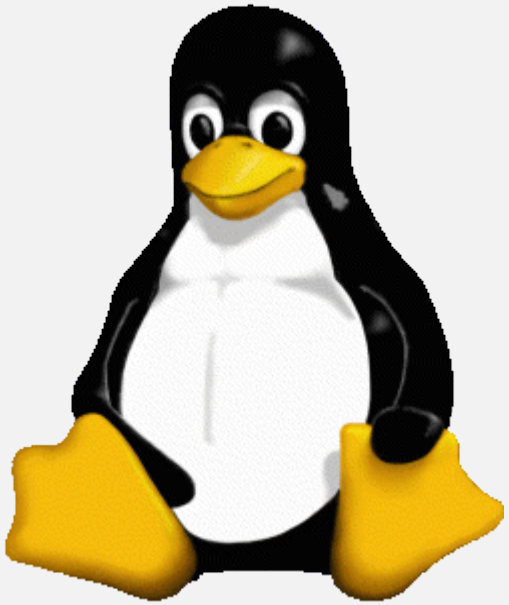
esse

LINUX TUTORIAL



Repolusk Jürgen

Technische Universität Wien – Security 183.124 – WS 2006/2007

esse

Inhalt

- Was ist Unix / Linux
- Unix Shells / bash
- Linux Benutzerbefehle
- Befehle zur Netzwerkverwaltung
- Shell Magic
- Weitere Informationsquellen

esse

Was ist UNIX?

- UNIX ist ein Mehrbenutzer-Betriebssystem.
- Anfang der 70er von Bell (Ken Thompson und Dennis Ritchie) entwickelt
- Im allgemeinen Sprachgebrauch bezeichnet man als Unix(e) Betriebssysteme die ihre Wurzeln in UNIX haben.

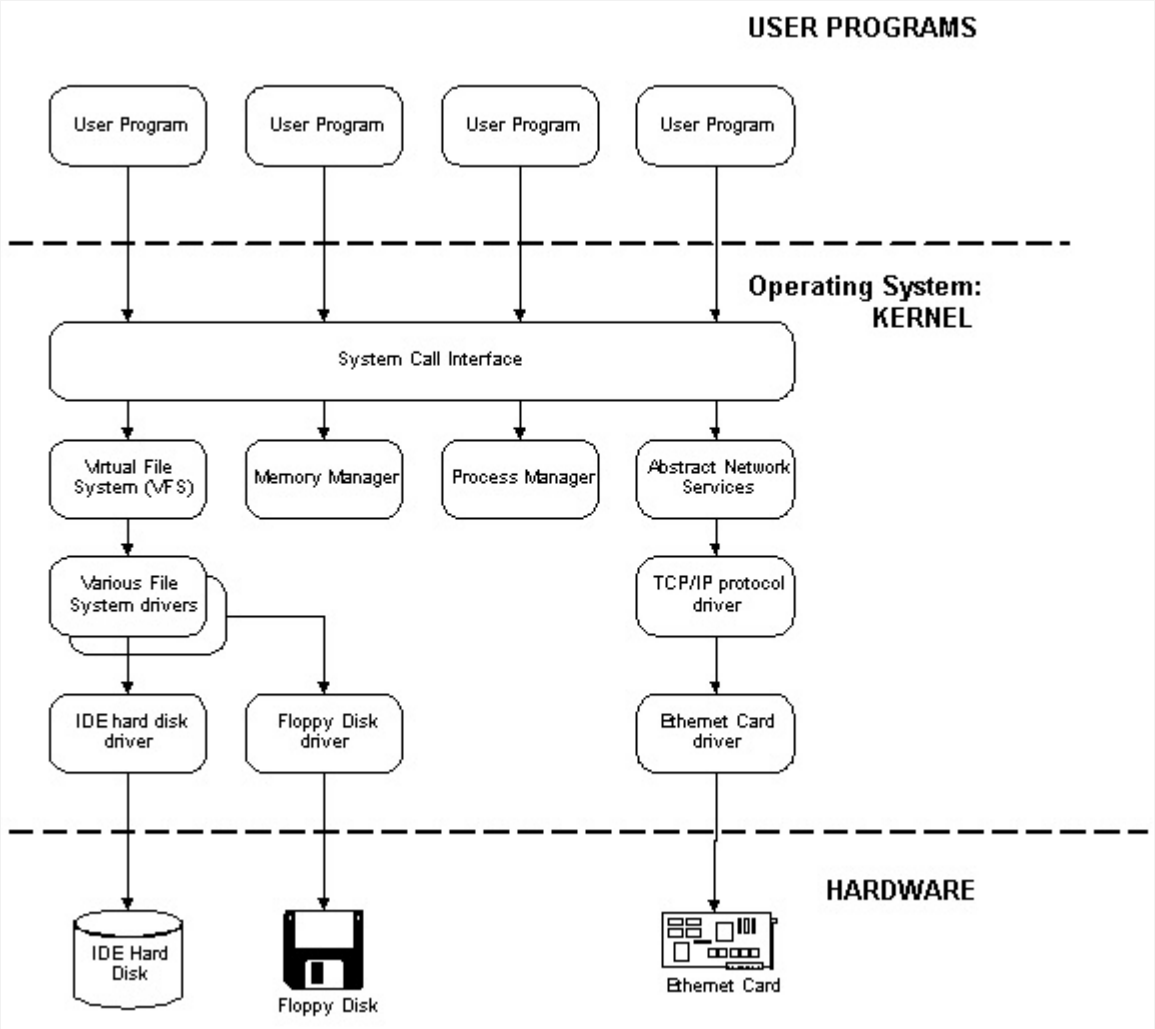
- Bekannte Abkömmlinge sind:
  - Linux
  - Free-, Open- und NetBSD
  - Solaris
  - SCO
  - AIX

