

Tag_1_Feb25

February 21, 2025

1 Linux

2 Tag 1

3 Kapitel 1

Linux ist eigentlich ein Kernel (das Gehirn des Computers) und die Distribution ist dann das Betriebssystem (operating system OS).

Die Distributionen enthalten den Kernel und weitere Anwendungen. Welche Anwendungen vorinstalliert sind, hängt davon ab wofür diese Distribution gedacht ist. Eine Distribution, die rein für den Betrieb von Servern gedacht ist, braucht nicht dieselben Programme wie eine für den Heimgebrauch.

Es gibt sogenannte Familien von Distributionen, die verschiedene Paketmanager benutzen:

Debian/Ubuntu/Mint benutzen dpkg, apt und apt-get um deb-Pakete zu installieren.

Red Hat/Fedora/CentOS benutzen yum und dnf um rpm-Pakete zu installieren.

Ubuntu:

```
apt-cache search figlet # suche nach dem Paket figlet
sudo apt-get install figlet # Installation des Pakets figlet
sudo apt-get remove figlet # Deinstallation des Pakets figlet
```

4 Kapitel 2

4.0.1 Shell

Shell: Das ist die Kommandozeile

Ubuntu nutzt die Bash Shell, es gibt aber noch z.B. zShell, cShell und so weiter. Wir benutzen aber für alles die Bash Shell.

4.0.2 Bash

Bash ist auch eine Programmiersprache, konzipiert um möglichst effizient mit Dateien und Verzeichnissen zu arbeiten.

Jeder Befehl wird in der Regel sofort ausgeführt, wenn man Enter drückt. Genau wie in Python gibt es builtin Befehle.

4.0.3 prompt

NUTZERNAME@HOSTNAME: MOMENTANES VERZEICHNIS \$

Für einen normalen Benutzer

ROOT@HOSTNAME: MOMENTANES VERZEICHNIS #

Für den Superuser/root

Das \$ und Nutzernamen zeigt einen normalen Benutzer an und die # und root einen Superuser

Eine ~ zeigt an, daß man sich im Home-Verzeichnis des Nutzers befindet.

5 Befehle

Es geht jetzt erstmal darum die Kommandozeile zu zeigen und wie man mit ihr arbeiten kann.

Wenn man einen Filemanager wie Nautilus, Nemo oder sogar den Windows Explorer hat, dann kann man mit diesem gewisse Sachen machen, wie Dateien kopieren, löschen, erstellen und so weiter. All das kann man mit der Kommandozeile aber auch und sogar effizienter. Denn im GUI Dateimanager kann man nur EIN Verzeichnis nach dem nächsten Erstellen aber in der Kommandozeile kann man mit `mkdir` mehrere Verzeichnisse und sogar Unterverzeichnisse mit EINEM Befehl erstellen.

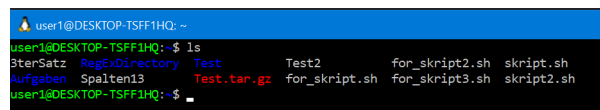
Zuerst werden einfach die häufigsten Datei- und Verzeichnisoperationen vorgestellt und beigebracht:

Typischer Befehlsaufbau:

Befehl OPTIONEN ARGUMENTE

Optionen verändern das Verhalten des Befehls und Argumente sind Dateien oder Verzeichnisse mit denen der Befehl arbeiten soll. Oft sind Optionen und/oder Argumente nicht notwendig.

5.0.1 ls - list



Dunkelblau sind Verzeichnisse, Rot sind Archive und Weiß sind alle anderen Dateien.

`ls` listet Inhalt des Verzeichnisses auf

`ls /expliziter Pfad`

Der explizite/absolute Pfad beginnt IMMER mit `/`!

`ls ~` oder `ls /home/USERNAME`

Gibt den Inhalt des Homeverzeichnisses aus EGAL in welchem Verzeichnis man gerade ist

`ls -a`

Versteckte directories/dateien (alles was einen Punkt davor hat ist versteckt (können trotzdem geöffnet werden)) `-a` ist eine Option von `ls`

`ls -l`

Gibt den Inhalt des Verzeichnisses im long format an:

```
drwxr-xr-x 1 user1 user1 4096 Aug 11 06:40 Music
-rw-r--r-- 1 user1 user1 1748 Aug 13 11:41 Datei1
```

d ganz Links zeigt ein VERZEICHNIS an.

