

Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

26. Oktober 2023

Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

1 Einführung

2 Kurzeinführung Unix

2.1 Grundkonzepte

2.2 Die Kommandozeile: Grundlagen

2.3 Dateisysteme

2.4 Ein- und Ausgabeströme

2.5 Pipes

2.6 Verzweigungen und Schleifen

3 Kurzeinführung TCP/IP

3.1 IP-Adressen

3.2 MAC-Adressen

3.3 TCP- und UDP-Ports

...

...



Änderungen
vorbehalten

1.3 Dateisysteme

- Dateien listen: `ls`
langes Listenformat: `ls -l`
rückwärts nach Zeit sortiert: `ls -lrt`
- Datei ausgeben: `cat hello.c`
- Datei anzeigen: `less hello.c`

1.3 Dateisysteme

- Arbeitsverzeichnis anzeigen: `pwd`
- Arbeitsverzeichnis wechseln: `cd script`
(*kein* Programm, sondern Shell-Befehl)
- übergeordnetes Verzeichnis: `cd ..`
- eigenes *Home*-Verzeichnis: `cd`
- Wurzelverzeichnis: `cd /`
- wieder zurück: `cd -`

```
cassini/home/peter/bo/2013ss/net/script> cd /usr/bin
cassini/usr/bin> cd ../lib
cassini/usr/lib> cd
cassini/home/peter>
```

1.3 Dateisysteme

- Dateien kopieren (*copy*): `cp`
- Dateien verschieben/umbenennen (*move*): `mv`
- Dateien löschen (*remove*): `rm`

```
cassini/home/peter> cp -p foo/test.txt
cp: missing destination file operand after `foo/test.txt'
Try `cp --help' for more information.
cassini/home/peter> cp -p foo/test.txt .
cassini/home/peter> mv test.txt bla.txt
cassini/home/peter> cat bla.txt
Dies ist ein Test.
cassini/home/peter> rm bla.txt
cassini/home/peter>
```

Aktuelles Verzeichnis: `.`

1.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

1.3 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



Benutzer (u – *user*) darf lesen und schreiben


1.3 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



Gruppe (g – *group*) darf lesen

1.3 Dateisysteme

- Zugriffsrechte

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```



alle anderen (o – *other*) dürfen lesen

1.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
...
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

- Zugriffsrechte ändern:

```
chmod o-r es-20191009.tex – Lesezugriff entziehen
chmod g+w es-20191009.tex – Schreibzugriff gewähren
chmod 640 es-20191009.tex – auf -rw-r----- setzen
```

1.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/home/peter/bo/2019ws/es/20191009> ls -l
...
-rw-r--r-- 1 peter peter 24523 Okt  8 21:47 es-20191009.tex
```

- Zugriffsrechte ändern:

```
chmod o-r es-20191009.tex – Lesezugriff entziehen
chmod g+w es-20191009.tex – Schreibzugriff gewähren
chmod 640 es-20191009.tex – auf -rw-r----- setzen
                        6   4   0
```

1.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2019ws/es/20191002> cat test2.txt
ls -l
cassini/home/peter/bo/2019ws/es/20191002> chmod +x test2.txt
cassini/home/peter/bo/2019ws/es/20191002> ls -l test2.txt
-rwxr-xr-x 1 peter peter 6 Okt  2 13:43 test2.txt
cassini/home/peter/bo/2019ws/es/20191002> ./test2.txt
insgesamt 4828
lrwxrwxrwx 1 peter peter      18 Apr 13  2016 csa2.jpg -> ..
-rw-r--r-- 1 peter peter 4619138 Okt  8 21:28 es-20191002.pdf
...
```

- ausführbare Textdateien: *Skripte*

hier: ausführbare Textdatei mit Shell-Befehlen
(ohne spezielle Kennung): Shell-Skript

1.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2019ws/es/20191002> cat test2.txt
ls -l
cassini/home/peter/bo/2019ws/es/20191002> chmod +x test2.txt
cassini/home/peter/bo/2019ws/es/20191002> ls -l test2.txt
-rwxr-xr-x 1 peter peter 6 Okt  2 13:43 test2.txt
cassini/home/peter/bo/2019ws/es/20191002> ./test2.txt
insgesamt 4828
lrwxrwxrwx 1 peter peter      18 Apr 13  2016 csa2.jpg -> ..
-rw-r--r-- 1 peter peter 4619138 Okt  8 21:28 es-20191002.pdf
...
```

- ausführbare Textdateien: *Skripte*

hier: ausführbare Textdatei mit Shell-Befehlen
(ohne spezielle Kennung): Shell-Skript

Kennung: 1. Zeile enthält `#!` und den Interpreter,
z. B. `#!/bin/bash`

1.3 Dateisysteme

- Datenträger in Verzeichnis *einhängen*: `mount`

