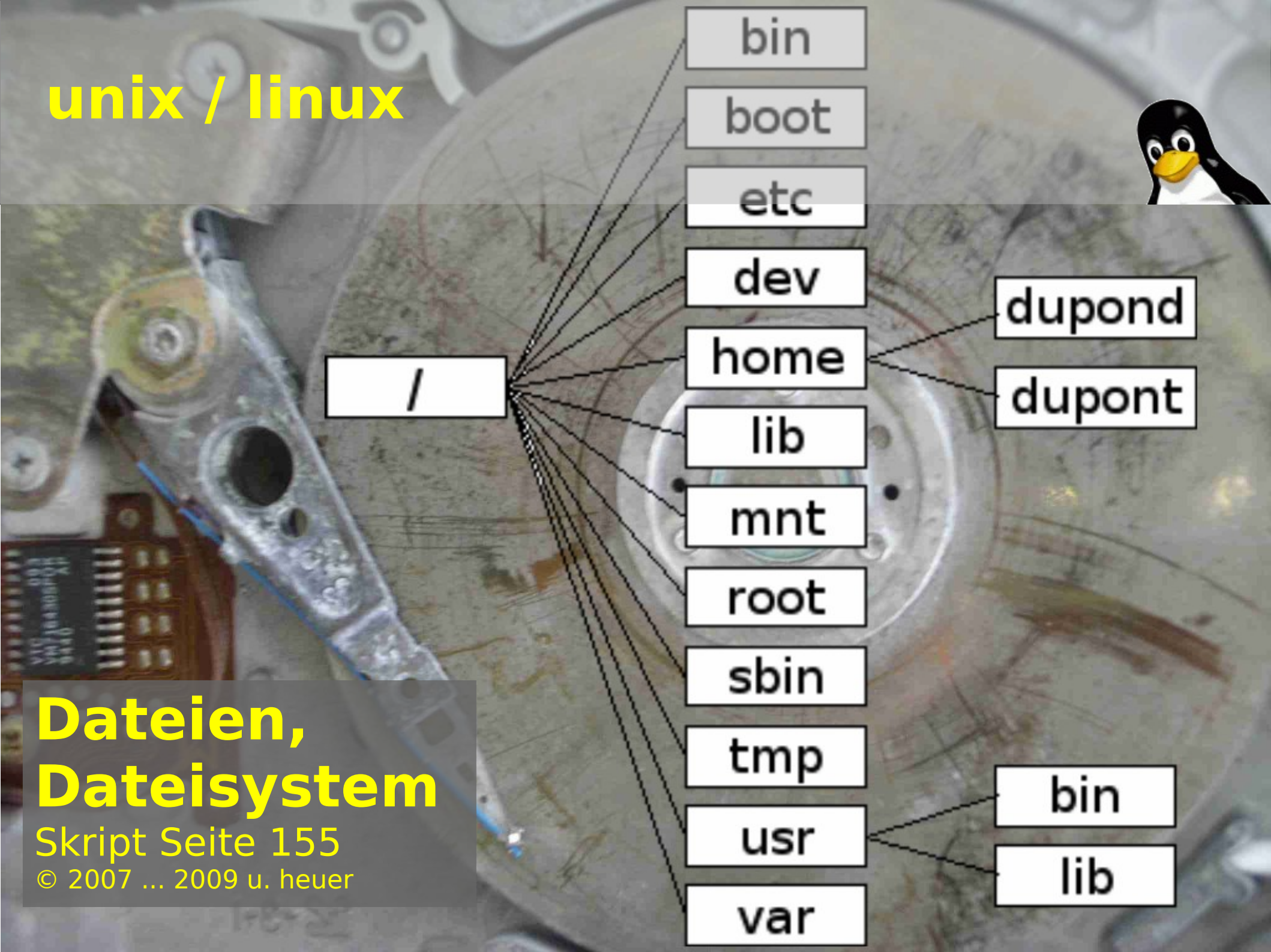
unix / linux



Dateien, Dateisystem

Skript Seite 155

Ziele

- Die Begriffe "Datei" und "Dateisystem" verstehen
- Die verschiedenen Dateitypen kennen
- Sich im Verzeichnisbaum eines Linux-Systems zurechtfinden
- Wissen, wie externe Dateisysteme in den Verzeichnisbaum eingebunden werden

Begriffe Datei

- **Datei**: (aka file)
  Eine Datei ist eine in sich geschlossene Sammlung von Daten.

