Linux 1

ID:	TT-2XE
Dokumenten URL:	http://docs.tx7.de/TT-2XE
Autor:	Tom Gries <tom@tx7.de>
Version:	5.0.0 vom 01.07.2017

>> Themen

- Definition Betriebssystem
- Geschichte von Linux
- Zentrale Begriffe
- An- und Abmelden an einem Linux-System
- Die wichtigsten Linux Befehle
- Referenzen

>> Definition Betriebssystem

Ein Betriebssystem (BS) oder englisch *Operating System* (OS) ist die Software, die die Ressourcen wie zum Beispiel Speicher und Ein- und Ausgabegeräte verwaltet und die Ausführung von Programmen steuert.

Betriebssysteme bestehen in der Regel aus einem Kernel, der die Hardware des Computers verwaltet und grundlegenden Systemprogrammen, die dem Start des Betriebssystems und dessen Konfiguration dienen. Ein wesentliches Unterscheidungsmerkmal von Betriebssystemen ist zum Beispiel die Multiuser Fähigkeit. Von Microsoft ist nur Windows NT/2000 Terminal Server ein Multiuser System.

>> Definition Betriebssystem

Die bekanntesten Betriebssysteme sind:

- UNIX
- Solaris
- Linux
- DOS
- Windows (95/98/NT/2000/XP/Vista/7/8)
- OS/2
- Mac (OS X)

>> Geschichte von Linux

Verglichen mit anderen Erfindungen, sind Computer noch relativ jung. In den 1960er und 70er Jahren bestand ein Rechner noch aus Röhren, Transistoren und Relais. Die Maschinen damals hatten ein kompaktes "Turnhallenformat". Ein Betriebssystem im heutigen Sinne kannten die Rechner damals noch nicht.

Im Laufe der Rechner-Evolution verabschiedete man sich von der reinen Job-Verarbeitung. Das Compatible Time-Sharing System (CTSS) war das erste Multiuser Betriebssystem. Entwickelt wurde es Anfang der 60er Jahre am MIT (Massachusetts Institute of Technology).

>> Geschichte von Linux

Ab 1963 wurde in Zusammenarbeit mit dem MIT, General Electric und den Bell Labs von AT&T Multics (Multiplexed Information and Computing Service) entwickelt, eine Weiterentwicklung des CTSS. Die Entwicklung wurde vom ARPA (Advanced Research Projects Agency) finanziell gefördert.

Multics war nicht sehr erfolgreich. Ab 1969 wurde an den Bell Labs von AT&T ein neues Betriebssystem entwickelt: Unics.

Der Name Unics mutierte im laufe der Jahre zu Unix und ab 1974 zu UNIX. Der (ursprüngliche) Name ist lediglich eine Anspielung auf Multics und hat keine weitere Bedeutung.

>> Geschichte von Linux

Aus dem "Ur UNIX" der Bell Labs entstanden im Laufe der Jahre so genannte Derivate. Die bekanntesten sind:

- BSD
- SunOS (Solaris)
- AIX von IBM
- HP-UX von Hewlett-Packard
- FreeBSD
- OpenBSD und
- Darwin (von Apple – Basis von Mac OS X)

>> Geschichte von Linux

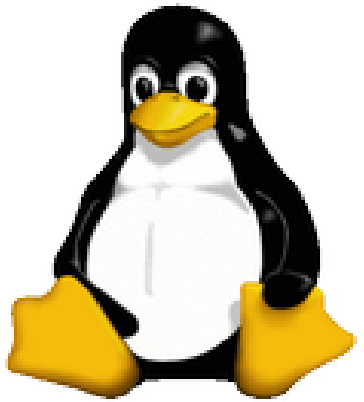
Basierend auf den Ideen und Konzepten entwickelte Linus Torvalds ein eigenes Betriebssystem: **Freax**. Allerdings hielt sich der Name nicht lange und wurde noch vor dem ersten offiziell verfügbaren Release (Version 0.02) vom 05.10.1991 von Ari Lemmke in **Linux** umbenannt. Ari verwaltete den FTP Server und ihm gefiel der Name Freax nicht.