```
cassini/home/peter> ls /media/usb1/  
cassini/home/peter> mount /media/usb1  
cassini/home/peter> ls /media/usb1/  
es-20191002.pdf  hello.c  hexapode  KIS-Bericht.pdf  
cassini/home/peter> umount /media/usb1  
cassini/home/peter> ls /media/usb1/  
cassini/home/peter>
```

1.3 Dateisysteme

- *Symbolische Verknüpfungen – symbolic links*

Verweis auf die eigentliche Datei

→ Wenn man die Datei löscht, zeigt der Link ins Leere.

Verknüpfung anlegen: `ln -s datei link`

(Richtung: wie bei `cp`)

Beispiel: `ln -s ../common/GNU-GPL-3 gpl.txt`

1.3 Dateisysteme

- *Symbolische Verknüpfungen – symbolic links*

Verweis auf die eigentliche Datei

→ Wenn man die Datei löscht, zeigt der Link ins Leere.

Verknüpfung anlegen: `ln -s datei link`

(Richtung: wie bei `cp`)

Beispiel: `ln -s ../common/GNU-GPL-3 gpl.txt`

- *Harte Verknüpfungen – hard links*

Dieselben Daten auf dem Datenträger

sind unter mehreren Namen verfügbar.

→ Wenn man einen löscht, sind die Daten noch da.

```
cassini/home/peter/bo/2019ws/es/20191002> ls -l
```

```
...
```

```
-rw-r--r-- 1 peter peter      1202 Okt  2 13:35 shell-06.tx
```

```
drwxr-xr-x 2 peter peter     4096 Okt  2 13:16 test
```



Anzahl der („harten“) Links auf diese Datei / dieses Verzeichnis

1.3 Dateisysteme

- `grep`: Dateien durchsuchen

```
cassini/home/peter/bo/2019ws/es/20191002> grep gcc *.txt  
shell-03.txt: cassini/...> gcc -Wall -O hello.c -o hello
```

1.3 Dateisysteme

- **find**: Dateien anhand ihrer Eigenschaften suchen

```
$ find . -name "*.txt"
```

```
./shell-06.txt
```

```
./shell-03.txt
```

```
./shell-05.txt
```

```
./test.txt
```

```
./test/test.txt
```

```
...
```

```
$ find . -name "*.txt" -perm /u+x
```

```
./test2.txt
```

```
$ find . -name "*.txt" -perm /u+x -exec ls -l {} \;
```

```
-rwxr-xr-x 1 peter peter 6 Okt  2 13:43 ./test2.txt
```

1.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

1.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

- Standard-Ausgabe an Datei anhängen

```
$ echo "Dies ist noch ein Test." >> test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

```
Dies ist noch ein Test.
```

1.4 Ein- und Ausgabeströme

- Fehler-Ausgabe in Datei umleiten

```
$ cat gibtsnicht.txt > fehler.txt
cat: gibtsnicht.txt: No such file or directory
$ cat fehler.txt
$ cat gibtsnicht.txt 2> fehler.txt
$ cat fehler.txt
cat: gibtsnicht.txt: No such file or directory
```

1.4 Ein- und Ausgabeströme

- Standard-Eingabe aus Datei lesen

```
$ bc
bc 1.06.95
Copyright [...] 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
2 + 2
4
$ echo "2 + 2" > test.bc
$ bc < test.bc
4
```

1.5 Pipes

Standard-Ausgabe von Programm A
wird zu Standard-Eingabe von Programm B

```
$ echo "2 + 2" | bc  
4
```

—→ sehr mächtiger „Baukasten“

1.5 Pipes

- `sed`: *stream editor*

Suchen und Ersetzen (und noch viel mehr)

```
$ echo "Schlimmer geht nimmer." | sed -e 's/nim/im/g'  
Schlimmer geht immer.
```


