

# Pfade durch Verzeichnisgestrüpp

Letzte Aktualisierung Monday, 7. May 2007

## Verzeichnisse

Wir bedienen uns zur besseren Illustration des ls-Befehls, der im Anschluss näher beschrieben wird!

```
user@host:~> ls
```

"ls" steht für das englische Wort listing. Dieser Befehl listet Ihnen also den Inhalt von Verzeichnissen auf. Der Inhalt eines Verzeichnisses ist entweder ein anderes Verzeichnis oder eine Datei.

Ohne Angabe einer Option bzw. eines Parameters, wird jenes Verzeichnis simpel aufgelistet, in dem Sie sich gerade befinden!

Aber wo befinden Sie sich gerade?

Erinnern Sie sich, wir haben schon den Befehl pwd kennengelernt und es ist an der Zeit, ihn hier einzusetzen ...

```
user@host:~> pwd
/home/user
user@host:~>
```

"pwd" steht für "print working directory - Gib das aktuelle Arbeitsverzeichnis aus!". Die Ausgabe dieses Befehls ergibt immer eine Absolute Pfadangabe.

/home/user ist das sogenannte "Home-Verzeichnis" des Benutzers namens "user". In GNU/Linux, Unix und/oder anderen Unix-Derivaten (z.B. FreeBSD) wird das Home-Verzeichnis des users mit ~ (=Tilde) abgekürzt! Deshalb sehen Sie auch im Eingabeprompt als Absolute Pfadangabe anstelle von /home/user eben dieses ~-Zeichen ausgegeben. Was Ihnen bei der Ausgabe von pwd völlig (als Linux-Neuling) abgehen wird ist die Angabe eines Laufwerksbuchstabens als oberste Hierarchiestufe!

Sie werden C:> oder D:> gewohnt sein, das von den Microsoft-Leuten eingeführt wurde. Absolute Pfadangabe bedeutet, dass die Pfadangabe vom obersten Hierarchiepunkt ausgehend bis zum aktuellen bzw. gewünschten Verzeichnisnamen reicht. Später werden wir auch noch die Relative Pfadangabe kennenlernen.

Jede Absolute Pfadangabe unter GNU/Linux beginnt mit einem "/" (Slash). Der Slash ist der "Verzeichnistrenner", und das ist ganz wichtig sich zu merken!

Der Slash trennt zwei Verzeichnisse voneinander ab, das bedeutet, dass JEDER Verzeichnisname bei einer Pfadangabe am Anfang und am Ende einen Slash stehen haben kann. Der Slash vor dem Verzeichnisnamen MUSS sein, der nach dem Verzeichnisnamen ist genau dann optional, wenn kein weiterer Verzeichnis- oder Dateinamen angegeben werden muss, weil der letzte Verzeichnisname das "Zielverzeichnis" der Absoluten oder Relativen Pfadangabe benennt (also das, wo Sie final hin wollten) bzw. der letzte Name die von Ihnen angepeilte (Ziel)Datei darstellt. Das mit dem Slash ist wichtig und sollte sich jeder einprägen. Am Anfang sicher gewöhnungsbedürftig.

Der Bereich links vom allerersten Slash ist die root-Partition. Root ist das englische Wort für Wurzel (in unserem Fall passte auch die Übersetzung "Ur-Grund") und eine Partition ist ein Teilbereich der Festplatte! Jedes GNU/Linux-System hat eine root-Partition. Ohne einer solchen geht nichts! Man könnte es mit einem Plafond vergleichen, in dem Haken eingeschraubt wurden. Der Plafond ist die root-Partition. Die

angeschraubten Haken sind die mit jeweils einem Namen versehenen und mit jeweils einem Slash von der Root-Partition getrennten "Mountpoints". Ein solcher "Haken" kann nun weitere Verzeichnisse am Haken haben, oder Dateien, einen Luster oder eine andere Partition, die in diesen Haken eingehängt wurde! Was immer er am Haken hat, wird auch durch einen Slash getrennt. Der Haken an dieser Geschichte ist, dass es keinen gibt! Wir werden uns später mit diesem Thema ernst auseinandersetzen! Grundsätzlich haben Sie aber das Bild vom Plafond mit seinen Haken und Ösen verstanden, oder?