  Beispiele: Text, Musik, Filme, Bilder, strukturierte Daten (Datenbank), Harddisk, ...

Begriffe Dateisystem

- **Dateisystem**: (aka filesystem)
  Ein Dateisystem legt fest, nach welcher Methode die Daten auf den Datenträgern angeordnet und verwaltet werden.
  Beispiele: ext2, ext3, ReiserFS, XFS, JFS, ...
- Die Hierarchie wie Dateien im Verzeichnisbaum abgelegt werden wird auch **Dateisystem** genannt.

Dateitypen

"**Alles ist eine Datei**".

Es werden 6 verschiedene Dateitypen unterschieden:

- Normale Files
- Verzeichnisse
- Symbolische Links
- Gerätedateien
- FIFO
- Unix-Domain-Sockets

plain files

Normale Files

Zu dieser Gruppe gehören Texte, Filme, Grafiken, Audiodaten, ...
aber auch ausführbare Programme.

`find: -type f;  ls -l:  -rwxrwxrwx`
`touch FILE`

directory

Verzeichnisse

Verzeichnisse strukturieren den Speicherplatz von einem Dateisystemen.

`find: -type` **`d`**`;  ls -l:  `**`d`**`rwxrwxrwx`

`mkdir DIRECTORY`

symbolic link

symbolische link

Symbolische Links verweisen auf andere Dateien.

`find: -type l;  ls -l:  lrwxrwxrwx`
`ln -s SOURCE [TARGET]`

devices

Gerätedateien

Devices sind die Schnittstelle zu den entsprechenden Treibern im Kernel.

Es gibt zwei unterschiedliche Devices:

- **Block Devices**
- **Character Devices**

Devices können mit mknod erstellt werden

block devices (buffered)

block devices arbeiten immer mit fixen 512B, 1024B, 2048B, ... Blöcken von Daten.

Beispielsweise werden von einer CD, Floppy, Harddisk immer Blöcke von 512Byte oder eine mehrfaches davon gelesen oder geschrieben.

block devices sind in der Regel nur bei den Massenspeicher Medien wie Harddisks, CDROM, DVD, Floppy, USB-Stick, ... verwendet.

`find: -type b;  ls -l:  brwxrwxrwx`
`mknod NAME b MAJOR MINOR`

character devices (unbuffered)

character devices können mit beliebig grossen, variablen Datenblöcken umgehen. Beispielsweise macht es keinen Sinn bei einer Tastatur darauf zu warten, bis der User 512 Tasten betätigt hat.

Alle Devices – mit Ausnahme der Massenspeicher – sind character devices

```
find: -type c;   ls -l:   crwxrwxrwx
mknod NAME c MAJOR MINOR
```

FIFO; aka named pipe

FIFO

Die FIFO verbindet 2 beliebige, lokale Prozesse miteinander - ähnlich wie die "|" bei de Shell.

Der Unterschied ist, dass bei einer FIFO Zugriffsrechte vergeben werden können und dass alle Prozesse die Möglichkeit haben die FIFO zu verwenden.

`find: -type p;  ls -l:  prwxrwxrwx`
`mknod NAME p`

Socket

Unix-Domain-Sockets verhalten sich ähnlich wie FIFOs.

Der Unterschied ist, dass sie sich wie die TCP/IP-Netzwerkschnittstelle verhalten, mit der Einschränkung, dass die beteiligten Prozesse auf dem gleichen Rechner laufen müssen.

`find: -type s;  ls -l:  srwxrwxrwx`

Der Socket wird vom verarbeitenden Program während der Laufzeit erstellt. Bei Programmende wird der Socket wieder gelöscht

Dateitypen

Typ	ls -l	ls -F	Anlegen mit …
Normale Datei	-		Diverse Programme
Verzeichnis	d	/	mkdir
sym. Link	l	@	ln -s
Gerätedatei	b,c		mknod
FIFO(named pipe)	p	|	mknod, mkfifo
Socket	s	=	

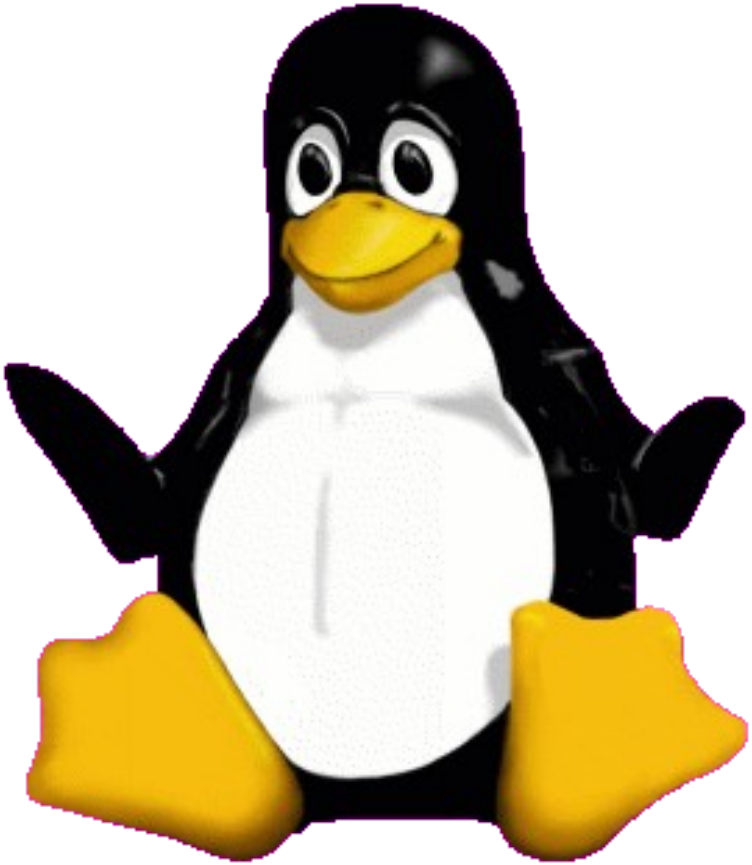
file

Welche Daten ein File enthält wird mit den '**magic bytes**' am Anfang vom File bestimmt.

Die meisten Fileformat schreiben ganz am Anfang vom File ein paar fixe Zeichen: Beispielsweise steht bei PDF-Dateien immer "%PDF-<version>" am Anfang der Datei.

Der Befehl `file [datei]` kann diese auslesen und anzeigen was für eine Art Files die Dateien sind.

Fragen ?



Übungen

8.1 Suchen Sie in Ihrem System nach Beispielen für die verschiedenen Dateitypen. (Tabelle 8.1 zeigt Ihnen, woran Sie die betreffenden Dateien erkennen können.)

Tipp: Verwenden Sie **`find`** um entsprechende Dateitypen zu finden.

Dateinamen

Unix verwendet intern die Inode-Nummern um Dateien zu unterscheiden.

Der Benutzer verwendet Dateinamen.

Die Abbildung von Dateienamen auf die Inode-Nummer erfolgt Mithilfe der Directory-Einträge.

Dateinamen

Ein Dateiname darf:

- maximal 255 Zeichen lang sein
- alle Zeichen ausser '**/**' und '**\0**' (Null Byte) enthalten.

Dateinamen

UNIX unterscheidet Gross- und Kleinschreibung

UNIX verlangt keine bestimmte Endung für einen bestimmten FileTyp.

☞ Die Verwendung von Endungen (extensions) ist jedoch sehr hilfreich und nicht verboten.

☞ Werden Dateien auf einem nicht UNIX-Filesystem gespeichert, sind oft die Regeln vom entsprechenden Filesystem bestimmend.

Dateinamen

Um Dateinamen portabel zu halten, sollten die Namen am besten nur folgende Zeichen enthalten: **`A-Za-z0-9`**

Zusätzlich können innerhalb vom Namen folgende Zeichen verwendet werden: **`+-._`**

Dateinamen

Neben dem Dateinamen ist auch die Position innerhalb von Dateibaum relevant.

Ein absoluter Pfad inkl. Dateinamen referenziert eine Datei eindeutig.

Ein relativer Pfad mit Dateiname kann mehrere unterschiedliche Dateien referenzieren - je nach dem was ihr aktuelles Verzeichnis ist.

➔ vergleiche Kapitel 5.2.1

Filesystem Hierarchy Standard

Wo welches File abgelegt wird ist im Filesystem Hierarchy Standard (FHS) beschrieben. [1]

Ber FHS beschreibt die erste Hierarchie Stufe und eine zweite Stufe unterhalb von den Verzeichnissen `/usr` und `/var`

Der Nutzen davon ist, dass sie sich auf allen UNIX-Systemen - die dem FHS folgend - sie sich problemlos zurecht finden. Sie finden die Dateien an dem Ort wo sie die Datei erwarten!

