch unabdingbar – niemand weiß alles, aber irgendwer hilft garantiert bei einem bestimmten Problem.

2 Überblick über die Installation

2.1 `SETUP.EXE` – oder wie?

Nein – nicht ganz so einfach. Die Installation muß ja auch auf Systemen funktionieren, die (noch) gar kein Betriebssystem installiert haben. Daher darf die Installation nicht auf irgendein System aufsetzen. Die Installation wird

- bereits mit einem (abgespeckten) Linux-System durchgeführt,
- wahlweise von Diskette, CD-ROM oder einer Festplatte aktiviert,
- auf Wunsch aus einem laufenden DOS-System vorgenommen
- und ist damit von keinerlei Voraussetzung abhängig.

Das Installationsprogramm besteht hierbei aus mehreren Teilen:

- Beim Booten von Diskette oder CD-ROM aus einem Lade-Programm, das die nachfolgenden Schritte initiiert. Es ist vergleichbar mit den ersten Schritten des Boot-Prozesses anderer Betriebssysteme. ⇒ LILO
- Beim Start aus DOS aus einem DOS-angeplaßten Lade-Programm, was unter Beendigung des Betriebssystems MS-DOS bzw. Windows das Installations-Linux läd und aktiviert. ⇒ LOADLIN
- Dem System-Kern (*Kernel*), der mit dem späteren lauffähigen System völlig identisch ist und der die Hauptarbeit in einem Linux-System übernimmt. ⇒ Kernel
- Und nicht zuletzt dem Installations-Programm, das eigentlich eine „normale" unter Linux erstellte Anwendung ist. Dieses Programm läuft mit vollen Administrations-Rechten, damit ein Aufspielen des Systems überhaupt möglich wird. ⇒ linuxrc YaST

2.2 Welche Funtionen übernimmt der Kernel?

Wie in Abbildung 1 zu sehen ist, greifen Prozesse niemals direkt auf die Hardware eines Computers zu, sondern bedient sich des Kernels der wiederum per Treiber auf die eigentliche Hardware zugreift. Das

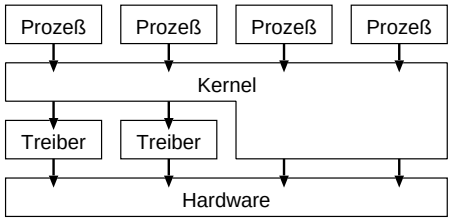

Abbildung 1: Aufbau eines Linux-Systems

Umgehen des Kernels durch trickreich geschriebene Programme ist unmöglich oder nur mit Administrations-Rechten möglich, wodurch Linux extrem sicher im laufenden Betrieb und kaum verwundbar ist.

Bei einem lauffähigen System zählen zu den Aufgaben des Kerns:

- Angestoßen durch den Boot- oder Ladevorgang Start des Systems mit allen Hilfsprogrammen, Hardwareansteuerung und Systemdiensten.
- Verwalten des Speichers
- Kontrolle jedes Zugriffs mittels der vergebenen Rechte
- Steuerung der Ausführung von Programmen (Prozesse)
- Umgang mit der Hardware (über Gerätetreiber)
- Verwaltung von Dateien auf Festplatten, CD-ROMs, Disketten, ...
- Elementarer Netzwerk-Support
- Durchführen von Statistiken, Abrechnungen, Protokollen
- Herunterfahren des Systems und Anhalten des Rechners

2.3 Geschafft: Der erste Boot-Vorgang

```
Linux version 2.2.10 (root@Mandelbrot.suse.de) (gcc version 2.7.2.3) #4 Tue Jul
20 17:01:36 MEST 1999
Detected 400922277 Hz processor.
Console: colour VGA+ 80x25
Calibrating delay loop... 799.54 BogoMIPS
Memory: 127820k/131072k available (1260k kernel code, 404k reserved (endbase 0xa
0000), 1544k data, 44k init)
VFS: Diskquotas version dquot_6.4.0 initialized
CPU: AMD AMD-K6(tm) 3D processor stepping 0c
Checking 386/387 coupling... OK, FPU using exception 16 error reporting.
Checking 'hlt' instruction... OK.
Checking for popad bug... OK.
POSIX conformance testing by UNIFIX
mtrr: v1.35 (19990512) Richard Gooch (rgooch@atnf.csiro.au)
PCI: PCI BIOS revision 2.10 entry at 0xfb2c0
PCI: Using configuration type 1
PCI: Probing PCI hardware
PCI: 00:38 [1106/0586]: Work around ISA DMA hangs (00)
Activating ISA DMA hang workarounds.
Linux NET4.0 for Linux 2.2
Based upon Swansea University Computer Society NET3.039
NET4: Unix domain sockets 1.0 for Linux NET4.0.
NET4: Linux TCP/IP 1.0 for NET4.0
```

Abbildung 2: Boot-Meldungen des Kernels

Wie in Abbildung 2 zu sehen ist, gibt der Kernel beim Booten eine Reihe von Meldungen über die gefundene Hardware aus sowie eine ganze Menge an Informationen zur Einstellung verschiedener Dienste, Prozesse, etc. Nach diesem Vorgang ist ein Linux-System zum Arbeiten bereit.

2.4 Arbeiten mit Linux

Bei der Anmeldung an einem Terminal wird zunächst eine Begrüßungsmeldung ausgegeben, auf die ein gültiger Benutzername sowie ein Paßwort einzugeben ist. Danach steht das System zum Eingeben von Kommandos zur Verfügung.

Damit Kommandos eingegeben werden können, wird nach dem Login eine shell (=Kommandointerpreter) gestartet, der Eingaben als Kommandos interpretiert.
Abbildung 3 zeigt einen solchen Verlauf.

Nach dem Ende einer Sitzung meldet sich ein Benutzer wieder ab. Dabei wird die shell wieder verlassen und das System zeigt erneut den Anmelde-Bildschirm (siehe Abbildung 3).

`/bin/logout`

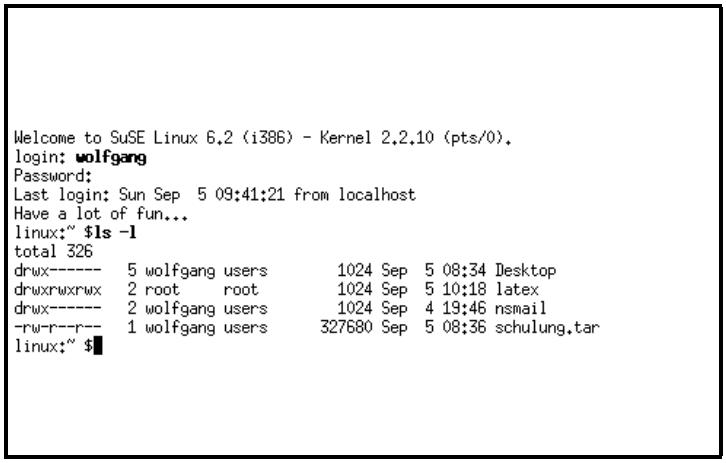

Abbildung 3: typische Anmeldung an einem Terminal

2.5 Abschluß: Herunterfahren des Systems

Ein Linux-Rechner darf **nie** einfach ausgeschaltet werden, da möglicherweise noch Daten in Puffern sind, die auf die Festplatte zurückgeschrieben werden müssen. Daher sind zum Abschalten besondere Kommandos erforderlich, die für ein kontrolliertes „Herunterfahren“ des Systems sorgen.

`/sbin/halt`

2.6 Besser: graphische Oberfläche

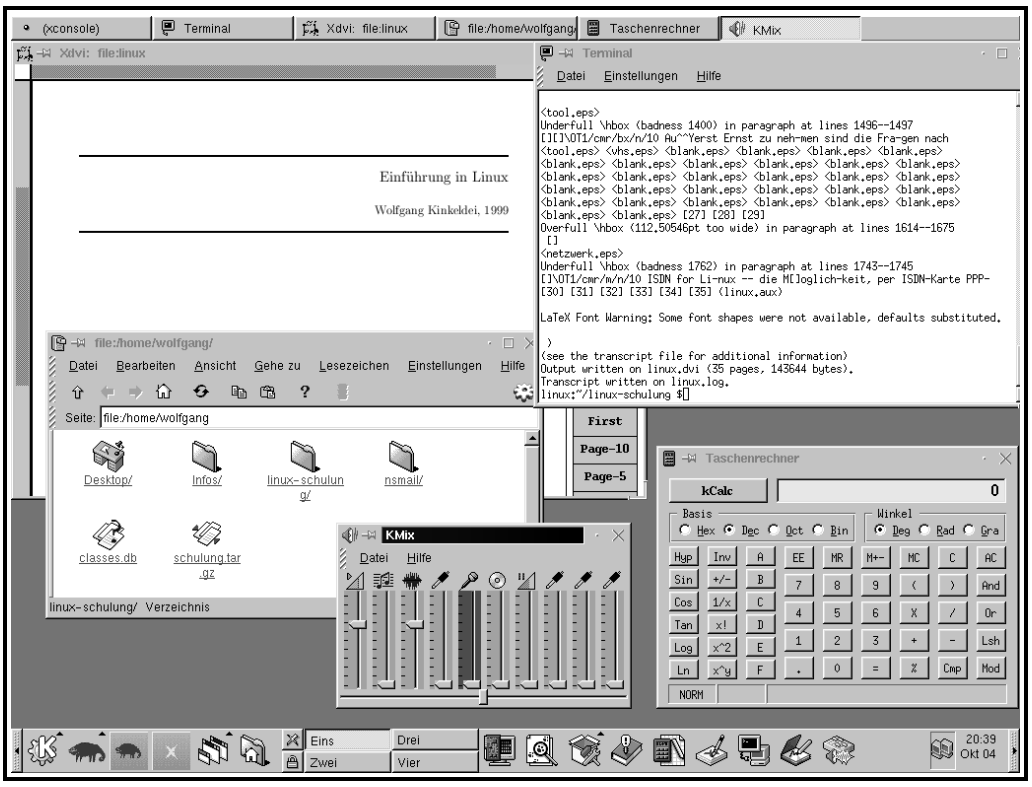

Abbildung 4: der KDE-Desktop mit einigen Programmen

Selbstverständlich läßt sich auch unter Linux graphisch arbeiten. Die verschiedenen Oberflächen sind genau so intuitiv wie die Konkurrenten. Nur unter Linux hat man die Wahl zwischen verschiedenen Oberflächen mit stark unterschiedlichen Möglichkeiten.

3 Installationsvorbereitung – Grundbegriffe

3.1 Allgemeines zu Festplatten

Bevor Dateien einen würdigen Platz finden, muß die Festplatte für Linux vorbereitet werden. Hierzu wird diese in Bereiche, sog. *Partitionen* eingeteilt werden.

⇒ `/dev/...` Gerätenamen

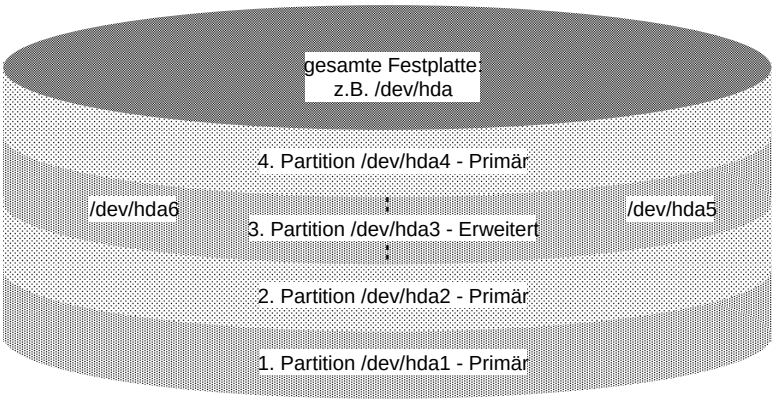

Abbildung 5: Einteilung einer Platte in Partitionen

Bei Intel PCs haben Partitionen folgende Eigenschaften:

- Jede Festplatte kann bis zu 4 *Partitionen* unterhalten. Jede Partition ist hierbei ein zusammenhängender Bereich, der durch die Angabe des Start- und End-Zylinders gekennzeichnet wird. Indirekt ergibt sich dadurch die Kapazität jeder einzelnen Partition.
- Eine Partition kann ein eigenes *Betriebssystem* aufnehmen, oder einem Betriebssystem einer anderen Partition als *Laufwerk* dienen.
- Genau eine der Partitionen ist die *aktive Partition*, die sich dadurch auszeichnet, daß das auf ihr befindliche Betriebssystem beim booten von dieser Platte gestartet wird.
- Je nach dem Verwendungszweck einer Partition wird die Partition mit einer *Identifikations-Nummer* dem späteren Verwendungszweck zugeordnet. Hiermit wird sichergestellt, daß jedes Betriebssystem nur auf seine eigenen Partitionen zugreift.
- Alle Angaben zu den Partitionen werden in der *Partitions-Tabelle* eingetragen, die bei jeder Festplatte an einer genau definierten Stelle sitzt und die eben nur 4 Partitionen aufnehmen kann.
- Sind 4 Partitionen zu wenig, so wird eine der vier möglichen als *Erweiterte Partition* angelegt, die dann beliebig viele logische Partitionen aufnehmen kann. Diese verhalten sich vom Prinzip her wie die sonst angelegten *Primären Partitionen*, werden intern nur anders verwaltet.
- Jedes Betriebssystem führt seine eigene interne Namensgebung zum Ansprechen einer Partition. Die Namensgebung von Linux ist in Abbildung 5 dargestellt, die Namensgebung unterschiedlicher Gerätetypen findet sich in Tabelle 1.
- Neben den Partitionen und der Partitionstabelle besitzt jede in einem PC eingesetzte Festplatte einen *Master-Boot-Record* (MBR), der typischerweise vom zuletzt installierten System mit dem zum Booten dieses Systems notwendigen Code besetzt wird.