Um nun alle im Plafond angebrachten Hakennamen sich anzeigen zu lassen, tippen Sie den ls-Befehl gefolgt vom allerersten Verzeichnistrenner, der die obersten Verzeichnisse von der root-Partition trennt.

```
user@host:~> ls /
```

```
bin dev fs lib lost+found misc mtp opt root usr work srv
boot etc home local media mnt work-local proc sbin tmp var
```

```
user@host:~> _
```

Die meisten dieser Namen sind nicht rein zufällig gewählt worden, und Sie werden sich schon gefragt haben, was diese zu bedeuten haben.

Root-Verzeichnis. Der namenlose Bereich links vom ersten Slash, ist der Beginn des Verzeichnisbaumes

*/bin/*

enthält wichtige Befehle, die auch zum Hochfahren des Systems benötigt werden sind hier abgelegt (bash, ls, mount ...)

*/boot/*

enthält statische files für den Boot-Loader (z.B. vmlinuz)

*/dev/*

Geräte-dateien (device-files), die Hardware-Komponenten repräsentieren.

*/etc/*

wichtige Dateien zur Systemkonfiguration, die auch beim Booten des Rechners Beachtung finden

*/home/*

Die (privaten) Verzeichnisse der Benutzer

*/lib/*

enthält "shared libraries" für dynamisch gelinkte Programme und die Kernel-Module

*/opt/*

optionale Software (KDE, GNOME, Netscape ...)

*/proc/*

Das Prozessdateisystem - bietet ein "Sicht-Fenster in den Kernel"

*/root/*

Home-Verzeichnis vom Superuser (/root/ NICHT mit root-partition verwechseln!)

*/sbin/*

enthält wichtige "System"-Kommandos, die dem Superuser gehören

/tmp/

Temporäre Dateien --- jeder user darf temporäre files in diesen Ordner legen

/usr/

Ist ein Basisverzeichnis für Anwendungsprogramme, Bibliotheken, Dokumentationen ...

/usr/bin/

Allgemein zugängliche Kommandos

/usr/local/

Lokale, von der Distribution unabhängige Erweiterungen

/usr/sbin/

Erneut Kommandos, die dem Superuser gehören

/usr/include/

Header-Dateien für den C-Compiler

/usr/share/doc/

Verschiedene Dokumentationsdateien

/usr/share/man/

Die Manual-Pages (Hilfetexte)

/usr/src/

Quelltexte der Systemsoftware

/usr/src/linux/

Der symbolische link zu den Kernelquellen (falls installiert)

/var/

Variable Daten: Mail und printer-spooler, log-files, pid-files, lock-files

/var/tmp/

Weitere Ablagemöglichkeit für temporäre Dateien

Bitte lassen Sie sich nicht dadurch verunsichern, dass Ihre Ausgabe am Bildschirm nicht deckungsgleich mit der hier gebotenen ist. Es ist alles okay mit Ihrem Computer!

Ich habe nur die wichtigsten aufgezählt. Sie werden wahrscheinlich auch ein sys oder ein media sehen, oder sogar ein work und mnt und vielleicht noch was anderes dazu... die oben aufgezählten waren mir am wichtigsten.

Was könnte es für einen Sinn haben, dass man weltweit diesen Standard etabliert hat, an den sich alle Distributoren mehr oder weniger brav halten?

Einerseits wird es für Sie als user oder superuser leichter sein, Dateien bzw. Verzeichnisse zu finden, weil ein xyz.conf file mit sehr hoher Wahrscheinlichkeit in /etc/ zu finden sein wird. Diese Vereinbarung dient einer besseren Orientierung durch das filesystem.

Andererseits wird es für Software-Entwickler so erst möglich, Programme (für GNU/Linux-Systeme) zu entwickeln.

Alle Linux-Distributoren halten sich an das Grundgerüst des "File-Hierarchy-Standards" (FHS)!

Für weiterführende Infos über den FHS besuchen Sie die Seite

<http://www.pathname.com/fhs/pub/fhs-2.3.html>

oder Sie laden sich den Inhalt als .txt-file oder pdf-file vom "Internet" auf Ihre Festplatte runter, indem Sie eines der files (txt, pdf, ...) auf der Seite <http://www.pathname.com/fhs/pub/> auswählen.

Das sind also mögliche Verzeichnis- oder Dateinamen gewesen (wir wissen das jetzt noch nicht) , die sich unmittelbar unterhalb der obersten Hierarchie - der root-Partition - befinden. Aus Gründen der Übersichtlichkeit sollten hier keine Dateien abgelegt werden!