[1] http://www.pathname.com/fhs/

Filesystem Hierarchy Standard

```
largo:/$ ls -l
total 74
drwxr-xr-x   2 root root  2880 Jun 16 17:47 bin
drwxr-xr-x   3 root root   944 Jun 10 13:34 boot
drwxr-xr-x  13 root root 39768 Jun 10 13:36 dev
drwxrwxr-x 143 root root  8568 Jun 16 17:47 etc
drwxr-xr-x  66 root root  1976 May 13 12:29 home
drwxr-xr-x  12 root root  4136 Jun 16 17:46 lib
drwxr-xr-x   2 root root  1656 Jun 16 17:46 lib64
drwxr-xr-x   4 root root   144 Jun 20  2005 media
drwxr-xr-x   2 root root    48 Nov  3  2003 mnt
drwxr-xr-x  20 root root   616 May  5 22:27 opt
dr-xr-xr-x 191 root root     0 Jun 10 13:35 proc
drwxr-xr-x  17 root root  1784 Jun 19 17:13 root
drwxr-xr-x   2 root root  5032 Jun 16 17:47 sbin
drwxr-xr-x   2 root root    48 Jun 20  2005 srv
drwxr-xr-x  11 root root     0 Jun 10 13:35 sys
drwxrwxrwt  10 root root  9792 Jun 19 22:15 tmp
drwxr-sr-x  16 root root   408 Jul 18  2005 usr
drwxr-xr-x  24 root root   600 May  6 13:41 var
```

/boot

Im Verzeichnis **/boot** liegt das Betriebssystem im engeren Sinne: vmlinuz ist der Kernel von Linux. Im Verzeichnis **/boot** finden sich ausserdem statische Dateien, die für den Bootlader (LILO oder GRUB) von Bedeutung sind.

/boot kann eine eigene, kleine Partition sein, um die 1024-Zylinder Grenze von älteren BIOS zu umgehen.

/bin

Unter **/bin** befinden sich die wichtigsten ausführbaren Programme (meist Systemprogramme), die unbedingt zum Starten des Systems notwendig sind.

Dazu gehören beispielsweise `cat, chgrp, mount, mkdir` …, elementare Linux-Kommandos, die in der Regel von allen Benutzern ausgeführt werden dürfen.

Programme, die beim Start nicht unbedingt gebraucht werden, finden sich unter **/usr/bin**

/sbin (System binaries)

Unter **/sbin** stehen Kommondos, die eigentlich nur root ausführen kann und wichtig sind um den Rechner zu booten.

"Normale" Benutzer können mit manchen dieser Programme Informationen abfragen.

Analog zu `/bin` gibt es auch ein Verzeichnis `/usr/sbin`, wo weitere System-Programme zu finden sind.

/lib (libraries; kernel modules)

Unter **/lib** befinden sich *shared libraries* der Programme die in `/bin` bzw `/sbin` liegen

Die Kernel-Module liegen unter **/lib/modules/$(uname -r)/**

/dev (devices)

In `/dev` und seinen Unterverzeichnissen befinden sich die Gerätedateien.

Die Gerätedateien sind die Schnittstellen zu den Gerätetreibern. Sie haben keinen "Inhalt" wie andere Dateien, sondern verweisen auf einen Treiber im Kernel.

Bei modern Distribution erscheinen im **/dev** nur die effektiv vorhanden Device-Files
siehe `udev(8)`

/dev (devices)

/dev/null	Virtueller Papierkorb
/dev/zero	NullBytes
/dev/random	Random Generator
/dev/urandom	Pseudo Random Generator
/dev/full	eine virtuelle Disk, die voll ist
/dev/mem	physikalisches Memory
/dev/kmem	virtuelles Memory
/dev/port	IO-Ports

/etc (system configuration)

Das Verzeichnis **/etc** ist sehr wichtig, denn hier befinden sich die Konfigurationsdateien für die allermeisten Programme.

In **/etc/inittab** und **/etc/init.d/*** beispielsweise stehen die meisten der system-spezifischen Daten, die zum Booten erforderlich sind.

Die wichtigsten Dateien werden hier etwas detaillierter erklärt. Mit wenigen Ausnahmen hat nur der Benutzer root Schreibrechte, aber jeder Benutzer Leserechte.