\$\$ ganz Links zeigt eine Datei an

rwxr-xr-x sind die Rechte für den Besitzer, die Gruppe und die restlichen Benutzer

1 Anzahl an Links
user1 Besitzer
user1 Gruppe
4096 Größe in Bytes
Aug 11 6:40 Änderungsdatum
Music Datei- oder Verzeichnisname

Rechte: r - Lesen

w - Schreiben

x - Ausführen

— - Rechte nicht erhalten

Beispiel: rwxr-xr-x

Besitzer: darf lesen, schreiben und ausführen (die ersten 3 Zeichen)

Gruppe: darf lesen und ausführen, kein Schreiben, weil statt w hat man dort - (die Mittleren)

Rest: darf lesen und ausführen, kein Schreiben, weil statt w hat man dort - (die letzten Drei)

ls -lt sortieren der long Liste nach der Zeit, NEUSTE Datei zuoberst

ls -ltr sortieren der long Liste nach der Zeit, NEUSTE Datei ganz unten wegen r = reverse

ls | less Der Output von ls wird an less weitergegeben und kann dann seitenweise gelesen werden.
Das ist hilfreich, wenn man ein Verzeichnis mit hunderten an Dateien auflistet

ls -lhs

```
4.0K -rw-r--r-- 1 user1 user1 1.7K Aug 13 11:41 Datei1
```

4.0K ist der Festplattenspeicherverbrauch der Datei 1.7K ist die tatsächliche Dateigröße

Der Rest ist identisch zu oben.

exit (schließt Linux)

which

sucht den Speicherort der ausführbaren Datei zu einem Befehl

Jeder Befehl ist quasi eine ausführbare Datei, außer den Builtin Befehlen

`sudo` (superuser DO (durch # gekennzeichnet)) fragt nach passwort und logt dich als root (admin) ein kannst dann buchstäblich alles machen.

Hoch und runter Pfeiltasten gehen die Bashhistory (Befehle, die du zuvor eingegeben hast.) durch
`clear` oder `Strg + L` löscht den Bildschirm

5.0.2 `cd` - Change Directory

`cd ..`

Geht einen Schritt rauf also von `/home/username/` zu `/home/`

`cd ~` oder `cd /home/NUTZERNAME`

Gehe ins Homeverzeichnis des Nutzers EGAL wo du gerade bist: `/home/NUTZERNAME/`
WINDOWS `c:\Benutzer\NUTZERNAME`

`cd /`

Ganz rauf auf root egal wo du bist

Identisch zu WINDOWS `c:\`

`cd /user/bin`

Gehe ins Verzeichnis `/usr/bin`

`cd ../Music`

Gehe aus dem momentanen Verzeichnis heraus und in das Verzeichnis `Music`

`../Music` ist ein RELATIVER Pfad.

`/user/bin` ist ein ABSOLUTER Pfad. Diese beginnen IMMER mit `/`

5.1 Hilfe mit `man` oder `-help`

Befehl `-help`

Ruft eine "kurze" Hilfe auf mit den häufigsten Optionen

`man` Befehl

`man` ruft intern den `less` Befehl auf welcher eine Navigation der Hilfedatei ermöglicht! Ruft die Hilfedokumentation von `Befehl` auf. Man kann dort mit `G` ganz nach unten ans Ende der Datei und mit `g` an den Anfang gehen. `e` oder `Pfeil runter` geht eine Zeile weiter runter, `y` oder `Pfeil rauf` geht eine Zeile rauf. `f` oder `Bild runter` geht eine Seite runter und `b` oder `Bild rauf` geht eine Seite rauf.

Mit `/` oder `?` startet man die vorwärts/rückwärts Suche innerhalb der Datei. Gibt man z.B. `/Linux` ein, dann sucht `less` nach dem Wort `Linux` in der Datei. Mit `n` kann man zum nächsten Treffer springen und mit `N` springt man wieder zurück.

Es gibt ein Hilfemenu in den durch `man` angezeigtem Bildschirm erreichbar durch `h`.

Abschnitt	Beschreibung
NAME	Name des Befehls und kurze Beschreibung
SYNOPSIS	Beschreibung der Befehlssyntax
DESCRIPTION	Beschreibung der Wirkung des Befehls
OPTIONS	Verfügbare Optionen
ARGUMENTS	Verfügbare Argumente
FILES	Hilfsdateien
EXAMPLES	Ein Beispiel für den Einsatz des Befehls
SEE ALSO	Querverweise zu verwandten Themen
DIAGNOSTICS	Warn- und Fehlermeldungen
COPYRIGHT	Autor(en) des Befehls
BUGS	Bekannte Fehler und Beschränkungen des Befehls

In der Praxis enthalten die meisten Manpages nicht alle diese Teile.

Kategorie	Beschreibung
1	Benutzerbefehle
2	Systemaufrufe
3	Funktionen der C-Bibliothek
4	Treiber und Gerätedateien
5	Konfigurationsdateien und Dateiformate
6	Spiele
7	Verschiedenes
8	Systemadministrator-Befehle
9	Kernel-Funktionen (nicht Standard)

Die Kategorien sind wichtig, wenn es Dateien mit gleichem Namen gibt, die eigene man pages haben. Z.B. gibt es `passwd` als Benutzerbefehl und auch als Konfigurationsdatei. Der Benutzerbefehl ist Kategorie 1 und die Konfigurationsdatei ist Kategorie 5. Jede Datei ist in GENAU einer Kategorie enthalten.

Will man die Hilfe für die Konfigurationsdatei öffnen, dann muß man `man 5 passwd` eintippen.

5.1.1 touch

`touch Dateiname(Dateiname2 Dateiname3)`

Ändert das "letzte Änderungsdatum" der Datei(en), ABER wenn die Datei(en) NICHT existiert(en), dann wird eine(oder mehrere) leere neue Datei(en) erstellt!

5.1.2 echo

`echo STRING`

Gibt den String auf dem Screen zurück

5.1.3 Redirection > und »

```
echo STRING > Dateiname
```

> erstellt eine Datei und fügt den String ein, wenn die Datei NICHT existiert, ansonsten wird die Datei ÜBERSCHRIEBEN

```
echo STRING » Dateiname
```

>> erstellt eine Datei, wenn Dateiname nicht existiert, ODER der String wird hinten an die Datei angehängt!

5.1.4 Multiline String

```
echo "Hallo 'Du Gummischuh'  
  nächste Zeile  
  und noch eine Zeile"
```

AUSGABE:

```
Hallo 'Du Gummischuh'  
nächste Zeile  
und noch eine Zeile
```

Benutzt man ein Anführungszeichen, dann kann man eine mehrzeiligen String durch die Eingabetaste erzeugen. Erst wenn das zweite Mal dasselbe Anführungszeichen erreicht wird, wird der String ausgegeben.

5.1.5 Anführungszeichen

Will man mit touch eine Datei mit einem Leerzeichen im Namen erstellen, dann gibt es dazu drei mögliche Wege. " " , ' ' um den Dateinamen herum oder das Leerzeichen mit "escapen".

Normalerweise hat das Leerzeichen die Bedeutung, daß Argumente und Optionen dadurch getrennt werden.

```
touch Hallo Welt
```

wird als der Befehl touch mit ZWEI Argumenten angesehen. Es werden dann diese Dateien erstellt oder ihr Datum geändert.

```
touch "Hallo Welt"  
touch 'Hallo Welt'  
touch Hallo\ Welt
```

In allen drei Fällen wird EINE Datei mit Namen 'Hallo Welt' erstellt.

Hier gibt es zwischen " und ' KEINEN Unterschied.

Bei der Verwendung von Variablen allerdings gibt es einen wichtigen Unterschied.

```
echo "Hallo $USER"
```

Hier wird einfach "Hallo user1" ausgegeben, wenn die Variable USER den Wert 'user1' enthält. Das \$ weist Bash an die Variable auszuwerten und ihren Wert auszugeben.

```
echo 'Hallo $USER'
```

gibt "Hallo \$USER" zurück.

Beim " verlieren bis auf \$, und ' alle Sonderzeichen ihre Bedeutung. Beim ' verlieren ALLE Sonderzeichen ihre Bedeutung. Deswegen wird hier die Variable NICHT durch ihren Wert ersetzt.

Eigene Variablen werden mit

```
variable=wert
```

definiert. KEINE LEERZEICHEN vor oder nach dem Gleichheitszeichen!

5.1.6 cat

Damit fügt man mehrere Dateien zusammen indem sie hintereinander gehängt werden. Das kann man dann in eine neue Datei umlenken. Am häufigsten wird cat aber zum Anzeigen des Inhalts von Dateien verwendet.

```
-n      Zeilennummern anzeigen
```

[]: