Dateien und Verzeichnisse

TECH STARTER

Agenda

1. Absolute und relative Pfade
2. Dateien auflisten
3. Dateien & Verzeichnisse erstellen, verschieben und löschen
4. Globbing

Dateien und Verzeichnisse

Absolute und relative Pfade

TECH STARTER

Dateien und Verzeichnisse

- ➔ Linux-Dateisystem ähnlich zu anderen Betriebssystemen
- ➔ Dateien = enthalten Daten (menschenlesbaren Text, ausführbare Programme, binäre Daten)
- ➔ Verzeichnisse = dienen zur Organisation des Dateisystems

Dateien und Verzeichnisse

```
$ tree

Documents
├── Mission-Statement.txt
└── Reports
    └── report2018.txt

1 directory, 2 files
```

- ➔ Hier ist *Documents* ein **Verzeichnis**
- ➔ **Datei** (*Mission-Statement.txt*) und **Unterverzeichnis** (*Reports*) in *Documents* enthalten
- ➔ *Reports* enthält wieder eine **Datei** *reports2018.txt*

Datei- und Verzeichnisnamen

- ➔ Können Groß- und Kleinbuchstaben, Ziffern, Leerzeichen und Sonderzeichen enthalten
- ➔ Achtung bei Sonderzeichen (Leerzeichen und $) `$ cd Mission\ Statements`
- ➔ Dateinamen können einen Suffix enthalten, das auf einen Punkt folgt
- ➔ z.B. .txt zeigt an, dass es sich um eine Klartextdatei handelt
- ➔ Achtung: In Linux hat das keine besondere Bedeutung. Es könnte jede Art von Daten enthalten sein

Aktuellen Standort ermitteln

```
user@hostname ~/Documents/Reports $
```

```
user@hostname ~/Documents/Reports $ pwd
/home/user/Documents/Reports
```

- ➔ / zeigt Beziehung zwischen Verzeichnissen an
- ➔ ~ steht für das Home-Verzeichnis

- ➔ Wir müssen wissen, wo wir uns befinden
- ➔ Eingabeaufforderung gibt das an
- ➔ Dieselbe Information gibt auch Befehl pwd an

Verzeichnisinhalt auflisten

```
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

- ➔ Mit ls
- ➔ ls enthält keine Informationen über das übergeordnete Verzeichnis bzw. den Unterverzeichnissen

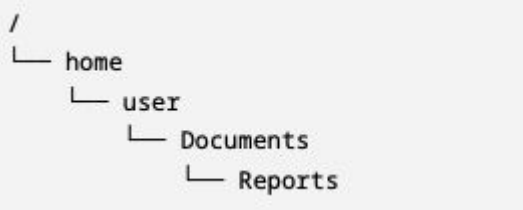
Aktuelles Verzeichnis wechseln

```
user@hostname ~ $ cd /home/user/Documents
user@hostname ~/Documents $ pwd
/home/user/Documents
user@hostname ~/Documents $ ls
Mission-Statement.txt Reports
```

```
User@hostname ~/Documents $ cd Reports
user@hostname ~/Documents/Reports $ pwd
/home/user/Documents/Reports
user@hostname ~/Documents/Reports $ ls
report2018.txt
```

➔ cd (change directory)

Absolute Pfade



- ➔ pwd gibt immer den absoluten Pfad an, d.h. Der Pfad enthält jeden Schritt des Pfades vom Ausgangspunkt des Dateisystems (/) bis zum aktuellen Punkt.
- ➔ Absolute Pfade beginnen immer mit einem /
- ➔ Absoluter Pfad enthält also immer alle Informationen, um von überall im Dateisystem zum jeweiligen Verzeichnis zu wechseln

Relativer Pfad

Ich besuche dich in deinem Haus und du sagst mir, dein Freund wohnt nebenan. Ich würde diese Angabe verstehen, weil sie relativ zu meinem aktuellen Standort ist. Wenn du mir die Beschreibung aber z.B. am Telefon gibst, dann wäre ich lost

- ➔ Mit cd Reports konnten wir deutlich weniger tippen
- ➔ Das ist ein relativer Pfad
- ➔ Relative Pfade beschreiben den Pfad immer in Bezug auf die aktuelle Position

Spezielle relative Pfade

```
user@hostname ~/Documents/Reports $ cd ..
user@hostname ~/Documents $ pwd
/home/user/Documents
```

```
user@hostname ~/Documents $ cd ../..
$ pwd
/home
```

- ➔ Wir können Pfadangaben verkürzen
- ➔ . steht für den aktuellen Standort
- ➔ .. steht für das Elternverzeichnis

Befehle

- ➔ **cd** Wechselt das aktuelle Verzeichnis.
- ➔ **pwd** Zeigt den Pfad des aktuellen Verzeichnisses.
- ➔ **ls** Listet den Inhalt eines Verzeichnisses auf und zeige die Eigenschaften von Dateien an.
- ➔ **mkdir** Erstellt ein neues Verzeichnis.
- ➔ **tree** Zeigt eine hierarchische Auflistung eines Verzeichnisbaums an.

Aufgabe

1. **Gebe für jeden der folgenden Pfade an, ob es sich um einen absoluten oder relativen Pfad handelt**
   a. **/home/user/Downloads**
   b. **../Reports**
   c. **/var**
   d. **docs**
   e. **/**

Aufgabe