1.5 Pipes

- `grep`: Standard-Eingabe durchsuchen
- `$(...)`: Output in Kommandozeile übernehmen

```
$ ls | grep slides
```

```
pgslides.sty
```

```
$ ls *.pdf | grep -v logo
```

```
es-20191002.pdf
```

```
Zeichen_123.pdf
```

```
$ ls -l $(ls *.pdf | grep -v logo)
```

```
-rw-r--r-- 1 ... 4619138 Okt 8 21:28 es-20191002.pdf
```

```
lrwxrwxrwx 1 ...          25 Okt 3  2016 Zeichen_123.pdf -> ...
```

1.6 Verzweigungen und Schleifen

```
$ if grep Blubb test.txt; then echo "gefunden"; \  
    else echo "nicht gefunden"; fi  
nicht gefunden  
$ for x in foo bar baz; do echo $x; done  
foo  
bar  
baz
```

1.6 Verzweigungen und Schleifen

```
$ if grep Blubb test.txt; then echo "gefunden"; \  
  else echo "nicht gefunden"; fi  
nicht gefunden  
$ for x in foo bar baz; do echo $x; done  
foo  
bar  
baz
```

Beispiel: Datei mit einer **while**-Schleife zeilenweise lesen,
Zeilennummern ergänzen und wieder ausgeben:

```
number=0  
cat test.txt \  
| while read line; do  
  number=$((number + 1))  
  echo "Zeile $number: $line"  
done
```

- **read**: Variable von Standardeingabe lesen
- **\$((...))**: arithmetischen Ausdruck auswerten

Übungsaufgabe

Schreiben Sie ein Shell-Skript, das aus einer selbst erstellten Textdatei (z. B. CSV) Daten extrahiert. Welche Daten, soll per Parameter angegeben werden.

Beispiele: Siehe [../20231018.p0/dbs-20231018.txt](#)

Hinweise:

- Parameter in Shell-Skripten: `$1`, `$2`, ...
- `man cut`
- `man head`
- `man tail`
- Sie dürfen voraussetzen, daß in der Textdatei gespeicherte Strings bestimmte Zeichen (z. B. das Trennzeichen) nicht enthalten.

Zusatzaufgabe

Schreiben Sie dasselbe Programm noch einmal in einer beliebigen Sprache, wobei die in der Textdatei gespeicherten Strings (nahezu) beliebige Zeichen enthalten dürfen. (Beispiel: Zwischen Anführungszeichen zählt auch das Trennzeichen als Bestandteil des Strings.)

Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

1 Einführung

2 Kurzeinführung Unix

2.1 Grundkonzepte

2.2 Die Kommandozeile: Grundlagen

2.3 Dateisysteme

2.4 Ein- und Ausgabeströme

2.5 Pipes

2.6 Verzweigungen und Schleifen

3 Kurzeinführung TCP/IP

3.1 IP-Adressen

3.2 MAC-Adressen

3.3 TCP- und UDP-Ports

...

...



Änderungen
vorbehalten

2.1 IP-Adressen

- `ip addr` (Linux)
 `ifconfig` (Unix allgemein)
 `ipconfig` (MS Windows)
- `ip addr add <Netz>`
- `ip link`
- `ping <IP-Adresse>`

```
# ifconfig
```

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    [...]
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.42.101  netmask 255.255.255.0
                                broadcast 192.168.42.255
    ether be:3f:ca:aa:7e:51 txqueuelen 1000  (Ethernet)
    [...]
```

2.1 IP-Adressen

IPv4-Adressen:

- 32 Bit
- dezimal, 4 Gruppen zu je 8 Bit (0–255), durch Punkte getrennt

IPv6-Adressen:

- 128 Bit
- hexadezimal, 8 Gruppen zu je 4 Hex-Ziffern, durch Doppelpunkte getrennt
- Führende Nullen dürfen weggelassen werden.
- Zwei Doppelpunkte bedeuten: Mit Nullen auffüllen.
- Literatur und Beispiel: <https://de.wikipedia.org/wiki/IPv6>

2.2 MAC-Adressen

MAC = Media Access Control

MAC-Adresse = Hardware-Adresse = Ethernet-Adresse

- `ip neigh`
`arp`

2.3 TCP- und UDP-Ports

- `nc <IP> <Port>`
Verbindung zu Programm <Port> auf Rechner <IP> aufnehmen
- `nc -l <Port>` oder `nc -p <Port> -l`
auf eingehende Verbindungen warten („lauschen“)
- TCP-Ports: Verbindungskonzept, Netzwerk prüft
- UDP-Ports: einzelne Pakete, Anwendung muß selbst prüfen
- ICMP: keine Ports, nur Rechner:
Erreichbarkeit, Eigenschaften der Übertragung

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse