Kostenloses eBook

LERNEN unix

Free unaffiliated eBook created from
Stack Overflow contributors.

#unix

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: unix

It is an unofficial and free unix ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official unix.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Unix

Bemerkungen

In diesem Abschnitt erhalten Sie einen Überblick darüber, was Unix ist und warum ein Entwickler es verwenden möchte.

Es sollte auch alle großen Themen innerhalb von Unix erwähnen und auf die verwandten Themen verweisen. Da die Dokumentation für Unix neu ist, müssen Sie möglicherweise erste Versionen dieser verwandten Themen erstellen.

Examples

Installation oder Setup

Detaillierte Anweisungen zum Einrichten oder Installieren von Unix.

Erste Schritte mit Unix online lesen: https://riptutorial.com/de/unix/topic/912/erste-schritte-mit-unix

Kapitel 2: Berechtigungen

Einführung

In Unix hat jede Datei bestimmte Berechtigungen wie Lesen, Schreiben und Ausführen. Ein Benutzer kann die Berechtigungen einer Datei mit dem Befehl 'chmod' ändern.

Bemerkungen

In UNIX gibt es drei Berechtigungen, um einer bestimmten Ebene Zugriff auf eine Datei oder einen Ordner zu gewähren.

Für Dateien:

- **Lesen** : Erlaubt dem Benutzer / der Gruppe / anderen, eine Datei zu lesen.
- **Schreiben** : Erlaube dem Benutzer / der Gruppe / anderen, eine Datei zu ändern.
- **Ausführen** : Erlaubt dem Benutzer / der Gruppe / anderen, eine Datei auszuführen (oder auszuführen).

Diese werden für Verzeichnisse geringfügig geändert:

- **Lesen** : Erlaube dem Benutzer / der Gruppe / anderen, die Namen der Dateien in einem Verzeichnis aufzulisten.
- **Schreiben** : Erlaubt dem Benutzer / der Gruppe / anderen, Dateien in einem Verzeichnis zu erstellen, zu löschen und umzubenennen.
- **Ausführen** : Dem Benutzer / der Gruppe / anderen Zugriff auf Dateimetadaten und Inhalte eines Verzeichnisses gewähren.

Diese Berechtigungen können mit den Buchstaben "r" zum Lesen, "w" zum Schreiben und "x" zum Ausführen dargestellt werden. Sie können auch numerisch dargestellt werden: 4 zum Lesen, 2 zum Schreiben und 1 zum Ausführen.

Examples

Ändern Sie die Berechtigungen einer Datei

```
> chmod 644 example.txt
> ls -l example.txt
-rw-r--r--  1 owner ogroup 57 Jul  3 10:13  example.txt
```

Der obige Befehl ändert die Dateiberechtigungen, damit der Eigentümer der Datei lesen und in eine Datei schreiben kann. Außerdem können Benutzer in der Gruppe des Besitzers und andere Benutzer im System die Datei lesen.

Grundlegendes zu Berechtigungen

`add.sh` , es gibt eine Datei, die wir ausführen `add.sh` , beispielsweise ein Bash-Skript namens `add.sh` Die `./add.sh` jedoch zu einem Berechtigungsfehler. Das Abrufen der Berechtigungen ist ein einfacher Prozess.

Geben Sie Folgendes ein, um die Berechtigungen einer Datei zu ermitteln:

`ls -l filename` oder in unserem Fall `ls -l ./add.sh`

Dies druckt Folgendes auf die Konsole:

```
-r--r--r-- 1 username groupname 0 Jan 4 12:00 add.sh
```

Lass uns anhalten und verstehen, was das bedeutet. Es gibt drei verschiedene Arten von Berechtigungen: Eigentümer, Gruppe und andere. Für jeden Berechtigungstyp gelten unterschiedliche Berechtigungen.

Es gibt auch drei Berechtigungsaktionen, die allgemein beschreiben, was ein Benutzer mit einer Datei genau tun kann. Dies sind: (Lesen: r, Schreiben: w, Ausführen: x).

Also zurück zu dieser Reihe von Strichen und Strichen. Jede Berechtigungsgruppe verfügt über drei potenzielle Fähigkeiten. Die Gruppen werden in der Reihenfolge `owner-group-others` und die Aktionen als `read-write-execute` .

Aber warten Sie, das heißt, es gibt ein zusätzliches Zeichen am Anfang der Zeichenfolge. Dies ist eigentlich das Dateideskriptor-Zeichen. Wir können sehen, dass dort `-` ist, aber andere Zeichen existieren für Verzeichnisse `(d)` , Sockets `(s)` , symbolische Links `(l)` usw.

Damit bleiben uns im Wesentlichen diese Informationen: `a file where owner, group, and others have read permissions. No other permissions granted.`

Lassen Sie uns dies ändern, damit der Eigentümer die Datei auch schreiben und ausführen kann. Hinweis: Je nach den Berechtigungen, kann es notwendig sein , um prepend `sudo` zu diesem Befehl.

```
chmod 744 add.sh
ls -l add.sh
```

Druckt aus

```
-rwxr--r-- 1 username groupname 0 Month time add.sh
```

Nun kann der Besitzer der Datei die Datei durch Eingabe ausführen

```
./add.sh
```

CHMOD-Berechnung

CHMOD-Berechnung

CHMOD ist binär. _ / _ _ _ / _ _ _ / _ _ = _ / 4 + 2 + 1/4 + 2 + 1/4 + 2 + 1 = 777 = _ / rwx / rwx /

rwx = 777 Daher ist _rwx = _ / 4 + 2 + 1 = 7

D / _ _ _ / _ _ _ / _ _ _ ('D' = Verzeichnis, eine andere Verwendung ist L = Link)

Also zB _rwxr_xr_x = _ / rwx / rx / r_x = 755

Chown

Um die own: group zu ändern, verwenden Sie den Befehl chown user: group

zB chown Besitzer: Gruppe oder wenn Besitzer und Gruppe gleich sind, können Sie Chown Besitzer verwenden: (da linus Besitzer: Gruppe voraussetzt).