```
$ sudo tree -F /

/
├── etc/
│   ├── network/
│   │   └── interfaces
│   ├── systemd/
│   │   ├── resolved.conf
│   │   ├── system/
│   │   ├── system.conf
│   │   ├── user/
│   │   └── user.conf
│   └── udev/
│       ├── rules.d/
│       └── udev.conf
└── home/
    ├── lost+found/
    └── user/
        └── Documents/

12 directories, 5 files
```

```
$ pwd
/etc/udev/rules.d
$ cd ../../systemd/user
$ cd ..
$ pwd
```

1. **Betrachte die folgende Dateistruktur. (Achtung: Verzeichnisse enden mit einem / wenn wir tree mit der Option -F nutzen). Die folgende Beispielausgabe zeigt keine vollständige Verzeichnisstruktur. Nutze sie aber zur Beantwortung der folgenden Fragen:**
   a. **Ein Benutzer gibt cd /etc/udev und ls -a ein. Wie lautet die Ausgabe?**
   b. **Unser aktueller Standort ist root (/). Wie gelangen wir ins Verzeichnis lost+found im Verzeichnis home?**
   c. **Unser aktueller Standort ist root (/). Wie kommen wir ins Verzeichnis /etc/network?**
   d. **Unser aktueller Standort ist /home/user/Documents. Wir gelangen wir ins Verzeichnis /home/user?**
   e. **Unser aktueller Standort ist /etc/systemd/system. Wir kommen wir ins Verzeichnis /home/user?**

Aufgabe

```
$ pwd
/etc/udev/rules.d
$ cd ../../systemd/user
$ cd ..
$ pwd
```

1. **Wie lautet die Ausgabe?**

Aufgabe

```
$ mkdir "this is a test"
$ ls
this is a test
```

1. Schreibe die folgenden Befehle in deine Shell. Mit welchen cd-Befehl könnten wir in das Verzeichnis wechseln?
2. Wiederhole das ganze, aber drücke nach cd this die TAB-Taste. Was siehst du in der Shell?
3. Versuche ein Verzeichnis zu erstellen, dessen Name das Zeichen \ enthält.

Dateien und Verzeichnisse

Dateien auflisten

Heimatverzeichnisse

```
$ tree -L 1 /
/
├── bin
├── boot
├── cdrom
├── dev
├── etc
├── home
├── lib
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```

- Die meisten dieser Verzeichnisse sind auf allen Linux-Systemen zu finden
  - Is in /bin
  - Systemkonfiguration durch Anpassung von Dateien in /etc
  - Systemprotokolle in /var
- Änderungen, die wie im Root-Dateisystem vornehmen, betreffen alle Benutzer
- Änderungen von Dateien im Root-Dateisystem erfordern Administratorrechte.

Heimatverzeichnisse

```
$ tree /home
/home
├── user
│   └── Documents
│       ├── Mission-Statement
│       └── Reports
│           └── report2018.txt
├── michael
│   ├── Documents
│   │   └── presentation-for-clients.odp
│   └── Music
```

Unter Linux ist /home ähnlich wie ein Mehrfamilienhaus: Viele Nutzer haben ihren eigenen, abgetrennten Bereich. Versorgung und Wartung des Gebäudes selbst liegt hingegen in der Verantwortung des Hausverwalters namens root.

TECH STARTER

```
$ tree -L 1 /home
/home
├── user
├── michael
└── lara
```

➔ Wir konzentrieren uns nun auf Verzeichnis /home

```
$ tree /home/user
user
└── Documents
    ├── Mission-Statement
    └── Reports
        └── report2018.txt
```

Besonderer relativer Pfad für das Heimatverzeichnis

➔ Wenn wir eine Terminalsitzung in Linux starten, sehen wir `user@hostname ~ $`

➔ Die Tilde ~ repräsentiert das Heimatverzeichnis

Linux ist einem Mehrfamilienhaus ähnlich, in dem viele Benutzer in /home wohnen. Der Bereich des Benutzers user wird sich folglich von dem des Benutzers michael unterscheiden. Das werden wir mit dem Befehl su (“switch user”) zeigen, der den Benutzer wechselt.

```
user@hostname ~ $ pwd
/home/user
user@hostname ~ $ su - michael
Password:
michael@hostname ~ $ pwd
/home/michael
```

➔ Bedeutung von ~ ändert sich je nach Benutzer: Für michael ist der absolute Pfad von ~ /home/michael, für lara ist er /home/lara

Relative-to-Home-Pfade

➔ Verwendung von ~ in Befehlen ist praktisch solange wir den Benutzer nicht wechseln

```
$ ls
Documents
$ cd Documents
$ ls
Mission-Statement
Reports
$ cd Reports
$ ls
report2018.txt
$ cd ~
$ ls
Documents
```

