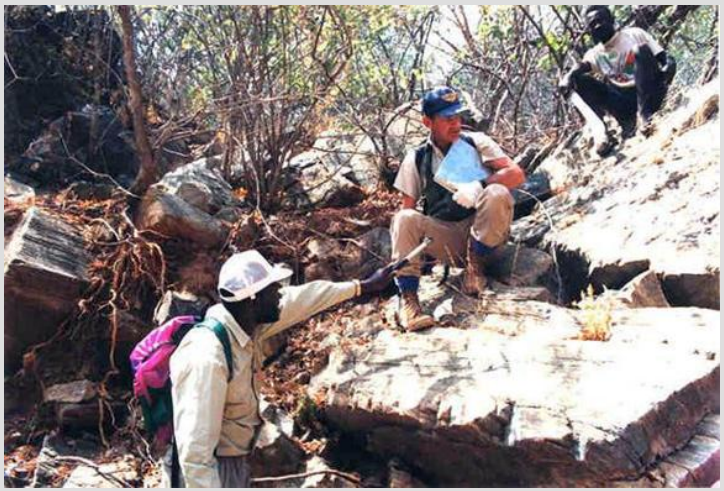
Datenverarbeitung in der Geophysik

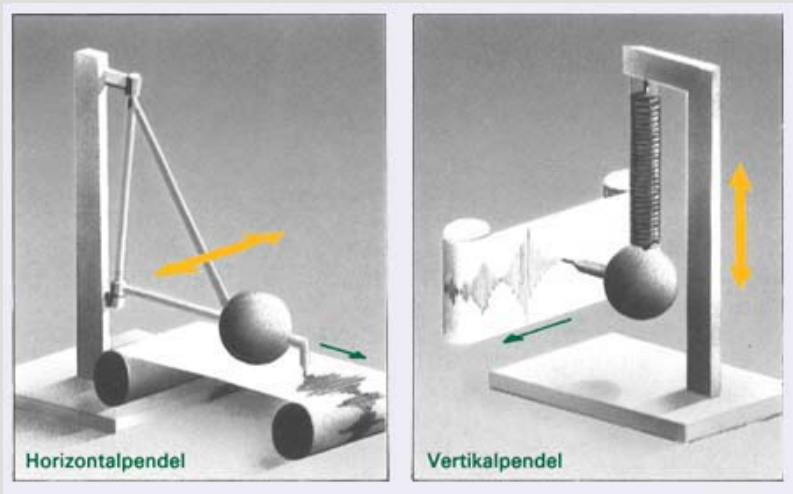
Jens Oeser

Geophysik
Department für Geo- und Umweltwissenschaften
Ludwig-Maximilians-Universität München

22. Dezember 2016

Datenverarbeitung in der Geophysik – FRÜHER





Datenverarbeitung in der Geophysik – HEUTE

- „Digitaltechnik“
- digitale Messwertenregistrierung (ADU)
- Datenauswertung/-darstellung am PC
- Datengewinnung immer häufiger durch Modellrechnungen auf HPC-Rechnern
- Datenvolumina wächst beständig

Mit was werden wir uns beschäftigen?

- Datenverarbeitung in den Geowissenschaften ist eng verknüpft mit Entwicklung der Rechentechnik
- Unix ist in den 80er Jahren Standardbetriebssystem
- Software häufig nur für Unix verfügbar
- Softwareentwicklung im wissenschaftlichen Umfeld auf Unix

$\Longrightarrow$

- GNU/Linux und Shell-Programmierung
- Generic Mapping Tools (GMT)
- Python

Mit was werden wir uns beschäftigen?