3.2 Festplatten unter Linux

Zur Installation von Linux müssen (besser: sollten) mindestens zwei Partitionen angelegt werden:

- Eine Partition zur Aufnahme der gesamten von Linux verwalteten Dateien – System, Programme und Anwender-Dateien. Diese Partition muß ausreichend dimensioniert sein (bei graphischer Oberfläche min. 1 GB). Der Linux-Mensch spricht vom sog. *root-Dateisystem*.
- Typischerweise (nicht zwingend) eine *swap-Partition*, die zur Auslagerung von Blöcken aus dem Hauptspeicher verwendet wird, sofern dieser knapp wird und noch Speicher benötigt werden sollte. Meist wird diese Partition doppelt so groß angelegt wie der vorhandene Hauptspeicher (RAM). Ab 64 MB RAM kann man evtl. auf den swap-Bereich verzichten, es ist aber nicht empfehlenswert.

Diese (und evtl. weitere) Partitionen können wahlweise und willkürlich auf die zur Verfügung stehenden Festplatten und deren Partitionen verteilt werden. Die einzige Einschränkung ist, daß die root-Partition

IDE-Festplatten	
/dev/hda	gesamte erste Festplatte (Master, 1. Kanal)
/dev/hdb	gesamte zweite Festplatte (Slave, 1. Kanal)
/dev/hdc	gesamte dritte Festplatte (Master, 2. Kanal)
/dev/hdd	gesamte vierte Festplatte (Slave, 2. Kanal)
/dev/hda1	erste Partition der ersten Platte
/dev/hda2	zweite Partition der ersten Platte
/dev/hda5	erstes logisches Laufwerk der erw. Partition
/dev/hda6	zweites logisches Laufwerk der erw. Partition

SCSI-Festplatten	
/dev/sda	gesamte erste Festplatte (niedrigste ID)
/dev/sdb	gesamte zweite Festplatte (nächsthöhere ID)
/dev/sda3	dritte Partition der ersten Platte

Tabelle 1: verwendete Gerätenamen für Festplatten

maximal bis zum Zylinder Nr. 1024 geht (BIOS-Problem), da sonst von dieser Partition möglicherweise nicht gebootet werden kann.

Der Sicherheit dienlich (wenn gleich schwer zu planen) ist die Aufteilung der Dateien auf mehrere Platten bzw. Partitionen. Hierbei werden alle Dateien eines bestimmten Unterverzeichnisses einfach auf eine andere Partition ausgelagert, bzw. von dieser an den Baum des root-Dateisystems „angehängt."

Die einzelnen Partitionen, aus denen ein Linux Dateisystem besteht (außer swap) werden zu einem Dateibaum zusammengefaßt, der im Root-Dateisystem seinen Ursprung findet (siehe Abbildung 6). Im Gegensatz zu anderen Systemen kennt Linux keine Gerätenamen oder Laufwerksbuchstaben. Es existiert immer genau *ein* Dateisystem, welches durch den Pfad-Namen „/" symbolisiert wird. Alle sonst in einem Linux-System vorhandenen Partitionen oder Geräte werden an entsprechenden Verzeichnissen (*mount-point*), die im root-Dateisystem angelegt werden, eingebaut (*mount*).

⇒ swap Bereich

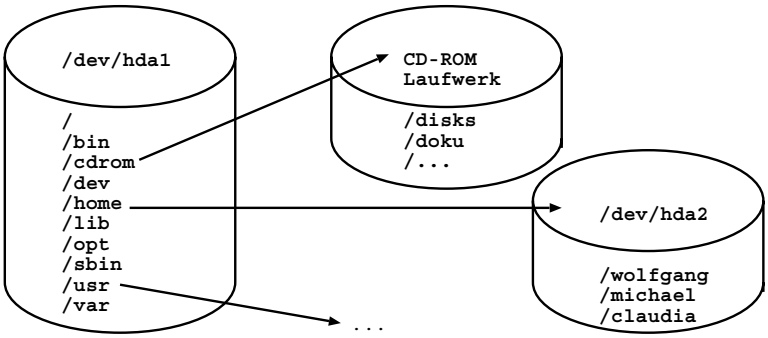

Abbildung 6: Verbinden von Dateisystemen

Bei der Partitionierung der Festplatte legt man gerne eigene Partitionen an für:

- Benutzerverzeichnisse (unter `/home/`)
- Anwendungsprogramme (unter `/usr/` oder `/opt/`)
- große variable Datenmengen wie Log-Dateien (unter `/var/`)

Hierbei ist jedoch eine genaue Planung der angestrebten Datenmengen erforderlich, da jede Partition eine exakt vorher zu definierende Größe haben muß und nicht nachträglich verändert werden kann.

Im Root Dateisystem müssen für jeden Mount-Point leere Verzeichnisse angelegt werden. Dies sieht auch „gelegentliches" Einbinden vor für:

/bin/mount
/etc/fstab

- MS-DOS Dateisysteme (z.B. `/dos/`, `/C/`)
- Disketten (z.B. `/floppy/`)
- CD-ROM Laufwerke (z.B. `/cdrom/`)
- sonstige Geräte (z.B. `/mnt/`)

3.3 Dateien

Dateien sind ein wichtiger Eckpfeiler für jedes UNIX-System – und damit auch für unser Linux. Für fast jeden Zweck gibt es Dateien, deren Zweck nicht unbedingt der konventionellen Bedeutung von Dateien entspricht. Grund für dieses Verhalten ist eine grundsätzliche Philosophie. Denn wenn ein System für jeden Zweck nur mit Dateien umgehen können muß, lassen sich viele Dinge innerhalb des Betriebssystems einfach wesentlich simpler umsetzten. Die „Kenndaten“ einer Datei sind in Tabelle 2 zusammengefaßt.

Für jeweils diese Zwecke gibt es Dateien:

- Der Kernel ist eine ganz normale Datei. Er wird zwar auf besondere Weise aktiviert, steht aber ganz normal neben allen anderen Dateien auf einer Festplatte.
- Anwenderdateien werden selbstverständlich wie gewohnt im Dateibaum abgelegt.
- Programme liegen im nicht gestarteten Zustand als „ausführbare“ Dateien irgendwo im Dateibaum.
- Jegliche Systemeinstellung oder -konfiguration wird durch eine oder mehrere Dateien vorgenommen, die unter bestimmten Namen an genau festgelegten Stellen des Dateibaums liegen. Solche Dateien sind meist Text-Dateien, die sich mit jedem Editor bearbeiten lassen.
- Jedes Verzeichnis (oder Ordner) ist eigentlich eine Datei – allerdings mit besonderer Bedeutung.
- Eine Verknüpfung (link) ist eine Datei.
- Für jedes Gerät, das an einen Rechner angeschlossen ist (Maus, Modem, Druckerport, CD-ROM, Festplatte...) gibt es eine Datei, über die das Gerät intern angesprochen wird.
- Sogar der Hauptspeicher (RAM) wird intern wie eine Datei behandelt.

Dateiname	Eine bis zu 255 Zeichen lange Buchstabensequenz, die theoretisch jeden Buchstaben einschließlich Sonderzeichen enthalten kann. Linux unterscheidet zwischen Groß- und Kleinbuchstaben. Jedes Sonderzeichen (einschließlich dem Punkt) ist Bestandteil des Namens. z.B. `datei`, `meine-datei`, `brief.txt`
Extension	Wahlweise einem Dateinamen angehängter Text, der durch einen Punkt vom gedachten Namen getrennt ist. Da der Punkt Bestandteil des Namens ist, spielt intern für Linux die Extension keine Rolle. Anwenderprogramme jedoch erwarten meist, daß ihre Dateien eine dem Programm passende Extension tragen.
Versteckt	Sind Dateien, deren Name mit einem Punkt beginnen. Solche Dateien werden – obwohl vorhanden – meist nicht angezeigt.
Rechte	Jede Datei besitzt einen *Eigentümer*, der wiederum Mitglied einer bestimmten *Gruppe* ist. Diese Angaben sowie entsprechende Zugriffsrechte für die gesamte Gruppe oder Außenseiter werden jeder Datei nit auf den Weg gegeben.
Zeit	Jede Datei führt mehrere Zeit-Stempel mit sich: Erstellungs-, Änderungs- und letztes Zugriffsdatum.
Daten	Natürlich ist es der Zweck einer Datei, Daten aufzunehmen.

Tabelle 2: Merkmale von Dateien

3.4 Pfade

Um eine Datei eindeutig identifizieren zu können, ist nebem dem Namen der Datei auch die Lage innerhalb des Verzeichnisbaums von Bedeutung. Dabei gibt es einige Konventionen:

- Die einzelnen „Etappen“ eines Pfades werden durch einen Schrägstrich (`/`) voneinander getrennt.
- Ein Pfad, der mit einem Schrägstrich (`/`) anfängt, beginnt in der Wurzel (root) des Dateibaums (*absoluter Pfad*) – fehlt der Schrägstrich am Anfang, beginnt der Pfad beim aktuellen Verzeichnis (*relativer Pfad*)
- Ein einzelner Punkt (`.`) kennzeichnet das aktuelle Verzeichnis
- Zwei Punkte (`..`) meint eine Ebene höher liegend als das aktuelle Verzeichnis (kann nicht über die Wurzel des Dateisystems hinausgehen)
- Die Tilde (`~`) bezeichnet das Home-Verzeichnis des angemeldeten Benutzers, was in aller Regel unter `/home/`*`benutzername`*`/` untergebracht ist.
  Ausnahme: Benutzer „root“ hat sein Home-Verzeichnis unter `/root/`.

Beispiele:

`/`	Die Wurzel des Dateibaums
`/home/wolfgang/linux.txt`	Absolute Pfadangabe vom Anfang des Baumes an
`/home/wolfgang/.emacs`	Absolute Pfadangabe zu einer versteckten Datei
`../kcc/kcc.cpp`	Relative Pfadangabe von der gegenwärtigen Stelle aus (Eine Ebene höher, dann ins Verzeichnis `kcc...`)
`./setup`	Datei setup im aktuellen Verzeichnis
`test.gif`	Datei im aktuellen Verzeichnis
`bilder/ich.jpg`	von aktueller Stelle ausgehend
`~/dok/brief.txt`	Angabe relativ zum eigenen Home-Verzeichnis
`/floppy/brief.doc`	Datei „`brief.doc`“ auf einer eingelegten und ins Dateisystem eingebundenen Diskette
`/cdrom/suse/a1/`	Das Verzeichnis „`/suse/a1`“ auf einer eingelegten und ins Dateisystem eingebundenen CD-ROM

3.5 Wichtige Verzeichnisse unter Linux

`/`	Das Wurzelverzeichnis (root directory); der Beginn des Verzeichnisbaums
`/bin/`	Enthält ausführbare Programme, die zum Hochlaufen und zum Betrieb des Systems erforderlich sind sowie einige essentielle Kommandos
`/boot/`	Enthält den Kernel sowie einige weitere für das Booten von Linux per LILO erforderliche Dateien
`/dev/`	Beinhaltet alle bekannten Geräte, die als Geräte-Dateien realisiert sind
`/etc/`	besteht aus Konfigurations-Dateien, die die Arbeitsweise des Rechners maßgeblich beeinflussen
`/home/`	Hier gibt es für jeden Benutzer, der sich anmelden kann ein eigenes Verzeichnis mit dessen Daten
`/lib/`	Shared Libraries für dynamisch gelinkte Programme, evtl. Kernel-Module
`/lost+found/`	hier landen bei Datenrettung entstehende „verlorene“ Dateien
`/opt/`	In dieses Verzeichnis werden zahlreiche große Systeme installiert (z.B. KDE, Gnome)
`/proc/`	Alle hier befindlichen Dateien und Verzeichnisse reflektieren Aktionen des Kernels und werden von vielen Utilities genutzt
`/root/`	Das Home-Verzeichnis des Benutzers „root“
`/sbin/`	Kommandos für den System-Administrator und zum Hochlaufen des Rechners
`/tmp/`	wird für (meist kleine) temporäre Dateien genutzt
`/usr/`	Die meisten Anwendungsprogramme sind hier. In diesem Zweig sind typischerweise keine veränderlichen Dateien enthalten. Damit kann eine Partition, die den `/usr` Zweig enthält als nur-lesbar gemountet werden
`/var/`	Dient der Ablage von (veränderlichen) Konfigurationsdateien und Protokollen

4 Genauer Ablauf der Installation

4.1 Daten sammeln

Bevor an eine Installation zu denken ist, gilt es alle für Linux relevanten Daten über den einzurichtenden Rechner zu sammeln:

- Art, Größe und Anschluß der Festplatte
  z.B. IDE, 6GByte, Controller 0 – Master
- Größe und Lage der evtl. zu erhaltenden (Windows-)Partition
  z.B. Partition 1: 2,5 GByte; Partition 2: 1 GByte (OS/2)
- Größe der unbenutzten oder zu löschenden Partition(en)
  z.B. nach Partition 2 sind 2,5 GB frei
- Alle Steckkarten mit ihren Daten zusammentragen (Hersteller, Modell, Besonderheiten, Einstellungen)
- Alle Peripherie-Elemente zusammenfassen (Maus, Monitor, Drucker,...)

4.2 Installations-Methode festlegen

Jetzt muß festgelegt werden, wie unser Installations-System gestartet werden soll:

- Von CD-ROM (sofern Hardware und BIOS dies zuläßt) – die bequemste und schnellste Methode.
- Von Diskette – eine einfache aber gemütliche Variante
- Von Festplatte unter MS-DOS – hier ist etwas Vorarbeit zu leisten, damit dies überhaupt funktioniert.

4.3 Erste Hürde – Partitionierung

Nach den ersten (noch trivialen) Fragen des Installationsprogramms muß die zu verwendende Platte partitioniert werden. Hierbei gilt es besonders vorsichtig vorzugehen, wenn bestehende Systeme erhalten werden sollen. Bitte nur die Linux-Partitionen anlegen und ändern, niemals versuchen, Veränderungen an bereits bestehenden Partitionen vorzunehmen!

Nach der Festlegung der Partitionen muß jeder Linux-Partition ein Pfadname zugeordnet werden, der die spätere Lage im Dateibaum wiederspiegelt.

- Genau eine Partition muß den Beginn des Baums darstellen – hier ist der Baum-Anfang (`/`) als *mount-point* anzugeben.
- Sollen weitere Partitionen fest in den Dateibaum eingebaut werden (vorherige Analyse der Datenmenge!!!), so sind diesen Partitionen die entsprechenden Pfade anzugeben (z.B. `/usr`, `/home`).

Jetzt werden die Partitionen formatiert, genauer gesagt Dateisysteme (jedoch noch ohne Dateien) angelegt. Die hier entstehenden Dateisysteme sind ausschließlich zur Aufnahme von Linux-Dateien geeignet, DOS und andere Betriebssysteme haben darauf keinen Zugriff!

4.4 Konfiguration bestimmen

Jetzt beginnt die Auswahl des Installations-Umfangs. Die von uns verwendete Distribution SuSE gestattet die Auswahl von mehreren vordefinierten Standard-Konfigurationen, in denen jeweils sorgsam abgestimmt einzelne Komponenten miteinander kombiniert installiert werden.

Intern besteht jede Konfiguration aus einer Liste einzelner *Pakete*, die jeweils ein gesamtes Programmpaket mit allen dazugehörigen Hifs-Dateien, Dokumentationen, Beispielen etc. enthalten. Jedes Paket ist jeweils als eine Datei auf der Installations-CD vorliegend. Folgende Paketformate sind üblich:

- RPM (Red Hat Package Manager) ist das meist verbreitete Paketformat. Diese Paketdateien sind erkennbar am Dateianhang `.rpm` und enthalten neben den Dateien noch zahlreiche Verwaltungs-Informationen (Abhängigkeiten zu anderen Paketen etc), die für die korrekte Installation hilfreich sind.
- DEB (Debian) ist vergleichbar zu RPM. Erkennbar an der Extension `.deb`.
- TAR (Tape Archive) ist ein 0 8 15 Unix-Pack-Dateiformat (vergleichbar zu WinZip), in dem ebenfalls mehrere Dateien (jedoch ohne Komprimierung) abgelegt sind. Hier jedoch gibt es keinerlei Verwaltungs-Information, was manchmal große Probleme wegen fehlender weiterer Dateien mit sich bringt. Hier wird die Extension `.tar` verwendet.

- Gepackte Archive sind TAR-Dateien, die mit einem UNIX Packprogramm komprimiert wurden. Sie sind an den Extensions `.tgz` oder `.tar.gz` schnell erkennbar.

Neben der Anwahl einer Standard-Konfiguration lassen sich Pakete auch einzeln dem Installationsumfang hinzufügen oder entfernen. Dies ist durch die hohe Zahl von mehr als 1000 Paketen zwar am Anfang recht mühsam, doch sind die einzelnen Pakete ind *Serien* eingeteilt, deren Titel eine grobe Auswahl nach einiger Routine schnell auffinden läßt.

4.5 Anschließende Fragen

Nach dem Einspielen der ausgewählten Pakete werden noch ein paar „Kleinigkeiten“ abgefragt, die die Konfiguration der Installation betreffen. Die meisten Fragen sind dabei ziemlich selbsterklärend.

Wichtig ist die korrekte Auswahl eines Kernels. Hier enthält die Liste eine Serie von Kernels, die jeweils auf eine bestimmte Hardware (Festplattencontroller) zugeschnitten ist. Meist wird wohl der „EIDE“ Kernel die korrekte Wahl sein.

Ernst zu nehmen ist die Abfrage der Boot-Konfiguration. Hier gibt es drei Möglichkeiten, den LILO (Linux Loader) zu installieren:

- Booten von Festplatte über den Master Boot Record (MBR). Hiermit werden alle sonst auf dem Rechner befindlichen Betriebssysteme außer Kraft gesetzt. Eine spätere Rekonstruktion ist nur schwer wieder möglich.
- Booten von Festplatte über den Boot-Sektor der Partition. Damit wird nur in der Root-Partition ein Boot-Lader installiert. Dieser kann allerdings nur von einem Boot-Manager angesprochen werden.
- Keine Installation von LILO. Bei dieser Einstellung muß der Boot-Prozeß über eine speziell herzustellende Diskette oder von MS-DOS aus aktiviert werden.

Nun startet das gerade installierte System und schließt die Installation mit dem ersten Start ab. Dieser Vorgang wird einige Minuten (ca. 5-10) dauern.

5 Ablauf der Installation

5.1 Booten von einem geeigneten Installations-Medium

Eine „normale“ Installation wird vorgenommen, indem nach dem Selbsttest des Rechners dieser von einer speziell präparierten Linux-Boot Diskette gestartet wird. Diese enthält einen voll lauffähigen Kernel, einige Treiber-Dateien und ein spezielles für die Installation notwendiges Programm.

Wichtig

Unsere Installation geht rechnerbedingt etwas anders. Von der Linux Installations-CD kopieren wir einige Dateien in ein neu zu erstellendes Verzeichnis. Dies wird im MS-DOS Fenster des auf dem Rechner vorhandenen Windows 98 durchgeführt.

```
C:\WINDOWS>cd \
C:\>md linux
C:\>cd linux
C:\LINUX>copy L:\suse\setup\loadlin.exe
C:\LINUX>copy L:\suse\images\eide01
C:\LINUX>copy L:\suse\images\initdisk.gz
```

Nun verlassen wir das MS-DOS Fenster von Windows, beenden Windows und starten im MS-DOS Modus (ohne diese Prozedur ist Linux nicht zum Laufen zu bewegen...)

```
C:\WINDOWS>cd \linux
C:\LINUX>loadlin eide01 initrd=initdisk.gz root=/dev/ram
```

Nach dem erfolgreichen Start des Ur-Linux und „linuxrc“ sind anzugeben:

- Die Sprache des Installations-Programmes ⇒ Deutsch
- Die Art des Bildschirms Farbe
- Die gewünschte Tastaturbelegung Deutsch

Im nachfolgenden Menü können Informationen angezeigt oder Module nachgeladen werden, was abhängig von der verwendeten Hardware (z.B. SCSI-Controller) erforderlich sein kann. Zum Abschluß wird der Menüpunkt „Installation / System starten“ gewählt.

5.2 Art der Aktivität wählen

hier bietet „linuxrc“ vier Möglichkeiten:

- Starten der Installation – der Normalfall ⇒ Starten
- Installiertes System booten – falls normal booten nicht klappen sollte
- Rettungssystem starten – ein „mini-System“ für Service-Zwecke
- Live-CD starten – ein voll lauffähiges System auf einer separaten CD

5.3 Quell-Medium wählen

In diesem letzten Menü-Punkt bieten sich vier Alternativen:

- CD-ROM – der übliche Weg ⇒ CD-ROM
- Netzwerk [NFS] – falls ein NFS-Server zur Installation zur Verfügung steht
- Netzwerk [FTP] – für die Installation von einem FTP-Server (evtl. Internet)
- Festplatte – von einer Platte, auf der die Installations-Daten vorliegen

Sind alle diese Fragen beantwortet und keine Fehler aufgetreten, dann wird das eigentliche Installationsprogramm „YaST“ gestartet, was im laufenden Linux-System jederzeit zur nachträglichen Umkonfiguration gestartet werden kann (allerdings nur von „root“). `/sbin/yast`

5.4 Ziel-Festplatte(n) vorbereiten

Die erste Hürde bei der Installation ist die Einrichtung der Partitionen. Soll das Dateisystem in mehrere Partitionen zerlegt werden, muß vorher eine Abschätzung der erforderlichen Datenmenge vorgenommen werden!
Für jede einzelne Festplatte ist hierbei festzulegen, welche Partitionen einzurichten sind. Dies ist in Abbildung 7 zu sehen.

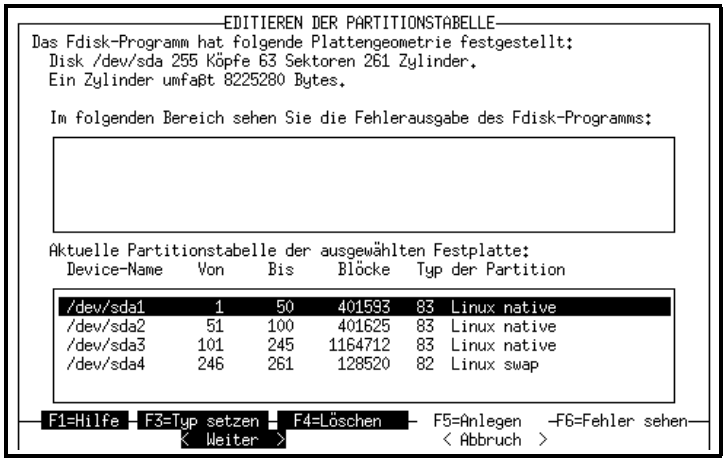

Abbildung 7: Partitionierung einer Festplatte

Dabei gilt es folgendes zu beachten:

- Niemals den Typ einer bestehenden Partition ändern, da sonst das auf diese Partition zugreifende Betriebssystem Probleme bekommt.
- Partitionen lassen sich nicht vergrößern oder verkleinern – nur indirekt durch Löschen und neu Anlegen einer Partition. Dabei gehen unweigerlich alle in dieser Partition gespeicherten Daten verloren.
- Eine Festplatte kann nur 4 Partitionen besitzen; sollen mehr als 4 Partitionen erzeugt werden, ist zumindest die 4. Partition als erweiterte Partition einzurichten, die dann logische Laufwerke aufnehmen kann.
- Eine Partition sollte als Swap-Partition eingerichtet werden, deren Größe typischerweise das doppelte des physikalisch vorhandenen Hauptspeichers beträgt, meist jedoch maximal 128MB.

Wir bedienen uns einer bereits angelegten Partition `hda6`, die wir für unsere Linux-Installation verwenden. Diese Partition ist ein logisches Laufwerk unter Windows, das wir entfernen und für zwei Partitionen (Linux und Swap) umdefinieren.

- zunächst wird die Partition gelöscht.
- Dann wird eine neue Partition angelegt. Größe ist der vorgegebene Bereich abzüglich 9 Zylindern, Typ ist *Linux native*.
- Daraufhin wird eine weitere Partition mit der restlichen Größe von 9 Zylindern angelegt, die den Typ *Linux swap* erhält.

5.5 Dateisysteme einrichten

Nach der Definition der Partitionen wird jeder Partition eine Lage innerhalb des Dateibaums zugeordnet. Dies wird anhand der Abbildung 8 deutlich gemacht. Neu einzurichtende Partitionen sollten formatiert werden, was nach Beendigung dieses Bildschirms erfolgt.

Da wir nur eine Partition für Linux definiert haben, wird diese Partition zum „root“ des Dateibaums, was wir erreichen, indem der „Mount-Point“ dieser Partition auf (`/`) gesetzt wird. Außerdem soll diese Partition formatiert werden, daher ist in das Feld „Formatieren“ der Eintrag auf (`Ja`) zu setzen.

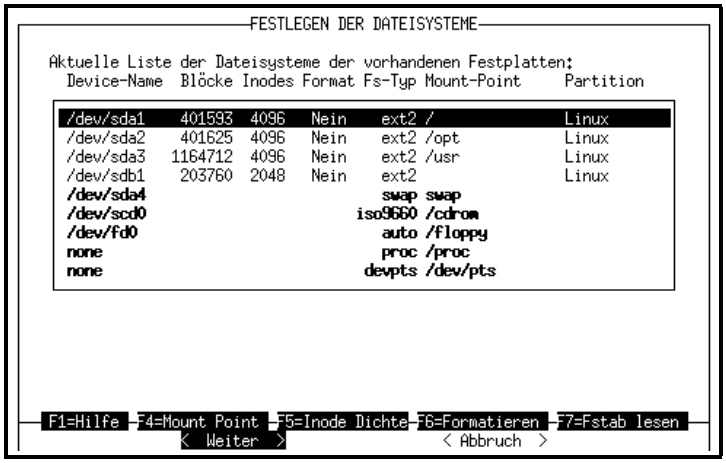

Abbildung 8: Festlegen des Dateibaums

5.6 Pakete zur Installation auswählen

Jetzt kann die eigentliche Installation beginnen. Hierbei bietet sich die einfache Möglichkeit vorgegebene Konfigurationen zunächst zu übernehmen (siehe Abbildung 9).

> Wir machen uns zunächst das Leben besonders einfach, indem wir eine der Standard-Konfigurationen, nämlich „SuSE Minimal System“ anwählen.
> Dadurch entfällt zunächst die umständliche Auswahl von verschiedenen Paketen...

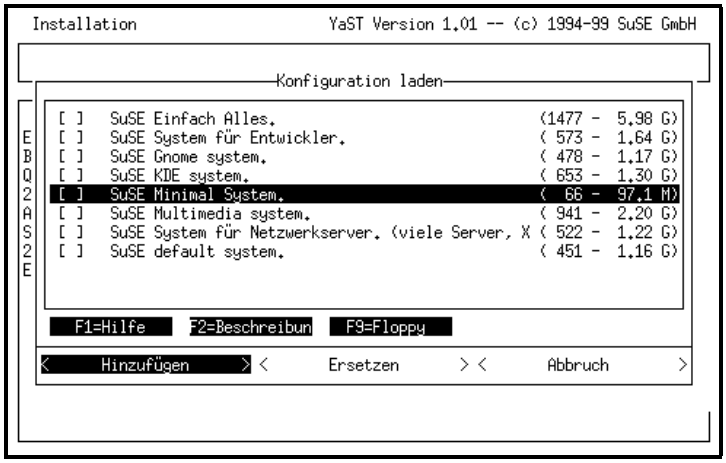

Abbildung 9: Auswählen von Standard-Konfigurationen

Anschließend (oder alternativ) können dann einzelne Pakete, die jeweils der Übersichtlichkeit halber in Gruppen zusammengefaßt sind, installiert werden. Hier ist gemäß den Abbildungen 10 und 11 zunächst die Serie und dann einzelne Pakete aus dieser Serie zu wählen.