/etc (system configuration)

Wichtige Files in /etc:

`/etc/fstab`	Filesystem Tabelle
`/etc/hosts`	Hosts IPAdressen Tabelle
`/etc/inittab`	Konfiguration vom **`init`** Prozess
`/etc/init.d/*`	Start-, Stop-Skripte
`/etc/issue`	allgemeine Hinweise die **vor** dem Login angezeigt werden.
`/etc/motd`	allgemeine Hinweise, die **nach** dem Login angezeigt werden.
`/etc/mtab`	mount Tabelle, was wurde gemountet
`/etc/passwd`	UserDatenbank

/opt (Add-on Software)

Dieses Verzeichnis ist dafür gedacht, dass Drittanbieter fertige Softwarepakete anbieten können, die sich installieren lassen sollen, ohne mit den Dateien einer Linux-Distribution oder den lokal angelegten Dateien zu kollidieren.

Solche Pakete belegen ein Unterverzeichnis **/opt/{Paketname}**

```
lrwxrwxrwx  1 root root  12 2007-02-10 00:20 httpd -> httpd-
2.0.59
drwxr-xr-x 13 root root 312 2006-06-11 01:01 httpd-2.0.58
drwxr-xr-x 13 root root 312 2006-09-10 00:57 httpd-2.0.59
```

/usr (statische, read-only)

Hier findet man alle möglichen Dateien,unter anderem Anwendungen die für das System nicht die höchste Priorität haben.

Der Name wird häufig als "Unix system resources" interpretiert

In **/usr** finden sich in diversen Unterverzeichnissen sämtliche Programme, die nicht direkt zum System gehören.

/usr (statische, read-only)

Die Dateien sind in Prozessor-Architektur abhängige Files (Programme, Libs) und Prozessor-Architektur unabhängige Files (Docu, man-pages, ...) unterteilt. Die Archiktur unabhängigen Files sind im Verzeichnis `/usr/share/` abgelegt.

Diese Unterteilung vereinfacht die Erstellung von Software-Paketen.

/usr (statische, read-only)

/usr/X11R6	X-Window System
/usr/bin	Systemprogramme
/usr/sbin	SystemAdminProgramme
/usr/lib	Shared Libraries
/usr/local	Lokale Installationen
/usr/share	Rechenarchitektur unabhängige Files
/usr/share/doc	Dokumentationen, HOWTOs
/usr/share/info	info Seiten
/usr/share/man	manPages
/usr/src	Source Code vom Kernel

/proc (Fenster zum Kernel)

Im /proc Verzeichnis finden Sie sämtliche Informationen zu den laufenden Prozessen und ausserdem weitere Informationen, die der Kernel über die Hardware des Rechners besitzt. In einigen Dateien finden Sie zum Beispiel eine komplette Hardwareanalyse des Rechners.

/proc (Fenster zum Kernel)

/proc ist ein PseudoFilesystem.
Es sieht aus wie ein Filesystem, es belegt aber keinen Platz auf der Festplatte.

Die Dateien werden vom Kernel generiert, wenn sie aus der "Datei" gelesen werden bzw. der Kernel speichert die Daten, die in eine Datei geschrieben werden als Konfigurations Parameter vom entsprechenden System ab.

/proc (Fenster zum Kernel)

`/proc/cpuinfo`	welche CPUs sind da
`/proc/devices`	vom Kernel unterstützte Devices
`/proc/interrupts`	Hardware Interrupts
`/proc/ioports`	belegte IO-Ports
`/proc/kcore`	Ein Abbild des ganzen Memory
`/proc/loadavg`	ein Mass für die Auslastung
`/proc/meminfo`	Wie wird das Memory verwendet
`/proc/net/`	Neztwerk Infos
`/proc/scsi/`	SCSI Informationen
`/proc/sys/`	Tuning des Kernels (!)
`/proc/version`	Kernel Version
`/proc/bus/`	Informationen zu den PCI-, USB-, ... Bus-Systemen im Rechner

/proc (Fenster zum Kernel)

Für jeden laufenden Prozess gibt es auch einen Eintrag in **/proc**. Die wichtigsten Files bzw Verzeichnisse sind folgende:

/proc/{PID}/	Prozess spezifische Daten:
cmdline	Befehlszeile
cwd	CWD vom Prozess
environ	Environment des Prozesses
exe	Programm das ausgeführt wird
fd/	Offene Dateien
status	Statusinformationen vom Prozess

/var (variable Dateien)

Hier befinden sich veränderliche Dateien, verteilt auf verschiedene Verzeichnisse.

Unter /var werden

/var/spool/	(Mail, Drucker)
/var/log/	Log-Dateien
/var/tmp/	temporäre Dateien
/var/caches	Caches
/var/run	Prozess Informationen

abgelegt

/tmp (temporäre Dateien)

Viele Programme brauchen temporären Speicherplatz. In **/tmp** können beliebige Programme temporäre Dateien ablegen.