- Datenverarbeitung in den Geowissenschaften ist eng verknüpft mit Entwicklung der Rechentechnik
- Unix ist in den 80er Jahren Standardbetriebssystem
- Software häufig nur für Unix verfügbar
- Softwareentwicklung im wissenschaftlichen Umfeld auf Unix

- GNU/Linux und Shell-Programmierung
- Generic Mapping Tools (GMT)
- Python

Unix und GNU/Linux

- Unix (Plural: Unices) ist der Oberbegriff für alle Betriebssysteme, die auf dem ursprünglichen Unix-Entwurf (AT&T) basieren
- Unix-Einfluss erstreckt sich auf nahezu alle heute verbreiteten Betriebssysteme
- Ideen und Konzepte von Unix finden sich überall wieder
- GNU/Linux oder die freien BSD-Systeme implementieren das Verhalten und die Schnittstellen von Unix neu und steuern eigene Erweiterungen bei
- wenn das Betriebssystem gemeint ist, muss man korrekterweise von „GNU/Linux“ sprechen, da immer der Kernel (Linux) und die Systemwerkzeuge (GNU) zusammen gemeint sind

Geschichte

- 1965 – 1. Konzept für ein Betriebssystem „Multics“
  - Hardwareanforderungen für diese Zeit zu hoch
  - Ritchie, Thompson, Mc Ilroy und Ossanna wollten nicht aufgeben
  - brauchten ein Mehrbenutzersystem
- 1969 – 1. Version von Unix in Assemblersprache durch Thompson erstellt (DEC PDP-7)
  - interessantes Projekt für die Bell Labs
  - Gründung der AT&T Unix Systems Group (später: Unix Systems Group)
- 1972-74 – Reimplementierung von Unix in C
  - Implementierung des Konzepts der Pipes
  - kostenfreier Vertrieb mit C-Compiler an Universitäten (BSD-Linie entwickelt sich)
  - AT&T versucht Unix gewinnbringend zu vermarkten (System V Linie von Unix ensteht)

- 1979 – Microsoft erwirbt Lizenz für Unix
  - Entwicklung von Xenix
  - Interesse durch DOS verloren
  - Rechteabgabe an SCO
- 80er – Unix wird dominierendes Betriebssystem an Universitäten
  - BSD - Berkeley System Distribution
  - Fülle an verschiedenen Derivaten
  - „Zeit der Unix-Kriege“
  - Notwendigkeit einer Standardisierung
- 1983 – Richard Stallman ist über Kommerzialisierung von Unix verärgert
  - Beginn der Arbeiten an einem eigenen, Unix ähnlichen Betriebssystem namens GNU (GNU's not Unix!)
- 1985 – Veröffentlichung des GNU Manifest durch Stallman
- freie Softwarebewegung wird immer stärker

- 1987 – Professor Andrew S. Tanenbaum
  - Entwicklung des Unix ähnlichen Betriebssystems „Minix“
  - vorwiegend für seine Lehre
- 1988 – Veröffentlichung des POSIX.1-Standard
  - heute IEEE-Standard mit der Nummer 1003.1
- 1991 – 21-jähriger finnischer Student Linus Benedict Torvalds
  - Beginn der Entwicklung der Unix artigen Betriebssystembasis für AT-386 Computer „Linux“

5. Oktober 1991 Newsgroup comp.os.minix: „As I mentioned a month ago, I'm working on a free version of a Minix-look-alike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be, depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02... but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it.“