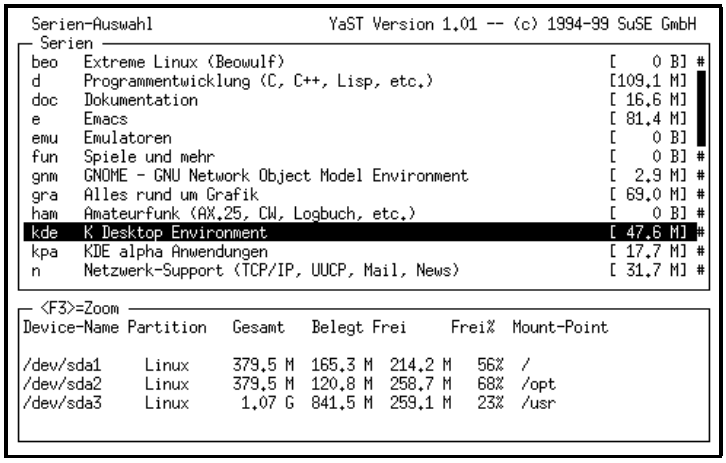

Abbildung 10: Auswählen einer Paket-Serie

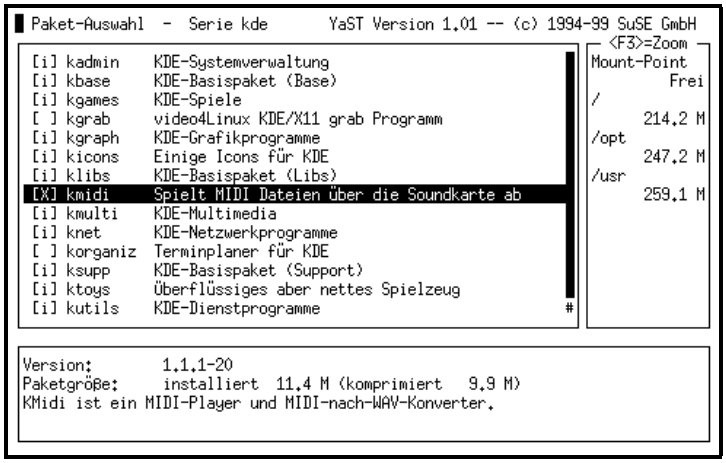

Abbildung 11: Wählen eines oder mehrerer Pakete

5.7 Post-Installation – erster Start

Sind alle Pakete korrekt installiert worden, so ist zum Abschluß noch zu bestimmen, welcher Kernel zum künftigen Start des Systems einzusetzen ist und wie das neue System aktiviert werden soll. Hierzu schlägt „YaST“ vor, den LILO zu installieren.

Da wir sicherstellen müssen, daß die vorliegenden Rechner nicht in Mitleidenschaft gezogen werden, dürfen wir **LILO nicht installieren**. Wir übergehen diesen Vorschlag und lassen das System starten (Möglicherweise erleben wir einen Systemabsturz – liegt an der erzwungenen Installation von DOS aus...)

Zuletzt wird das gerade installierte System (allerdings ohne LILO) gestartet. Dabei wird sofort „YaST“ gestartet, damit letzte Einrichtungen (Drucker, Netzwerk, Benutzer, etc) vorgenommen werden können.

Um auf unseren vorliegenden Rechnern Linux starten zu können, ist erneut das Beenden von Windows und eine Umschaltung in den MS-DOS Modus notwendig. Danach kann Linux so gestartet werden (darf natürlich als `.BAT` Datei realisiert sein):

```
C:\WINDOWS>cd \linux
C:\LINUX>loadlin eide01 root=/dev/hda6
```

c‘est tout.

Bei den ersten Starts unseres neuen Linux Systems ist etwas Vorsicht geboten, da wir noch keine Benutzer angelegt haben und uns als „root“ anmelden müssen. Da dieser Benutzer keinerlei Einschränkungen kennt, ist also hier eine besonders hohe Gefahr von Fehlern mit nicht rekonstruierbaren Situationen möglich.
Eine solche Konstellation sollte in der Praxis vermieden werden – Der erste Start sollte also zumindest dazu dienen, sich selbst als „normalen“ Benutzer anzulegen, der für die typischen arbeiten mit dem System verwendet wird. Nur bei Administrativen Arbeiten sollte der Benutzer „root“ eingesetzt werden.

Aufpassen!

6 Die Linux Shell

6.1 Anatomie von Kommandos

Nach dem Anmelden an Linux ergibt sich typischerweise ein Bildschirmaufbau, wie er in Abbildung 12 gezeigt ist (die fett gedruckten Passagen entsprechen den Eingaben des Benutzers).

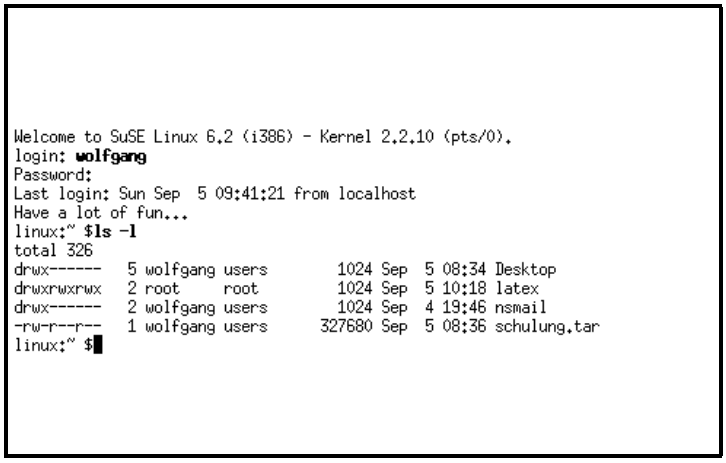

Abbildung 12: Anmeldung und ein Shell-Befehl

Meldet man sich an einem Text-Bildschirm an oder aktiviert man in der graphischen Oberfläche ein Terminal, so gelangt man zu einer *Shell*, einem Kommando-Interpreter, der Buchstabeneingaben wie einen Befehl interpretiert und sich dementsprechend verhält. Nur wenn ein gültiger Befehl eingegeben wurde, verhält sich eine shell vernünftig.

Die in der Shell eingegebenen Befehle haben grundsätzlich folgenden Aufbau:

- Das erste eingegebene Wort (zusammengeschriebene Buchstaben-Sequenz) ist der Name eins Kommandos. Die meisten Kommandos lesen sich wie kryptische Abkürzungen englischer Worte. Kommandos sind meist klein geschrieben
  z.B. `cat`, `ls`, `mkdir`, `ifconfig`, `SuperProbe`
- Sollen dem Kommando weitere Angaben mit auf den Weg gegeben werden, so ist mindestens ein Leerzeichen nach dem Kommando anzugeben, wonach dann die einzelnen *Parameter* (ebenfalls durch Leerzeichen miteinander getrennt) folgen.
  z.B. `datei.txt`, `/home/wolfgang/test.gif`
- Es gibt besondere *Optionen*, die mit einem Strich (`-`) beginnen und meist mit nur einem Buchstaben direkt nachfolgend angegeben werden. Solche Optionen sind keine Parameter im Sinne von Dateinamen, sondern schalten das Kommando in einen besonderen Modus mit anderen Auswirkungen, als der „normale“ Befehl
  z.B. `-l`, `-F`, `-c10`
- Alternativ lassen sich Optionen auch durch zwei Striche (`--`) gefolgt von einem ausgeschriebenen Schlüsselwort angeben. Hierbei wird die gleiche Wirkung wie in der Kurzfassung erreicht, doch der Lern- und Tippaufwand ist höher...
  z.B. `--long`, `--force`

Wurde ein Kommando eingegeben, so wird mit einem Druck auf den Zeilenschalter das Kommando abgesandt und ausgeführt. Sollte das Kommando nicht gültig sein, so wird eine Fehlermeldung ausgegeben. Falls ein Parameter falsch oder ungültig ist, gibt es ebenfalls eine Meldung...

Vielleicht mag diese Methode steinzeitlich erscheinen – Dieser Weg einen Computer zu bedienen hat zwar seinen Lernaufwand und macht nicht so viel Freude wie eine graphische Oberfläche, doch bei Problemen mit einem Rechner ist das meist der einzige Weg, der noch funktioniert... Deswegen werden wir sehr viel mit dieser Methode arbeiten, damit Ihnen eine Administration des Systems auch unter schwierigen Umständen möglich wird.

6.2 Die Eingabe-Aufforderung

Die nach dem login gestartete Shell gibt als Erkennungsmerkmal eine Eingabeaufforderung aus. Diese wird als *prompt* bezeichnet und besteht (sofern nicht verändert) aus

- dem Rechnernamen (z.B. `linux` oder `meinrechner`)
- gefolgt von einem Doppelpunkt (`:`)
- dem aktuellen Verzeichnis (z.B. `/usr/src`, `~` oder `~/briefe`)
- und dem eigentlichen Prompt-Symbol (`#`) oder (`$`)

Nach dieser Aufforderung lassen sich Kommandos eingeben. Eine Veränderung (kurzfristig oder dauerhaft) des Prompts ist jederzeit durchführbar und kann in den Start-Dateien des Systems oder eines Benutzers geändert werden.

6.3 Spezielle Zeichen

Die nachfolgenden Zeichen haben bei der Kommandoeingabe in der shell eine Sonderbedeutung und sollten daher weder in Kommando, noch in Operionen oder in Dateinamen vorkommen:

```
( ) < > { } | & ; ! # $ \ ` ' ~ "
```

6.4 Besondere Tasten bei der Eingabe (auszugsweise)

← →	Cursor bewegen
↑ ↓	scrollen in der History (bisherige Befehle)
←–	Rückwärts löschen eines Zeichens
⇌	eingegebenes Fragment (Kommando/Verzeichnis/Dateiname) vervollständigen
↩	Eingabezeile absenden
Strg + C	laufenden Job abbrechen
Strg + D	Ende eines Eingabe-Stroms
Strg + Z	laufenden Job anhalten und zur shell zurückkehren

6.5 Suchmuster

Zahlreiche Befehle gestatten die Angabe eines „Musters“ stellvertretend für mehrere Dateien, die dann nicht einzeln genannt werden brauchen – sie müssen lediglich auf das angegebene Suchmuster passen. Üblicherweise ist überall dort, so eine einzelne Datei als Parameter erlaubt ist, auch die Angabe eines Suchmusters gestattet. Hierbei gelten folgende Konventionen:

- Ein Stern (`*`) kann keinen, oder beliebig viele beliebige Zeichen treffen; der Stern ist auch mehrfach am Anfang, mitten oder am Ende eines Suchmusters erlaubt.
- Ein Fragezeichen (`?`) meint genau einen Buchstaben, der beliebig sein kann aber vorhanden sein muß.
- In eckige Klammern (`[ ]`) können die an dieser Stelle erlaubten Buchstaben oder Bereiche (`[a-z]`) angegeben werden, nach denen gesucht wird.
  Ist das erste Zeichen innerhalb der Klammerung ein Ausrufezeichen (`!`), so dürfen die in der Klammer genannten Buchstaben an dieser Stelle nicht auftreten.

Beispiele:

`*`	alle Dateien und Verzeichnisse
`*.*`	alle Dateien mit (mindestens) einem Punkt irgendwo (auch ganz am Anfang oder am Ende)
`brief?.doc`	trifft z.B. `brief1.doc`, `briefx.doc` nicht aber `brief11.doc`
`brief.???`	trifft z.B. `brief.doc`, `brief.txt` nicht aber `briefe.doc`
`brief[a-z]*`	trifft z.B. `briefa.doc`, `briefb` nicht aber `brief1.txt`
`kap[!1-4]`	Alle Dateien, denen nicht die Ziffern 1-4 nach „kap“ folgt.

6.6 Quoting – vereinfacht

Da Suchmuster von der Shell *expandiert* werden (tatsächlich erfolgt als Parameterübergabe die Liste aller Dateien) und bestimmte Sonderzeichen spezielle Bedeutungen haben, sind besondere Vorkehrungen zu

treffen, damit Sonderzeichen und -sequenzen als solche erhalten bleiben. Hier ein kleiner Auszug der Möglichkeiten (weitere in der Online-Hilfe, siehe Abschnitt 7.6 auf Seite 22):

- Der *Backslash* (\) sorgt für das Erhalten des nachfolgenden Zeichens oder – am Zeilenende gesetzt – für die Fortsetzung der aktuellen Zeile in der nächsten eingegebenen Zeile, nach deren Ende das eingegebene Kommando erst komplett ausgewertet wird.
- In *Anführungszeichen* (") gesetzte Zeichen werden mit Ausnahme der Sondersequenzen ($), (`) und (\) nicht interpretiert, sondern bleiben erhalten.
- Zwei weitere Anführungszeichen – (`) und (´), die nur für spezielle Zwecke vor allem bei der Programmierung der Shell eingesetzt werden. Deren Verwendung sollte in unseren Fällen nicht notwendig sein...

6.7 Darstellung von Kommandos im Text

Auf den folgenden Seiten wird folgende Syntax eingesetzt:

cmd	(Name des Kommandos fett)
[-l]	(eine wahlweise Option)
[-a|-b]	(eine weitere Option mit Alternative)
name	(symbolischer Parameter kursiv)

7 Reise durch das Dateisystem

7.1 Navigation zwischen Verzeichnissen

pwd	aktuelles Verzeichnis ausgeben

cd	Verzeichnis wechseln
[-]	zum zuletzt benutzten Verzeichnis gehen
[*verz*]	in dieses Verzeichnis wechseln – ohne: home)

mkdir	ein neues Verzeichnis erstellen
verz	Name des zu erstellenden Verzeichnisses

rmdir	ein (leeres) Verzeichnis löschen
verz	Name des zu löschenden Verzeichnisses

7.2 Umgang mit Dateien

ls	auflisten von Verzeichnisinhalten
[-a]	alle Dateien (auch unsichtbare!)
[-l]	weitere Datei-Informationen (siehe Abbildung 12)
[*name*]	Name des Verzeichnisses bzw. Suchmuster

In der normalen (kurzen) Form des Befehls `ls` werden alle geforderten Dateien (Suchmuster!) spaltenweise mit alphabetischer Sortierung je Spalte und gleichbreiten ausgeglichenen Spalten angezeigt. Meist wird der Typ der Datei durch unterschiedliche Färbung dargestellt.

- Verzeichnisse sind **blau**,
- Links (Verknüpfungen) sind **weiß**,
- Programme sind **rot** und
- Dateien meist **weiß** abgebildet.