Berechtigungen online lesen: https://riptutorial.com/de/unix/topic/6394/berechtigungen

Kapitel 3: Erste Schritte mit Unix-Befehlen

Einführung

In diesem Thema werden grundlegende Unix-Befehle beschrieben.

Examples

Eine nicht erschöpfende Liste von Unix-Befehlen

```
$man <command>
```

Zeigt die Online-Handbuchseiten für den Befehl an

```
$clear
```

Löscht den Bildschirm des Terminals

```
$pwd
```

Gibt den Namen des Arbeitsverzeichnisses zurück

```
$echo <string>
```

Schreibt den String in die Standardausgabe

```
$printf <string>
```

Formatieren und Drucken der Zeichenfolge Beispiel: print $ PATH $ printf "% s \ n" $ PATH

```
$uptime
```

Zeigen Sie, wie lange das System läuft

```
$which <program>
```

Suchen Sie eine Programmdatei im Pfad des Benutzers

```
$whereis <program>
```

Überprüft die Standard-Binärverzeichnisse nach den angegebenen Programmen und druckt die

DATEIEN

```
$cd [directory]
```

Verzeichnis wechseln Häufig verwendete Verzeichnissymbole:

- . : Aktuelles Verzeichnis
- .. : Übergeordnetes Verzeichnis
- ~: Ausgangsverzeichnis
- / : Wurzelverzeichnis

```
$ls
```

- $ ls -a: Versteckte Dateien anzeigen
- $ ls -l: Lange Liste anzeigen
- $ ls -1: Zeigt nur den Dateinamen pro Zeile an
- $ ls -h: Vom Menschen lesbares Format

  $ file Dateityp festlegen (zB gzip)

DATEIEN LESEN

```
$more
```

Zeigt den Inhalt einer Datei Bildschirm für Bildschirm an
Leertaste: Zum nächsten Bildschirm blättern; b = vorheriger Bildschirm
eingeben: Eine Zeile scrollen
h: Hilfe für mehr
q: Beenden Sie die Hilfe

```
$less <file>
```

Weniger ist ein Programm, das mehr ähnlich ist, aber es ermöglicht sowohl eine Rückwärtsbewegung in der Datei als auch eine Vorwärtsbewegung

```
$cat <file>
```

Liest Dateien sequenziell und schreibt sie in die Standardausgabe

```
$head [-number] <file>
```

Erste Zeilen einer Datei anzeigen

```
$tail [-number] <file>
```

Zeigt den Inhalt der Datei oder standardmäßig die Standardeingabe der Standardausgabe an

- $ tail -f: Zeigt Änderungen in der Datei in Echtzeit an

```
$touch <file>
```

Legt die Modifikations- und Zugriffszeiten von Dateien fest. Wenn eine Datei nicht vorhanden ist, wird sie mit Standardberechtigungen erstellt

```
$tee <file>
```

Kopiert die Standardeingabe in die Standardausgabe. Drücken Sie Strg-D, um das Hinzufügen von Inhalten zu beenden

- $ tee -a: Hängt die Ausgabe an die Dateien an, anstatt sie zu überschreiben

```
$mkdir <directory>
```

Erstellen Sie ein Verzeichnis

```
$wc
```

Zeigt die Anzahl der Zeilen, Wörter und Bytes an, die in jeder Eingabedatei oder Standardeingabe enthalten sind

- wc -l: Zeilen zählen
- wc -w: Wörter zählen
- wc -m: Zeichen zählen

```
$diff <file1> <file2>
```

Vergleichen Sie zwei Dateien Zeile für Zeile. Druckt nur die verschiedenen Zeilen.

```
$locate <file>
```

Suchen Sie nach Dateien auf der Festplatte

- $ locate -q: Fehler unterdrücken

```
$find <path> <expression> <action>
```

Suchen Sie nach Dateien nach Name oder Inhalt

- $ find -name: Suche nach Dateiname
- $ find -size <+/- n> Beispiel: Finden Sie Dateien im aktuellen Verzeichnis, die größer als 10k $ find sind. -größe +10

```
$rm <file or directory>
```

Löschen Sie <Datei oder Verzeichnis>

- rm -f: Bestätigung überspringen
- rm -i: Bestätigen Sie jede Löschung
- rm -r: Rekursiv

Beispiel: Löschen Sie die und ihren Inhalt $ rm -r

```
$mv <source_file> <target_file>
```

Benennt eine Datei um

```
$mv <source_file> <target_directory>
```

Verschiebt eine Datei

- $ mv -i: Nicht vorhandene Dateien überschreiben
- $ mv -r: Rekursiv

Beispiel: Verzeichnis in der Hierarchie $ mv nach oben verschieben.

```
$cp
```

Datei / Verzeichnis innerhalb desselben Rechners kopieren (scp-Befehl zum Kopieren auf einen entfernten Rechner verwenden)

- $ cp -i: Nicht vorhandene Dateien überschreiben
- $ cp -r: rekursiv

Beispiel: Eine Datei kopieren und umbenennen

```
$cp <file_name> <new_file_name>
```