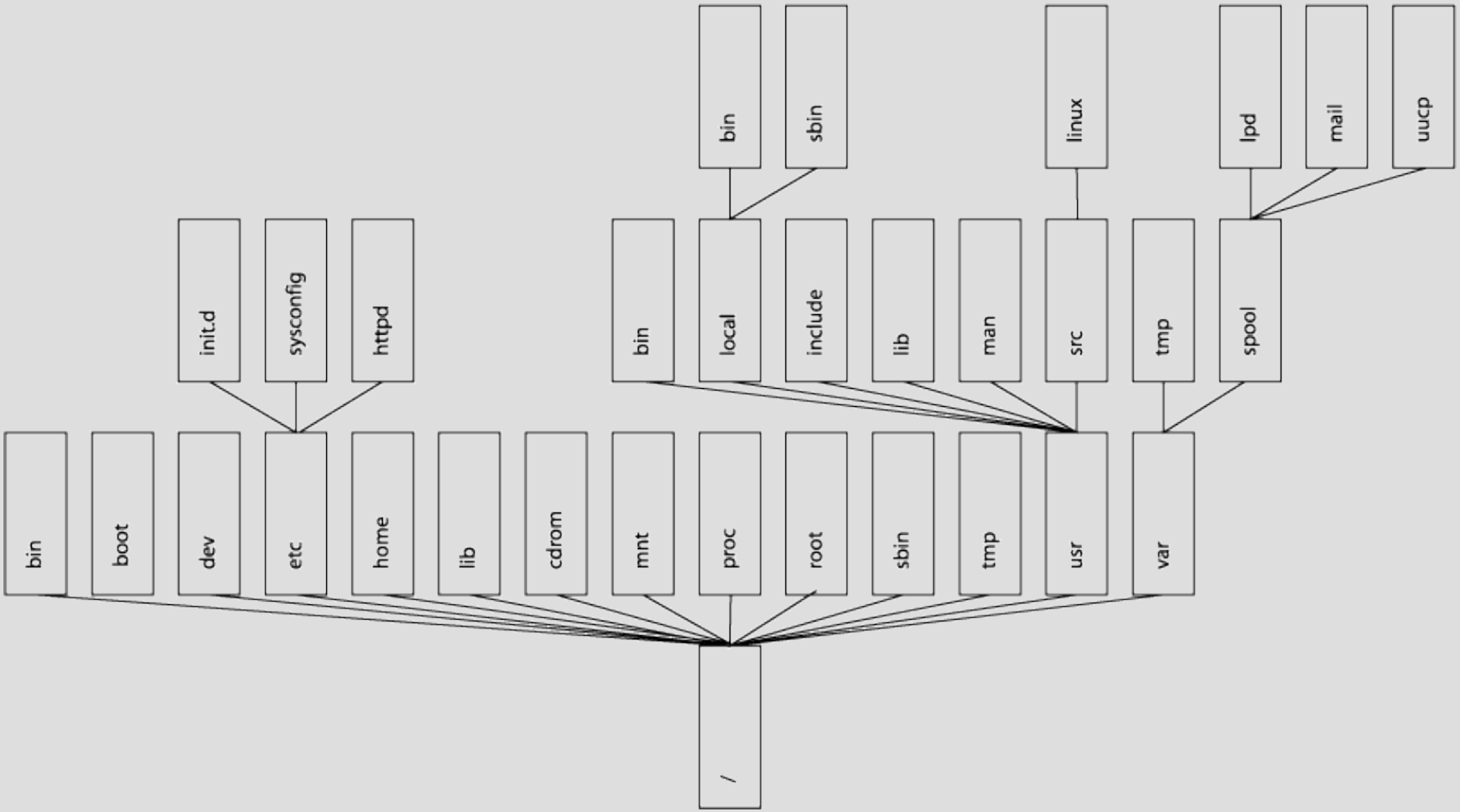
- 1993 – mehr als 100 Programmierer arbeiten am Linux Kernel mit
  - schnelle Entwicklung hält weiterhin an
  - aktueller Linux Kernel 4.8 (04.10.2016)
- GNU-Projekt provitiert erheblich vom Linux Kernel
- damit ist erstmals ein komplett freies Betriebssystem erhältlich
- Fülle an freien Softwareprojekten existiert heute
  - LibreOffice
  - Apache
  - KDE
  - GNOME
  - GIMP
  - ...

Literatur

- Linux – Das umfassende Handbuch (J. Plötner und S. Wendzel)
- Linux/Unix – Grundlagenreferenz (Helmut Herold)
- Advanced Bash-Scripting Guide
- LRZ – Hochschulschriften
- LRZ – Kurse GNU/Linux
- Suchmaschinen
- Linux – die Reise des Pinguins (3sat – Dokumentation zu Linux)