In der langen Form des Befehls `ls` werden die Dateien jeweils zeilenweise dargestellt, wobei der Typ der Datei auch an anderen Merkmalen erkennbar ist. Hierbei erfolgt die Auflistung folgender Informationen:

- Datei-Typ und Berechtigungen. Der Typ der Datei ist in der ersten Spalte angezeigt und kann folgende Inhalte annehmen:
  - `-` kennzeichnet eine „normale“ Datei
  - `d` bezeichnet ein Verzeichnis (directory)
  - `l` wird bei Links eingesetzt
  - `b` macht ein Block-Gerät (Festplatte, Diskette, CD) erkennbar
  - `c` steht für ein Zeichen-Gerät (Terminal, Seriell, Parallel)
  - `f` wird bei FIFO-Puffer Dateien genutzt

  Die Berechtigungen werden durch die Buchstaben `rwx` oder ein Minuszeichen angegeben. Näheres dazu unter Abschnitt 10.2 auf Seite 27
- Eine Referenz-Zahl, die angibt, wie oft intern Verweise auf diese Datei erfolgen
- Name des Eingentümers dieser Datei
- Name der Gruppe, der die Datei gehört
- Größe der Datei in Byte
- Das Datum und Zeit der letzten Änderung
- Name der Datei – bei Links auch Angabe des Verweises

`cp`	Kopieren von Dateien
[`-r`]	Verzeichnisse mit Inhalt rekursiv kopieren
`name`	Dateiname/Suchmuster oder mehrere einzelne Dateien
`ziel`	Zieldatei oder Zielverzeichnis

`mv`	Bewegen oder umbenennen von Dateien
[`-i`]	interaktiv: vor dem Überschreiben fragen
`name`	Dateiname oder mehrere einzelne Dateien
`ziel`	Zieldatei oder Zielverzeichnis

`rm`	Löschen von Dateien (bzw. Verzeichnissen)
[`-i`]	interaktiv: vor dem Löschen fragen
[`-r`]	Verzeichnisse rekursiv mit Inhalt löschen (Gefahr!!)
`name`	Datei(en)/Suchmuster bzw. Verzeichnis(se) (`-r`)

`ln`	Erstellen eines Verweises (Link)
[`-i`|`-f`]	interaktiv (i) oder Zieldateien überschreiben (f)
[`-s`]	Symbolischen Link erzeugen; sonst Hard-Link
`name`	Ursprungs-Datei (oder mehrere bei Zielverzeichnis)
`ziel`	Ziel-Datei oder Ziel-Verzeichnis

Im Gegensatz zum Kopieren wird bei einem Link lediglich ein Verweis auf eine Datei hergestellt, die an ihrem ursprünglichen Ort verbleibt.

- Ein Symbolischer Link erzeugt eine neue Datei, deren Inhalt der Pfad zur Originaldatei ist.
- Bei einem Hard-Link entsteht lediglich ein Verzeichniseintrag, der auf das Original verweist.
- Jeder Zugriff auf den Link (außer Löschen bei Sym-Link) greift auf das Original zurück.

7.3 Datei-Inhalte ansehen

`cat`	Inhalt der Datei ausgeben
[*`name`*]	Name der Datei – ohne: stdin

`more`	Inhalt der Datei seitenweise anzeigen
[*`name`*]	Name der Datei – ohne: stdin

`less`	Inhalt der Datei blätterbar und scrollbar anzeigen
[*`name`*]	Name der Datei – ohne: stdin

7.4 Ein- Ausgabe-Umleitung

Diese Funktion macht Unix-ähnliche Betriebssysteme besonders leistungsfähig, denn mehrere Befehle können auf diese Weise miteinander „verkettet“ werden. Diesen Befehlen liegt ein recht simples Schema jedes einzelnen Prozesses zugrunde: Die Tatsache, daß jeder Prozeß eine *Standard-Eingabe* (stdin) und eine *Standard-Ausgabe* (stdout) sowie einen *Standard-Fehler* (stderr) Kanal besitzt, die geändert werden können.

`kommando <datei`	Standard-Eingabe von *datei* holen
`kommando >datei`	Standard-Ausgabe in *datei* umlenken
`kommando >>datei`	Standard-Ausgabe an *datei* anhängen
`kommando1 | kommando2`	Ausgabe von k1 geht an Eingabe von k2 (Pipe)
`kommando 0<datei`	Alternative zu `kommando <datei`
`kommando 1>datei`	Alternative zu `kommando >datei`
`kommando 2>datei`	stderr umleiten z.B. nach stdout (`&1`)

7.5 Filterkommandos

Im Zusammenhang mit Ein- und Ausgabeumleitung kommen immer wieder kleine Hilfsprogramme ins Spiel, die ihre Eingabe nach bestimmter Verarbeitung wieder ausgeben. Man spricht hier von Filterprogrammen. Hier eine kleine Auswahl:

`sort`	Eingabe sortieren und wieder ausgeben
[`-r`]	absteigend (rückwärts) sortieren
[`-f`]	Groß-/ Kleinschreibung ignorieren
[`+|- feld`]	Beginn (+) und Ende (-) der Sortierung (Feld-Nr)
[*datei*]	Dateiname – ohne: stdin

`head`	die ersten Zeilen einer Datei ausgeben
[`-anzahl`]	Anzahl der Zeilen
[*datei*]	Dateiname – ohne: stdin

`tail`	die letzten Zeilen einer Datei ausgeben
[`-anzahl`]	Anzahl der Zeilen
[*datei*]	Dateiname – ohne: stdin

`tee`	Datenstrom duplizieren in Datei und stdout
[`-a`]	Anhängen der Daten an die Datei
datei	Datei, in die dupliziert wird

`grep`	innerhalb des Datenstroms nach Text suchen
[`-e text`]	Ausdruck nach dem gesucht wird (meist in Anführungszeichen)
[*datei*]	Dateiname – ohne: stdin

7.6 Nützliche Kommandos

`find`	Suchen nach Dateien
pfad	Beginn der Suche an dieser Stelle
[`-name`]	Suche nach Namen
name	Name nach dem gesucht wird (am besten in Anführungszeichen gesetzt)

`man`	On-line Hilfefunktion (Beenden mit `Q`)
kommando	zu diesem Kommando

`apropos`	Suche einer Hilfe-Seite
such	nach diesem Suchbegriff

help	Hilfefunktion der shell
[*kommando*]	für dieses Kommando (ohne: alle Kommandos gelistet)

df	Auflistung der Mount-Points des Dateibaums

du	Benutzte Plattenbelegung erfragen – Angaben in KByte
[*name*]	wahlweise Datei- oder Verzeichnis-Angabe, sonst aktuelles Verz.

7.7 Shell-Variablen (Environment)

Jedes Programm (also auch die Shell) führen eine Reihe von Variablen mit, auf die laufende Programme zugreifen können. Beim Start eines Programmes werden alle Variablen vom startenden Programm (parent) kopiert und an den Kind-Prozeß übergeben. Diese Variablen lassen sich aus der Shell mit einigen einfachen Befehlen anzeigen oder ändern:

set Alle Shell-Variablen werden aufgelistet. Da diese Listen sehr lang sind ist eine Pipe mit `less` nicht verkehrt.

PS1="\w $" Die Variable PS1 (Befehls-Prompt) zeigt nun das aktuelle Verzeichnis gefolgt von einem Dollar-Zeichen.

PRINTER=picasso Die Variable PRINTER wird auf „picasso“ gesetzt.

echo "Drucker $PRINTER eingestellt" Zugriff auf eine Variable in einem Kommando. Alternativ kann der Name der Variablen in geschweifte Klammern ({ }) gesetzt werden, wenn danach kein Leerzeichen folgt, was als Trennzeichen dient.

8 Umgang mit Prozessen

8.1 Allgemeines

Jeder Prozeß besteht aus drei Teilen:

- dem eigentlichen Programm
- einem Bereich für Daten des Programms
- der Umgebung (environment) des Prozesses (in Form von Variablen)

Der Prozeß selbst besitzt zur Laufzeit im System eine Reihe von „Kenndaten“:

- eine eindeutige Prozeßnummer (PID)
- einen Elternprozeß (Prozeß der diesen startete)
- eine Benutzer- und Gruppen-Nummer (UID, GID)
- eine Priorität sowie Angaben zur Lauf- und Rechenzeit
- ein aktuelles Verzeichnis
- eine Liste an geöffneten Dateien (auch stdin, stdout, stderr)

8.2 Hintergrund-Jobs

Ein Job kann auf zwei Arten in den Hintergrund gestellt werden:

- durch Anhängen eines „&“ an den Jobnamen (genauer: an das Ende des Kommandos mit Parametern)
- durch Eingabe von Strg + Z bei laufendem Job (Job ist suspendiert, d.h. wartend)

und kann anschließendes weiterverarbeitet werden. ein paar Kommandos zur Prozeßverwaltung:

ps	Auflistung von Prozessen
[-l]	umfangreiche Anzeige – mehr Informationen
[-a]	Alle Prozesse – auch anderer Benutzer
[-x]	auch Prozesse ohne Terminal (daemons)

kill	Signal an einen oder mehrere Prozesse senden
-signal	Nummer oder Name des Signals (ohne: SIGTERM)
prozess	PID des Prozesses (evtl. mehrere)

bg	angehaltenen Job im Hintergrund weiterverarbeiten
prozess	PID des Prozesses oder laufende Nr. (%1,%2,...)

fg	angehaltenen Job im Vordergrund weiterverarbeiten
prozess	PID des Prozesses oder laufende Nr (%1,%2,...)

jobs	Übersicht über die laufenden / angehaltenen Jobs

nice	Beeinflussen der Priorität des folgenden Prozesses
[-n *zahl*]	der nice-Faktor (-20=langsam...19=schnell)
kommando	dieses Kommando wird so ausgeführt

nohup	Wenn Elternprozeß stirbt, läuft der folgende weiter
kommando	dieses Kommando wird so ausgeführt

9 Wie geht es weiter (Hausaufgabe)

9.1 Kommandos in Folge ausführen

kommando1 && kommando2 Kommando2 wird nur dann ausgeführt, wenn kommando1 ohne Fehler lief (status=0).

kommando1 || kommando2 Kommando2 wird nur dann ausgeführt, wenn kommando1 einen Fehler produzierte (status≠0)

kommando1 ; kommando2 beide Kommandos werden unabhängig von ihrem Status-code nacheinander ausgeführt

(kommando1 ; kommando2) > datei beide Kommandos in Folge innerhalb einer eigenen shell mit gemeinsamer Ausgabe-Umleitung ausführen

{ **kommando1 ; kommando2 ;** } **> datei** beide Kommandos in Folge ohne eigene shell mit gemeinsamer Ausgabe-Umleitung ausführen

9.2 Erleichterungen durch die alias-Funktion

per alias kann eine Buchstabenfolge, die in einer Eingabezeile eines Kommandos auftaucht, in eine andere Buchstabenfolge umgeändert werden; damit lassen sich häufig einzugebende Kommandos vereinfachen, z.B.:

alias md=mkdir Anstelle von `mkdir` kann ab jetzt `md` verwendet werden.

9.3 Weitere Informationen

...finden sich in Massen in der man-Page zur `bash`-Shell. Hier ist genauestens Beschrieben, welche Shell-Variablen welchen Zweck haben, wie man alle Parameter verändert, welche Tastenkombinationen was erreichen u.v.m...

10 Benutzerverwaltung

Zur Vergabe von Rechten muß jeder Zugriff auf eine Resource (Datei, Verzeichnis, Gerät,...) auf Erlaubnis hin geprüft werden. Zu diesem Zweck ist vor dem Arbeiten mit Linux eine entsprechende Anmeldung erforderlich. Bei der Anmeldung ergibt sich:

- ein Benutzereintrag sowie
- eine Gruppenzugehörigkeit

Intern werden Benutzer und Gruppen als Zahlen codiert, die für die entsprechenden Prüfungen herangezogen werden. Jeder Benutzer ist zunächst einer Gruppe fest zugeordnet, kann aber (Erlaubnis vorausgesetzt) zu einer anderen Gruppe wechseln.

Für jeden Benutzer muß ein Eintrag (d.h. eine Zeile) in den Dateien `/etc/passwd` (und evtl. in `/etc/shadow`) vorliegen, damit eine Anmeldung möglich ist. Die Gruppe, der der Benutzer angehört, muß in `/etc/group` /ung evtl. in `/etc/gshadow`) eingetragen sein.

Neu anzulegende Benutzer sollten eine UID>100 und neue Gruppen sollten GID>100 besitzen, da Zahlen unterhalb dieser Grenzen für interne Zwecke reserviert sind.

Benutzer- und Gruppeneinträge lassen sich mit einem Editor oder – komfortabler mit einigen Hilfsprogrammen erledigen.

useradd	Benutzer hinzufügen
[-c *komm*]	Kommentarfeld-Eintrag
[-d *home*]	Home-Verzeichnis; default: /home/*login-name*
[-e *ablauf*]	Ablaufdatum YYYY-MM-DD
[-f *tage*]	Tage bis Sperrung nach Ablauf
[-g *gruppe*]	Gruppe
[-G *gruppen*]	alternative Gruppe(n)
[-m]	Erstellen des Home-Verzeichnisses
[-k *skel*]	Daten aus skel kopieren
[-p *pw*]	das verschlüsselte Passwort; default: gesperrt
[-s *shell*]	Shell
[-u *uid*]	UID für den Benutzer
[-o]	erforderlich, wenn UID nicht eindeutig
login	Login-Name

usermod	Benutzerdaten ändern – fast wie oben

userdel	Benutzerdaten löschen
[-r]	Home-Verzeichnis samt Inhalt löschen
login	Login-Name

chsh	Ändern der Login-Shell eines Benutzers
[-s *shell*]	die Shell (muß in /etc/shells verzeichnet sein)
login	name des Benutzers; ohne: eigenes Benutzerkonto

passwd	Ändern eines Passwortes
login	Benutzername; ohne: eigenes Benutzerkonto

10.1 Gruppen verwalten

Standardgemäß werden alle neu anzulegenden Benutzer der Gruppe *users* (GID=100) zugeordnet. In einer Netzwerkumgebung mag es jedoch zur sinnvollen Vergabe von Rechten geeineter sein, mehrere Benutzer mit ähnlichen Rechten jeweils einer eigenen Gruppe zuzuordnen, die dann auf bestimmte Verzeichnisse innerhalb des Verzeichnisbaums besondere Rechte – oder Einschränkungen erteilt bekommen.

groupadd	Hinzufügen einer Gruppe
[-g *gid*]	GID der Gruppe
[-o]	erforderlich wenn GID nicht eindeutig ist
gruppe	Name der Gruppe

groupmod	Verändern von Gruppen-Informationen
[-g *gid*]	GID der Gruppe
[-o]	erforderlich wenn GID nicht eindeutig ist
gruppe	Name der Gruppe

groupdel	Löschen einer Gruppe
gruppe	Name der zu löschenden Gruppe

10.2 Zugriffsrechte regeln

chmod	Ändern von Zugriffsrechten
[-c|-v]	Ausgeben der geänderten (c) oder aller Dateien (v)
[-f]	keine Fehlermeldung(en) ausgeben
[-R]	Rekursiv alle Unterverzeichnisse durchlaufen
recht	Nummer oder Bezeichnung des Rechts
datei(en)	Datei- oder Verzeichnisname(n)

Intern werden die Zugriffsrechte über Zahlen gespeichert, von denen die ersten 12 Bit entsprechende Rechte setzen oder verbieten. Die einzelnen Bits lassen sich mit einzelnen Buchstaben oder Oktalzahlen zu- oder Abschalten. Die Bedeutung der einzelnen Bits sind in Tabelle 3 zusammengefaßt.