Beispiel: In das Verzeichnis kopieren

```
$cp <file_name> <directory_name>
```

Beispiel: Ein Verzeichnis kopieren und umbenennen

```
$cp –R <directory> <new_directory>
```

Beispiel: Kopieren Sie alle Dateien eines bestimmten Typs in ein Verzeichnis

```
$cp *.txt <directory>
```

```
$ln –s <file> <link name>
```

Erstellen Sie einen Alias (Link) zu einer Datei

- $ ln -s: Softlink erstellen (Ein Link, der auf mehreren Computern funktioniert)

```
$sort <file>
```

Sortiere den Inhalt einer Datei -r Sortiert die Sortierung -n

Beispiel: sortieren Sie das und schreiben Sie das Ergebnis in sortieren.txt $ sortieren uniq -u> sortierte.txt

```
$uniq [-ucd] filename(s)
```

Sucht nach doppelten Zeilen. Daten müssen zuerst sortiert werden

- $ uniq -d: Zeigt nur eine Kopie der doppelten Zeilen
- $ uniq -u: Zeigt nur Zeilen an, die nicht doppelt vorhanden sind
- $ uniq -c: Gibt jede Zeile mit einer Anzahl von Vorkommen aus. Beispiel: Zeigt Benutzern, die mehr als einmal verbunden sind, $ who | cut -d '' -f1 | sortieren | uniq -d

```
$grep <pattern> <file_name>
```

Druckt Zeilen, die eine Übereinstimmung für ein Muster enthalten.

- $ grep -i: Führt das Vergleichen von Groß- und Kleinschreibung durch
- $ grep -v: Druckt alle Zeilen, die keine Regex enthalten
- $ grep -r: Durchsucht die aufgelisteten Unterverzeichnisse rekursiv und gibt Dateinamen mit Vorkommen des Musters aus
- $ grep -I: Schließt binäre Dateien aus

```
$tr “string1” [“string 2”]
```

Tool suchen und ersetzen. tr nimmt nur die Eingabe von Pipes und Weiterleitungen an. Es akzeptiert keine Dateien als Eingabe.

- tr -d: Löscht alle Vorkommen aller CHARACTERS in string1

Beispiel: Drucken Sie a.txt auf dem Bildschirm, nachdem Sie alle Vorkommen von ";" $ cat a.txt | gelöscht haben tr -d ";"

- tr -s: Ersetzen Sie Vorkommen durch ein einzelnes Zeichen

Beispiel: $ echo “SSSS SS” | tr -s "S" "S"

```
$tar
```

Erzeugt und bearbeitet Streaming-Archivdateien. Diese Implementierung kann Teer-, pax-, cpio-, zip-, jar-, ar- und ISO-Images extrahieren und Tar-, pax-, cpio-, ar- und Shar-Archive erstellen.

FESTPLATTENNUTZUNG

```
$du [file or directory]
```

Zeigt die Verwendung des Dateisystemblocks für jede Datei oder jedes Verzeichnis an. Wenn keine Datei / Verzeichnis angegeben ist, wird die Blocknutzung des aktuellen Verzeichnisses angezeigt.

- $ du -a: Dateien und Verzeichnisse (standardmäßig nur Verzeichnisse)
- $ du -h: Vom Menschen lesbares Format Beispiel: Anzeige der Datenträgerauslastung, sortiert, nur MB-Dateien $ du -h | grep -i "m \ t" | sort -n Beispiel: Finden Sie die 10 größten Dateien / Verzeichnisse $ du -a / var | sort -n -r | Kopf -n 10

```
$df
```

Freier Speicherplatz anzeigen

- $ df -h: Vom Menschen lesbares Format

REDIRECTIONS & PIPES

> Standardausgabe umleiten. Datei nicht überschreiben, wenn sie existiert

>! Standardausgabe umleiten Datei überschreiben, falls vorhanden

> & Standardausgabe und Standardfehler umleiten

Beispiel: Befehlsausgabe in eine Datei umleiten

```
$ls > result.txt
```

Verwenden Sie die Datei> / dev / null, um die Fehlernachricht zu löschen

Beispiel: Suchen Sie eine Datei mit dem Namen my_file_name und drucken Sie das Ergebnis nach ~ / find.txt. Fehler ausblenden (zB "Berechtigungen verweigert")

```
$find / -name my_file_name.* > /dev/null > ~/find.txt
```

<Standardeingabe umleiten

>> Standardausgabe anhängen <Befehl> >>: Ausgabe an das Ende eines vorhandenen anhängen

<Befehl> <: Leiten Sie die Eingabe aus einer Datei auf einen Befehl um

| Standardausgabe auf einen anderen Befehl umleiten (Pipe)

Beispiel: Zeigen Sie paginierte Details der laufenden Prozesse an

```
$ps –ex | more
```

| : Pipe-Ausgabe von command1 als Eingabe von command2 (Wenn eine Ausgabedatei in der Mitte einer Pipe gewünscht wird, verwenden Sie den T-Befehl)

Beispiel: Zählen Sie die Anzahl der verbundenen Benutzer

```
$who | wc –l
```

PROZESSE

```
$ps
```

Aktive Prozesse anzeigen

- $ ps -e: Zeigt Informationen zum Prozess an
- $ ps -x: Versteckte Prozesse anzeigen

Beispiel: Suchen Sie Prozesse anhand des Namens $ grep -l <Prozessname_regex>

```
$kill [–signal] pid
```