- Unix History: http://www.levenez.com/unix/history.html

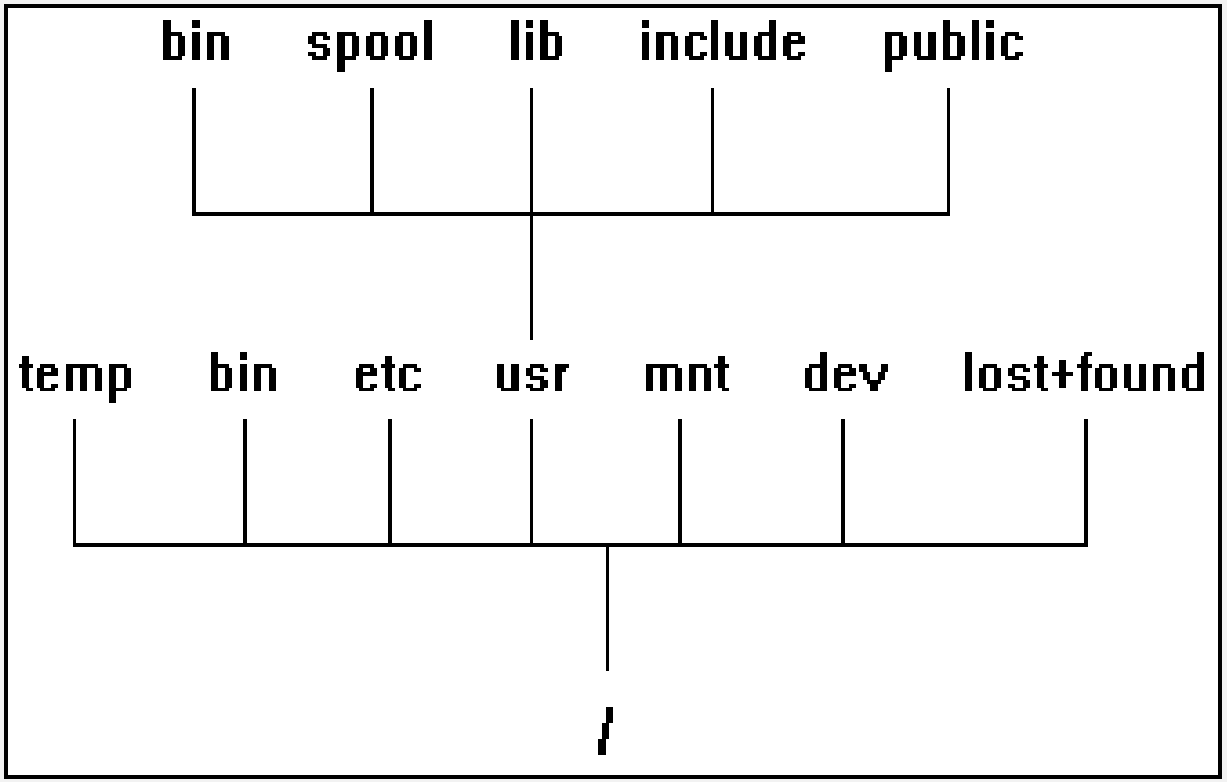
Was ist LINUX?

- Linux ist eine “freier” Kern für Computer Betriebssysteme
- Name leitet sich vom Gründer Linus Torvalds und Unix ab
- 1991 erstmals öffentlich verfügbar
- Kern wurde ursprünglich von Linus geschrieben und später unter einer freien Lizenz veröffentlicht
- Dadurch beteiligten sich weltweit viele Programmierer an der Entwicklung von Linux unter der Koordination von Linus
- Meist kommt mit einem Linux ein GNU System zum Einsatz, dass die zugehörige Infrastruktur liefert. Man spricht auch meist von GNU/Linux
- Bekannte Distributionen:
  - Debian
  - Red Hat
  - SuSE, Mandrake, Gentoo, ...

Aufbau des Linux Kernel



Linux Dateibaum



Shells

Shell ist eine CLI (Command line interface)

Vorteile:
- Schnelle & direkte Steuerung von Funktionen/Services
- Gute Automatisierbarkeit durch Skripte

Anwendungen für eine Shell:
- Interaktive Benutzung
- Anpassung der eigenen Arbeitsumgebungen
- Programmierung

Bash: bourn again shell

- Teil des GNU Projekts
- Wortspiel als “wiedergeborene” und als “wieder einmal (eine)” Shell
- Programmiert von Brian Fox und Chet Ramey Ende der 80er
- Kompatibel zur orginalen Bourne Shell (sh)
- Im Funktionsumfang erheblich erweitert
- Beherrscht Großteil der ksh und versteht Teile von csh, wie z.B.
  - Command History,
  - Directory-Stack,
  - die $RANDOM-Variable
  - und die POSIX-Form der Command Substitution '$(...)'.
- Die Bash ist die Standard-Shell (/bin/sh) auf vielen Linux-Systemen und wurde auf fast alle Unix-Systeme portiert.

esse

Dokumentation von Befehlen

- **man** - System Dokumentation
- Dokumentation von Befehlen:
  z.B. man printf
- Dokumentation von C includes:
  z.B. man 3 printf
- Finden von man-pages durch **apropos**
- Infopages mittels **info**

Arbeiten mit dem Dateisystem

- **pwd** gibt das aktuelle Arbeitsverzeichnis aus
- **ls** liefert Informationen über Dateien
- **cd** Verzeichniswechsel
- **mkdir** erzeugt ein Verzeichnis
- **rm** löscht eine Datei / Verzeichnis
- **rmdir** löscht ein leeres Verzeichnis

Verzeichnis- und Dateibefehle

- **ln** erzeugt einen Link
- **cat** dient zur Ausgabe von Dateien
- **cp** kopiert Dateien / Verzeichnisse
- **mv** dient zum Verschieben / Umbennnen
- **file** gibt den Typ eines Files aus
- **touch** erzeugt ein File mit leerem Inhalt

Beispiele

```
jvr@rocksolid ~ $ mkdir security
jvr@rocksolid ~ $ touch security/lva.txt
jvr@rocksolid ~ $ cp /home/jvr/lvb.txt security/
jvr@rocksolid ~ $ cd security
jvr@rocksolid ~/security $ ln -s lvb.txt lvc.txt
jvr@rocksolid ~/security $ ls -ltr
total 4
-rw-r--r--  1 jvr users  0 Oct 11 18:00 lva.txt
-rw-r--r--  1 jvr users 63 Oct 11 18:02 lvb.txt
lrwxrwxrwx  1 jvr users  7 Oct 11 18:08 lvc.txt -> lvb.txt
jvr@rocksolid ~/security $ mv lvb.txt lvd.txt
jvr@rocksolid ~/security $ cat lvc.txt
cat: lvc.txt: No such file or directory
jvr@rocksolid ~/security $ cd
jvr@rocksolid ~ $ rm security/*
jvr@rocksolid ~ $ rmdir security
```