➔ Beachte, dass Benutzer eine neue Sitzung stets im Homeverzeichnis beginnen

➔ cd ~ bewirkt dasselbe wie cd (mit Leerzeichen)

➔ Wir können die Home-Verzeichnisse anderer Benutzer angeben, indem wir den Benutzernamen direkt nach der Tilde angeben

```
$ ls ~michael
Documents
Music
```

Versteckte Dateien und Verzeichnisse

```
$ ls -a ~
.
..
.bash_history
.bash_logout
.bash-profile
.bashrc
```

- ➔ ls -a hat uns alle Dateien und Verzeichnisse aufgelistet, einschließlich versteckter Dateien und Verzeichnisse
- ➔ Versteckte Dateien und Verzeichnisse beginnen immer mit einem Punkt
- ➔ i.d.R. gibt es im Home-Verzeichnis viele versteckte Dateien
  - ◆ Benutzerspezifische Konfigurationseinstellungen

Long-list Option

```
$ ls -l
-rw-r--r-- 1 user staff      3606 Jan 13  2017 report2018.txt
```

- **-rw-r—r--** Dateityp und Berechtigungen der Datei. Beachte, dass eine normale Datei mit Bindestrich beginnt und ein Verzeichnis mit d.
- **1** Anzahl der Links zur Datei.
- **user staff** Gibt den Eigentümer der Datei an, in diesem Fall also user. Zudem ist die Datei der Gruppe staff zugeordnet.
- **3606** Größe der Datei in Bytes.
- **Jan 13 2017** Zeitstempel der letzten Änderung an der Datei.
- **report2018.txt** Name der Datei.

TECH STARTER

- Befehl ls hat viele Optionen, um sein Verhalten zu ändern
- -l erstellt eine long list. D.h. Dateien und Verzeichnisse belegen in der Ausgabe jeweils eine Zeile, und es werden zusätzliche Informationen über jede Datei und jedes Verzeichnis angezeigt.

Weitere ls-Optionen

- **ls -lh** Die Kombination von “langer Liste” (“long list”) mit “menschenlesbaren” (“human readable”) Dateigrößen gibt uns nützliche Suffixe wie M für Megabyte oder K für Kilobyte.
- **ls -d */** Die Option -d listet Verzeichnisse auf, aber nicht deren Inhalt. Kombiniert mit */ zeigt sie nur Unterverzeichnisse und keine Dateien.
- **ls -lt** Kombiniert “lange Liste” mit der Option, nach “Zeitpunkt der letzten Änderung” zu sortieren. Dateien mit den letzten Änderungen stehen oben, Dateien mit den ältesten Änderungen unten, wobei die Reihenfolge auch umgekehrt werden kann.
- **ls -lrt** Kombiniert “lange Liste” mit “Zeitpunkt der letzten Änderung” und -r für “umgekehrte Sortierung” (“reverse order”), so dass Dateien mit den letzten Änderungen am Ende der Liste stehen. Neben der Sortierung nach “Zeitpunkt der letzten Änderung” sind auch “Zeitpunkt des letzten Zugriffs” oder nach “Zeitpunkt der letzten Statusänderung” möglich.
- **ls -lX** Kombiniert “lange Liste” mit der Sortierung nach Dateiendungen (“eXtension”), um z.B. alle Dateien, die mit .txt oder mit .jpg enden, zusammenzufassen.
- **ls -S -S** sortiert nach Dateigröße, so wie -t nach Zeit oder -X nach Dateiendung, wobei die größten Dateien an erster Stelle stehen und die kleinsten zuletzt. Inhalte von Unterverzeichnissen sind übrigens von der Sortierung ausgenommen.
- **ls -R** Die Option -R bewirkt für den Befehl ls, dass er eine rekursive Liste ausgibt. Was bedeutet das?

Rekursion in Bash

➔ Rekursion bezeichnet eine Situation, in der “etwas in sich selbst definiert” ist. Rekursion ist ein sehr wichtiger Begriff in der Informatik, aber hier ist seine Bedeutung viel einfacher.

```
$ ls ~
Documents
```

➔ ls gibt uns Dateien und Verzeichnisse von einem Ort. Wir wollen jetzt aber auch die Unterverzeichnisse einschließen.

```
$ tree /home/user
user
└── Documents
    ├── Mission-Statement
    └── Reports
        └── report2018.txt

$ ls -R ~
/home/user/:
Documents

/home/user/Documents:
Mission-Statement
Reports

/home/user/Documents/Reports:
report2018.txt
```

Rekursion in Bash

➔ Rekursion bezeichnet eine Situation, in der “etwas in sich selbst definiert” ist. Rekursion ist ein sehr wichtiger Begriff in der Informatik, aber hier ist seine Bedeutung viel einfacher.

