Shell

LUKAS LEIPOLD

Shell

- Oder “Konsole” suchen im “Application Finder”
- Auch Terminal, Bash oder Kommandozeile genannt
- Bash nimmt Befehle entgegen und führt diese aus

Befehle I

cd	change directory	ssh	secure shell (remote)
mkdir	make directory	scp	secure copy
pwd	print working directory	less	Dateien anzeigen
rm	remove	cut	spaltenweises Extrahieren
mv	move	wc	word count
cp	copy	grep	Suche nach RegEx
ls	list	wget	herunterladen von
ln	link		FTP/HTTP-Servern
man	manual	chmod	Rechte ändern
cat	concatenate		
sort			
uniq	Entfernt doppelte Einträge		

Befehle II

```
tar             echo
gzip            head
diff            tail
tree            more
sync            find
vi/vim          date
ps              Unlink
kill            rmdir
```

https://wiki.ubuntuusers.de/Shell/Befehlsübersicht/

man <Befehl> zeigt Hilfe an und listet alle zusätzlichen Optionen

apropos <Suchwort> alle Manpages nach Suchwort durchsuchen

Pipelines – Pipes – |

- Aneinanderreihung von Befehlen
- Pipesymbol: |
- Syntax: `Befehl_1 | Befehl_2 | Befehl_3 > pipes.out`
- **Output** von `Befehl_1` dient als **Input** für `Befehl_2` …
- “>” – “Umleiten” des Outputs in eine beliebige Datei
  - Beispiel:
    ```
    find . –name *.txt > all_text_files.out
    ```
- “<“ – Dateiinhalt als Input für ein Programm verwenden

Beispiel – Pipes

- **cat:** Anzeigen/ Zusammenhängen von Dateien

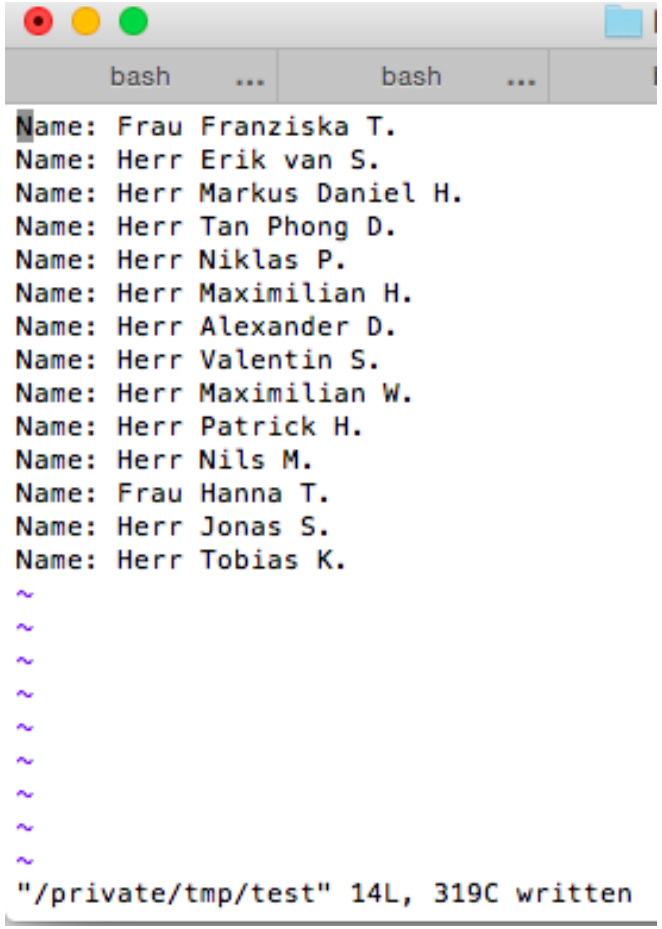
- **cut**: Spaltenweises Ausschneiden
  - **-d** Trennzeichen zwischen den Spalten (default tab)
  - **-f** Spaltennummer(n), die man selektieren will