Merkmale von GNU/Linux

- GNU/Linux besteht aus einem Kernel
  - dieser allein hat die Kontrolle über Geräte und Prozesse
  - Kernel stellt Systemaufrufe als Schnittstelle für Programme zur Verfügung
  - Vielzahl von Programmen vervollständigen das Betriebssystem
- Dateisystem ist als hierarchisches Verzeichnis mit beliebigen Unterverzeichnissen organisiert
  - umgekehrt baumartig
  - „/“ ist Wurzelverzeichnis (Root-Verzeichnis)


/
bin
boot
dev
etc
home
lib
cdrom
mnt
proc
root
sbin
tmp
usr
var
init.d
sysconfig
httpd
bin
local
include
lib
man
src
tmp
spool
bin
sbin
linux
lpd
mail
uucp

- ein Grundprinzip von GNU/Linux und Unix
  - „Alles ist eine Datei!“
  - dieser verallgemeinerte Dateibegriff gehört zum Wesen
  - einfache und einheitliche Schnittstelle für die verschiedenen Anwendungen
  - unerreicht einfache Ein-/Ausgabeumleitung in diese Dateien
  - Verkettung von mehreren Programmen über Pipes
- alle Unix artigen Betriebssysteme enthalten eine grafische Benutzeroberfläche (X11)
- Unix ist historisch eng mit der Programmiersprache C verbunden

- zu wichtigsten Merkmalen von GNU/Linux und Unix gehören
  - hohe Stabilität
  - Multiuser
  - Multitasking
  - Multithreading
  - Speicherschutz
  - virtueller Speicher
  - TCP/IP Netzwerkunterstützung
  - voll ausgebaute Shell
  - Vielzahl von Programmen für fast alle Aufgabenstellungen
  - hervorragende Scripting-Eigenschaften

Benutzung eines GNU/Linux Systems – Zugang

- Benutzerkonto besteht in der EDV häufig aus einem Benutzernamen mit Benutzerpasswort und zusätzlichen Informationen
- Authentifizierung (Benutzerüberprüfung) erfolgt durch Eingabe von Benutzernamen und -passwort
- Authorisierung (Rechtezuweisung) erfolgt danach anhand der zusätzlichen Informationen
- An- und Abmeldung ist über eine grafischen Benutzeroberfläche (Windowmanager) oder in der Textkonsole möglich

Benutzung eines GNU/Linux Systems – Benutzerklassen

- Benutzereinteilung in Klassen ermöglicht genaue Rechtezuweisung
- für jeden Benutzer ist die Unix Welt in 3 Klassen unterteilt
  - u = login **u**ser
  - g = **g**roup
  - o = **o**thers
- alle Benutzer werden über eine **UID** (**u**ser **id**entification number) identifiziert, die eindeutig vom Benutzernamen abhängt
- Benutzer verfügen noch über eine oder mehrere **GID** (**g**roup **id**entification number), mit dieser wird Zugehörigkeit zur Gruppe festgelegt (Rechteteilung mit anderen Mitgliedern der Gruppe)
- alle Benutzer die nicht in dieser Gruppe sind, gelten als der Rest der Unix Welt (**o**thers)
- für u, g und o können Schreib-, Lese- und Ausführungsrechte gesetzt werden, die als Dateiattribute gespeichert werden

Benutzung eines GNU/Linux Systems – Kommandosyntax

- abgesehen von wenigen Ausnahmen hat ein Unix-Kommando den Syntax

```
# KOMMANDO -OPTIONEN PARAMETER
```

- Einführung in die wichtigsten Befehle ist Gegenstand des nächsten Kapitels
- Optionen beeinflussen die Ausführung eines Befehls
  - Kurzform für Optionen `-a`
  - Langform für Optionen `--all`
- Parametern sind meist Dateinamen

Benutzung eines GNU/Linux Systems – Dokumentation

- jedes System verfügt über eine eingebaute Dokumentation/Hilfe

```
# man KOMMANDO
```