Prozess abbrechen

Einige der am häufigsten verwendeten Signale:

- 3: quit (quit)
- 9: KILL (nicht einfangbarer, nicht zu vernachlässigender Kill)
- 15: TERM (Software-Beendigungssignal)

```
$top
```

Sortierte Informationen zu Prozessen anzeigen und aktualisieren Siehe Liste der möglichen Schlüssel auf den Manpages. Allgemeine Schlüssel sind: CPU, Threads, Ports

- $ top -o

```
$htop
```

Anzeige und Aktualisierung sortierter Informationen zu Prozessen

NUTZERBERECHTIGUNG

```
$sudo <command>
```

Führen Sie den Befehl als Superuser aus

```
$su
```

(Ersatzbenutzer) öffnet eine Sitzung als Administrator

```
$exit
```

Wurzel verlassen

```
$whoami
```

Anzeige der effektiven Benutzer-ID

```
$who
```

Alle verbundenen Benutzernamen drucken

```
$passwd
```

Ändere das Passwort

```
$chmod <who> <operation> <permissions> <file or directory name>
```

Ändern Sie den Besitzer- / Gruppenzugriff auf eine Datei oder ein Verzeichnis

Wer: u Benutzer; g Gruppe; o anderes; ein alles oben

Bedienung: + hinzufügen; - Löschen; = set (bedeutet auf nichts zurücksetzen und nur das festlegen, was angegeben wurde)

Berechtigungen: rwx

Beispiel: Fügt der Gruppe Lese- / Ausführungsberechtigungen hinzu

```
$chmod g +rx <file>
```

Beispiel:

```
$chmod 743 <file>
```

Beachten Sie, dass Sie für das $ cd in ein Verzeichnis die x-Berechtigungen benötigen

Erste Schritte mit Unix-Befehlen online lesen: https://riptutorial.com/de/unix/topic/9848/erste-schritte-mit-unix-befehlen

Kapitel 4: Grundlegende Konsolenbefehle

Examples

pwd - Arbeitsverzeichnis drucken

```
$> pwd
/home/myUserHome
$> cd ..
$> pwd
/home
```

druckt den aktuellen Pfad zur Konsole.

pushd / popd (aktuelles Verzeichnis auf Stack speichern und zum Ziel / Pop-Verzeichnis)

```
$ pwd
/home/bob/somedir1/somedir2/somedir3

$ pushd /home/bob/otherdir1/otherdir2
/home/bob/otherdir1/otherdir2 /home/bob/somedir1/somedir2/somedir3

$ popd
/home/bob/somedir1/somedir2/somedir3

$ pushd /usr
/usr /home/bob/somedir1/somedir2/somedir3

$ pushd /var
/var /usr /home/bob/somedir1/somedir2/somedir3

$ popd
/usr /home/bob/somedir1/somedir2/somedir3

$ pwd
/usr

$ popd
/home/bob/somedir1/somedir2/somedir3

$ pwd
/home/bob/somedir1/somedir2/somedir3
```

Dateibearbeitungsbefehle

Liste der Befehle, die hier eingeführt werden:

```
ls     #view contents of a directory
touch  #create new file
mkdir  #create new directory
cp     #copy contents of one file to another
```

```
mv     #move file from one location to another
rm     #delete a file or directory
```

ls Beispiele

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     test.cpp
```

zeigt das aktuelle Verzeichnis

```
jennifer@my_computer:~/Desktop$ ls c++\ projects
DNA_analysis.cpp        encryption.cpp  pool_game.cpp
```

zeigt das Verzeichnis "C ++ - Projekte". Leerzeichen in Dateinamen werden als "\" eingegeben.

Berühren Sie Beispiel

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     test.cpp
jennifer@my_computer:~/Desktop$ touch ruby_test.rb
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     ruby_test.rb    test.cpp
```

mkdir beispiel

```
jennifer@my_computer:~/Desktop$ mkdir ruby
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     ruby    ruby_test.rb    test.cpp
jennifer@my_computer:~/Desktop$ cd ruby
jennifer@my_computer:~/Desktop/ruby$ ls
<nothing>
jennifer@my_computer:~/Desktop/ruby
```

Es wird nicht wirklich `<nothing>` gedruckt. Es ist nur, wie ich darstelle, dass es nichts ausgibt

cp beispiele

```
jennifer@my_computer:~/Desktop/ruby$ cd ..
jennifer@my_computer:~/Desktop$ cp test.cpp c++_test.cpp
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby_test.rb
test.cpp
```

Dies ist, wenn das letzte Argument für `cp` , in diesem Fall "c ++ _ test.cpp", kein vorhandenes Verzeichnis ist. `cp` erstellt eine Datei mit dem Namen "c ++ _ test.cpp", deren Inhalt mit dem von "test.cpp" identisch ist. Wenn c ++ _ test.cpp bereits vorhanden war, hätte `cp` den vorherigen Inhalt gelöscht, bevor der Inhalt von "test.cpp" kopiert wurde.

```
jennifer@my_comptuer:~/Desktop$ ls ruby
<nothing>
jennifer@my_computer:~/Desktop$ cp ruby_test.rb ruby
jennifer@my_computer:~/Desktop$ ls ruby
```

```
ruby_test.rb
```

Dies geschieht, wenn das letzte Argument für `cp` , in diesem Fall "Ruby", ein Verzeichnis ist. `cp` erstellt eine Datei mit demselben Namen wie "ruby_test.rb", aber im Verzeichnis "ruby".