Dateien in dieser Ebene wären sehr unüblich und unerwünscht! Ausser den im FHS angeführten Verzeichnissen können Sie auch andere Ihrer Wahl anlegen - je mehr, desto unübersichtlicher!

Der "bin" Haken ist also von der root-Partition und von seinem Gehalt mittels eines Slashes abgetrennt. Listen Sie doch einmal auf, was "bin" beinhaltet! (Die folgende Ausgabe beginnt unterhalb des Eingabeprompts und endet oberhalb des nächsten Eingabeprompts "a Stückl weida unten")

```
user@host:~> ls /bin/
Mail arch ash awk
bash bunzip2 bzip2
bzip2recover cat chgrp chmod
chown compress cp cpio
csh cut date dd
df dialog dircolors dmesg
dnsdomainname domainname du echo
ed egrep false fgrep
free ftp gawk gawk-3.1.4
getopt getoptprog grep gunzip
gzip head hostname ipmask
kill killall ksh ln
loadkeys login ls lsmod
lsmod.old mail mkdir mkfifo
mknod more mount mt
mt-GNU mt-st mv netstat
nisdomainname ping ping6 ps
pwd red rksh rm
rmdir rpm rzip sed
setterm sh shred sleep
sln stty su sulogin
sync tar tar-1.13 tar-1.15.1
tcsh telnet touch true
umount uname which ypdomainname
zcat zsh zsh-4.2.1
user@host:~>
```

Die Ausgabe muss bei Ihnen nicht exakt so aussehen. Es hängt nämlich der Inhalt der obigen Ausgabe auch vom Satz an Programmpaketen ab, welche auf Ihrem Computersystem lokal installiert wurden.

... oder das Verzeichnis /home/

```
user@host:~> ls /home/
...
...
```

```
user@host:~>
```

Unsere Eingabe (Keyboard) kommt über Kanal 0, und die Ausgabe auf dem Bildschirm kommt über Kanal 1.  
Jeder

Befehl hat nach seiner Ausführung einen Rückgabewert, der uns Auskunft darüber gibt, ob der Befehl erfolgreich ausgeführt wurde, oder nicht. Dieser Rückgabewert hat zwei Zustände: "OK" oder "NICHT OK" (true oder false). Bei erfolgreicher Ausführung ist der Rückgabewert 0 (=OK, true). Ansonsten (=NICHT OK, false) von 0 verschieden, z.B. 1, 13, 255

...

Die Befehlsfolge zur Ausgabe des Rückgabewertes lautet:

```
user@host:~> echo $?  
0  
user@host:~>
```

Dieser Rückgabewert bezieht sich immer nur auf den unmittelbar zuvor ausgeführten Befehl! In unserem Beispiel also "ls /home/".

Lassen Sie uns bewußt (oder unbewußt, weil Sie schon am einschlafen sind) einen Fehler produzieren, indem wir um die Ausgabe des Inhaltes eines Verzeichnisses bitten, das es gar nicht gibt:

```
user@host:~> ls /xyz/  
ls: /xyz: No such file or directory  
user@host:~> _
```

Das ist also eine Fehlermeldung, die uns vom System über Kanal 2 "ausgespuckt" wurde. Und was sagt uns das? Den Luster(haken) namens xyz gibt es nicht, oder den hat es gar nie gegeben. Die Abfrage des Rückgabewertes wird den Fehlschlag bestätigen ...

```
user@host:~> echo $?  
1  
user@host:~>
```

Sie sehen - der Rückgabewert des zuletzt eingetippten Befehls ist 1, für uns gleichbedeutend mit "Fehlschlag".

Der Befehl echo wird gerne zur Ausgabe von Texten auf den Bildschirm eingesetzt, und wir werden immer wieder "echo" einsetzen..

```
user@host:~> echo Hallo  
Hallo  
user@host:~> echo $?  
0  
user@host:~> echo Alea iacta est!  
Alea iacta est!  
user@host:~> _
```

Wieder schön zu beobachten:

Die Eingabe erfolgt übers Keyboard durch Kanal 0, die Eingabe erscheint uns am Ausgabegerät (Monitor, Braillezeile) quasi als Kontrolle des Getippten, was über Kanal 1 kam, genauso wie die Ausgabe des echo-Befehls auch über Kanal 1 ans Ausgabegerät geliefert wird. Es schadet überhaupt nicht, sich das immer wieder ins Gedächtnis zurück zu rufen!