- jede Manualseite ist unterteilt in einzelne Abschnitte
  - NAME – Name des Kommandos
  - SYNOPSIS – Syntaxbeschreibung
  - DESCRIPTION – ausführliche Beschreibung des Kommandos
  - OPTIONS – Liste aller Optionen mit einer kurzen Beschreibung
  - COMMANDS – Anweisungen für interaktive Programmen
  - FILES – Dateien, die mit dem Kommando zusammenhängen
  - SEE ALSO – Hinweise auf verwandte Kommandos und Manualeinträge
  - DIAGNOSTICS – Liste der Fehlermeldungen
  - EXAMPLE – Beispiele zum Aufruf des Kommandos
  - BUGS – bekannte Fehler und Schwierigkeiten
- die Befehle `info` und `apropos` liefern weitere Informationen

Benutzung eines GNU/Linux Systems – Systemnutzung

- in heutiger „Fensterwelt“ wirkt Befehlseingabe im Terminal kompliziert
  - alle Benutzeraufgaben können auch ohne Kommandoeingabe gelöst werden
  - mit dem erhöhtem Komfort ist auch ein verminderter Fahrspaß verbunden
- Unix-Systeme sind wie Baukästen
  - eine Menge an Programme steht zur Verfügung
  - diese sind für verschiedenste Aufgaben miteinander kombinierbar
- es ist nicht immer notwendig die aufgeblähte Programme zu benutzen, wenn die kleinen, spezialisierten Programme geschickt kombiniert werden können

- Terminal (Shell) nimmt die Befehle
  - entgegen
  - verarbeitet diese
  - startet Programm
  - gibt Ergebnis aus
- Kommandoeingabe erfolgt entsprechend des erwähnten Syntax
  - Form eines Befehls ist nicht nur durch Syntax des aufgerufenen Kommandos, sondern auch durch Syntax der im Terminal benutzten Shell bestimmt
  - vor Ausführung erfolgt das Parsens (Reihe von Veränderungen an der Eingabe)
  - diese Änderungen haben nichts mit dem aufgerufenen Kommando zu tun, sondern werden von der Shell nach den immer gleichen syntaktischen Regeln vorgenommen (zahlreiche Vereinfachungen bei der Kommandoeingabe möglich)

Benutzung eines GNU/Linux Systems – Dateien

- aus Benutzersicht ist eine Datei immer eine Folge logisch zusammenhängender Informationen
- aus Systemsicht aber nur eine endliche Folge von Bytes
- unter Unix gibt es die folgenden wesentlichen Dateiarten
  - normale Datei ≡ Datei, die Daten, Text oder Programm enthält
  - Dateiverzeichnis ≡ Datei, die Verweise auf weitere Dateien enthält
  - Gerätedatei ≡ Datei, die Verweis auf physikalisches Gerät enthält
  - Pipe ≡ Datei, die der systeminternen Prozesskommunikation dient
- jeder Benutzer hat genau ein „HOME Directory“

```
# echo $HOME
```

- jeder Benutzer hat während der Sitzung ein „Working Directory“

```
# echo $PWD
```

- Zugriff auf Datei erfolgt immer über den Dateinamen
- Dateinamen ist frei wählbar
  - allerdings setzen einige Programme Namenskonventionen voraus
- Datei wird über den absoluten oder relativen Pfad angesprochen
- die selbe Datei kann im Verzeichnisbaum auf verschiedenen Ästen liegen und über verschiedene Namen verfügen
- jede Datei hat eine Reihe von Attributen
  - Dateiname
  - Dateipfad
  - Dateityp
  - UID und GID des Besitzers
  - Zugriffsrechte
  - Dateilänge
  - Knotennummer
  - Anzahl der Links, Verweise auf diese Datei
  - Datum der Erstellung, letzten Änderung und des letzten Zugriffs

- Benutzer kann die Attribute ändern
- Bearbeitung einer Gruppen von Dateien durch sog. Jokerzeichen möglich
  - `?` genau ein beliebiges Zeichen
  - `*` beliebig viele Zeichen
  - `[abc]` genau eines der angegebenen Zeichen
  - `[a-d]` ein Zeichen aus dem angegebenen Bereich
  - `[!abc]` keines des angegebenen Zeichen

Benutzung eines GNU/Linux Systems – Shell

Was ist eine Shell?