mv Beispiele

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby_test.rb
test.cpp
jennifer@my_computer:~/Desktop$ mv ruby_test.rb ruby\ test.rb
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby test.rb
test.cpp
```

Dies geschieht, wenn das letzte Argument für `mv` , in diesem Fall "ruby test.rb", kein vorhandenes Verzeichnis ist. Die Datei "ruby_test.rb" wurde in "ruby test.rb" umbenannt. Wenn "ruby test.rb" bereits vorhanden war, wurde es überschrieben. Beachten Sie, dass Leerzeichen ein " vorangestellt ist.

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby test.rb
test.cpp
jennifer@my_computer:~/Desktop$ ls c++\ projects
DNA_analysis.cpp        encryption.cpp  pool_game.cpp
jennifer@my_computer:~/Desktop$ mv test.cpp c++\ projects
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby test.rb
jennifer@my_computer:~/Desktop$ ls c++\ projects
DNA_analysis.cpp        encryption.cpp  pool_game.cpp   test.cpp
```

Dies geschieht, wenn `mv` ein bereits vorhandenes Verzeichnis ist. Die Datei "test.cpp" wird in das Verzeichnis "C ++ - Projekte" verschoben.

rm Beispiele

```
jennifer@my_computer:~/Desktop$ ls
c++ projects    c++_test.cpp    Research Paper.docx     ruby    ruby test.rb
jennifer@my_computer:~/Desktop$ rm c++_test.cpp
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     ruby    ruby test.rb
```

c ++ _ test.cpp wurde gelöscht

```
jennifer@my_computer:~/Desktop$ rm c++\ projects
rm: cannot remove 'c++ projects': Is a directory
jennifer@my_computer:~/Desktop$ ls
c++ projects    Research Paper.docx     ruby    ruby test.rb
```

`rm` hat eine zusätzliche Anforderung zum Löschen von Verzeichnissen

```
jennifer@my_computer:~/Desktop$ rm -rf c++\ projects
```

```
jennifer@my_computer:~/Desktop$ ls
Research Paper.docx     ruby    ruby test.rb
```

`-rf` muss hinzugefügt werden, um ein Verzeichnis zu löschen.

Um mehr über `ls` zu erfahren, geben Sie den Befehl `ls --help` . `touch --help` Sie zur `touch` `touch --help` . Ebenso bei allen 6 hier genannten Befehlen. Dadurch wird eine ausführliche Erläuterung der Verwendung ausgedruckt, ohne etwas zu erstellen oder zu löschen.

CD-Befehl, Verzeichnisse erklärt

```
michael@who-cares:~$
```

Das Symbol `~` nach dem `who-cares:` ist das aktuelle Verzeichnis. `~` bedeutet eigentlich das Heimatverzeichnis der Person. In diesem Fall ist das `/home/michael` .

```
michael@who-cares:~$ cd Downloads
michael@who-cares:~/Downloads$
```

Sucht im aktuellen Verzeichnis nach `Downloads` und erstellt dann das aktuelle Verzeichnis.

```
michael@who-cares:~/Downlaods$ cd /var
michael@who-cares:/var$
```

Da dieses Verzeichnis mit einem `/` hat, müssen Sie im Stammverzeichnis nach dem Verzeichnis `var` suchen. Für diejenigen, die von Windows kommen, entspricht das Stammverzeichnis `C:\` . Verzeichnisse, die mit `/` beginnen, heißen **"absolute Verzeichnisse"** und Verzeichnisse, die nicht als **"relative Verzeichnisse" bezeichnet werden.**

```
michael@who-cares:/var cd lib/dbus
michael@who-cares:/var/lib/dbus$
```

Das `/` in der Mitte bedeutet `cd lib` und sobald dies erledigt ist, `cd dbus` in einem Befehl.

```
michael@who-cares:/var/lib/dbus$ cd .
michael@who-cares:/var/lib/dbus$
```

`.` bedeutet eigentlich "das aktuelle Verzeichnis". Der Befehl `cd .` ist im Grunde nutzlos, aber `.` ist nützlich für andere Dinge.

```
michael@who-cares:/var/lib/dbus$ cd ..
michael@who-cares:/var/lib$
```

`..` bedeutet eigentlich "das übergeordnete Verzeichnis des aktuellen Verzeichnisses". `cd ..` bedeutet also "ein Verzeichnis nach oben navigieren".

```
michael@who-cares:/var/lib$ cd ../log/apt
michael@who-cares:/var/log/apt$
```

`.` und `..` kann auch Teil der `/` -Kette sein. Es gibt auch keine Begrenzung, wie lange es dauern kann.

```
michael@who-cares:/var/log/apt$ cd /dev/bus
michael@who-cares:/dev/bus$
```

Die `/` chain kann sogar existieren, wenn das Verzeichnis bei root beginnt.

```
michael@who-cares:/dev/bus$ cd /
michael@who-cares:/$
```

`cd /` bringt Sie in das Stammverzeichnis. Ich frage mich, was passiert, wenn Sie `cd ..` ... hier ... (keine Sorge, es ist sicher)

```
michael@who-cares:/$ cd home
michael@who-cares:/home$ cd michael
michael@who-cares:~$
```

Jeder Benutzer hat ein Verzeichnis für seine Sachen im Heimatverzeichnis. Wenn sich das aktuelle Verzeichnis im Heimatverzeichnis befindet, wird dieser Teil des Namens, in diesem Fall `/home/michael` , durch `~` .

```
michael@who-cares:~$ cd sys
michael@who-cares:/sys$ cd ~/Desktop
michael@who-cares:~/Desktop$ cd ~/..
michael@who-cares:/home$
```

`~` kann auch Teil der `/` -Kette sein. Es kann sogar in derselben Kette wie `..` . Wenn das Verzeichnis mit `~` beginnt, ist es ein absolutes Verzeichnis, als würde es mit `/` .

