

# Linux - Grundlagen

by

Dr. Günter Kolousek

- ▶ Unix-artiges Betriebssystem
  - ▶ Linus Torvalds
- ▶ Kernel
- ▶ Distribution
  - ▶ Betriebssystemkern (nukleus, kernel)
  - ▶ Systembibliotheken (system libraries)
    - ▶ Prozess mittels Systemaufrufen (system-call)
  - ▶ Systemprogrammen (system utilities): Benutzer

# Linux Distributionen

- ▶ Slackware: Urgestein, extrem stabil (seit 4/1993!)
- ▶ Redhat: kommerziell, hohe Verbreitung in den USA
  - ▶ freier Ableger: Fedora
- ▶ openSUSE: stabil, benutzerfreundlich
- ▶ Gentoo: selbst übersetzen!
  - ▶ für Fortgeschrittene
- ▶ ArchLinux: "Keep It Simple"
  - ▶ rolling distribution, d.h. mit rolling releases
  - ▶ für Fortgeschrittene

# Linux Distributionen – 2

- ▶ Manjaro: basierend auf ArchLinux
  - ▶ für Anfänger und Fortgeschrittene
  - ▶ eigene Repositories
- ▶ Debian: extrem stabil, robustes Paketmanagement, breite Plattform-Unterstützung, "einfaches" Upgrade, sehr sicher (seit 8/1993)
- ▶ Ubuntu: wie Debian, Schwerpunkt auf Bedienbarkeit
  - ▶ LUbuntu, Xubuntu, Kubuntu
- ▶ Void Linux: *kein* systemd, rolling releases, klein, schnell
  - ▶ für Fortgeschrittene

# Linux – Charakterisierung

- ▶ GNU: Gnu is Not Unix, Projekt um ein vollständig freies Betriebssystem zu entwickeln, Richard Stallman, 1984
- ▶ UNIX: eingetragenes trademark der Open Group
- ▶ unixartig
  - ▶ jedoch kein UNIX-Derivat (da nicht vom UNIX Quelltext abgeleitet)
- ▶ UNIX-konforme BS: Solaris (Oracle), Mac OSX (Apple), AIX (IBM), HP-UX (HP)

# Unix-Philosophie

- ▶ "von Programmierern für Programmierer"
- ▶ kleine Programme mit speziellen Funktionen, die miteinander kombiniert werden können.
- ▶ effizientes arbeiten mit dem System
  - ▶ Shell mit kurzen Befehlen
  - ▶ keine Rückfragen, sondern Parameter und Optionen
  - ▶ nur Fehlerfall wird gemeldet und nicht die erfolgreiche Abarbeitung ("Das Backup ist fertig. Bestätigen sie mit OK")
- ▶ einheitliche Schnittstellen
  - ▶ Datei ist eine allgemeine Abstraktion für: Dateien, Verzeichnisse, Geräte, Datenströme (Pipes, Message queues, Netzwerk)
  - ▶ Netzwerktransparenz: X11, syslogd, NFS

# Linux – Anwendungen

- ▶ PC, Server, Mainframe, Supercomputer (498 von 500!)
- ▶ Webserver! (Apache, Nginx,...)
- ▶ Embedded, Appliance
  - ▶ Router, Access Points, Switches
  - ▶ SAT-Receiver, Media-Center (z.B. Amazon Fire TV), Fernseher,...
  - ▶ Webcam, Action-Cams, Navis (z.B. TomTom, Bosch, Citroen, Mazda,...)
  - ▶ Drucker
  - ▶ Smartphone (Android)
- ▶ Intel, ARM, MIPS, PowerPC

# Multitasking/multiuser

- ▶ Jeder Benutzer hat einen Namen und einen Identifier
- ▶ Jeder Benutzer gehört genau zu einer Gruppe (primary group)
- ▶ Jeder Benutzer kann zu mehreren zusätzlichen Gruppen gehören (secondary group)
- ▶ Jede Gruppe hat einen Namen und einen Identifier
- ▶ Der Systemadministrator hat den Namen "root" und gehört zur Gruppe "root".



# Linux-Dateisystemhierarchie

- ▶ Alles eine Datei, wie z.B.:
  - ▶ Verzeichnis
  - ▶ Geräte
  - ▶ Kommunikationskanäle
- ▶ Wurzelverzeichnis
- ▶ Pfad
  - ▶ getrennt durch /
  - ▶ absolute Pfade beginnen bei der Wurzel
  - ▶ relative Pfade beginnen im aktuellen Verzeichnis
- ▶ Dateisysteme können an beliebiger Stelle im Dateisystembaum eingehängt werden (mount).
  - ▶ Partition oder Netzwerk
  - ▶ Linux üblich: Ausfallsicherheit, Performance, Sicherheit, Wartbarkeit.

# Linux-Dateisystemhierarchie – 2

- ▶ / ... Wurzel (root)
- ▶ /bin ... Programme, die jedem Benutzer zur Verfügung stehen, jedoch schon zum Systemstart benötigt werden.
- ▶ /boot ... enthält Dateien, die für den Bootprozess notwendig sind. Hier befindet sich normalerweise der Kernel.
- ▶ /dev ... enthält Gerätedateien (Devices). Auch: /dev/null
- ▶ /etc ... Konfigurationsdateien
- ▶ /home ... enthält HOME-Verezeichnisse der Benutzer
- ▶ /lib ... enthält die für den Systemstart und die Systemprozesse notwendigen dynamischen Bibliotheken (shared libraries).

# Linux-Dateisystemhierarchie – 3

- ▶ `/media` ... wird oft verwendet, um die externen Datenspeicher wie z.B. Floppy-Disk, CD-ROM bzw. USB-Sticks zu "mounten".
- ▶ `/root` ... HOME Verzeichnis des Benutzers root
- ▶ `/sbin` ... Programme, die dem root-Benutzer vorbehalten sind, jedoch schon zum Systemstart benötigt werden.
- ▶ `/tmp` ...dient Programmen zur temporären Ablage von Dateien. Dieses Verzeichnis wird in der Voreinstellung bei jedem Systemstart gelöscht.
- ▶ `/usr` ... enthält in der Regel nur lesbare Dateien, die auch mit anderen Hosts geteilt werden können.
- ▶ `/var` ... enthält variable Daten wie Logdateien, Druckerspools

# Booten von Linux

- ▶ Begriff BOOT: bootstrap (Lasche bei Lederschuhe). Bei anderen Architekturen wird auch die Abkürzung IPL (initial program load) verwendet.
- ▶ Prinzipieller Ablauf
  1. Kernel wird vom Bootmanager geladen
  2. Kernel wird als Urprozess (init-Prozess) mit der Prozess-ID 1 ausgeführt
  3. Weitere Prozesse werden gemäß Konfigurationsdateien gestartet

# Benutzeroberflächen von Linux

- ▶ Shell ("Muschel, Hülle, Schale um Betriebssystem") ist ein Kommandointerpreter, z.B.:
  - ▶ sh ... Bourne-Shell
  - ▶ bash ... Bourne-Again-Shell ist Standardshell unter Linux!
  - ▶ zsh ... viele Verbesserungen gegenüber bash
  - ▶ fish ... eine der leistungsfähigsten Shell (nicht kompatibel zu bash)
  - ▶ xonsh ... auf Python basierend; extrem leistungsfähig
- ▶ X-Windows
  - ▶ Fenstersystem
  - ▶ API (freie Implementierungen: XFree86, XOrg)
  - ▶ (Netzwerk-)Protokoll: X-Terminal!
- ▶ Fenstermanager (window manager): i3, fvwm,...
- ▶ Desktop: Xfce, Gnome, KDE,...