Beim booten wird **/tmp** gelöscht oder neu formatiert. Speichern sie also NIE Daten, die wichtig sind in **/tmp** !

/srv

Es gibt keinen klar definierten Ort, wo Web-, FTP- oder andere Server ihre Daten holen bzw speichern sollen.

/srv ist ein Versuch dies nachzuholen.

Das FHS-Dokument definiert dass in **/srv** die lokalen Daten enthält, die von diesem Rechner angeboten werden (per http, https, ftp, ...)

Alternativ sind diese Daten oft im Home-Verzeichnis eines entsprechenden funktionellen Users.

/media

Das Verzeichnis **`/media`** enthält weitere, leere Verzeichnisse, etwa **`cdrom`**, **`floppy`**, **`usbstick`**, ... die als Mountpoint für die entsprechenden Medien dienen.

Je nach Distribution, Window-Manager, ... können weitere Unterverzeichnisse vorhanden sein bzw. bei bedarf erstellt werden.

/mnt

Dieses leere Verzeichnis dient zum kurzfristigen Einbinden weiterer Dateisysteme.

Bei manchen Distributionen, etwa denen von Red Hat, können sich hier (und nicht in /media) Verzeichnisse als Mountpunkte für CD-ROM, Floppy, ... befinden.

/home

Unter dem Verzeichniss **`/home`** befinden sich die Home-Directories aller Benutzer.

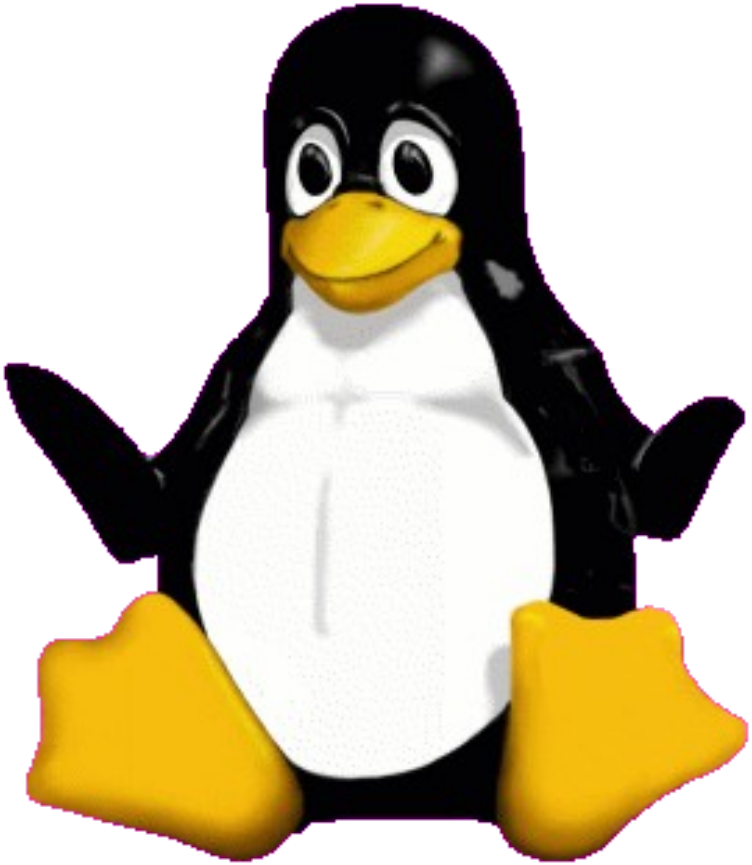
Die einzige Ausnahme davon ist der User `root`, der ein eigenes Verzeichnis (/root) hat.

/root

Unter /root befindet sich das Home-Directory vom SuperUser root.

Der User root besitzt immer ein **lokales** Home-Directory, so dass er immer auf seine Daten zurück greifen kann, auch wenn /home von einem anderen Rechner gemountet wird.

Fragen ?



Übungen

8.2 Wie viele Programme enthält Ihr System an den "gängigen" Plätzen?

8.3 Wird grep mit mehr als einem Dateinamen als Parameter aufgerufen, gibt es vor jeder passenden Zeile den Namen der betreffenden Datei aus.