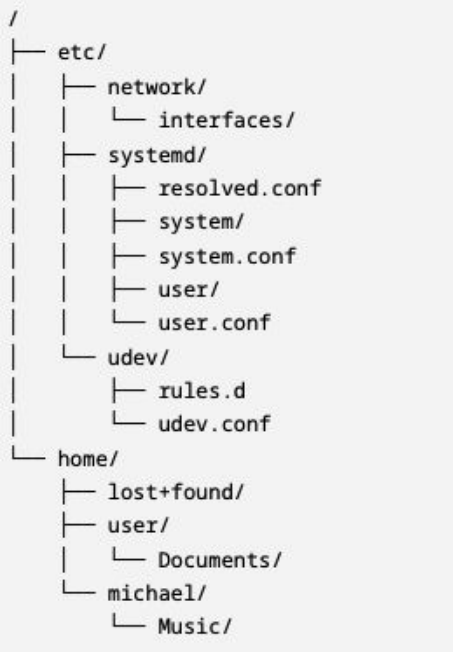
ls -R = “Führe ls hier aus und wiederhole den Befehl in jedem Unterverzeichnis, das du findest”

```
$ tree /home/user
user
└── Documents
    ├── Mission-Statement
    └── Reports
        └── report2018.txt

$ ls -R ~
/home/user/:
Documents

/home/user/Documents:
Mission-Statement
Reports

/home/user/Documents/Reports:
report2018.txt
```

Befehle

- ➔ **-a (all)** Gibt alle Dateien/Verzeichnisse aus, einschließlich der versteckten.
- ➔ **-d (directories)** Gibt alle Verzeichnisse aus, nicht deren Inhalt.
- ➔ **-h (human readable)** Gibt Dateigrößen in einem menschenlesbaren Format aus.
- ➔ **-l (long list)** Liefert zusätzliche Details mit einer Datei/einem Verzeichnis pro Zeile.
- ➔ **-r (reverse)** Kehrt die Reihenfolge einer Sortierung um.
- ➔ **-R (recursive)** Listet jede Datei, einschließlich der Dateien in allen Unterverzeichnissen.
- ➔ **-S (size)** Sortiert nach Dateigröße.
- ➔ **-t (time)** Sortiert nach dem Zeitpunkt der letzten Änderung.
- ➔ **X (eXtension)** Sortiert nach Dateiendung.

Aufgabe



1. Welcher Befehl wechselt in das Verzeichnis network unabhängig vom aktuellen Standort?
2. Welchen Befehl kann user eingeben, um von /etc/udev ins Verzeichnis Documents zu wechseln? (kürzester)
3. Welchen Befehl kann user eingeben, um ins das Verzeichnis music des Benutzers michael zu wechseln? (kürzester)

Aufgabe

```
drwxrwxrwx  5 eric eric  4.0K Apr 26  2011 China/
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0066.jpg
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0067.jpg
-rwxrwxrwx  1 eric eric  1.6M Jul 18  2011 img_0074.jpg
-rwxrwxrwx  1 eric eric  1.8M Jul 18  2011 img_0075.jpg
-rwxrwxrwx  1 eric eric   46K Jul 18  2011 scary.jpg
-rwxrwxrwx  1 eric eric  469K Jan 29  2018 Screenshot from 2017-08-13 21-22-24.png
-rwxrwxrwx  1 eric eric  498K Jan 29  2018 Screenshot from 2017-08-14 21-18-07.png
-rwxrwxrwx  1 eric eric  211K Jan 29  2018 Screenshot from 2018-01-06 23-29-30.png
-rwxrwxrwx  1 eric eric  150K Jul 18  2011 tobermory.jpg
drwxrwxrwx  6 eric eric  4.0K Apr 26  2011 Tokyo/
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 081.jpg
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 085.jpg
-rwxrwxrwx  1 eric eric  944K Jul 18  2011 Toronto 152.jpg
-rwxrwxrwx  1 eric eric  728K Jul 18  2011 Toronto 173.jpg
drwxrwxrwx  2 eric eric  4.0K Jun  5  2016 Wallpapers/
```

1. Betrachte die Ausgabe von ls -lh. Beachte, dass Verzeichnisse mit einem d am Zeilenanfang gekennzeichnet sind
   a. Welche Datei steht zu Beginn, wenn wir den Befehl ls -lrS ausführen?
   b. Beschreibe, welche Ausgabe wir von dem Befehl ls -ad */ erwarten

Aufgabe

```
drwxrwxrwx  5 eric eric  4.0K Apr 26  2011 China/
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0066.jpg
-rwxrwxrwx  1 eric eric  1.5M Jul 18  2011 img_0067.jpg
-rwxrwxrwx  1 eric eric  1.6M Jul 18  2011 img_0074.jpg
-rwxrwxrwx  1 eric eric  1.8M Jul 18  2011 img_0075.jpg
-rwxrwxrwx  1 eric eric   46K Jul 18  2011 scary.jpg
-rwxrwxrwx  1 eric eric  469K Jan 29  2018 Screenshot from 2017-08-13 21-22-24.png
-rwxrwxrwx  1 eric eric  498K Jan 29  2018 Screenshot from 2017-08-14 21-18-07.png
-rwxrwxrwx  1 eric eric  211K Jan 29  2018 Screenshot from 2018-01-06 23-29-30.png
-rwxrwxrwx  1 eric eric  150K Jul 18  2011 tobermory.jpg
drwxrwxrwx  6 eric eric  4.0K Apr 26  2011 Tokyo/
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 081.jpg
-rwxrwxrwx  1 eric eric  1.4M Jul 18  2011 Toronto 085.jpg
-rwxrwxrwx  1 eric eric  944K Jul 18  2011 Toronto 152.jpg
-rwxrwxrwx  1 eric eric  728K Jul 18  2011 Toronto 173.jpg
drwxrwxrwx  2 eric eric  4.0K Jun  5  2016 Wallpapers/
```