Linux ist keine Weiterentwicklung von UNIX, sondern eine eigenständige Entwicklung. Das Maskottchen des Linux Betriebssystems ist der vollgefressene, glückliche Pinguin Tux. Der Name entstand als Ableitung von **T**orvalds **U**ni**x**. Der Pinguin wurde vermutlich in Anlehnung an *Tuxedo* – englisch für Smoking gewählt.

>> Geschichte von Linux

Linux besteht aus dem Kernel – dem eigentlichen Linux – und vielen Tools für die Verwaltung von Ressourcen. Tools und Kernel werden von vielen Anbietern als so genannte Distribution zusammen gestellt. Die Bekanntesten sind:

- Debian
- Red Hat
- Suse
- Ubuntu
- Fedora
- Knoppix (Live-CD - basierend auf Debian)
- Kali Linux (Nachfolger von BackTrack)
- Android



Das Linux Maskottchen Tux

Terminal

- Ein Terminal ist ein Endgerät, dass aus einem Bildschirm und einer Tastatur besteht und Benutzern einen direkten Zugriff auf einen Computer ermöglicht. Terminals sind über lokale Kabel oder ein Netzwerk an einen Computer angeschlossen.

- Heutzutage sind Terminals überwiegend durch Terminal Emulationen ersetzt worden. Solche Emulationen sind eigenständige Programme, die auf einem PC laufen.

Shell

- Eine Shell ist ein Programm, dass als Schnittstelle zwischen einem Benutzer und einem Betriebssystem arbeitet. Der deutsche Begriff ist "Kommandozeileninterpreter". Also eine Kommandozeile, die Befehle entgegen nimmt, auswertet und ausführt.

- Die verbreiteste Shell unter Linux ist die Bash (Bourne Again Shell – die "Wiedergeburt").

- Die Bezeichnungen *Prompt*, *Command-Prompt* und *Eingabeaufforderung* werden unter DOS/Windows synonym verwendet.

>> An- und Abmelden an Linux

Bevor man an einem Linux System arbeiten kann, muss man sich an dem System anmelden (Authentication). Dies besteht auch heute noch überwiegend mit einem Benutzernamen und einem Passwort. An die Authentizität sind verschiedene Berechtigungen verbunden (Authorization).

Zum Anmelden an ein Linux System kann man sich direkt an den Rechner begeben und die dort angeschlossene Tastatur und Monitor zum Anmelden benutzen oder eine Terminalemulation von einem beliebigen PC über das Netzwerk. Eine Anmeldung über echte Terminals ist auch möglich, allerdings sind diese wie bereits erwähnt nur noch sehr selten anzutreffen.

>> An- und Abmelden an Linux

Unter einem anderen Linux System kann man von der Shell mit z. B. TELNET oder SSH eine Verbindung zu einem entfernten Rechner aufbauen. Beide, also TELNET und SSH sind sowohl Protokolle als auch Applikationen. Die Applikationen beinhalten eine Terminalemultion.

Der Rechner (Server), mit dem man sich verbinden will, muss das entsprechende Protokoll unterstützen. Das ist bei SSH auf einem Linux System fast zu 100% der Fall.

Da der Datenverkehr bei TELNET unverschlüsselt ist, wird heutzutage fast ausschließlich nur noch SSH eingesetzt.

>> An- und Abmelden an Linux

- **Anmelden mit telnet**
  - `$ telnet tx7.de`
- **Anmelden mit ssh**
  - `$ ssh tom@tx7.de`
  - `$ ssh -l tom tx7.de`

Alternative Schreibweise

>> An- und Abmelden an Linux

Den Prompt, den man nach dem Anmelden sieht, ist der Prompt von der remote Shell. Welche Shell das ist, ist von dem remote System abhängig, also welche default Shell dem User zugewiesen wurde.

Oft wird er Name im Prompt mit angegeben. Mit

```
$ hostname
```

erfährt man den Hostnamen des Systems. Wenn keine Verbindung zu einem remote System besteht, erhält man den eigenen (lokalen) Hostnamen.

>> An- und Abmelden an Linux

Windows hat kein SSH an Bord. Hier muss man zu externen Tools greifen. Das verbreiteste Tool für TELNET und SSH unter Windows ist PuTTY. Als portable Version steht KiTTY zur Verfügung.

Im Anmeldedialog von PuTTY gibt man lediglich den Namen (oder die IP Adresse) des remote Systems und das entsprechende Protokoll (SSH – 22) an. Das Protokoll ist bei den neueren Versionen von PuTTY schon vorausgewählt.

Anmeldedialog von Putty

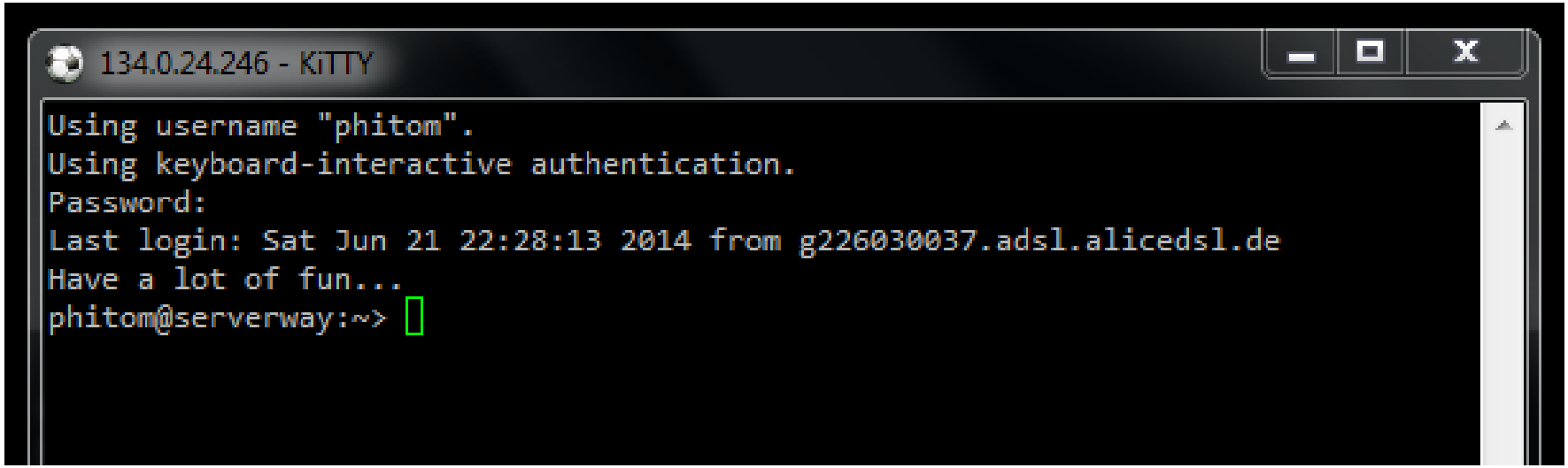
>> An- und Abmelden an Linux

Aber unabhängig, ob man sich von einem Linux Desktop oder einem Windows Rechner an ein Linux System per SSH über das Netzwerk anmeldet. Nach dem erfolgreichen Anmelden bekommt man eine Shell zugewiesen. Wir werden hier nur die Bash behandeln.

Das Erscheinungsbild ist auch in beiden Fällen identisch: meistens weißer Text auf schwarzem Hintergrund.

Im folgendem kennzeichnet ein vorangestelltes $ Symbol die Eingabeaufforderung. Andere Zeichen dafür sind zum Beispiel das > oder # Zeichen.

>> An- und Abmelden an Linux



>> An- und Abmelden an Linux

Zum Abmelden und beenden einer Sitzung gibt man am Prompt lediglich

```
$ exit
```

ein. Damit wird die Sitzung beendet und die Verbindung geschlossen.

Unter Linux ist die Groß-/Kleinschreibung zu beachten. Die Befehle *exit* und *Exit* sind nicht identisch. Den Befehl *Exit* gibt es nicht. Interaktive Anwendungen, wie z. B. *man*, beendet man mit *q*, einige mit <strg>+c. Den Editor *vi* mit der Sequenz *[ESC] : q*.

>> Die wichtigsten Linux Befehle

Unter Linux haben alle Befehle grundsätzlich den Aufbau

```
$ Befehl [-Optionen] [Argumente]
```

Die Optionen beginnen üblicherweise mit einem - und stehen im Gegensatz zu DOS/Windows vor den Argumenten. Optionen beeinflussen das Verhalten eines Kommandos (Befehls) und können einzeln als auch zusammengefasst angegeben werden. Die folgenden beiden Befehle sind beispielsweise identisch:

```
$ ls -l -h -a
```

und

```
$ ls -lha
```

>> Die wichtigsten Linux Befehle

Für viele Argumente (nicht Optionen) können so genannte Wildcards eingesetzt werden. Der Stern "*" steht dabei für eine beliebige Anzahl von beliebigen Zeichen und das Fragezeichen "?" für genau ein beliebiges Zeichen.

Die Tilde "~" ist die Kurzschreibweise für das HomeDirectory, also dem eigenen Directory.

Der einzelne Punkt "." referenziert das Directory, in dem man sich gerade befindet und die zwei Punkte ".." das in der Hierarchie darüber liegende Directory.

>> Die wichtigsten Linux Befehle

Die Bash unterstützt *Commandline completion* und hat eine umfangreiche History.

Mit der Commandline completion können Befehle, Verzeichnisse und Dateinamen mit Hilfe der Tabulatortaste vervollständigt werden. Dazu einfach die ersten Buchstaben eingeben und durch drücken der Tabulatortaste vervollständigen. Wenn der Name nicht eindeutig ist, muss man ein zweites mal die Tabulatortaste drücken und es wird eine Auswahl an Möglichkeiten angezeigt.

- Man muss dann die Eingabe nur so weit vervollständigen, bis der Name eindeutig ist.

>> Die wichtigsten Linux Befehle

Die History erreicht man über die Pfeil nach oben bzw. Pfeil nach unten Taste. Hiermit blättert man also durch die letzten Befehle. Wie viele Befehle gespeichert werden, ist abhängig von der Einstellung. Üblicherweise sind es 500 oder 1.000 Befehle.

Auf die History wird noch bei den Befehlen eingegangen. Die History Funktion ist Bestandteil der Bash.

man

Der Befehl *man* steht für *manual* und liefert (meistens) umfangreiche Informationen zu einem Befehl. Um Hilfe über den Befehl *man* selbst zu bekommen, gibt man am Prompt einfach

```
$ man man
```

ein. Wie man sieht, ist der Befehl, über den man Informationen möchte, hier als Parameter angegeben.

>> Die wichtigsten Linux Befehle

pwd

Der Befehl *pwd* steht für *print working directory* und gibt das working directory aus, also das directory, in dem man sich gerade befindet.

```
$ pwd
```

ls

Der Befehl *ls* steht für *list* und listet Dateien und Verzeichnisse.

```
$ ls
```

ls (cont.)

Für diesen Befehl stehen etliche Optionen zur Verfügung. Als Argument können Dateinamen und Verzeichnisse übergeben werden. Ohne Argument(e) wird das aktuelle Verzeichnis aufgelistet. Weiter oft benutze Optionen sind

```
$ ls -l
$ ls -lh
$ ls -lha
```

Dabei steht -l für *long listing*, -h für *human readable* und das -a für *all*, also auch versteckte Dateien.

history

Gibt alle eingegebenen Befehle aus (nicht nur von der aktuellen Sitzung).

```
$ history
```

Die History kann man mit

```
$ history -c
```

löschen.

cd

Der Befehl *cd* steht für *change directory* und wechselt das aktuelle Verzeichnis (working directory). Als Argument gibt man hier absolute oder relative Pfade an.

```
$ cd /tmp
$ cd ../conf
```

Das erste Beispiel hat dabei eine absolute Pfadangabe, das Zweite eine Relative.

cd (cont.)

Um von einem beliebigen working directory in sein HomeDirectory zu wechseln, gibt es mehrere Möglichkeiten:

```
$ cd
$ cd ~
$ cd $HOME
$ cd /home/username
```

Das letzte Beispiel hat wieder eine absolute Pfadangabe, Der absolute Pfad zum HomeDirectory ist von System zu System unterschiedlich.

echo

Gibt das angegebene Argument aus.

```
$ echo Hallo Welt
```

oder

```
$ echo "Hallo Welt"
```

Mit der Option -e können auch Steuerzeichen angegeben werden, z. B. für einen Zeilenumbruch.

```
$ echo -e "Hallo\nWelt"
```

mkdir

Der Befehl *mkdir* steht für *make directory* und erstellt ein neues Verzeichnis. Als Argument gibt man hier absolute oder relative Pfade an.

```
$ mkdir /home/user/conf
$ mkdir ~/tmp
```

Das erste Beispiel hat dabei eine absolute Pfadangabe, das Zweite eine Relative.

rmdir

Der Befehl *rmdir* steht für *remove directory* und löscht ein Verzeichnis. Als Argument gibt man hier absolute oder relative Pfade an.

```
$ rmdir /home/user/conf
$ rmdir ~/tmp
```

Das erste Beispiel hat dabei eine absolute Pfadangabe, das Zweite eine relative. Die Verzeichnisse, die mit *rmdir* gelöscht werden sollen, müssen leer sein.

rm

Der Befehl *rm* steht für *remove* und löscht Dateien oder Verzeichnisse. Im Gegensatz zu *rmdir* löscht *rm* auch nicht leere Verzeichnisse (auch rekursiv).

```
$ rm datei.txt
$ rm -rf ~/tmp
```

Das erste Beispiel löscht eine Datei, das Zweite ein Verzeichnis rekursiv. (Anm.: Hier ist ~/tmp ein Verzeichnis).

rm (cont.)

Wenn der Superuser root

```
$ rm -rf /
```

ausführt, werden **alle** Dateien und Verzeichnisse ohne Rückfrage auf den in das System eingebundenen Festplatten gelöscht. Diese Operation ist irreversibel. Das Gleiche gilt natürlich für relative Pfadangaben, also z. B.

```
$ rm -rf ..
```

vom root Homedirectory */root* aus.

cp

Der Befehl *cp* steht für *copy* und kopiert Dateien oder Verzeichnisse. Als Argument gibt man hier absolute oder relative Pfade an.

```
$ cp /etc/httpd/httpd.conf ~/backup
$ cp -R /etc/httpd/ ~/backup/
```

Das erste Beispiel kopiert die Datei *httpd.conf* in das Verzeichnis *~/backup*. Das Zweite kopiert das komplette Verzeichnis *httpd* in das Verzeichnis *~/backup*.

mv

Der Befehl *mv* steht für *move* und verschiebt Dateien oder Verzeichnisse. Als Argument gibt man hier absolute oder relative Pfade an.

```
$ mv ~/conf ~/backup
```

Dieses Beispiel verschiebt die Datei oder das Verzeichnis ~/conf in die Datei oder das Verzeichnis ~/backup. Ob es sich hier um Dateien oder Verzeichnisse handelt, ist nicht ersichtlich. Verzeichnisse können mit einem abschließen-den Slash / gekennzeichnet werden, also z. B.

```
$ mv ~/conf ~/backup/
```

whoami

Der Befehl *whoami* steht für *who am I* und zeigt den Usernamen an, mit dem man gerade angemeldet ist.

```
$ whoami
```

id

Der Befehl *id* steht für *identity* und gibt die reelle und aktive User ID aus.

```
$ id
```

cat

Der Befehl *cat* steht für *concatenate (verketten)* und verkettet mehrere Dateien und gibt dies nach stdout aus. Wenn nur eine Datei angegeben wird, wird der Inhalt einfach angezeigt – ein häufiger Einsatzzweck.

```
$ cat datei-1 [datei-2 … [datei-n]]
```

Das Ergebnis kann durch "redirecten" auch in eine neue Datei geschrieben werden.

which

Der Befehl *which* zeigt den absoluten Pfadnamen eines (Shell) Kommandos an.

```
$ which rmdir
```

zeigt den absoluten Pfadnamen des Befehls *rmdir* an. Es können nur Pfadnamen von Kommandos gefunden werden, die in einem Verzeichnis der Umgebungsvariable $PATH liegen.

less

Der Befehl *less* zeigt längere Texte seitenweise an. Es ist der Nachfolger von *more* (less is a little bit more) und kann im Gegensatz zu more auch rückwärts blättern und nach Wörtern suchen.

```
$ less /etc/apache/httpd.conf
```

zeigt die Apache Konfigurationsdatei seitenweise an. Man kann *less* mit der taste [q] verlassen.

>> Referenzen

SelfLinux:

http://docs.tx7.de/TT-W3S

Das Linux-Buch von Michael Kofler:

http://docs.tx7.de/TT-HW4