Das letzte, was Sie versuchen sollten: Geben Sie `cd` ohne Verzeichnis ein.

welche

Verwenden Sie den Befehl `which` , um festzustellen, wo auf Ihrem System eine ausführbare Datei in Ihrem Pfad vorhanden ist:

```
$ which python
$
```

Wenn keine Antwort vorliegt, ist diese ausführbare Datei in Ihrem Pfad nicht vorhanden. Das System gibt Ihnen einfach eine neue Eingabeaufforderung ohne Fehlermeldung zurück. Wenn die ausführbare Datei in Ihrem Pfad vorhanden ist, wird das Verzeichnis angezeigt, in dem es tatsächlich vorhanden ist:

```
$ which ls
/bin/ls
```

Dies kann hilfreich sein, um festzustellen, warum das Verhalten nicht den Erwartungen entspricht,

indem Sie sicherstellen, dass Sie die Version der ausführbaren Datei ausführen, von der Sie glauben, dass Sie sie sind. Wenn Sie beispielsweise sowohl Python 2 als auch Python 3 installiert haben, werden beide möglicherweise durch Eingabe von `python` in das Terminal ausgeführt. Die tatsächlich ausgeführte ausführbare Datei kann jedoch anders aussehen als erwartet. Wie oben gezeigt, funktioniert dieser Befehl für jeden Standard-Unix-Befehl, der alle von einzelnen ausführbaren Dateien unterstützt wird.

Grundlegende Unix-Befehle

```
$pwd
```

Zeigt das aktuelle Arbeitsverzeichnis an.

```
$who
```

Zeigt alle angemeldeten Benutzer an.

```
$who am i
```

Zeigt den Benutzernamen des aktuellen Benutzers an.

```
$date
```

Zeigt das aktuelle Systemdatum an

```
$which <command>
```

Zeigt den Pfad des angegebenen Befehls. Beispielsweise zeigt "$ which pwd" den Pfad des Befehls "pwd".

```
$file <file_name>
```

Zeigt den Typ der angegebenen Datei (reguläre Datei, Verzeichnis oder andere Dateien)

```
$cal
```

Zeigt den Kalender des aktuellen Monats an.

```
$bc
```

Zeigt die mathematische Berechnung zwischen zwei Ganzzahlen von Floaten. Zum Beispiel gibt "$ bc 2 + 3" die arithmetische Summe von 3 und 5 zurück.

```
$ls
```

Listet den Inhalt des Verzeichnisses auf.

- $ ls -l: Listen im langen Format.
- $ ls -c: Ausgabe in mehreren Spalten.
- $ ls -f: Listet den Dateityp auf.
- $ ls -r: Rekursive Auflistung aller angetroffenen Unterverzeichnisse.
- $ ls -a: Zeigt alle Dateien einschließlich versteckter Dateien an.
- $ ls -i: Listet alle Dateien mit ihrer I-Node-Nummer auf.

```
$grep [options] <pattern> <input_file_names>
```

Druckt Zeilen, die eine Übereinstimmung für ein Muster enthalten.

- $ grep -i: Führt das Vergleichen von Groß- und Kleinschreibung durch
- $ grep -v: Druckt alle Zeilen, die keine Regex enthalten
- $ grep -r: Durchsucht die aufgelisteten Unterverzeichnisse rekursiv und gibt Dateinamen mit Vorkommen des Musters aus
- $ grep -I: Schließt binäre Dateien aus

Grundlegende Konsolenbefehle online lesen:
https://riptutorial.com/de/unix/topic/4262/grundlegende-konsolenbefehle

Kapitel 5: Sehen Sie sich die Manual Pages an

Examples

Anzeigen der Handbuchseite für einen Systembefehl

```
man <command>
```

Dadurch wird die Handbuchseite für den angegebenen Befehl angezeigt.

Zum Beispiel zeigt `man ping` :

```
PING(8)                   BSD System Manager's Manual                   PING(8)

NAME
     ping -- send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS
     ping [-AaCDdfnoQqRrv] [-b boundif] [-c count] [-G sweepmaxsize]
          [-g sweepminsize] [-h sweepincrsize] [-i wait] [-k trafficclass]
          [-l preload] [-M mask | time] [-m ttl] [-P policy] [-p pattern]
          [-S src_addr] [-s packetsize] [-t timeout] [-W waittime] [-z tos]
          host
     ping [-AaDdfLnoQqRrv] [-b boundif] [-c count] [-I iface] [-i wait]
          [-k trafficclass] [-l preload] [-M mask | time] [-m ttl] [-P policy]
          [-p pattern] [-S src_addr] [-s packetsize] [-T ttl] [-t timeout]
          [-W waittime] [-z tos] mcast-group

DESCRIPTION
     The ping utility uses the ICMP protocol's mandatory ECHO_REQUEST datagram
     to elicit an ICMP ECHO_RESPONSE from a host or gateway.  ECHO_REQUEST
     datagrams (``pings'') have an IP and ICMP header, followed by a ``struct
     timeval'' and then an arbitrary number of ``pad'' bytes used to fill out
     the packet.  The options are as follows:

...
```

Während der Anzeige der Manpage kann gesucht werden. Wenn Sie einen Schrägstrich (`/`) gefolgt vom Suchbegriff eingeben, springen Sie zum ersten Vorkommen des Begriffs. Beispiel: `/ping`

Wenn Sie anschließend `N` drücken, wird zum nächsten Vorkommen gesprungen. `Shift+N` springt zum vorherigen Vorkommen.