Suchfunktionen

- **which** listet alle Executeables im Userpath
- **whereis** listet alle Executeables
- **find** sucht nach definierter Maske
- **grep** sucht nach Strings in Files

Beispiele

```
jvr@rocksolid ~ $ which which
/usr/bin/which
jvr@rocksolid ~ $ whereis find
find: /usr/bin/find /usr/X11R6/bin/find /usr/bin/X11/find
/usr/man/man1/find.1.gz /usr/man/man1p/find.1p.gz
/usr/share/man/man1/find.1.gz /usr/share/man/man1p/find.1p.gz
jvr@rocksolid ~ $ find / -name passwd
/bin/passwd
/etc/pam.d/passwd
/etc/passwd
/usr/bin/passwd
jvr@rocksolid ~ $ grep root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/bin/bash
jvr@rocksolid ~ $ apropos apropos
apropos (1) - locate commands by keyword lookup
```

esse

Dateibefehle

- **diff** vergleicht Files
- **tail** gibt Ende einer File aus
- **head** gibt Anfang einer File aus
- **wc** zählt Wörter / Zeilen
- **cut** schneidet Zeichen / Wörter aus Files

Dateibefehle

- **paste** fügt Zeilen zusammen
- **md5** erzeugt eine Checksum vom File
- **echo** schreibt auf die Standardausgabe
- **sort** sortiert
- **tar** erzeugt Archive
  mit z als Option - komprimiert Files

Beispiele

```
jvr@rocksolid ~ $ diff /etc/passwd passwd
1c1,2
< root:x:0:0:root:/root:/bin/bash
---
> root:x:0:0:root:/root:/bin/false
> foot:x:0:0:root:/root:/bin/bash
jvr@rocksolid ~ $ head -n 4 passwd
root:x:0:0:root:/root:/bin/false
foot:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/false
daemon:x:2:2:daemon:/sbin:/bin/false
jvr@rocksolid ~ $ wc -l /etc/passwd passwd
  44 /etc/passwd
  45 passwd
  89 total
jvr@rocksolid ~ $ md5sum /etc/passwd passwd
028cbf575d53c3c13c2a30fbeb925107  /etc/passwd
15676c8f24761f8a30193ecef85fa1b9  passwd
```

Beispiele

```
jvr@rocksolid ~ $ tail -n 1 /etc/passwd | cut -b 1-20
messagebus:x:101:408
jvr@rocksolid ~ $ echo "eine einfache nachricht"
eine einfache nachricht
jvr@rocksolid ~ $ echo -n "eine einfache nachricht"
eine einfache nachrichtjvr@rocksolid ~ $
jvr@rocksolid ~ $ paste /etc/passwd passwd | head -n 2
root:x:0:0:root:/root:/bin/bash root:x:0:0:root:/root:/bin/false
bin:x:1:1:bin:/bin:/bin/false   foot:x:0:0:root:/root:/bin/bash
jvr@rocksolid ~ $ tar czf etc.tar.gz /etc/
jvr@rocksolid ~ $ tar xzf etc.tar.gz
```

User & Systemverwaltung

- **su** Benutzerwechsel
- **chown** Ändert File Benutzer- und Gruppe
- **chmod** Ändert die File-modi
- **who** Wer ist seit wann eingeloggt
- **env** Umgebungsvariablen

esse

User & Systemverwaltung

- **last** Letzte Logins
- **uptime** Letzter Besuch im Serverraum :-)
- **uname** Kernelname / Version
- **useradd** Hinzufügen eines Users
- **userdel** Löschen eines Users

Netzwerkbefehle

- **ifconfig** Verwaltung v. Netzwerk-devices
- **ping** Sinnvolles von ICMP
- **traceroute** Zeigt die Route zu einem Host
- **telnet** Unsichere Verbindung
- **ftp** FTP Client

Netzwerkbefehle

- **ssh** Secure Shell
- **netcat** Schweizertaschenmesser bzgl. TCP/IP
- **nmap** Portscanner
- **lynx** Textbasierter Web-Browser

esse

Netzwerkbefehle

- Warum lynx?
- Tauglichkeit zur Durchführung von Security Analysen
- Sicherer da kein Exploids durch JavaScript oder ActiveX Code ausgeführt werden können.

Beispiele

```
# traceroute stud3.tuwien.ac.at
traceroute to stud3.tuwien.ac.at (193.170.75.13), 30 hops max, 40 byte packets
 1  pnsgw1.demo.tuwien.ac.at (128.131.200.1)  44.657 ms  23.683 ms  5.913 ms
 2  winnetou-792.kom.tuwien.ac.at (128.131.192.1)  51.300 ms  4.227 ms  7.877
ms
 3  sw-d-7.kom.tuwien.ac.at (128.131.5.2)  31.937 ms  25.202 ms  14.911 ms
 4  sw-d-4.kom.tuwien.ac.at (192.35.243.37)  51.705 ms  12.733 ms  19.520 ms
 5  stud3.tuwien.ac.at (193.170.75.13)  8.733 ms  11.686 ms  6.033 ms
```

Beispiele

bash-3.00# nmap -sS [redacted]
Starting nmap 3.50 (http://www.insecure.org/nmap/) at 2005-10-14 14:54 CEST
Interesting ports on [redacted]
(The 1653 ports scanned but not shown below are in state: filtered)
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
80/tcp open http
110/tcp open pop3
443/tcp closed https
Nmap run completed -- 1 IP address (1 host up) scanned in 84.783 seconds

Shell Symbole

- > Ausgabe umleiten
- >> Ausgabe an Datei anhängen
- < Eingabe umleiten
- << „Here“-Dokument (Eingabe umleiten)
- & Prozess im Hintergrund starten
- | Ausgabe weiterleiten
- && Und (Return-Wert ausschlaggebend)
- || Oder (Return-Wert ausschlaggebend)

Shell Symbole

- $0 Befehlsname
- $$ Prozessnummer
- $var Wert der Variable var
- $n n-tes Argument
- $* Alle Argumente als einfaches Wort
- ; Trennt Befehle in einer Zeile
- # Kommentar

esse

Umleitungen der Ausgabe

- **1 > filename**
  Leitet die Standardausgabe nach filename um
- **1 >> filename**
  Leitet die Standardausgabe nach filename um und erweitert diese
- **2 > filename**
  Leitet die Standardfehlerausgabe nach filename um
- **2 >> filename**
  Leitet die Standardfehlerausgabe nach filename um und erweitert diese
- **& > filename**
  Leitet die Standardausgabe und Standardfehlerausgabe nach filename um

Shell Magic

- **!command**
  Führt den letzten Befehlsaufruf von command mit den zuletzt benutzten Argumenten auf
- **!**
  Führt den letzten Befehl aus
- **!-2**
  Führt den vorletzten Befehl aus
- **!-n**
  Führt den n-en vorergehenden Befehl aus

esse

Shell Magic

- **!^** oder auch **!:1**
  Erstes Argument des letzten Befehls
- **!:2**
  Zweites Argument des letzten Befehls
- **!:n**
  n-tes Argument des letzten Befehls
- **!:$**
  letztes Argument
- **!-2:0 !:$**
  Führt den vorletzten Befehl mit dem letzten Argument des letzten Befehls aus

Telnet

- Sinnvolles mit telnet:

```
jvr@rocksolid ~ $ telnet stud3.tuwien.ac.at 22
Trying 193.170.75.13...
Connected to stud3.tuwien.ac.at.
Escape character is '^]'.
SSH-1.99-3.2.3 SSH Secure Shell (non-commercial)
```

esse

History

- Man arbeitet auf einem Rechner rum und will nicht das die history nach dem ausloggen geschrieben wird.
- Mit echo $$ die PID der aktuellen Shell ausgeben lassen
- Zweite Shell öffnen
- Mit kill PID die erste Shell töten
- Suprise! Keine history der ersten Shell vorhanden.

Weitere Informationen

- http://www.kefk.net/SelfLinux/html/bash_basic.html
- http://www.tnt-computer.de/yanip/lbefehle.html
- http://www.ruwela.de/
- http://www.linuxfibel.de/kapitel2.htm
- http://suedpol.physik.ruhr-uni-bochum.de/beschreibung/
- http://www.linux.org

- Man pages!

esse

EXIT

```
echo thanxs for all the fish
touch somemagic
ls | grep somemagic | cut -c 4 > !:1
security=`!-3 | cut -c 5,21`t
$(echo "`cat somemagic`$security  0")
```