



Linux – ein Crashkurs

- Anmelden, environment, shell, Prozesse
- User-Kommandos
- User, Groups und Permissions
- Filesystem-Struktur (was ist wo)
- „Wie Ei“ – der schönste Editor der Welt
- Administration (Kommandos vs. YAST)
- Wie verschaffe ich mir Informationen?
- Ein Notfall – was nun?



Erstmal: Was gibt es sonst?

- UNIX System V: Bell Laboratories, jetzt Rechte bei NOVELL gelandet (wo auch SuSE inzwischen ist)
- BSD UNIX: Derivat der UCB, portabel (OpenBSD)
- SCO UNIX: R.I.P.
- HP-UX: Spezialität für HP Server Hardware (RISC)
- IBM AIX: Spezialität für IBM Server Hardware (RISC)
- **SUN Solaris 10: Für SUN Sparc und INTEL**
- MINIX: Ein spezielles Betriebssystem für die Lehre
- **Linux: Neuimplementierung, von MINIX inspiriert**



Die Arbeits-Umgebung

- Nach dem Einloggen arbeitet man in einer Session, die ein bestimmtes „Environment“ besitzt und an ihre „Kinder“ vererbt
- Das Environment zeigt man an mit „env“
- Dazu gehören das aktuelle Arbeitsverzeichnis, der Username, das Home-Verzeichnis
- Das erste Programm in einer Session (ohne GUI) ist eine „shell“ als User Interface



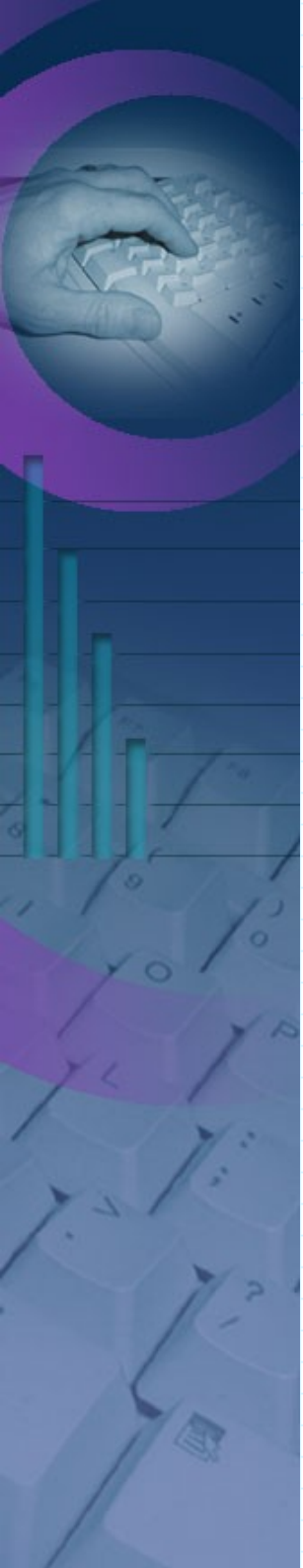
Was ist eine „shell“?

- Zeichen- und zeilenorientierte Bedieneroberfläche (CUI – character UI)
- Interface zum Filesystem und zu den Linux/Unix System-Funktionen
- Beispiel bash: Guter und portabler Standard bei akzeptablem Komfort
- Beispiel tcsh: Weniger portabel, dafür mehr Komfort (bash entwickelt sich aber dorthin)
- cmd.exe: Das Pendant unter Windows



Kommandos für User (1)

- `cd:` Arbeitsverz. anzeigen/wechseln
- `ls:` Auflisten von Dateien
- `date:` Datum und Zeit anzeigen
- `who:` Wer ist am System angemeldet?
- `less:` Datei seitenweise ausgeben
- `grep:` Regular Expressions suchen
- `find:` Dateien im Verzeichnisbaum finden



Ein/Ausgabe-Datenströme

- Jeder Linux-Prozess erhält per default beim Start drei I/O-Kanäle:
 - stdin (Eingabe-Datenstrom)
 - stdout (Ausgabe-Datenstrom)
 - stderr (Error-Ausgabe-Datenstrom)
- Durch eine „pipe“ | kann man diese Ströme miteinander verknüpfen, z.B.:

```
ls -l | less
```
- Man kann dies beliebig fortsetzen!