1. Führe den Befehl ls -lh in einem Verzeichnis aus, das Unterverzeichnisse enthält. Beachte die angezeigte Größe der Verzeichnisse. Scheint dir die Dateigröße korrekt? Entspricht sie dem Inhalt aller Dateien in diesem Verzeichnis?
2. Probiere mal du -h als Befehl aus und beschreibe die Ausgabe
3. Auf vielen Linux-Systemen kann man ll eingeben und wir erhalten dieselbe Ausgabe wie bei ls -k. Beachte jedoch, dass ll kein Befehl ist. man 11 wird beispielsweise darauf hinweisen, dass keine entsprechende Manpage existiert. Es ist ein Beispiel für einen Alias. Inwiefern können Aliase nützlich sein?

Übung

1. Verwende die Man Page cp, um herauszufinden, wie man eine Kopie einer Datei erstellt und die Berechtigungen und Änderungszeiten mit dem Original übereinstimmen
2. Was bewirkt der Befehl rmdir -p? Experimentiere damit und erkläre, wie er sich von rm -r unterscheidet
3. Führe den folgenden Befehl NICHT AUS: rm -ri /* (NICHT AUSFÜHREN!!!!). Was würde der machen?
4. Ist es möglich, außer mit -i zu verhindern, dass mv Zieldateien überschreibt?
5. Erkläre den Befehl cp -u

Dateien & Verzeichnisse erstellen, verschieben und löschen

Dateien & Verzeichnisse erstellen, verschieben und löschen

- ➔ Befehlszeile als effektivster Weg, schnell Dateien und Verzeichnisse zu erstellen, verschieben und löschen
- ➔ Groß- und Kleinschreibung beachten

```
$ cd /
$ ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
$ cd ETC
bash: cd: ETC: No such file or directory
$ pwd
/
$ cd etc
$ pwd
/etc
```

Verzeichnisse erstellen

```
$ mkdir linux_essentials-2.4
$ ls
Desktop  Documents  Downloads  linux_essentials-2.4
```

```
$ mkdir -p creating moving copying/files copying/directories deleting/directories
deleting/files globs
```

- ➔ mkdir
- ➔ entweder ein oder mehrere Verzeichnisse
- ➔ Achtung beim Erstellen von Unterverzeichnissen: mkdir -p erstellt auch das übergeordnete Verzeichnis

Dateien erstellen

```
$ touch globs/question1 globs/question2012 globs/question23 globs/question13
globs/question14
$ touch globs/star10 globs/star1100 globs/star2002 globs/star2013
```

- ➔ touch
- ➔ Erstellt leere Datei
- ➔ Wenn wir das auf eine bestehende Datei anwenden, ändert sich der Inhalt nicht, aber der Zeitstempel

Dateien erstellen

```
$ touch globs/question1 globs/question2012 globs/question23 globs/question13
globs/question14
$ touch globs/star10 globs/star1100 globs/star2002 globs/star2013
```

Mit cat können wir uns die Datei ausgeben lassen. Diese müsste initial leer sein

Mit echo und dem Operator > können wir was reinschreiben

```
$ echo hello > question15
$ cat question15
hello
```

- ➔ touch
- ➔ Erstellt leere Datei
- ➔ Wenn wir das auf eine bestehende Datei anwenden, ändert sich der Inhalt nicht, aber der Zeitstempel

echo zeigt Text auf der Kommandozeile an. Das Zeichen > weist die Shell an, die Ausgabe eines Befehls in die angegebene Datei (statt ins Terminal) zu schreiben, was dazu führt, dass die Ausgabe von echo, in diesem Fall hello, in die Datei question15 geschrieben wird. Das ist nicht spezifisch für echo, das geht mit jedem anderen Befehl.

Dateien erstellen

```
$ touch globs/question1 globs/question2012 globs/question23 globs/question13
globs/question14
$ touch globs/star10 globs/star1100 globs/star2002 globs/star2013
```

Mit cat können wir uns die Datei ausgeben lassen. Diese müsste initial leer sein

Mit echo und dem Operator > können wir was reinschreiben

```
$ echo hello > question15
$ cat question15
hello
```

- ➔ touch
- ➔ Erstellt leere Datei
- ➔ eine
- ➔ anwenden,
- ➔ halt nicht,
- ➔ bel

Vorsicht bei der Verwendung von >! Wenn die genannte Datei bereits existiert, wird sie überschrieben!

echo zeigt Text auf der Kommandozeile an. Das Zeichen > weist die Shell an, die Ausgabe eines Befehls in die angegebene Datei (statt ins Terminal) zu schreiben, was dazu führt, dass die Ausgabe von echo, in diesem Fall hello, in die Datei question15 geschrieben wird. Das ist nicht spezifisch für echo, das geht mit jedem anderen Befehl.