- **sort:** Sortiert Zeilen einer Datei

- **uniq:** Anzeigen/ Filtern doppelter Einträge
  - **-c** Ausgabe der Anzahl doppelter Einträger

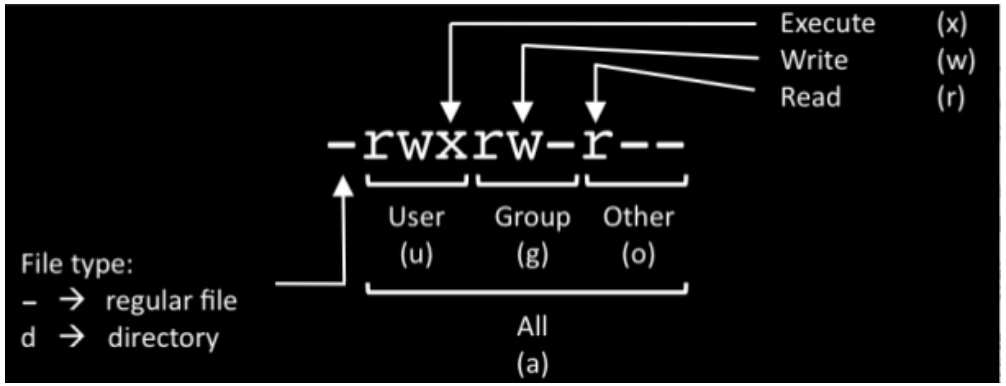
cat /private/tmp/test | cut -f2 -d " " | sort | uniq –c

Selektiert die 2. Spalte der Datei „test“ , sortiert und zählt wie viele Männer und Frauen in der Datei enthalten sind.



Dateirechte

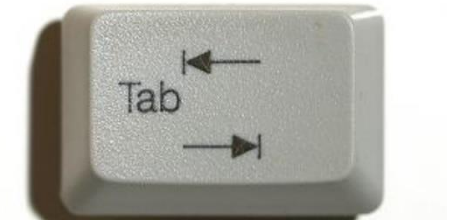
- Rechte setzen mit:
  `chmod <Berechtigung> <File>`
- <Berechtigung>: [ugoa][+-=][Art der Berechtigung]



- Ausführbare Shell Skripte:
  chmod 755 script.sh
  `./script.sh` oder `bash script.sh` führt Skript aus
- Binäre Rechte: r = 4, w = 2, x = 1 -> Aufsummieren für einzelne Benutzergruppen
  - `chmod 761 test.txt` oder `chmod u+rwx,g+rw,o+x text.txt`

Arbeiten mit der Shell – Tipps

- Autovervollständigung von Verzeichnissen/Dateien: **Tab Tab Tab**!!!
- Befehl: `history` -> listet die letzten eingegebenen Befehle
  - Pfeil hoch/ runter – Taste
  - STRG + R -> Listet Chronik
- Pos1 / Ende – Taste -> springen zum Anfang / Ende der aktuellen Zeile
- STRG + C -> Abbruch des aktuellen Befehls/ Programms
- “q” zum beenden von man-Pages
- STRG + Shift + T -> Öffnet weiteren Tab in aktueller Shell
- “clear” oder STRG + L -> räumt Shell auf
- Fortgeschritten:
  - Persönliche Einstellungen (z.B. Änderungen in $PATH) in `~/.zshrc_local` (selbst erstellen)



Dateiverzeichnisse - Pfade

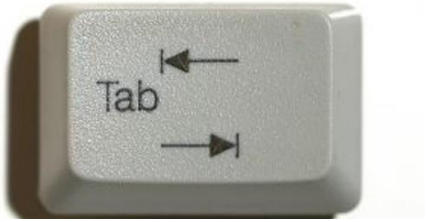
- Arbeitsverzeichnis
  - benutzername@rechnername:<arbeitsverzeichnis>
  - "~" bezeichnet das HOME-Verzeichnis
- Absolutes Verzeichnis
  - Geben Dateipfad vom Wurzelverzeichnis aus an
  - Beginnt immer mit "/"
  - Befehl: pwd

```
leipoldl@dachpilz:~/tutorium (0) [17:53:40] % pwd
/home/l/leipoldl/tutorium
leipoldl@dachpilz:~/tutorium (0) [17:54:23] % cd ../
leipoldl@dachpilz:~ (0) [17:54:39] %
```