Bit	Oktal	Abk.	Bedeutung
12	4xxx	u⊗s	User-ID des Eigentümers bei Ausführung
11	2xxx	g⊗s	Group-ID der Gruppe bei Ausführung
10	1xxx	t	Sticky-Bit – andere haben keinen Zugriff
9	x4xx	u⊗r	Leseberechtigung für Besitzer
8	x2xx	u⊗w	Schreibberechtigung für Besitzer
7	x1xx	u⊗x	Ausführberechtigung für Besitzer
6	xx4x	g⊗r	Leseberechtigung für Gruppe
5	xx2x	g⊗w	Schreibberechtigung für Gruppe
4	xx1x	g⊗x	Ausführberechtigung für Gruppe
3	xxx4	o⊗r	Leseberechtigung für Andere
2	xxx2	o⊗w	Schreibberechtigung für Andere
1	xxx1	o⊗x	Ausführberechtigung für Andere
		a⊗?	Alle Benutzer: wie ugo
		?⊗X	x falls schon einmal vorhanden
		⊗	+ hinzufügen, - löschen, oder = alleine setzen

Tabelle 3: Zugriffsrechte

chown	Eigentümer von Dateien ändern
name	Benutzername oder -nummer
datei(en)	Datei- oder Verzeichnisname(n)

chgrp	Gruppenzugehörigkeit von Dateien ändern
gruppe	Gruppenname oder -nummer
datei(en)	Datei- oder Verzeichnisname(n)

11 Anpassungen am System

11.1 Anpassungen mit „YaST“

Ein Teil der notwendigen Änderungen an einem laufenden Linux-System läßt sich mit dem Installationsprogramm „YaST“ vornehmen, das vom Benutzer „root“ per Eingabe von `yast` jederzeit aufgerufen werden kann. Wie in Abbildung 13 zu sehen ist, bieten sich hier zahlreicht Möglichkeiten. Die Benutzung ist – von einigen Kleinigkeiten abgesehen – ziemlich geradlinig.

`/sbin/yast`

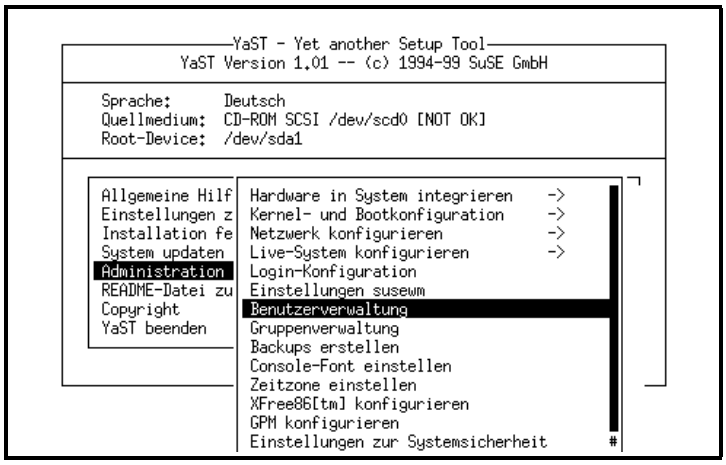

Abbildung 13: Administrations-Möglichkeiten von YaST

11.2 Manuelle Anpassungen – allgemein

Um manuell in ein laufendes System eingreifen zu können, gilt es, jeweils die passende der (leider sehr zahlreich vorhandenen) Konfigurations-Dateien zu verändern. Alle Konfigurations-Dateien sind im Verzeichnis `/etc` untergebracht und tragen meist einen Namen, der mit dem in Verbindung stehenden Programm etwas gemein hat. Eine kleine Übersicht ist der Tabelle 6 zu entnehmen.

⇒ `/etc/...`

Zur Änderung dieser Dateien ist die Benutzung eines Text-Editors erforderlich. Hier stehen etliche Pakete zur Verfügung, daher hier eine kleine Auswahl:

- Der Klassiker unter den Editoren ist **vi**, von dem es unter Linux etliche Varianten gibt. Für den Einstieg etwas hinderlich ist die Tatsache, daß er zwischen einem Kommando- und einem Eingabe-Modus unterscheidet.
  Einige Kommandos des vi sind in Tabelle 4 zusammengefaßt.
- Die große Konkurrenz dazu ist der **emacs**, der sich immer im Editier-Modus befindet, jedoch ebenfalls durch seine zahlreichen Optionen am Anfang sehr gewöhnungsbedürftig ist.
  Tabelle 5 listet einige Befehle des Emacs Editors auf.
- Ein Wordstar-ähnlicher Editor ist **joe**, der eher simpel ist, aber eben die Befehle eines „Klassikers“ benutzt – vielleicht eine Hilfe für alle TURBO-Pascal Fans...
- Extrem primitiv ist der Editor **edy** – kann nicht viel, dafür besitzt er aber einfachste Tastenkombinationen und ein Menü mit diversen Sonderbefehlen, was ihn zumindest am Anfang leicht in der Handhabung macht.

11.3 Manuell Pakete einspielen

Manche Programme sind leider nur auf manuellem Wege in ein Linux-System zu bekommen. Dabei treten eine Reihe verschiedener Datei-Typen auf, die jeweils unterschiedlich zu behandeln sind:

Endung	Erstellt mit	Entpacken mit
`.Z`	`compress`	`uncompress`
`.gz`	`gzip`	`gunzip`
`.tar`	`tar cf`	`tar xf`
`.tgz`	`tar cfz`	`tar xfz`
`.rpm`	`rpm`	`rpm`

Befehle im Befehlsmodus (Modus beim Aktivieren)	
`i`	Eingabemodus: Zeichen am Cursor einfügen
`r`	Ersetzen des Zeichens am Cursor
`R`	Eingabemodus: Ersetzen des bestehenden Textes
`o` | `O`	Eingabemodus: nach(o) oder vor(O) aktueller Zeile einfügen
[*zahl*]`x`	aktuelles oder *zahl* Zeichen löschen
[*zahl*]`dd`	aktuelle oder *zahl* Zeilen löschen
`J`	Folgezeile am Zeilenende anfügen
`u`	letzten Befehl rückgängig machen
`ZZ`	Editieren beenden und Datei schreiben
`:q[!]`	Editor verlassen – zwangsweise (!)

Befehle im Eingabemodus	
Esc	Beendet den Eingabemodus
Return	Neue Zeile
Rück	letztes Zeichen löschen

Tabelle 4: Einige (wenige) Befehle des vi

???	intuitive Editier-Kommandos
`Strg`	Beginnen einer Kommando-Sequenz (C-)
`Alt`	Beginnen einer Kommando-Sequenz (M-)
`Esc`	Beginnen einer Kommando-Sequenz (M-)
`C-x C-c`	Beenden des Emacs-Editors
`C-x C-s`	Speichern der aktuellen Datei
`C-x C-f`	Öffnen einer neuen Datei
...	...

Tabelle 5: Ein paar Informationen zum emacs

Die hierbei zu benutzenden Kommandos besitzen folgende Syntax:

`compress`	Komprimieren von Dateien
`datei`	Name der zu komprimierenden Datei

`uncompress`	Dekomprimieren von Dateien
`datei`	Name der zu entpackenden Datei

`gzip`	Komprimieren von Dateien
[`-` *`zahl`*]	Qualität der Kompression 1...9
`datei`	Name der zu komprimierenden Datei

`gunzip`	Dekomprimieren von Dateien
`datei`	Name der zu entpackenden Datei

`tar`	Packen oder entpacken von Archivdateien
[`c`|`x`|`t`]	Erzeugen(c), Extrahieren(x) oder Ansehen(t) von Archiven
[`f`]	Dateiname folgt – sonst Gerätename (Bandlaufwerk)
[`z`]	zusätzlich (de)komprimieren
`datei`	Name der zu entpackenden Archivdatei
[*`dateien`*]	Packen: Dateien sind anzugeben

`rpm`	Installation aus einem RPM-Archiv
[`-i`]	Installations-Modus
`datei`	Name der zu installierenden Archivdatei

/etc/passwd	Benutzer-Namen, UID, Passworte usw.
/etc/group	Gruppen-Namen, GID usw.
/etc/shadow	Benutzer-Datei des Shadow-Systems
/etc/gshadow	Gruppen-Datei des Shadow-Systems
/etc/conf.modules	Angaben über ladbare Kernel-Module
/etc/ld.so.conf	Verzeichnisse der dynamischen Libraries
/etc/lilo.conf	Einstellungen des Linux-Loaders (LILO)
/etc/syslog.conf	Einstellungen des Syslog-Daemon
/etc/DIR_COLORS	Farb-Information und Optionen für das ls Kommando; Alternative: ~/.dircolors
/etc/fstab	gibt alle Mount-Points der verschiedenen Dateisysteme sowie Berechtigungen für andere Benutzer an
/etc/inittab	Zuordnung zwischen den Runlevels und deren Aktionen
/etc/inputrc	Vereinbarung von Tastenkombinationen für die Eingabezeile
/etc/issue	Begrüßungsmeldung beim Login
/etc/printcap	Verwalten der zur Verfügung stehenden Drucker
/etc/profile	enthält ein Start-Script für jeden Benutzer, das zum Start einer Shell aktiviert wird
/etc/shells	Shells, die ein Benutzer wählen kann
/etc/rc.d/	Enthält Start- und Stop-Scripte für die Runlevels
/etc/skel/	Standard-Dateien für neue Benutzer

Tabelle 6: Einige Systemdateien

11.4 Drucker einrichten und benutzen

Lokal an einem Linux-PC angeschlossene Drucker werden zumeist an einer der parallelen Schnittstellen angeschlossen. Diese werden nach ihrer Hardware-Adresse fest einem Namen zugeordnet und nicht – wie unter MS-DOS – immer mit einer logischen Nummer versehen:

Gerät	DOS-Name	IRQ	IO-Port	Bemerkung
/dev/lp0	LPT3:	5	3BC - 3BE	Hercules Karte
/dev/lp1	LPT1:	7	378 - 37A	meist on-board
/dev/lp2	LPT2:	5	278 - 27A	meist on-board

In der Regel werden Drucker vom BSD-Warteschlangen-System aus angesteuert, welches per „Yast“ auch bequem eingerichtet werden kann. Zu diesem Zweck werden Druckerwarteschlangen eingerichtet, von denen es auch mehrere für einen Drucker geben kann. Für jede Warteschlange muß es geben:

Paket: lprold[n]

- einen Eintrag in der Datei /etc/printcap (Definition der Warteschlange mit einem Namen und zahlreichen Parametern)
- ein Spool-Verzeichnis (meist unter /var/spool/lpd/), in dem die Druck-Dateien abgelegt werden

Bei SuSE-Linux sorgt zunächst das Programm apsfilter für die korrekte Behandlung aller eingehenden Druckjobs. Dies wird dadurch erreicht, daß die Dateien auf ihren Inhalt hin untersucht. Die Weiterverarbeitung wird an ein geeignetes Programm weitergegeben, das eine PostScript-Datei ersellt. Diese wird dann von gs (Postscript-Interpreter GhostScript) in eine direkt vom Drucker druckbare Datei umgesetzt. Diese wird dann gedruckt.

Paket: aps[ap]

Paket: gs[ap]

Wahlweise kann apsfilter auch über ein eigenes SETUP Programm eingerichtet werden, was nach kurzer Einarbeitung meist problemlos funktioniert. Die mit diesem Programm erstellte Konfigurationsdatei /etc/printcap sollte dann nicht von Hand geändert werden...

/var/lib/apsfilter/SETUP

Die Ansteuerung von Druckern im Netzwerk erfordert meist das „händische“ editieren der Text-Datei /etc/printcap und manuelles Anlegen des Spool-Verzeichnisses.

Zur Ansteuerung (bereits eingerichteter) Drucker gibt es folgende Kommandos:

lpr	Senden einer Datei an eine Drucker-Warteschlange
[-P *drucker*]	Angabe der Warteschlange (ohne: PRINTER-Variable)
datei	zu druckende Datei (ohne: stdin)

lpq	Anzeige der Jobs einer Drucker-Warteschlange
[-P *drucker*]	Angabe der Warteschlange (ohne: PRINTER-Variable)

lprm	Entfernen eines Jobs einer Drucker-Warteschlange
[-P *drucker*]	Angabe der Warteschlange (ohne: PRINTER-Variable)
nummer	interne Nummer des Jobs (zu erfahren per lpq)

lpc	Steuerungen vornehmen an Warteschlangen

12 Die graphische Oberfläche X-Windows

12.1 Konzeptionelles

Um in den Genuß einer graphischen Oberfläche zu kommen, muß auf dem Linux-Rechner zunächst einen *X-Server* installiert haben, der sich um die gesamte Steuerung des Grafik-Bildschirms kümmert. Der X-Server ist ein speziell auf die Grafikkarte (Chipsatz!) angepaßtes Programm, welches zusätzliche Parameter per Installation bekommen muß, damit aufgrund der Kombination Grafikkarte - Monitor eine korrekte Anzeige möglich ist.

⇒ X-Server

Vorsicht! Zerstörungsgefahr

Der Server hat dabei grob folgende Aufgaben:

- Bildschirm, Maus und Tastatur in „Besitz“ nehmen
- Kommunikation mit den X-Clients, d.h. den Anwendungsprogrammen in einem speziellen X-Protokoll (wahlweise per Netzwerk oder lokal)
- Zur Verfügung stellen einer Programmierschnittstelle, derer sich die Clients bedienen, um graphische Anzeigen zu erreichen
- Verwalten aller graphischen Ressourcen (Fenster, Schriften, . . .)
- Zeichen-Kommandos ausführen

⇒ X-Clients

Da der Server „nur“ das blanke graphische Werkzeug darstellt, benötigt eine graphische Oberfläche noch sogenannte *Window-Manager*, die sich um die eigentliche Oberfläche kümmern. Davon gibt es reichlich; die populärste, da anderen Systemen am ähnlichsten und besonders intuitiv bedienbar ist das *K Desktop Environment*, dessen Entwicklung auch zum Großteil aus deutschen Landen stammt.

⇒ Window-Manager

⇒ KDE

12.2 Installation von X-Windows

Zunächst sind folgende Pakete zu installieren:

- Das X11 Grundpaket
- evtl. 100dpi Bildschirm-Schriften
- evtl. das XF86Setup Programm
- bei SuSE evtl. das komfortable Setup-Programm sax
- auf jeden Fall den X VGA-Server (wird von sax und XF86Setup benötigt)
- den Server, der zur eigenen Graphik-Karte paßt (Grafik-Chip!)
- mindestens einen Window-Manager (oder KDE)
- entsprechende X-Anwendungsprogramme

Nach dieser Arbeit geht es an die Konfiguration, die im VGA-Modus per XF86Setup, im Textmodus mit xf86config oder am komfortabelsten mit sax erfolgt. Alle Konfigurationsmöglichkeiten haben eine Gemeinsamkeit: sie erstellen eine Datei (/etc/XF86Config), in der die aus der Konfiguration gewonnen Werte abgelegt sind.

XF86Setup

> **Äußerst Ernst zu nehmen sind die Fragen nach dem Bildschirm, da hier eine echte Zerstörungsgefahr steckt, wenn der Bildschirm übertaktet werden sollte.**

Nach getaner Konfiguration kann X-Windows erstmals mit startx aktiviert werden. Ein graphischer Bildschirm (bei KDE: ein entsprechender Desktop) taucht auf.

startx

Den geringsten Widerstand erhalten wir bei der Installation durch folgende Schritte:

- Im Setup-Programm „Yast“ stellen wir unter der Hardware-Option die Maus auf „PS/2“ ein und aktivieren die Mausunterstützung. Dadurch steht im Folgenden für die Maus ein Gerät namens `/dev/mouse` zur Verfügung.
- Damit sich die Grafikkarte voll entfalten kann benötigen wir aus der Serie „X Server“ den Server „Mach 64“.
- Anschließend starten wir das Konfigurations-Programm `XF86Setup`, welches im VGA-Modus und bereits mit Maus-Unterstützung nachfolgend Fragen stellt, die wir geduldig beantworten.
- Zum Schluß kann X mit dem Kommando `startx` gestartet werden – Ein uns fast bekannter Desktop taucht auf...

12.3 Arbeiten mit X-Windows

Einige Tastenkombinationen erleichtern die Arbeit mit X-Windows:

Strg + Alt + ←	Abbrechen des X-Servers
Strg + Alt + F1-6	wechsel zur Textconsole 1 bis 6
Strg + Alt + F7	zurück zu X-Windows
Strg + Alt + ZF+	Wechseln zur nächsten Auflösung
Strg + Alt + ZF-	Wechseln zur vorherigen Auflösung

Wahlweise kann X-Windows bereits beim Hochfahren des Rechners aktiviert werden (nach erfolgreichem Test!!); hierzu muß lediglich der Runlevel auf 3 gesetzt werden – oder per „YaST“ die Login-Konfiguration auf „graphisch“ gesetzt werden. Ein typischer KDE-Desktop ist in Abbildung 14 zu sehen.

⇒ graphisches Login

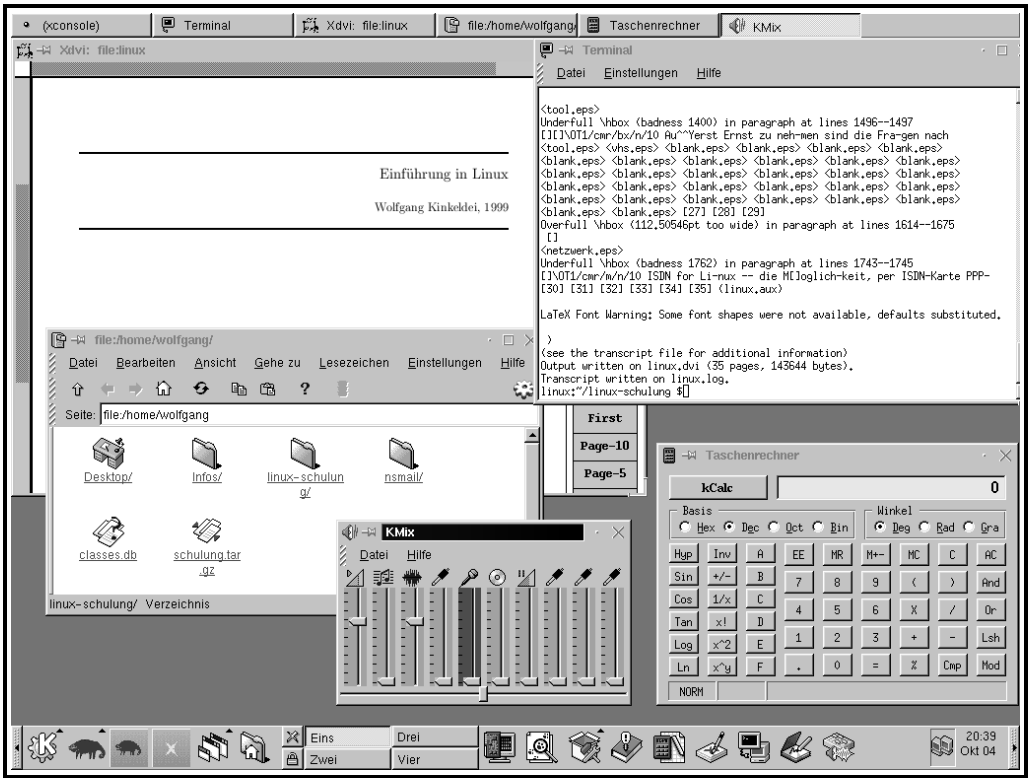

Abbildung 14: der KDE-Desktop mit einigen Programmen

13 Linux im Netzwerk

13.1 TCP/IP Grundlagen

Das TCP/IP (Transmission Control Protocol / Internet Protocol) wurde eigens für große Netzwerke entwickelt. Da es besonders auf UNIX Systemen zum Einsatz kam, ist Linux besonders mit diesem Protokoll besonders befähigt.

TCP/IP-Netzwerke haben einige wissenswerte Eigenschaften:

- Jeder Rechner (genauer: jeder Knotenpunkt) erhält eine eigene sog. *IP-Nummer*, die immer aus vier Zahlen im Bereich von 0...254 liegen muß, z.B.: `192.168.0.5`. Alle verwendeten IP-Nummern müssen weltweit eindeutig sein (Ausnahme: private IP Nummern)
  private Nummern sind z.B. `192.168.0.0` – `192.168.255.255` ⇒ `/etc/hosts`
- Die Adresse `127.0.0.1` ist für den eigenen Rechner (*localhost*) reserviert und liegt in einem (pseudo-)Netzwerk für lokalen Datentransfer
- Von der ersten Zahl abhängig ist nur die erste, die ersten beiden oder die ersten drei Zahlen die *Netzwerk-Nummer*, z.B. `192.168.0` – die restliche(n) Zahlen die *Knoten-Nummer*, z.B. `5`
- Innerhalb eines Netzes lassen sich Subnetze bilden, was durch eine weitere Zahl, die *Subnetz-Maske*, z.B. `255.255.255.0` erreicht wird. (dabei wird die Subnetz-Maske mit der IP-Nummer Und-Verknüpft, die verbleibenden Zahlen sind die [Sub]Netzwerk-Nummer)
- Alle Rechner innerhalb des selben Netzwerkstranges haben vereinbarungsgemäß eine gemeinsame Subnetz-Maske und gehören somit zum gleichen Netzwerk
- Die Empfänger-Adresse, die nur aus Binär-1 besteht (z.B. `255`) ist die *Broadcast-Adresse*. Wenn eine Netzwerk-Nachricht an diese Adresse gerichtet wird, ist sie für alle Teilnehmer des Netzwerkes interessannt.
- Die einzelnen Protokolle von TCP/IP richten ihre Netzwerk-Nachrichten, die in einzelnen Teilen (*Pakete*) versandt werden, nicht nur an einen bestimmten Empfänger, sondern auch an einen bestimmten *Port* (=Anschluß); dabei ist jedem Protokoll ein bestimmter Port zugeteilt. Man spricht auch von einer *Socket* anstelle von Port. ⇒ `/etc/services`
- Alle Netzwerk-Nachrichten, die an Teilnehmer außerhalb des eigenen Netzwerks gerichtet sind, werden an ein *Gateway* übermittelt, welches dann die Nachrichten weiterleitet. Ist der Adressat nicht bekannt, so wird die Nachricht an das *Default-Gateway* gesandt. Bei diesem Weiterleitungs-Vorgang spricht man auch von *Routing*.
- Die im Internet üblichen Namen wie `www.linux.de` sind nur eine menschlich verständlichere Form für IP-Adressen. Diese *Domain-Namen* werden vom *Name-Resolver* innerhalb des Kernels in IP-Adressen umgewandelt. Möglicherweise bedient dieser sich eines oder mehrerer *Domain Name Server*. Daher ist bei der Einrichtung eines ans Internet angeschlossenen Rechners meist eine solche Angabe erforderlich.

13.2 Netzwerk Interna

Damit Linux mit dem Netzwerk umgehen kann, sind einige Geräte angelegt bzw. noch anzulegen, hinter denen entsprechende Programme (entweder im Kernel oder als Daemons) für die Ansteuerung der diversen Übertragungsmedien stecken. Die häufig verwendeten Bezeichnungen für gängige Medien sind in Tabelle 7 aufgelistet. ⇒ devices

device	Verwendungszweck
lo	lokale Loopback-Adresse (meist 127.0.0.1)
eth0	erste Ethernet-Steckkarte
eth1	zweite Ethernet-Steckkarte, usw.
ippp0	erstes ISDN Interface für PPP Zugriff
ppp0	erstes Analog-Modem für PPP Zugriff
plip0	erste Schnittstelle für Parallel-Line IP
sl0	erste Serial-Line IP Schnittstelle

Tabelle 7: ein paar übliche Netzwerk-devices

Eine weitere Hürde beim Umgang mit dem Netzwerk-Subsystem von Linux sind zahlreiche Daemon-Programme, die von – teilweise mehreren – Text-Dateien gestuert ihre Arbeit verrichten. Nur wenn dir richtigen dieser Daemons installiert und per Installationsprogramm oder von Hand richtig konfiguriert

sind, wird das Netzwerk-System korrekt arbeiten. Um hier zumindest einen Anhaltspunkt zu erhalten, sind in Tabelle 8 die wichtigsten Dienste und deren Dateien zusammengefaßt. Allerdings enthält diese Tabelle nur einen wirklich essentiellen Teil der Möglichkeiten, die sich bieten und nur wenige Details. Daher ist die Lektüre der man-Pages oder der entsprechenden HowTos unumgänglich.

Etwas Vorsicht ist bei einigen dieser Dateien geboten, da teilweise Installationsprogramme (z.B. „YaST“) hier ebenfalls eingreifen. Daher ist nach einer zusätzlichen Installation manche Konfigurations-Datei erneut auf ihren vorherigen Zustand „verbogen“.

13.3 Einfache Netzwerke aufbauen

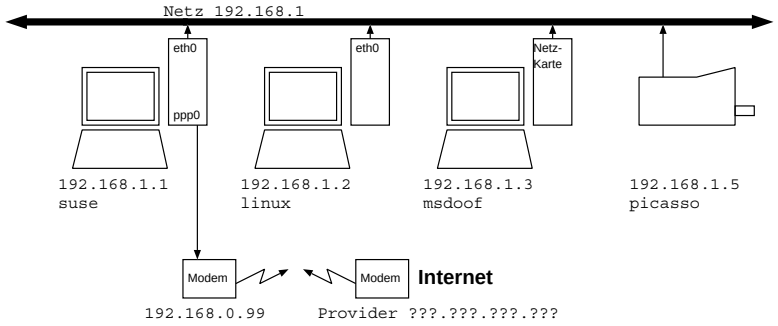

Abbildung 15: Aufbau eines einfachen Netzwerks

Wie in Abbildung 15 zu sehen ist, erhält jeder Rechner (und Drucker) eine eigene IP-Nummer, über die ein Zugriff auf die jeweiligen Geräte erfolgen kann. Alle Rechner im gleichen Netzwerkstrang haben als Netzwerk-Anteil die gleiche Nummern-Folge (Netzwerk-Nummer `192.168.1`).
Zu beachten gilt es, bei allen Rechnern (außer „suse“) die IP-Nummer `192.168.1.1` als default Gateway für Zugriffe auf andere Netze als das eigene einzurichten und am Rechner „suse“ als default Gateway `192.168.0.99` (Interface `ppp0`) zu definieren, damit alle einen Zugriff auf das Internet haben.

Die meisten dieser Einstellungen werden sicherlich im Installationsprogramm per „Administration“ vorgenommen; dennoch könnten einige der nachfolgenden Kommandos (Fast alle möglichen Parameter unterschlagen...) zur Fehlersuche ganz hilfreich sein:

`ifconfig`	Einstellen oder Abfragen eines Netzwerk-Interfaces
[*`device`*]	Abzufragendes Device – ohne: alle Devices

`route`	Einstellen oder Abfragen der Kernel Routing-Tabelle
[`-n`]	Numerische Ausgabe anstelle von IP-Nummern (wo möglich)

`ping`	Senden von „Testsignalen“ mit Empfangsbestätigung
[*`ziel`*]	an diese Zieladresse (IP oder Name)

`netstat`	Zahlreiche Informationen zum Netzwerk-Status
[`-n`]	Numerische Ausgabe anstelle von IP-Nummern (wo möglich)
[`-t|-u|-w|-x`]	nur TCP(t), UDP(u), RAW(w) oder UNIX(x) Sockets
[`-a`]	Auflistung aller (auch nicht aktiver) Sockets mit Statusangabe

`netstat`	Weitere Informationen zum Netzwerk-Status
[`-i`]	Statistik über Netzwerk-Interfaces anzeigen
[`-a`]	Auflistung aller (auch nicht aktiver) Schnittstellen

13.4 Ausblick: Was geht noch alles?

Alle Netzwerk-Möglichkeiten aufzuzählen wäre sicher unmöglich. Doch ein paar typische Anwendungsmöglichkeiten sollten nicht fehlen. Daher sind in Tabelle 9 einige Pakete mit ihrem Verwendungszweck aufgelistet.