Umbenennen von Dateien

```
$ touch file1 file22
$ echo file3 > file3
$ echo file4 > file4
$ ls
file1  file22  file3  file4
```

```
$ mv file22 file2
$ ls
file1  file2  file3  file4
```

- ➔ mv
- ➔ Verschiebt oder benennt eine Datei um
- ➔ Achtung: Wenn wir eine bestehende Datei angeben, wird diese mit der anderen überschreiben → wenn wir mv -i angeben, erzwingen wir vor der Ausführung eine Bestätigung

```
$ touch file4 file5
$ mv -i file4 file3
mv: overwrite 'file3'? y
```

Verschieben von Dateien

```
$ cd ~/linux_essentials-2.4/moving
$ mkdir dir1 dir2
$ ls
dir1  dir2  file1  file2  file3  file5
```

```
$ mv file1 dir1
$ ls
dir1  dir2  file2  file3  file5
$ ls dir1
file1
```

- ➔ mv
- ➔ Das letzte Argument von mv ist das Zielverzeichnis (wenn’s ein Verzeichnis ist, wird’s verschoben, ansonsten umbenannt)
- ➔ Mit mv können wir auch Verzeichnisse verschieben und umbenennen

```
$ ls
dir1  dir2  file5
$ ls dir1
file1
$ mv dir1 dir3
$ ls
dir2  dir3  file5
$ ls dir3
file1
```

Löschen von Dateien und Verzeichnissen

```
$ cd ~/linux_essentials-2.4/moving
$ ls
dir2  dir3  file5
$ rmdir file5
rmdir: failed to remove 'file5': Not a directory
$ rm file5
$ ls
dir2  dir3
```

```
$ rmdir deleting
rmdir: failed to remove 'deleting': Directory not empty
$ ls -l deleting
total 0
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 files
```

- ➔ rm löscht Dateien und Verzeichnisse
- ➔ rmdir löscht Verzeichnisse (aber nur leere)
- ➔ rm funktioniert nur bei normalen Dateien
  - ◆ -r löscht alle Unterverzeichnisse
  - ◆ -i erfordert eine Benutzerbestätigung

TECH STARTER

```
$ ls
copying  creating  deleting  globs  moving
$ rm deleting
rm: cannot remove 'deleting': Is a directory
$ ls -l deleting
total 0
drwxrwxr-x. 2 emma emma 6 Mar 26 14:58 directories
$ rm -r deleting
$ ls
copying  creating  globs  moving
```

```
$ rm -ri moving
rm: descend into directory 'moving'? y
rm: descend into directory 'moving/dir2'? y
rm: remove regular empty file 'moving/dir2/file2'? y
rm: remove regular empty file 'moving/dir2/file3'? y
rm: remove directory 'moving/dir2'? y
rm: descend into directory 'moving/dir3'? y
rm: remove regular empty file 'moving/dir3/file1'? y
rm: remove directory 'moving/dir3'? y
rm: remove directory 'moving'? y
```

Kopieren von Dateien und Verzeichnissen

```
$ cd ~/linux_essentials-2.4/copying
$ ls
directories  files
$ cp /etc/nsswitch.conf files/nsswitch.conf
$ cp /etc/issue /etc/hostname files
```

```
$ cd ~/linux_essentials-2.4/copying/files
$ ls
hostname  issue  nsswitch.conf
$ cat hostname
mycomputer
$ cat issue
Debian GNU/Linux 9 \n \l

$ cp hostname issue
$ cat issue
mycomputer
```

- ➔ cp
- ➔ Letztes Argument ein Verzeichnis → Kopie der vorherigen Argumente innerhalb dieses Verzeichnisses
- ➔ Beide Operanden Dateien → Überschreiben der zweiten Datei mit Kopie der ersten
- ➔ Wenn wir Verzeichnisse kopieren wollen, dann mit cp -r

```
$ cp -r files directories
```

Aufgabe

```
$ pwd
/tmp
$ find
.
./outfiles
./outfiles/text
```

1. Gegeben sei die folgende Umgebung (links). Markiere die Verzeichnisse, die der Befehl mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today erzeugen würde
   a. /tmp
   b. /tmp/outfiles
   c. /tmp/outfiles/text
   d. /tmp/outfiles/text/today
   e. /tmp/infiles
   f. /tmp/infiles/text
   g. /tmp/infiles/text/today

Lösung

```
$ pwd
/tmp
$ find
.
./outfiles
./outfiles/text
```

1. Gegeben sei die folgende Umgebung (links). Markiere die Verzeichnisse, die der Befehl mkdir -p /tmp/outfiles/text/today /tmp/infiles/text/today erzeugen würde

/tmp	
/tmp/outfiles	
/tmp/outfiles/text	
/tmp/outfiles/text/today	X
/tmp/infiles	X
/tmp/infiles/text	X
/tmp/infiles/text/today	X

Aufgabe

Was bewirkt -v bei mkdir, rm und cp?

Lösung

```
$ rm -v a b
removed 'a'
removed 'b'
$ mv -v a b
'a' -> 'b'
$ cp -v b c
'b' -> 'c'
```

Was bewirkt -v bei mkdir, rm und cp?

- Typischerweise schaltet -v die ausführliche Ausgabe ein. Es bewirkt, dass die jeweiligen Programme ausgeben, was sie tun, während sie es tun:

Aufgabe

Was passiert, wenn wir versehentlich versuchen, drei Dateien auf der gleichen Befehlszeile in eine bereits vorhandene Datei zu kopieren, anstatt in ein Verzeichnis?

Lösung

```
$ touch a b c d
$ cp a b c d
cp: target 'd' is not a directory
```

Was passiert, wenn wir versehentlich versuchen, drei Dateien auf der gleichen Befehlszeile in eine bereits vorhandene Datei zu kopieren, anstatt in ein Verzeichnis?

- cp weigert sich, etwas zu tun
- Es gibt eine Fehlermeldung

Aufgabe

Was passiert wenn wir mit mv ein Verzeichnis in sich selbst verschieben?

Lösung

```
$ mv a a
mv: cannot move 'a' to a subdirectory of itself, 'a/a'
```

Was passiert wenn wir mit mv ein Verzeichnis in sich selbst verschieben?

- Wir erhalten eine Fehlermeldung, die uns mitteilt, dass mv das nicht tun kann

Aufgabe

Verwende die Man Page cp, um herauszufinden, wie man eine Kopie einer Datei erstellt und die Berechtigungen und Änderungszeiten mit dem Original übereinstimmen.

Lösung

Verwende die Man Page cp, um herauszufinden, wie man eine Kopie einer Datei erstellt und die Berechtigungen und Änderungszeiten mit dem Original übereinstimmen.

- Mit -p

```
$ man cp
-p     same as --preserve=mode,ownership,timestamps
--preserve[=ATTR_LIST]
              preserve the specified attributes (default: mode,ownership,time-
              stamps), if  possible  additional  attributes:  context,  links,
              xattr, all
```

Aufgabe

Was bewirkt der Befehl rmdir -p? Teste ein wenig. Erkläre, wie er sich von rm -r unterscheidet.

Lösung

```
$ find
.
./a
./a/b
./a/b/c
$ rmdir -p a/b/c
$ ls
```

Was bewirkt der Befehl rmdir -p? Teste ein wenig. Erkläre, wie er sich von rm -r unterscheidet.

- rmdir ähnlich wie mkdir -p
- Wenn wir einen Baum mit leeren Verzeichnissen übergeben, werden alle entfernt

Aufgabe

Bitte nicht ausführen:
Was würde in der Theorie der Befehl rm -ri /* bewirken?

Lösung

Bitte nicht ausführen:
Was würde in der Theorie der Befehl rm -ri /* bewirken?

- Es werden alle Dateien und Verzeichnisse entfernt, die von deinem Benutzerkonto beschreibbar sind. Dazu gehören auch alle Netzwerk-Dateisysteme

Aufgabe

Ist es möglich, außer mit -i z verhindern, dass mv Zieldateien überschreibt?

Lösung

```
$ cat a
a
$ cat b
b
$ mv -n a b
$ cat b
b
```

Ist es möglich, außer mit -i z verhindern, dass mv Zieldateien überschreibt?

- Mit der Option -n bzw. –no-clobber verhindert, dass mv Dateien überschreibt

Aufgabe

Was macht cp -u?

Lösung

```
$ ls -l
total 24K
drwxr-xr-x 123 emma student  12K Feb  2 05:34 ..
drwxr-xr-x   2 emma student 4.0K Feb  2 06:56 .
-rw-r--r--   1 emma student    2 Feb  2 06:56 a
-rw-r--r--   1 emma student    2 Feb  2 07:00 b
$ cat a
a
$ cat b
b
$ cp -u a b
$ cat b
b
$ cp -u a c
$ ls -l
total 12
-rw-r--r-- 1 emma student 2 Feb  2 06:56 a
-rw-r--r-- 1 emma student 2 Feb  2 07:00 b
-rw-r--r-- 1 emma student 2 Feb  2 07:00 c
```

TECH STARTER

Was macht cp -u?

- Die Option -u bewirkt, dass cp eine Datei nur dann kopiert, wenn das Ziel fehlt oder älter als die Quelldatei ist

Dateien und Verzeichnisse

Globbing

Globbing

- ➔ Einfache Musterabgleichsprache
- ➔ Es gibt bestimmte Musterabgleichzeichen:
  - ◆ * Entspricht einer beliebigen Anzahl von Zeichen, einschließlich kein Zeichen
  - ◆ ? Entspricht genau einem beliebigen Zeichen
  - ◆ [] Entspricht einer Klasse von Zeichen

```
$ cd ~/linux_essentials-2.4/globs
$ ls
question1   question14  question2012  star10    star2002
question13  question15  question23    star1100  star2013
$ ls star1*
star10  star1100
$ ls star*
star10  star1100  star2002  star2013
$ ls star2*
star2002  star2013
$ ls star2*2
star2002
$ ls star2013*
star2013
```

```
$ ls
question1   question14  question2012  star10    star2002
question13  question15  question23    star1100  star2013
$ ls question?
question1
$ ls question1?
question13  question14  question15
$ ls question?3
question13  question23
$ ls question13?
ls: cannot access question13?: No such file or directory
```

TECH STARTER

Globbing

- ➔ Einfache Musterabgleichsprache
- ➔ Es gibt bestimmte Musterabgleichzeichen:
  - * Entspricht einer beliebigen Anzahl von Zeichen, einschließlich kein Zeichen
  - ? Entspricht genau einem beliebigen Zeichen
  - [] Entspricht einer Klasse von Zeichen

```
$ ls
file1  file2  file3  file4  file5  file6  file7  filea  fileb  filec
$ ls file[1-2]
file1  file2
$ ls file[1-3]
file1  file2  file3
```

```
$ ls file[^a]
file1  file2  file3  file4  file5  file6  file7  fileb  filec
```

^ = alles außer

Globbing

```
$ ls file[[:digit:]]
file1  file2  file3  file4  file5  file6  file7
$ touch file1a file11
$ ls file[[:digit:]a]
file1  file2  file3  file4  file5  file6  file7  filea
$ ls file[[:digit:]]a
file1a
```

- Zeichenklassen
  - [:alnum:] Buchstaben und Zahlen.
  - [:alpha:] Groß- oder Kleinbuchstaben.
  - [:blank:] Leerzeichen und Tabs.
  - [:cntrl:] Steuerzeichen, z.B. Backspace, Glocke, NAK, Escape.
  - [:digit:] Zahlen (0123456789).
  - [:graph:] Alle graphischen Zeichen (alle Zeichen außer ctrl und Leerzeichen)
  - [:lower:] Kleinbuchstaben (a-z).
  - [:print:] Druckbare Zeichen (alnum, punct und das Leerzeichen).
  - [:punct:] Interpunktionszeichen, d.h. !, &, ".
  - [:space:] Whitespace-Zeichen, z.B. Tabs, Leerzeichen, Zeilenumbrüche.
  - [:upper:] Großbuchstaben (A-Z).
  - [:xdigit:] Hexadezimale Zahlen (normalerweise 0123456789abcdefABCDEF).

Befehle

- ➔ **cat** Liest und gibt den Inhalt einer Datei aus.
- ➔ **cp** Kopiert Dateien oder Verzeichnisse.
- ➔ **echo** Gibt eine Zeichenkette aus.
- ➔ **find** Geht durch einen Dateisystembaum und sucht nach Dateien, die einem bestimmten Satz von Kriterien entsprechen.
- ➔ **ls** Zeigt Eigenschaften von Dateien und Verzeichnissen und listet den Inhalt eines Verzeichnisses auf.
- ➔ **mkdir** Erstellt neue Verzeichnisse.
- ➔ **mv** Verschiebt Dateien oder Verzeichnisse und benennt sie um.
- ➔ **pwd** Gibt das aktuelle Arbeitsverzeichnis aus.
- ➔ **rm** Löscht Dateien oder Verzeichnisse.
- ➔ **rmdir** Löscht Verzeichnisse.
- ➔ **touch** Erstellt neue leere Dateien oder aktualisiert den Änderungszeitstempel einer bestehenden Datei.

Aufgabe

Wie würden wir alle Dateien in einem aktuellen Verzeichnis löschen, die mit old beginnen?

Lösung

```
$ rm old*
```

Wie würden wir alle Dateien in einem aktuellen Verzeichnis löschen, die mit old beginnen?

- Verwende einen Glob old* mit rm

Aufgabe

1. Welche der folgenden Dateien würden mit log_[a-z]_201?_*_01.txt übereinstimmen?
   a. log_3_2017_Jan_01.txt
   b. log_+_2017_Feb_01.txt
   c. log_b_2007_Mar_01.txt
   d. log_f_201A_Wednesday_01.txt
2. Erstelle ein paar Globs, die der folgenden Liste von Dateinamen entsprechen:

```
doc100
doc200
doc301
doc401
```

Lösung

```
$ ls log_[a-z]_201?_*_01.txt
log_f_201A_Wednesday_01.txt
```

1. Welche der folgenden Dateien würden mit log_[a-z]_201?_*_01.txt übereinstimmen?
   a. log_3_2017_Jan_01.txt
   b. log_+_2017_Feb_01.txt
   c. log_b_2007_Mar_01.txt
   d. log_f_201A_Wednesday_01.txt

   → log_[a-z] entspricht log_, gefolgt von jedem Kleinbuchstaben, so dass sowohl log_f_201A_Wednesday_01.txt als auch log_b_2007_Mar_01.txt übereinstimmen. _201? passt auf jedes einzelne Zeichen, daher passt nur log_f_201A_Wednesday_01.txt. Schließlich passt *_01.txt zu allem, was mit _01.txt endet, also passt unsere verbleibende Option.

Lösung

```
doc*
doc[1-4]*
doc?0?
doc[1-4]0?
```

1. Erstelle ein paar Globs, die der folgenden Liste von Dateinamen entsprechen:

```
doc100
doc200
doc301
doc401
```