Holen Sie sich den Dateipfad für eine Handbuchseite

```
$ man -w find
/usr/share/man/man1/find.1.gz
```

```
$ man -w printf
/usr/share/man/man1/printf.1.gz

$ man -w man
/usr/share/man/man1/man.1.gz
```

Suchen Sie nach einer Handbuchseite

Sie können nach `man` Seiten suchen, die in ihrer Beschreibung eine bestimmte Zeichenfolge enthalten.

```
man -k <string>
```

Zum Beispiel:

```
man -k unzip
```

Könnte wiederkommen:

```
man -k unzip
IO::Uncompress::Bunzip2(3pm) - Read bzip2 files/buffers
IO::Uncompress::Gunzip(3pm) - Read RFC 1952 files/buffers
IO::Uncompress::Unzip(3pm) - Read zip files/buffers
PerlIO::gzip(3pm)        - Perl extension to provide a PerlIO layer to gzip/gunzip
gzip(1), gunzip(1), zcat(1) - compress or expand files
IO::Uncompress::Bunzip2(3pm) - Read bzip2 files/buffers
IO::Uncompress::Gunzip(3pm) - Read RFC 1952 files/buffers
IO::Uncompress::Unzip(3pm) - Read zip files/buffers
PerlIO::gzip(3pm)        - Perl extension to provide a PerlIO layer to gzip/gunzip
bzip2(1), bunzip2(1)     - a block-sorting file compressor, v1.0.6 bzcat -
decompresses files to stdout bzip2recover - recovers data from damaged bzip2 files
funzip(1)                - filter for extracting from a ZIP archive in a pipe
unzip(1)                 - list, test and extract compressed files in a ZIP archive
unzipsfx(1)              - self-extracting stub for prepending to ZIP archives
```

Eine Manpage in einem anderen Abschnitt finden

Manchmal ist ein Begriff in mehreren Abschnitten des Handbuchs definiert. Standardmäßig zeigt `man` nur die erste gefundene Seite an, was für Programmierer ärgerlich sein kann, da C-Funktionen in einem späteren Abschnitt als Befehle und Systemaufrufe dokumentiert werden. Verwenden Sie den folgenden Befehl, um alle Seiten anzuzeigen, die einem Namen entsprechen:

```
$ man -wa printf
/usr/share/man/man1/printf.1.gz
/usr/share/man/man1p/printf.1p.gz
/usr/share/man/man3/printf.3.gz
/usr/share/man/man3p/printf.3p.gz
```

Um die Seite in einem bestimmten Abschnitt anzuzeigen, platzieren Sie sie einfach vor dem Begriff:

```
man 3 printf
```

Lesen Sie eine manuelle Datei mit dem Mann

Dies ist dasselbe wie beim Lesen eines Handbuchs für einen Befehl:

```
man /path/to/man/file
```

Sehen Sie sich die Manual Pages an online lesen: https://riptutorial.com/de/unix/topic/442/sehen-sie-sich-die-manual-pages-an

Kapitel 6: Überblick über Unix

Einführung

1969 in AT & T Bell Labs von Ken Thomson als Einzelbenutzer-Betriebssystem entwickelt. Ursprünglich in Assembler geschrieben. Als Mehrbenutzer-Betriebssystem entwickelt. später 1973 in C umgeschrieben 1974 für Bildungszwecke lizenziert POSIX (Portable Operating System for Unix) wurde 1974 entwickelt

