

# Computer & GNU/Linux Einführung

## Teil 3

Simon Haller, Sebastian Stabinger, Benjamin Winder

Inst. für Informatik  
[IFI]

September 26, 2013

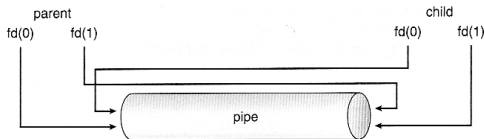
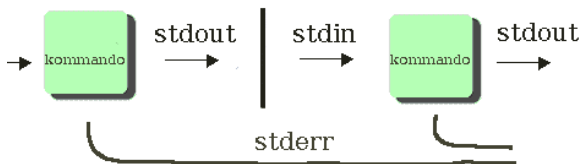


# Elementares

## Pipes (Weiterleitungen)

Die Shell kann die Ausgabe eines Kommandos als Eingabe für ein weiteres verwenden.

```
$ kommando1 < kmd1.in | kommando2 > kmd2.out
```



# Wichtige Kommandos

`grep` (*nach einer Weiterleitung*)

Nach einer Pipe-Weiterleitung ist das Kommando `grep` [SUCHBEGRIFF] sehr praktisch:

```
$ cat LangeDatei | grep [SUCHBEGRIFF]
```

`grep` sucht nach einem Suchmuster/begriff und gibt nur Zeilen in dem dieses(r) vorkommt aus.



# Wildcards

## mögliche Wildcards

- ▶ \* ... für beliebig viele oder kein Zeichen.
- ▶ ? ... für genau ein Zeichen.
- ▶  $[n - m]$  ... für genau ein Zeichen aus dem Bereich  $n$  bis  $m$ .
- ▶  $\{n, m\}$  ... für genau ein Zeichen  $n$  oder  $m$ .

$n$  und  $m$  können beliebige alphanumerische Zeichen sein.



# Wildcards

## Beispiel

```
mkdir test{1,2,3}
```

legt die Verzeichnisse

- ▶ test1
- ▶ test2
- ▶ test3

an.



# Sequentielle Kommandoausführung

## Möglichkeiten

```
$ kommando1 ↵
```

```
$ kommando2 ↵
```

```
$ kommando3 ↵
```

```
$ kommando1; kommando2; kommando3 ↵
```



# Kommandogruppierung

## Möglichkeiten

```
$ ( kommando1; kommando2 )
```

```
$ ( kommando1; kommando2 ) > out.file 2> err.file
```



# Bedingte Ausführungen

## Möglichkeiten

`$ kommando1 && kommando2`

Kommando1 wird ausgeführt, falls dieses normal beendet wird,  
wird Kommando2 ausgeführt

`$ kommando1 || kommando2`

Kommando1 wird ausgeführt, falls dieses fehlschlägt wird  
Kommando2 ausgeführt





# Bedingte Ausführungen

## Aufgabe

Eine Datei in ein Verzeichnis kopieren und bei Erfolg die Originaldatei löschen.



# Bedingte Ausführungen

## Aufgabe

Eine Datei in ein Verzeichnis kopieren und bei Erfolg die Originaldatei löschen.

## Lösung

```
$ cp -rvp Datei Irgendwohin && rm Datei
```

Wenn der Kopiervorgang erfolgreich war, dann wird die Originaldatei gelöscht.



# Kommandos in Kommandos

STDOUT eines Kommandos als Argument für ein anderes Kommando

Idee: die Ausgabe/Ergebnis von Kommando2 (z.B. eine Liste an Dateien) wird an ein Kommando1 als Argumente übergeben.

Varianten:

`cmd1 `cmd2 Argumente``      `cmd2` steht zwischen 'backticks' (*sh*)

`cmd1 $(cmd2 Argumente)`      (*ksh, bash*)



# Kommandos in Kommandos

## Beispiel

```
cp `find . -name '*txt' ` tmp/
```

Suche in alle Verzeichnissen ausgehend von . alle Dateien die mit txt Enden und kopiere sie nach tmp/.