# Grundlagen einer Shell

- ▶ Prompt, meist \$ oder #
- ▶ Wildcardzeichen (globbing)
  - ▶ \*
  - ▶ ?
  - ▶ [], z.B. [abc]\*
- ▶ Hilfe zu Befehlen
  - ▶ man ... für Programme
  - ▶ help ... für shell builtins
- ▶ Vervollständigung von Dateinamen und Befehlen → Tab
- ▶ wichtige Steuerzeichen
  - ▶ CTRL-C ... Abbruch
  - ▶ CTRL-D ... Dateiende
  - ▶ CTRL-Z ... aktueller Prozess in Hintergrund

# Shell- & Umgebungsvariable

- ▶ Zugriff mittels \$
- ▶ Zuweisung mittels =
- ▶ \$HOME
- ▶ \$USER
- ▶ \$PATH
- ▶ Umgebungsvariable
  - ▶ mittels export

- ▶ . ... aktuelles Verzeichnis, Arbeitsverzeichnis (working directory)
- ▶ . . ... übergeordnetes Verzeichnis
- ▶ Pfad des Benutzers mittels ~
  - ▶ auch ~ko
- ▶ "Versteckte" Dateien: beginnen mit einem .



# stdin, stdout, stderr

Umleitung mittels <, >, >>, 2>, &>, 2>>, |

```
ls *.c > cfiles.txt
```

```
tee < test.txt test_copy.txt | less
```

```
cat test?.txt >> out.txt
```

```
make 2> errors.txt > out.txt
```

```
date; ls > contents.txt # nur Ausgabe von ls!
```

```
(date;ls) > contents.txt # eigener Prozess!
```

```
(ls -y; ls) &> contents.txt # stdout & stderr!
```

- ▶ Textdatei: Abfolge von z.B. bash-Befehlen.
- ▶ 1. Zeile kann Pfad zum ausführenden Programm beinhalten:  
`#!/bin/bash`
  - ▶ → shebang
  - ▶ auch z.B.: `#!/usr/bin/env python`
- ▶ Aufruf bewirkt Ausführung in einem neuen Prozess außer es wird das Skript mittels `source` aufgerufen.
- ▶ Ausführung eines Skriptes im aktuellen Prozess mittels:  
`source DATEINAME` oder `. DATEINAME`. Bei `source` handelt es sich wieder um ein bash builtin.
- ▶ Bash-Konfigurationsdatei: `~/.bashrc`

# Wichtige Befehle

- ▶ `exit`
  - ▶ auch CTRL-D
- ▶ `echo`
  - ▶ `-n` → kein Zeilenumbruch
- ▶ `clear`
- ▶ `type ... builtin` oder nicht?
- ▶ `man`
  - ▶ auch: `man man`
- ▶ `help ...` für builtins
- ▶ `which`
- ▶ `file`
- ▶ *Option* `--help`!

# Wichtige Befehle – 2

- ▶ pwd
- ▶ cd
  - ▶ cd -
- ▶ ls
  - ▶ -a
  - ▶ -d ... Verzeichnisinhalt!
  - ▶ -l
  - ▶ -R ... rekursiv
  - ▶ -1 ... eine Datei pro Zeile
- ▶ cat
  - ▶ -n ... line numbers
- ▶ less

# Wichtige Befehle – 3

- ▶ `mkdir`
  - ▶ `-p` → parents
- ▶ `cp`
  - ▶ `-R` ... rekursiv
- ▶ `rm`
  - ▶ `-R`
  - ▶ `-f` (keine Nachfrage)
- ▶ `rmdir`
  - ▶ `-p` ... auch die Eltern (parents)
    - ▶ `rmdir -p a/b/c`
    - ▶ `rmdir a/b/c; rmdir a/b; rmdir a`
- ▶ `mv`

# Wichtige Befehle – 4

- ▶ touch
- ▶ chmod
  - ▶ -R ... rekursiv
  - ▶ chmod 0755 datadir
  - ▶ chmod 0764 projects.txt
  - ▶ chmod u=rwx,g=rw,o=r projects.txt
  - ▶ chmod g+x projects.txt
- ▶ stat ... Infos zur Datei
- ▶ ln
  - ▶ -s ... symbolischer Link
- ▶ alias
  - ▶ alias ll='ls -l'

# Wichtige Befehle – 5

- ▶ `tar`
  - ▶ `-c ... create`
  - ▶ `-v ... verbose`
  - ▶ `-x ... extract`
  - ▶ `-t ... toc`
  - ▶ `-z ... komprimieren mit gzip`
- ▶ `gzip, gunzip`
- ▶ `df`
  - ▶ `-h ... human readable`
- ▶ `ssh`
- ▶ `scp`
  - ▶ `scp xxx.txt ko@ifssh...:public`

# Wichtige Befehle – 6

- ▶ `ps`
  - ▶ `-e ...` alle Prozesse
  - ▶ `-f ...` full format
- ▶ `kill`
  - ▶ `kill -l ...` Signalnamen
  - ▶ `kill -kill 4711`
  - ▶ `kill %1 ...` sendet SIGTERM
    - ▶ `kill -term %1, kill -SIGTERM %1, kill -15 %1`
- ▶ `fg`
- ▶ `bg`
- ▶ `jobs`
- ▶ `&`



# Dateirechte

Recht	Datei	Verzeichnis	Beispiel
read	Inhalt ansehen	Verzeichnis auflisten	chmod ugo=r datei chmod 0444 datei
write	Inhalt ändern	Dateien anlegen, löschen, umbenennen	chmod ugo=w datei chmod 0222 datei
execute	Binärdateien ausführen Skripte: auch Leserecht notwendig	in Verzeichnis wechseln, auf Inhalt zugreifen	chmod ugo=x datei chmod 0111 datei
set-user-id	unter Kennung des Eigentümers ausführen		chmod ugo=x,u+s datei chmod 4111 datei
set-group-id	unter Kennung der Gruppe ausführen	Neue Dateien und Verzeichnisse ("erben" set-group-id) → gleiche Gruppe wie Verzeichnis	chmod ugo=x,g+s datei chmod 2111 datei
sticky		nur Eigentümer einer Datei kann diese löschen	chmod ugo=rwx,o+t verz chmod 1777 verz

- ▶ "neuere" Dateien von Quelle zum Ziel (mod. time, size)
- ▶ überträgt nur Änderungen!!!
- ▶ `rsync -rn workspace backup workspace` → backup
  - ▶ `-n` do nothing `--dry-run`
- ▶ `rsync -rn workspace/ backup` Inhalt von workspace
- ▶ `rsync -a workspace/ workspace.bak`
  - ▶ `-r` rekursiv
  - ▶ `-p` Rechte der Dateien behalten
  - ▶ `-t` Zeiten behalten
  - ▶ `-g` Gruppe behalten
  - ▶ `-o` Eigentümer behalten
  - ▶ `-l` symbolische Links als symbolische Links kopieren

# rsync - 2

- ▶ `--exclude=*.bak` bzw. `--exclude-from=no.txt`
- ▶ `-h` ... menschenfreundliche (h ... human) Ausgabe
  - ▶ z.B. 5K
- ▶ `--delete` löscht Datei im Ziel, wenn nicht in Quelle
- ▶ `-z` ... komprimieren der Daten bei Übertragung
- ▶ `-v` ... "verbose"
- ▶ Beispiele
  - ▶ `rsync -a workspace/ workspace.bak`
  - ▶ `rsync -a workspace/ maxi@ifssh.htlwrn.ac.at:workspace.bak`

# Entferntes Dateisystem einbinden

- ▶ Einbinden eines entfernten Verzeichnisses in den lokalen Verzeichnisbaum (engl. mount)
- ▶ mount: `sshfs user@host:dir mountpoint`
  - ▶ `sshfs ko@ifssh...:public/nvs5 nvs5`
    - ▶ `sshfs -o ServerAliveInterval=15 ...`
  - ▶ lokales Verzeichnis muss existieren
- ▶ unmount: `fusermount -u mountpoint`
  - ▶ `fusermount -u nvs5`