Name	Aufgabe
/sbin/ifconfig	wird benötigt (i.d.R. von Startscripten), um die einzelnen Netzwerk-Interfaces einzurichten und deren Parameter festzulegen. Ohne Parameter gestartet liefert es Informationen über die zur Verfügung stehenden (und installierten) Schnittstellen
(Name Resolver)	dieses fest im Kernel untergebrachte Programm sorgt für die Umsetzung von Rechnernamen (z.B. `mein.linux.de`) in IP-Adressen.
`/etc/resolv.conf`	Ist die zentrale Konfigurations-Datei des Resolvers. Sie legt einige Standards sowie den/die zu benutzende(n) Domain-Name-Server (DNS) fest.
`/etc/hosts`	In dies Datei werden typischerweise die lokalen Rechner eingetragen. Sie ist eine der Säulen des Resolvers. Bei kleinen Netzen ohne DNS genügt diese Datei, die dann natürlich auf jedem Rechner alle Rechner des Netzwerkes erfassen muß.
`/etc/host.conf`	Legt fest, in welcher Reihenfolge (Host-Datei und DNS) nach Namen gesucht werden soll.
/sbin/inetd	„lauscht“ an bestimmten IP-Ports (Sockets) bis eine Kontaktaufnahme erfolgt. Dann wird ein (vom Port abhängiges) entsprechendes Programm aktiviert, welches die weitere Arbeit übernimmt.
`/etc/inetd.conf`	enthält eine Liste der einzelnen „abzuhörenden“ Ports mit der dazugehörigen Aktivität. Soll ein bestimmter Dienst nicht mehr zur Verfügung stehen, so ist die entsprechende Zeile auszukommentieren.
`/etc/services`	vereinbart für jeden prinzipiell verfügbaren (unabhängig einer Erlaubnis) Service eine Port-Nummer mit dazugehörigem Namen. In der Datei `/etc/inetd.conf` wird auf die hier vereinbarten Namen der Services zurückgegriffen.
(Kernel Routing Tabelle)	Der Kernel muß entscheiden, über welches Netzwerk-Interface ein Netzwerkpaket weitergegeben werden soll. Dies setzt natürlich mindestens zwei Interfaces voraus (`lo` und `eth0` wären zwei).
/sbin/route	dient zum Bearbeiten der im Kernel gespeicherten *statischen* Routingtabelle und kann deren Inhalt auch anzeigen lassen.
`/etc/route.conf`	**Nur bei SuSE-Linux:** Hier werden die standardmäßig in die Kernel Routing Tabelle einzutragenden Routen abgelegt. Diese werden bei jedem Systemstart gesetzt.
(allgemeine Dateien)	Nachfolgend noch einige Dateien, die im Einzelfalle interessannt sein können
`/etc/protocols`	Festlegung verschiedener Protokolle mit Identikations-Nummern. Im Normalfall keine Änderung erforderlich.
`/etc/networks`	Diese Datei funktioniert ähnlich wie `/etc/hosts`, wobei hier nur die Namen der einzelnen Netzwerke mit der zugehörigen Netzwerk-Nummer verzeichnet sind. Wird von manchen Programmen (z.B. `/sbin/route`) benutzt, um anstelle der Nummer den Namen des Netzwerks anzuzeigen.
`/etc/securetty`	Hier sind alle Terminals verzeichnet, an denen sich „root“ anmelden darf.
`/etc/hosts.allow`	Enthält eine Liste aller Rechner (IP-Nummern), denen bestimmte Dienste gestattet sein sollen.
`/etc/hosts.deny`	Enthält Dienste, die bestimmten Rechnern nicht gestattet sind. Sichere Konfigurationen sperren hier alle Dienste für alle Rechner und gestatten in `/etc/hosts.allow` nur denjenigen Rechnern diejenigen Dienste, die erwünscht sind.
`/etc/hosts.equiv`	Beinhaltet eine Liste vertrauenswürdiger Rechner und Benutzer, die ohne Paßwort-Abfrage bestimmte Dienste (Kommandos mit „r“ am Anfang) ausführen dürfen.
Hinweis:	SuSE-Linux verhält sich teilweise etwas eigenwillig, da einige der Konfigurationsdateien bei zahlreichen Aktivitäten, die durch „YaST“ entstehen, verändert werden. Diese Eigenmächtigkeit hat zwar für Einsteiger Vorteile, ist aber bei gewollten Eingriffen in Dateien hinderlich. Einige der Eingriffe lassen sich durch Ändern der Konfigurationsdatei im Menuepunkt „Administration“ von YaST auf Wunsch abschalten.

Tabelle 8: einige Netzwerk Programme und Dateien

Paket	Zweck
routed	Ein Routing-Daemon, der sich per RIP (Routing Information Protocol) mit anderen Routern austauscht; dient zum intelligenten Routen in größeren Netzwerken.
gated	Ein weiterer Routing-Daemon, der etwas intelligenter als `routed` dynamisches Routing erlaubt.
bind	Domain-Name Server – ersetzt die Datei `/etc/hosts`, was in größeren Netzwerken oder bei Rechnern, die mit dem Internet dauerhaft verbunden sind, notwendig ist.
ppp	gestattet das Einwählen eines Rechners zu einem Provider und damit Zugriff zum Internet herzustellen. Weitere zusätzliche Pakete zur Automatisierung sind allerdings erforderlich.
i4l	ISDN for Linux – die Möglichkeit, per ISDN-Karte PPP-Verbindungen herzustellen. Nicht ganz einfach zu konfigurieren, aber sehr leistungsfähig.
sendmail	Ein sehr universelles Email-Programm.
samba	Bietet Windows Rechnern einen Zugriff auf die Linux-Maschine. Erlaubt Datei- und Drucker-Dienste, sehr differenzierte Benutzer-Rechte bis hin zu einem Domain-Controller.
swat	Ein nettes Hilfs-Programm für Samba – bietet die Konfiguration von Samba von jedem anderen Rechner aus per Web-Browser.
apache	Der meistverwendete HTTP-Server. Nicht allzu kompliziert zu administrieren und sehr leistungsfähig.
NFS	Eine ganze Serie von Programmen, die gemeinsame Benutzer-Daten (*Yellow-Pages*), verbunden mit entsprechenden Rechten etc. verwalten. Sinnvoll, wenn viele Linux- oder Unix-Rechner in einem Netzwerk auf gemeinsame Daten zugreifen müssen.
atalk	Dieses Programmpaket ermöglicht es, einen Linux-Rechner als Datei- und Drucker-Server für Apple-Macintosh Clients zu betreiben.

Tabelle 9: Ein paar Netzwerk-Pakete

14 Problemlösungen

14.1 Platten-Probleme

Wird ein Linux-System immer ordnungsgemäß heruntergefahren, so tauchen Festplattenprobleme eher selten auf. Beim mutwilligen „Abwürgen“ des Rechners oder bei Stromausfällen hingegen sind Probleme durchaus möglich. Gegenüber anderen Betriebssystemen ergeben sich durch das verzögerte Schreiben von Blöcken auf Datenträger leider etwas schneller Inkonsistenzen in den Festplatten-Einträgen. Diesen Tribut zahlt man leider für die sonst so hohe Geschwindigkeit, mit der Linux seine Schreib-Operationen durchführt.
Daher spricht aus dieser Sicht einiges für die starke Partitionierung der Festplatte – ein Fehler in einer Partition hat meist mit den anderen Partitionen nichts zu tun. So lassen sich einzelne Bereiche der Festplatte gegeneinander schützen.

Sind einmal Fehler aufgetreten, so gilt es, ein „Rettungssystem“ zu starten, was

- als bootfähige Diskette,
- von der bootfähigen CD-ROM oder
- über LOADLIN.EXE vom DOS-Modus des PC aus

geschehen kann. Das zentrale Wesen des Rettungssystems ist, daß es ein von der Festplatte unabhängiges Linux-System ist, das zumindest die für die Rettung (einschließlich Backup) notwendigen Programme bereitstellt. In aller Regel werden keine Partitionen des zu rettenden Systems aktiviert – dies muß manuell geschehen. Hier einige Kommandos:

mount	Dateisystem in den Dateibaum „hängen“
[-t *typ*]	Angabe des Typs (msdos, minix, ext2, iso9660...)
[-r|-w]	Dateisystem nur lesbar(r) oder beschreibbar(w) mounten
device	Gerätebezeichnung z.B. `/dev/hda5` oder `/dev/cdrom`
verz	Verzeichnis für den Mount-Vorgang

umount	Dateisystem aus dem Dateibaum entfernen
verz	Verzeichnis für den Unmount-Vorgang

sync	Alle Puffer auf die Datenträger schreiben

e2fsck	Dateisystem (Typ ext2) testen und reparieren
[-c]	vorher nach defekten Blöcken suchen und diese markieren
[-f]	Test in jedem Fall durchführen, auch wenn scheinbar OK
[-p]	Automatische Reparatur ohne Rückfragen
device	Gerätebezeichnung des zu testenden Dateisystems z.B. `/dev/hdb1`

14.2 ein paar Kernel-nahe Kommandos

So sicher Linux mittlerweile ist, so individuell unterschiedlich können einzelne Hardware Zusammenstellungen sein. Daher ist es nicht schlecht, einige Informationen über den Kernel, seine internen Vorgänge und sein Verhalten erfahren zu können. Ein Weg zum Kernel geht über das Linux-Dateisystem, genauer gesagt, das `/proc/` Verzeichnis, in dem in Form zahlreicher Dateien und Verzeichnisse sehr viel Information über den Kernel zu erfahren ist. Einige dieser Dateien sind in Tabelle 10 zusammengefaßt.

⇒ `/proc/...`

Darüberhinaus kann man dem Kernel mit einigen Kommandos „auf die Finger“ schauen:

dmesg	Kernel-Meldungen beeinflussen (ohne Parameter: ansehen)
free	Speicher-Benutzung erfahren (in KByte)
lsdev	Information über alle Geräte auflisten
lsmod	Auflisten aller Module

`cpuinfo`	liefert Daten über den Prozessor
`devices`	listet alle Treiber für Geräte auf
`dma`	alle benutzten DMA-Kanäle
`ioports`	alle IO-Ports aller erkannten Karten
`interrupts`	alle Interrupts der erkannten Karten
`pci`	Informationen zu PCI-Karten
`ide/`	alles rund um IDE-Treiber, Geräte
`scsi/`	rund um SCSI-Geräte

Tabelle 10: Ein paar Dateien des `/proc/` Dateisystems

`procinfo` Viel Informationen über Kernel-Aktivität

`free` Speicher-Benutzung erfahren (in KByte)

Und zu guter Letzt werden viele Aktivitäten in sog. Log-Dateien abgelegt. Diese existieren für das System allgemein (`/var/log/messages`) oder für einzelne Programme, die viel mitprotokollieren (wie httpd, isdn,...). Diese Dateien können zur Fehleranalyse sehr wertvoll sein, da Fehler meist in Form von Log-Einträgen vorhanden sind, die zumindest über den Fehlerzustand etwas verraten.

⇒ `/var/log/...`

Literaturhinweise

Allgemeine Linux-Literatur

- Michael Wielsch: Linux (Data Becker)
  im Data-Becker Stil geschriebene brauchbare Einführung zu fast allen Themen rund um Linux; nicht besonders tief gehend, aber sehr breit angelegt
- Matt Welsh & Lar Kaufman: Running Linux (O´Reilly & Associates)
  von der Installation bis hin zu Hinweisen einzelner Programme viele Beschreibungen, die zumindest einen Einblick in die Vielfalt einzelner Programme geben
- Marco Budde: Linux HOWTOs (MITP-Verlag)
  Zu den gängigen Themen finden sich hier sehr gut aufbereitet umfassende Anleitungen
- Jochen Hein: Linux Systemadministration (Addison Wesley)
  Alles rund ums Einrichten und Warten von Linux
- Matthias Hölzer, Dr. Bernhard Röhrig: KDE (C&L)
  Bedienung und Programmierung des KDE werden hier erläutert

Netzwerk-Bücher

- Olaf Kirch: Linux Network Administrators Guide (O´Reilly & Associates)
  Gute Einführung in TCP/IP sowie genaue Erklärung aller notwenigen Dienste und Möglichkeiten sowie eingehende Erklärung der jeweiligen Installations-Prozeduren
- Hal Stern: Managing NFS and NIS (O´Reilly & Associates)
  Wer reine Unix-Netze aufbauen will findet hier was er braucht, sonst unnötig
- Paul Albritz & Cricket Liu: DNS and BIND (O´Reilly & Associates)
  Pflichtlektüre für jeden, der seinen Linux-Rechner als Server mit dem Internet verbinden will
- W. Richard Stevens: TCP/IP Illustrated Vol. 1.3 (Addison-Wesley)
  Sehr genaue Erklärung aller TCP/IP Protokolle sowie deren Programmierung (Band 2-3)
- Michael Santifaller: TCP/IP und NFS in Theorie und Praxis (Addison-Wesley)
  Überblick über TCP/IP und NFS-Protokolle (sehr technisch)

Wer programmieren will

- Nikolaus Schüler: Der GCC Compiler (bhv Verlag)
  Einige Bedienungshinweise zum Gnu C Compiler und notwendigen Zusatzprogrammen
- Richard Petersen: Linux Programmers Reference (Osborne)
  Hier werden in der Hauptsache die Shells und nebenbei einige weitere Programmierwerkzeuge beleuchtet

Interna, Kernel

- Michael Beck u.a.: Linux Kernel Programmierung (Addison-Wesley)
  Wer den Kernel besser verstehen möchte findet hier zumindest einiges über interne Datenstrukturen

Motivations-Lektüre

- Jennifer Edstorm & Marlin Eller: Barbarians led by Bill Gates (Henry Holt & Co.)
  Wer mehr über die interna von Microsoft wissen will und Lachmuskeln hat...