Programme und Prozesse

- Ein **Programm** ist eine Datei mit ausführbarem Code (z.B. `/usr/bin/vi`, `/bin/bash`)
- Bei Aufruf des Programmes (z.B. von der Shell aus) lädt der **Loader** (`/bin/ld`) das Programm so in den Speicher, dass es ausgeführt werden kann.
- Diese Instanz im Speicher bezeichnet man als **Prozess**, er erbt das Environment der Shell, ist aber ansonsten abgeschirmt!



User-Kommandos (2)

- `ps`: Liste der laufenden Prozesse im eigenen „User-Workspace“
- `top`: Sortierte dynamische Prozess-Anzeige
- `w`: who mit Details
- `sar`: Auf UNIX Systemen zur Performance-Messung verfügbare Tools
- `vmstat`: Virtual Machine Status (BSD)
- `netstat`: Netzwerks-Zustände anzeigen



User, Groups und Permissions

- Ein User hat einen Login-Namen und eine uid (eine Nummer) -> `/etc/passwd`
- Zusätzlich ist er Mitglied einer oder mehrerer Gruppen; Gruppen haben einen Namen und eine gid (Nummer) -> `/etc/group`
- Jede Datei und jedes Verzeichnis hat separate Zugangs-Parameter („permissions“) für den Eigentümer (uid+gid), die Gruppe (nur gid) und den „Rest der Welt“.
- Beispiel: `rw-r--r--` oder `rwxr-xr-x` usw.
- Der „Super-User“ ist root (uid=0) und darf ALLES



Filesystem-Struktur

- Das Filesystem ist hierarchisch
- Es gibt keine „Laufwerke“
- Alles beginnt am „falschen Ende“ bei root
- Das Wurzel-Verzeichnis (root) wird mit einem einfachen / abgekürzt
- Der slash ist ein slash und kein backslash
- Ansonsten werden Verzeichnisse und Pfadnamen wie bei Windows notiert



Filesysteme managen

- Vereinfacht ist ein Filesystem eine organisierte Partition auf eine Platte
- Filesysteme werden beim Start des Systems eingehängt (`mount`, `/etc/fstab`)
- Zuerst wird das root-Filesystem „gemountet“
- Dann werden darin die anderen Filesysteme eingehängt
- Anzeigen mit `mount` oder `df` (mit Füllgrad)



Filesystem-Struktur (2)

- / (root-FS) ist der Startpunkt. Es enthält:
 - /etc Systemverwaltung (kein eigenes Filesystem!)
 - /bin allgemeine Programme (link: /usr/bin)
 - /sbin Programme für das System (link: /usr/sbin)
 - /tmp Temporäre Dateien
- Ggf. eigene Filesysteme sind:
 - /var Häufig veränderte Files
 - /opt Zusätzliche Software-Produkte
 - /usr Alles was User brauchen
 - /home Die Arbeitsverzeichnisse der User



Wichtige Verzeichnisse und Dateien

- Spezielle Dateinamen und -typen:
 - . und .. sind Shortcuts für das aktuelle und das darüberliegende Verzeichnis
 - „Harte“ Links sind beliebig viele Aliases für eine Datei
 - Symbolische Links sind für Anwendungen *transparente* Verweise auf Dateien an anderem Ort (anders als bei Windows!)
- Spezielle Verzeichnisse:
 - lost+found: Hier werden verwaiste Datenblöcke aufbewahrt (siehe `fsck`)



Wichtige Verzeichnisse und Dateien 2

- `/var/log`: Logfiles, z.B. `/var/log/messages`
- `/etc/init.d`: Skripte, die beim Booten ausgeführt werden
- `/etc/passwd`: User-Namen und uid
- `/etc/shadow`: Passworte zu uid's (cryptet)
- `/etc/fstab`: Filesysteme und Mountpoints
- `/etc/inittab`: Steuert den init-Prozess, der nach dem Booten des Kernels die Kontrolle über das System übernimmt
- `/etc/motd`: Message of the Day



Texteditor vi(m)

- Aufruf: `vi <filename>`
- Es gibt den Kommando- und den Edit-Mode
- Navigieren mit h, j, k, l oder evtl. Pfeiltasten
- Edit-Kommandos: i, a, c, s ...
- Edit-Mode -> Kommando-Mode mit ESC
- Verlassen und ggf. Speichern mit :x <enter>
- Gewöhnungsbedürftig, aber effizient
- Alternativen: jove, emacs, pico ...



Administrator-Aufgaben

- Shutdown / Boot
- Logfiles beobachten
- Filesysteme beobachten und pflegen
- Netzwerks-Informationen pflegen
- Netzwerks-Dienste beobachten und pflegen
- User und Gruppen anlegen
- Performance beobachten und optimieren
- Sicherheit aufrechterhalten



Admin-Möglichkeiten

- Editieren von Konfigurations-Files unter /etc (dazu muss man die Files kennen):
 - Alles kann eingestellt und optimiert werden
 - Man braucht Manuals
 - Man kann viel kaputtmachen
- GUI/CUI-Tools wie YAST oder webmin:
 - Begrenzte Möglichkeiten
 - Intuitive Bedienung
 - Keinesfalls risikolos, also trotzdem ... (b.w.)



Wo kriegt man Infos her?

- **RTFM:** Read The Fabulous Manual
- Kommando: `man <thema>` (z.B. `man ls`)
- Doc-Files liegen oft unter **`/usr/share/doc`**
- Wichtiges Hilfsmittel: howto Files (auch dort)
- Ansonsten steht praktisch alles im Internet
- Der Leitsatz muss sein:
 - ◆ **ERST lesen,**
 - ◆ **DANN planen,**
 - ◆ **DANN machen!**



Ein Notfall – was nun?

- Alt-Ctrl-Del bei einem Server?
- Informationen holen:
 - Logfiles, z.B. `/var/log/messages` und Konsole
 - `w`, `who`, `ps`, `top`, `df`, `mount`, `du`
 - `netstat`, `iptraf`, `ethereal`
- Massnahmen (als root):
 - Platten aufräumen: `rm` löscht, `mv` verschiebt
 - CPU-Fresser loswerden: `kill -<SIG> <pid>`
 - Netzwerk abklemmen: `ipconfig <if> down`



Boot-Konzept

- Durch den Bootsektor (MBR) auf dem ersten Systemlaufwerk wird der Bootstrap-Loader (grub, lilo) geladen
- grub/lilo liest ein Config-File und richtet eine Ramdisk ein (Basis-System, Treiber)
- grub/lilo bootet dann den Linux-Kernel
- Der Kernel startet ein initiales Programm, typischerweise /sbin/init, das den Rest erledigt



Nach dem Booten

- `/sbin/init` liest die Datei `/etc/inittab`
- Runlevel werden durchlaufen
- `/etc/init.d` Skripte werden ausgeführt
- tty's werden aktiviert
- GUI (KDE, gnome) wird gestartet (auch über Skripte in `/etc/init.d`)
- Login ist möglich



Übungsaufgabe

- SuSE Linux 10 auf den PCs im NATKOM Labor installieren (vorhandene Linux-Partition überschreiben ohne die Windows-Installation zu zerstören)
- Vorher Vorgehensweise planen:
 - Netzwerks-Adressen besorgen (Windows)
 - Root-Passwort vereinbaren
 - Zusatz-User vereinbaren
 - Platten-Aufteilung überlegen