Examples

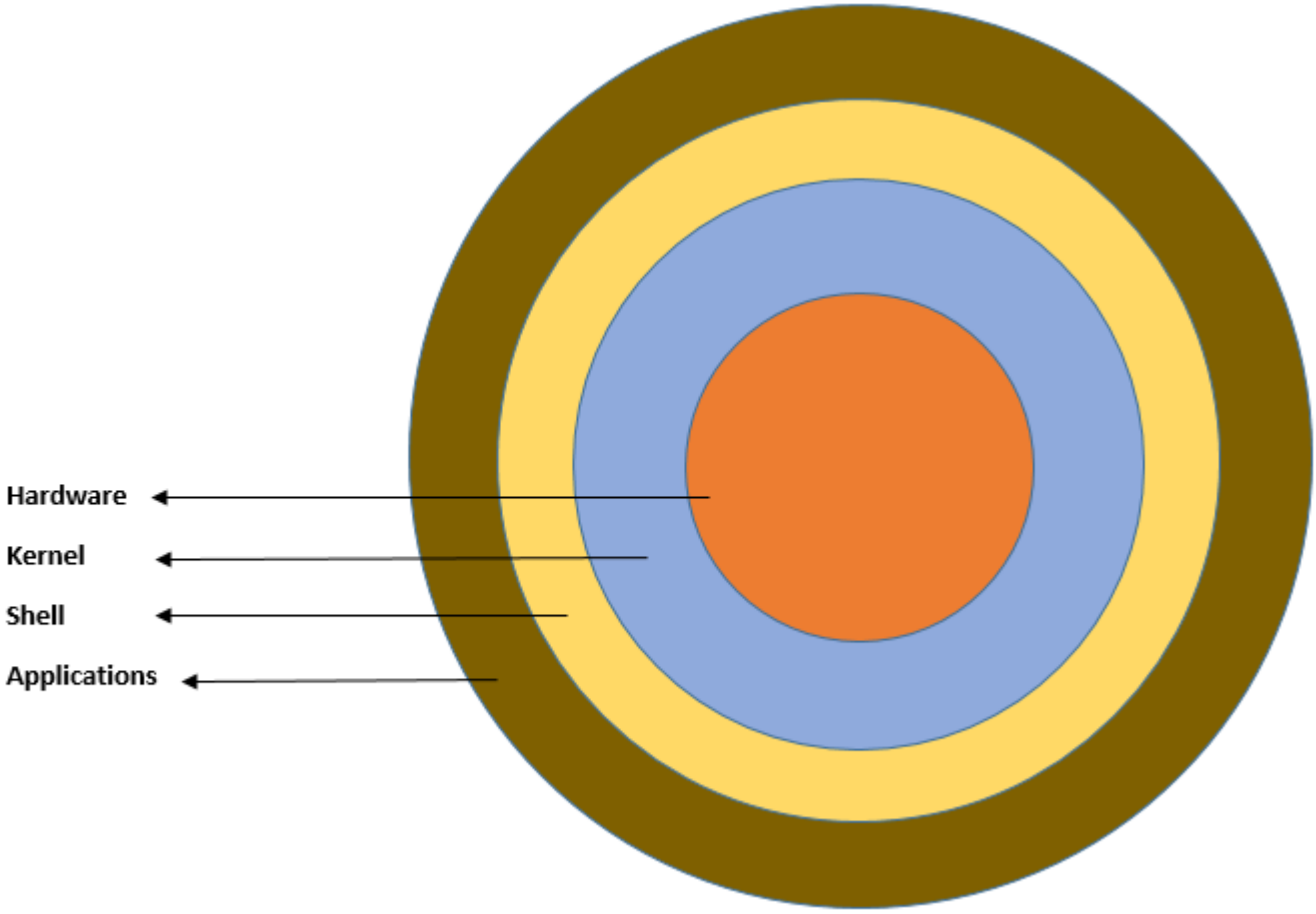
Unix-Geschmacksrichtungen

- AIX von IBM
- Solaris von Sun Microsystems
- HP-UX von Hewlett Packard
- IRIX von Silicon Graphics, Inc
- FreeBSD von Free BSD Group
- GNU / Linux von Open Source Movement
- SCO Unix von der Santa Cruz Operation Inc

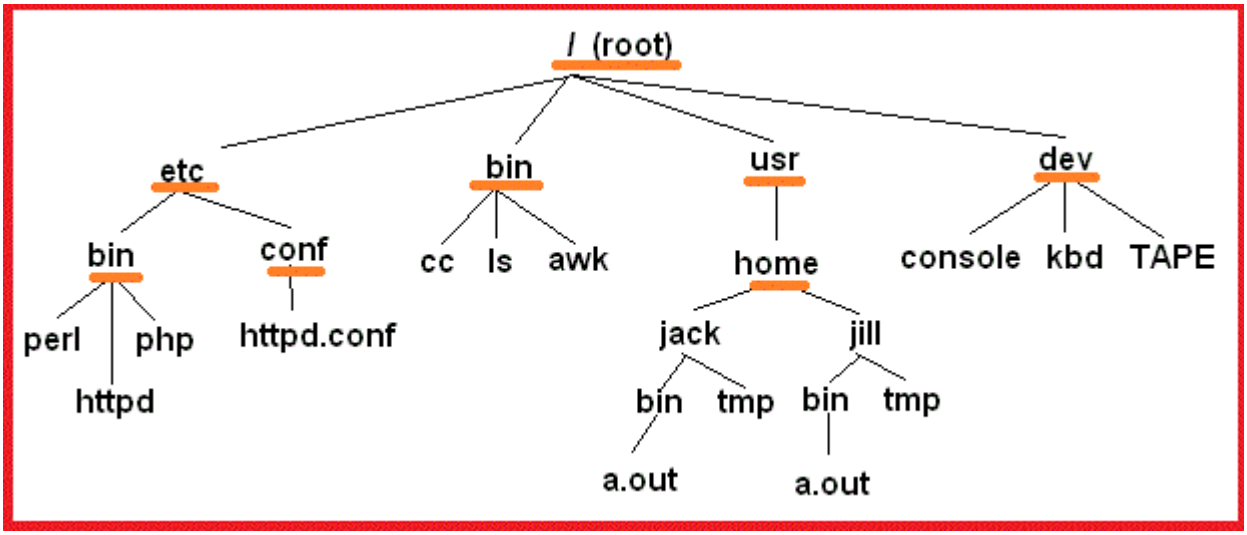
Funktionen von UNIX

- Mehrbenutzer
- Multi-Tasking
- Interaktiv
- Schale
- Sicherheit
- Hierarchisches Dateisystem

Unix-Architektur



Unix-Dateisysteme



Überblick über Unix online lesen: https://riptutorial.com/de/unix/topic/9338/uberblick-uber-unix

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Unix	Community
2	Berechtigungen	Anand C, Chris Forrence, Joey Chatterjee, rama chandra sunkara, William Carron
3	Erste Schritte mit Unix-Befehlen	eli-bd
4	Grundlegende Konsolenbefehle	Anand C, Boon, Dirk Schumacher, eli-bd, Jean-Baptiste Yunès, Nathaniel Ford, SarcasticSully
5	Sehen Sie sich die Manual Pages an	Benjamin W., drunken_monkey, intboolstring, Jahid, Simon Jester, Srinidhi, Will
6	Überblick über Unix	Anand C