# SSH (Secure Shell)

zid-gpl.uibk.ac.at

```
ssh [OPTIONEN] USERNAME@HOST [Kommando]
```

z.B.:

```
ssh user@zid-gpl.uibk.ac.at
```

-Y ... Sicheres X-Forwarding

-l ... Login-Name



# SSH-Keys

## SSH Schlüssel erzeugen

```
ssh-keygen -t rsa -b 2048
```

## SSH Public Key verwenden (Beispiel)

```
scp ~/.ssh/id_rsa.pub zid-gpl.uibk.ac.at:~/  
ssh zid-gpl.uibk.ac.at  
cat id_rsa.pub >> .ssh/authorized_keys && rm id_rsa.pub  
exit  
ssh -i .ssh/id_rsa zid-gpl.uibk.ac.at
```



# SSH-Keys (II)

## SSH-Config

Erstelle das File:

```
$HOME/.ssh/config
```

```
Host zid-gpl
```

```
    HostName zid-gpl.uibk.ac.at
```

```
    Port 22
```

```
    User username
```

```
    IdentityFile ~/.ssh/PRIVATEKEYFILE
```



# Weitere Befehle

`touch [DATEI]`

leere Dateien erstellen oder das Änderungsdatum einer vorhandenen Datei auf “jetzt” setzen.

## Aufgabe

Erstelle eine Datei, die vor drei Jahren erstellt wurde.





# Weitere Befehle

## touch [DATEI]

leere Dateien erstellen oder das Änderungsdatum einer vorhandenen Datei auf “jetzt” setzen.

## Aufgabe

Erstelle eine Datei, die vor drei Jahren erstellt wurde.

## Lösung

```
touch -t 201001171100 test.file
```

Mit der Option `-t` kann anstatt der aktuellen Zeit eine Andere vorgegeben werden.



# Wichtige Kommandos

`find [PFAD] [OPTIONEN] [SUCHAUSDRUCK]`

Finden von Dateien.

Die wichtigsten Suchkriterien:

- `-atime` Wann wurde zuletzt auf die Datei zugegriffen
- `-mtime` Wann wurde die Datei zuletzt geändert
- `-newer` [DATEI] selbsterklärend
- `-user` [USER] gehört bestimmten user
- `-name` [NAME] suche nach name.
- `-size` [*n*] wobei *n* in 512 Byte-Blöcken angegeben wird.



# Wichtige Kommandos

`find [PFAD] [OPTIONEN] [SUCHAUSDRUCK]`

Finden von Dateien.

Die wichtigsten Suchkriterien:

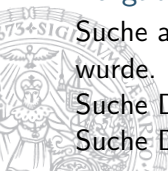
- `-atime` Wann wurde zuletzt auf die Datei zugegriffen
- `-mtime` Wann wurde die Datei zuletzt geändert
- `-newer [DATEI]` selbsterklärend
- `-user [USER]` gehört bestimmten user
- `-name [NAME]` suche nach name.
- `-size [n]` wobei  $n$  in 512 Byte-Blöcken angegeben wird.

## Aufgaben

Suche alle Dateien in deinem HOME auf die Heute zugegriffen wurde.

Suche Dateien die vor über einer Woche modifiziert wurden.

Suche Dateien die grösser als 1M sind.



# Wichtige Kommandos

## Lösungen

- `-atime -1` Auf die Datei wurde Heute zugegriffen
- `-mtime +7` Letzte Änderung der Datei ist älter als eine Woche
- `-size +2000` Datei ist grösser als 1M.



# Wichtige Kommandos

`tar [OPTIONEN] [DATEI]` (*tape archiver*)

Archivieren und Komprimieren von Dateien

- `c` create (erstellen)
- `f` file (Datei / Archiv verwenden)
- `r` append (Dateien an das Archiv anhängen)
- `t` list (Dateien aus dem Archiv auflisten)
- `x` extract (auspacken)
- `z` zip (zip-Komprimierung)

## Beispiele

```
tar czf Mein_Archiv.tar.gz ZuArchivierendeDateien
tar tvf Mein_Archiv.tar.gz
tar xzf Mein_Archiv.tar.gz
```



# Kommandos

`date [Optionen] (datum)`

`date` gibt das aktuelle Datum aus.

Optionen:

- `%a` abk. des Wochentags
- `%b` abk. des Monatnamens
- `%H` Stunde (0..23)
- `%M` Minute (0..60)
- ...

Beispiel:

```
date +%T
```



## Aufgabe

Erstelle ein Backup mit Dateien die Heute im HOME-Verzeichniss (und den Unterverzeichnissen) geändert wurden.

Es sollen alle Dateien die mit Nicht beginnen exkludiert werden.

Der Dateiname des Backups soll die aktuelle Zeit beinhalten (z.B.: *backup-Thu.26.09.2013-12.25.16.tar.gz*)

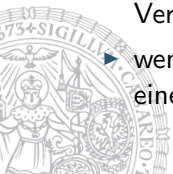


## Lösung

```
tar czf backup-$(date +%a.%d.%m.%Y-%H.%M.%S).tar.gz \  
$(find . -mtime -1 ! -name 'Nicht*' ! -name '.' ! -name '.BACKUP')
```

## Erklärung

- ▶ `$( cmd )` ... die Auswertung des `cmd` aus der Subshell wird an das 'übergeordnete' Kommando weitergegeben
- ▶ `date +%a.%d.%m.%Y-%H.%M.%S` ... gibt das Datum im gewünschten Format aus
- ▶ im `find` Kommando werden zusätzlich noch das aktuelle Verzeichnis und das `.BACKUP` Verzeichnis ausgenommen
- ▶ wenn `find` keine Dateien findet, gibt das Kommando `tar` einen Fehler aus





# Prozess

## Was ist ein Prozess

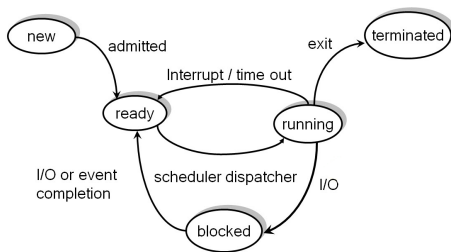
- ▶ ein exekutierendes Programm
- ▶ Ressourcen: Speicher, Files, CPU-Zeit
- ▶ Betriebssystem ist verantwortlich für:
  - ▶ Erzeugen und Terminieren von Prozessen
  - ▶ Scheduling von Prozessen
  - ▶ Synchronisierung von Prozessen
  - ▶ Kommunikation zwischen Prozessen
  - ▶ Behandlung von Deadlocks



# Prozess

## aus Sicht des Betriebssystems

- ▶ PCB (Prozess-Control-Block)
  - ▶ Prozess-ID (PID)
  - ▶ Adressraum für den Prozess
  - ▶ Priorität
  - ▶ andere Informationen wie: wann ist der Prozess zuletzt gelaufen, wieviel Rechnerzeit hat er verbraucht, ...
- ▶ Scheduler



# Prozess

## Steuerung von Prozessen / Jobkontrolle

fg (ForeGround)

bg (BackGround, zuerst mit *strg* + *z* job stoppen)

ps (Prozessstatus) & pstree

jobs (laufende Jobs anzeigen)

kill (prozess mit prozessid beenden)

nohup (no hangup)

top (oder htop)



## Speziellere Shellvariablen

- \$? Exit-Status des zuletzt beendeten Kommandos (Vordergrund)
- \$\$ Prozeß-ID der aktuell ausgeführten Shell
- #! Prozeß-ID des zuletzt gestarteten Hintergrundprozesses



## Aufgabe

- ▶ Finde den aktuellen status deiner offenen shell
- ▶ Was für CPU und Speicherinformationen finden sich in `/proc`
- ▶ Was macht der Befehl `grep -ri 'xxx' /etc/s* 2> /dev/null | less`
- ▶ Was ist der Unterschied zu `grep -ri 'xxx' /etc/s* 2> &1 | less`



# Kommandos

`dirname [Datei]`

Verzeichnisnamen von Datei ausgeben.

```
dirname [Datei]
```

Beispiel:

```
dirname /VER/ZEICHNIS/DATEINAME
```

liefert

```
/VER/ZEICHNIS
```



# Kommandos

`basename [Datei]`

Verzeichnis vom Dateinamen entfernen

```
basename [Datei]
```

Beispiel:

```
basename /VER/ZEICHNIS/DATEINAME
```

liefert

```
DATEINAME
```



# References I

Haller, S. (2010). Lpccs introduction course.

Silberschatz, A., P. Galvin, and G. Gagne (2010). Operating system concepts (8 ed.). Wiley.