- Relatives Verzeichnis
  - Beschreibt Verzeichnis ausgehend (relativ) vom **aktuellen** Arbeitsverzeichnis
  - ".." bezeichnet Verzeichnis eine Ordnerebene höher
  - "." bezeichnet aktuelles Arbeitsverzeichnis
- Versteckte Dateien beginnen immer mit "." (z.B. "`~/.zshrc`")

Arbeiten mit der Shell – Tipps

- Autovervollständigung von Verzeichnissen/Dateien: **Tab Tab Tab**!!!
- Pfeil hoch/ runter – Taste : die letzten Befehle
- Pos1 / Ende – Taste -> springen zum Anfang / Ende der aktuellen Zeile
- STRG + C -> Abbruch des aktuellen Befehls/ Programms
- “q” zum beenden von man-Pages
- STRG + Shift + T -> Öffnet weiteren Tab in aktueller Shell
- “clear” oder STRG + L -> räumt Shell auf
- Fortgeschritten:
  - Persönliche Einstellungen (z.B. Änderungen in $PATH) in `~/.zshrc_local` (selbst erstellen)



Bash Skripte

- 1. Zeile = Shebang Zeile:
  - `#!/bin/bash`
- # Kommentare
- Variablen:
  - `message=“hallo welt“`
    - Wichtig: Keine Leerzeichen zwischen Variable und „=“ Zeichen!
  - `echo “$message“`
    - Ausgabe der Variablen
- Befehle ausführen:
  - `$(cat test.txt)`
  - Abspeichern in einer Variablen durch:
    - `content=$(cat test.txt)`

Arrays und Rechnen

- array=(Dies sind 4 Werte)
- array[1]=“das“
- array+=(und diese Einträge werden hinzugefügt)
- Rechnen:
  - x=“62“ # also ein String
  - `echo $(($x+2)) # 64`
  - Potenz:
    `$((10**2)) #100`

If – else

```
1  if [ Test-Bedingung ]
2    then
3      Befehl...
4    elif [ Test-Bedingung ]
5      then
6        Befehl...
7    else
8      Befehl...
9  fi
```

```
1  #!/bin/bash
2  echo "Wie ist Ihr Name?"
3  read ANTWORT
4  if [ "$ANTWORT" == "root" ]
5      then
6          echo "Hallo, Administrator."
7      else
8          echo "Hallo, Anwender."
9  fi
```

Leerzeichen bei if und [Bedingung] beachten !!!

For Schleife

```
for variable in "Parameterliste"
  do
    Befehl1
    Befehl2
    usw.
done
```

```
for ((Anfangswert;Bedingung;Operation))
do
    Befehl1
    Befehl2
    usw.
done
```

```
#!/bin/bash
# for-schleife
for wort in "Hallo" "du" "Welt" "da" "draußen"
  do
    echo "$wort"
done
```

Parameter übergeben

- Parameter durch Whitespaces getrennt
  - script.sh par1 par2 par3
  - Parameter 1 – 9 befinden sich in den Variablen:
    $1 $2 $3 … $9

```
1  #!/bin/bash
2  # willkommen
3  if [ $1 == "Pingu" ]
4    then
5      echo "Hallo, kleiner Pingu!"
6    else
7      echo "Hallo, $1"
8  fi
```

```
./willkommen.sh Pingu
```

```
Hallo, kleiner Pingu!
```

```
./willkommen.sh Peter
```

```
Hallo, Peter
```

Funktionen

```
1  funktionsname(){
2  Befehle...
3  }
```

- z.B in der `~/.zshrc_local` Datei:

```
bioclient(){
      ssh  -p24 leipoldl@bioclient1.bio.ifi.lmu.de
}
```

  - Durch Aufruf von „bioclient“ in der Konsole wird der Code in dieser Funktion ausgeführt