Dies ist möglicherweise ein Problem, wenn man grep mit einem Shell-Dateisuchmuster (etwa "*.txt") aufruft, da das genaue Format der grep-Ausgabe so nicht vorhersehbar ist und Programme weiter hinten in einer Pipeline deswegen durcheinander kommen können.

Wie können Sie die Ausgabe des Dateinamens erzwingen, selbst wenn das Suchmuster nur zu einem einzigen Dateinamen expandiert?

(Tipp: Eine dafür nützliche "Datei" steht im Verzeichnis /dev.)

Übungen

8.4 Das Kommando "**`cp bla.txt /dev/null`**" tut im wesentlichen nichts, aber das Kommando "**`mv bla.txt /dev/null`**" – entsprechende Zugriffsrechte vorausgesetzt – ersetzt **`/dev/null`** durch **`bla.txt`**.

Warum?

Übungen

8.5 Welche Softwarepakete stehen auf Ihrem System unter `/opt`?

Welche davon stammen aus der Distribution und welche von Drittanbietern?

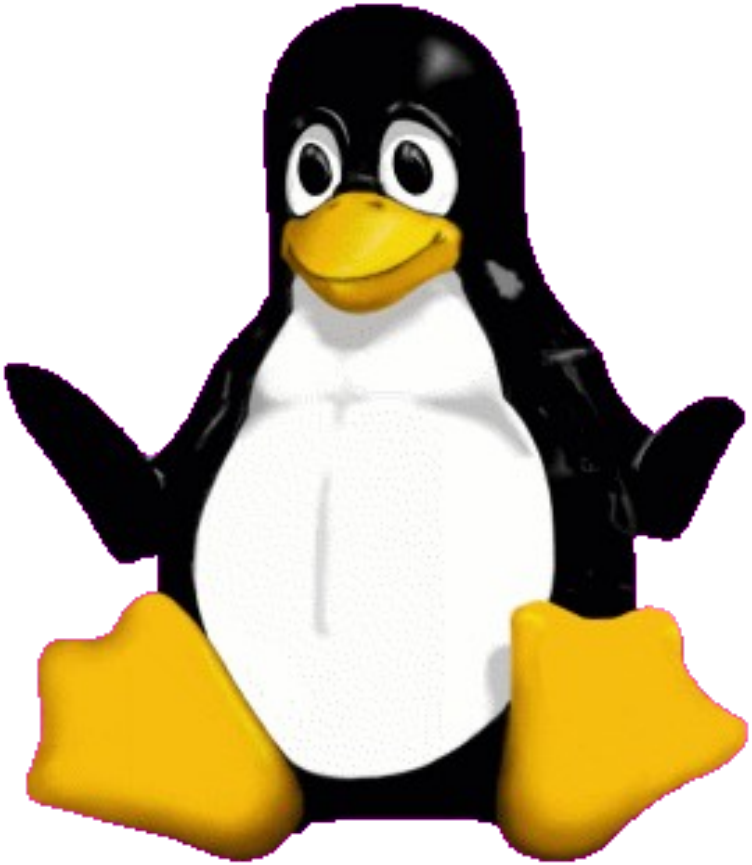
Sollte eine Distribution eine "Schnupperversion" eines Drittanbieter-Programms unter /opt installieren oder anderswo?

Was halten Sie davon?

Übungen

8.6 Warum ist es unklug, Sicherheitskopien des Verzeichnisses **`/proc`** anzulegen?

Fragen ?



mount

Die Datenträger können direkt via den entsprechenden Device-Files angesprochen werden.
Das ist OK Medien ohne FileSystem wie beispielsweise bei Bandlaufwerken.

Bei Medien, die ein FilesSystem enthalten muss der Kernel die Zugriffe steuern. Nur in Ausnahmefällen sollen sie auf das Medium zugreifen!

Dem Kernel muss wissen an welchem Ort im Verzeichnisbaum ein Datenträger montiert werden soll.

Diesen Vorgang nennt man **mounten**.

mount

Der Befehl **mount** dient zum einhängen der Datenträger in den Verzeichnisbaum.

```
mount {Optionen} device directory
```

oder wenn es einen Eintrag in /etc/fstab gibt

```
mount {Optionen} device | directory
```

Beispielsweise:

```
mount -t msdos /dev/fd0 /mnt/floppy
```

mount

Grundsätzlich kann nur der SuperUser den Befehl mount verwenden.

In der Datei `/etc/fstab` kann spezifiziert werden, dass auch ein User **`mount`** für einzelne Dateiträger (Floppy, CDROM, USB-Stick, ...) ausgeführt werden kann.

Um einen Mount-Vorgang rückgängig zu machen verwenden Sie den Befehl **`umount`**:

```
umount device | directory
```

umount

Entfernen dürfen sie den Datenträger erst nach dem er unmountet ist. Beachten sie dies nicht, so können die Daten auf dem Träger zerstören werden.

Graphische Oberflächen wie KDE oder Gnome "überwachen" die speziellen Devices.
Wenn sie einen Datenträger finden, so mounten sie diesen automatisch.
Zum umounten bieten die Oberflächen entsprechenden Menüpunkte an.

/etc/fstab

In der Datei /etc/fstab sind die mount-Optionen für das lokale System abgelegt:

```
# /etc/fstab: static file system information.
#
#<file system> <mount point> <type>  <options> <dump> <pass>
/dev/mapper/largo-swap  none    swap     sw          0   0
proc                    /proc   proc    defaults     0   0
/dev/sda1               /       reiserfs notail      0   1
/dev/mapper/largo-var   /var    reiserfs defaults    0   1
/dev/mapper/largo-home  /home   reiserfs defaults    0   1

#
/dev/fd0   /media/fd0   auto   user,noauto,exec,umask=000  0 0
/dev/cdrom /media/cdrom auto   user,noauto,exec,ro         0 0
```

mtools

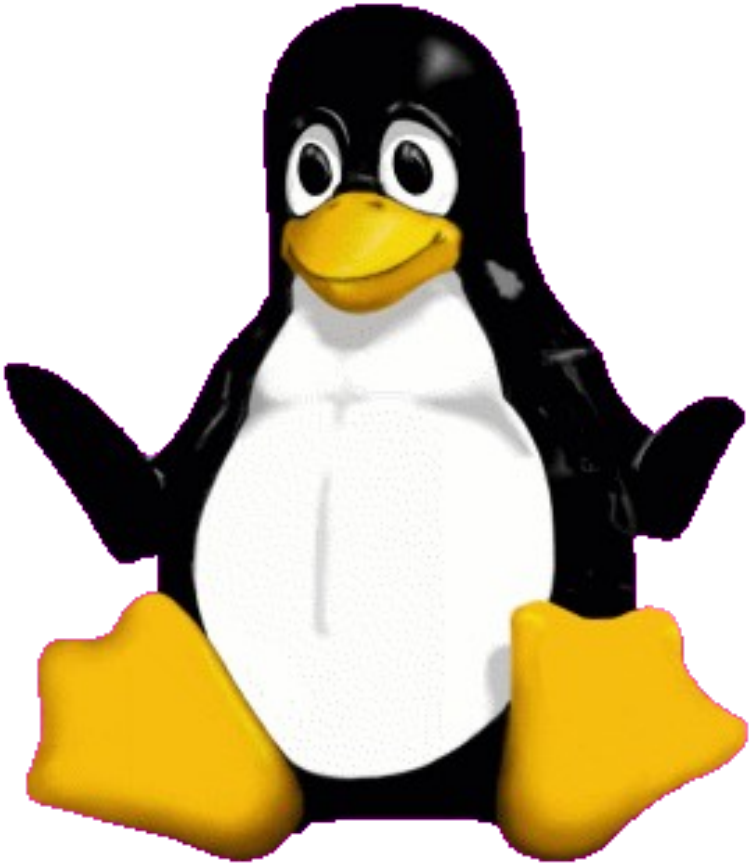
Um mit DOS-Disketten umzugehen können auch die **`mtools`** verwendet werden.

`mtools` implementiert die DOS-Befehle nach. Um diese zu verwenden muss man immer ein 'm' vorne anhängen:

copy ➔ mcopy
del ➔ mdel
format ➔ mformt
...

Lesen sie mtools(1). darin finden Sie auch Links ins Web

Fragen ?



Übungen

8.7 Legen Sie eine Diskette ins Laufwerk, hängen Sie sie ein, kopieren Sie eine Datei (zum Beispiel `/etc/passwd`) auf die Diskette und hängen Sie die Diskette dann wieder aus.

8.8 Lesen Sie in der Dokumentation zu mount(8) über die Mount-Option `user` nach. Welche Einschränkungen werden für von Benutzern einhängbare Medien gemacht und warum?

8.9 Probieren Sie die Mtools aus (siehe mtools(1)) und vergleichen Sie ihre Handhabung mit dem Einhängen von DOS-formatierten Disketten in den Linux-Dateibaum. Welche Methode ist Ihnen lieber?