- ein Programm, mit dessen Hilfe das System die Benutzerbefehle verstehen kann
- daher oft als Befehls- oder Kommandointerpreter bezeichnet (Shell interpretiert das Kommando)
- eine erfolgreiche Anmeldung startet immer die eigene Login-Shell
- gerade aktive Shell:

```
gkd12ca@cip50:~$ echo $SHELL
```

Welche Shells gibt es?

- einfache Shells:
  - Bourne-Shell (`sh`)
  - Korn-Shell (`ksh`)
  - C-Shell (`csh`)
- erweiterte Shells:
  - Bourne-Again-Shell (`bash`)
  - T-C-Shell(`tcsh`)
  - Z-Shell (`zsh`)

Wie kann ich zwischen den verschiedenen Shells wechseln?

```
gkd12ca@cip50:~$ echo $SHELL
/bin/bash
gkd12ca@cip50:~$ tcsh
cip50:~>
cip50:~> exit
gkd12ca@cip50:~$ ksh
$
$ exit
```

Was ist der Prompt?

- die Aufforderung zur Eingabe eines Kommandos
- je nach Shell kann sich das Aussehen des Prompts unterscheiden
- Ändern des Prompts durch die Variable `PS1`

```
gkd12ca@cip50:~$ ls
Desktop  maildir
gkd12ca@cip50:~$ echo $PS1
${debian_chroot:+($debian_chroot)}\u@\h:\w\$
gkd12ca@cip50:~$ PS1="abc123# "
abc123# ls
Desktop  maildir
abc123#
```

Was ist eine Variable?

- im allgemeinsten Sinne einfach ein Behälter für Rechengrößen (Werte)
- eine Variable
  - wird durch einen Namen bezeichnet
  - hat bestimmte Adresse im Speicher
  - Wert kann sich ändern
- in den Shells gibt es nur einen Typ von Variablen
- Variablen können beliebig angelegt und geändert werden
- Anmeldung initialisierte Umgebungsvariablen
  - beeinflussen das Verhalten der Shell
  - beeinflussen die Befehle/Kommandos

Welche Umgebungsvariablen sind initialisiert/gesetzt?

```
gkd12ca@cip50:~$ env
TERM=xterm
SHELL=/bin/bash
USER=gkd12ca
LOGNAME=gkd12ca
HOME=/home/cip/gkd12ca
PWD=/home/cip/gkd12ca
LANG=en_GB.UTF-8
MAIL=/var/mail/gkd12ca
PATH=/usr/local/bin:/usr/bin:/bin:/usr/games
DISPLAY=localhost:10.0
```

Welche Programmtypen gibt es?

- wie sie wissen, werden aus der Shell die Programme gestartet
- es gibt zwei unterschiedliche Gruppen von Programmen
  - auf der Festplatte tatsächlich abgelegte Programme
  - in die Shell „reinprogrammierte“ (shellintern)
- möchte man Einzelheiten zum Typ erfahren:

```
gkd12ca@cip50:~$ type mkdir
mkdir is /bin/mkdir
gkd12ca@cip50:~$ type echo
echo is a shell builtin
gkd12ca@cip50:~$ type ls
ls is aliased to ‘ls --color=auto’
```

- möchte man zu einem Kommando auch den Pfad erfahren:

```
gkd12ca@cip50:~$ which echo
/bin/echo
gkd12ca@cip50:~$ which gnuplot
/usr/bin/gnuplot
```

- gibt man nur `echo` ein $\rightarrow$ shellinterne Variante
- gibt man `/bin/echo` ein $\rightarrow$ Programm aus entsprechendem Pfad
- je nach genutzter Variante kann sich Funktionalität leicht unterscheiden

Wie werden Kommandos aneinander gereiht?

Stellen Sie sich vor, Sie möchten fünf verschiedene Befehle hintereinander starten, die aber jeweils eine unbestimmt lange Zeit benötigen werden. Eventuell müssen Sie so stundenlang vor dem Computer sitzen und warten, bis ein Befehl nach dem anderen durchgelaufen ist, um den jeweils folgenden zu starten.

- die Shell kann mehr als nur ein Kommando pro Befehl ausführen
- ein Befehl endet erst durch die Eingabetaste
- ein Kommando wird von weiteren getrennt durch das Semikolon

```
gkd12ca@cip50:~$ echo $HOME; ls Desktop; date -R
/home/cip/gkd12ca
Datenverarbeitung.pdf  Home.desktop  System.desktop
Sun, 01 Nov 2009 16:00:42 +0100
```

- weiter nur bei Erfolg

```
gkd12ca@cip50:~$ dtae -R && ls $HOME
-bash: dtae: command not found
```

- weiter nur bei Fehlschlag

```
gkd12ca@cip50:~$ dtae -R || echo "Hilfe ein Fehler."
-bash: dtae: command not found
Hilfe ein Fehler.
```

Multi-Line-Kommandos

- oft erstrecken sich Kommandoeingaben über mehrere Zeilen
- durch Backslash-Operator (\) kann aktuelle Zeile in nächster Zeile fortgesetzt werden

```
gkd12ca@cip50:~$ find /usr/share/WindowMaker/ \
>  -type f -name BlueImage.jpeg
/usr/share/WindowMaker/Backgrounds/BlueImage.jpeg
gkd12ca@cip50:~$ find /usr/share/WindowMaker/ \
> -type f \
> -name \
> BlueImage.jpeg
/usr/share/WindowMaker/Backgrounds/BlueImage.jpeg
```

Arbeiten mit Verzeichnissen – Pfade

- ein Pfad gibt einen Weg durch den hierarchischen Verzeichnisbaum hin zu einem bestimmten Ziel an
- ein vollständiges Verzeichnis wie `/home/cip/gkd12ca` beschreibt also auch einen Pfad, der angibt, wie man zu eben diesem Verzeichnis gelangt
- unter GNU/Linux kann man Pfade auf zwei unterschiedliche Arten angeben
  - absoluter Pfad
    - Pfad vom Wurzelverzeichnis ausgehend
    - beginnt immer mit einem Slash `/`
  - relativer Pfad
    - Pfad vom aktuellen Verzeichnis ausgehend
    - beginnt mit Unterverzeichnis oder `../`

- aktuelles Verzeichnis

```
gkd12ca@cip50:~$ echo $PWD
/home/cip/gkd12ca
gkd12ca@cip50:~$ pwd
/home/cip/gkd12ca
```

- Verzeichniswechsel

```
cd [PFAD]
```

- `[PFAD]`
  - wird durch einen absoluten oder relativen Pfad ersetzt

Die folgenden Zeichen besitzen in der Shell eine Sonderbedeutung:

- `.`
  - das aktuelle Verzeichnis
- `..`
  - das nächst höhere Verzeichnis
- `-`
  - das letzte Verzeichnis
- `~`
  - das eigene Heimatverzeichnis
- `$HOME`
  - das eigene Heimatverzeichnis

```
gkd12ca@cip50:~$ cd ..
gkd12ca@cip50:/home/cip$ cd ..
gkd12ca@cip50:/home$ pwd
/home
gkd12ca@cip50:/home$ cd ../usr/bin
gkd12ca@cip50:/usr/bin$ pwd
/usr/bin
gkd12ca@cip50:/usr/bin$ cd ~
gkd12ca@cip50:~$ cd /usr/bin
gkd12ca@cip50:/usr/bin$ pwd
/usr/bin
gkd12ca@cip50:/usr/bin$ cd $HOME/Desktop
gkd12ca@cip50:~/Desktop$ pwd
/home/cip/gkd12ca/Desktop
gkd12ca@cip50:~$
```

- Betrachten von Verzeichnissen

```
ls [OPTIONEN] [PFAD]
```

- `[PFAD]`
  - wird durch einen absoluten oder relativen Pfad ersetzt
- `[OPTIONEN]`
  - -a
    - kompletten Inhalt anzeigen
  - -l
    - Informationen im Langformat
  - -r
    - Reihenfolge der Ausgabe umkehren
  - -t
    - zuletzt modifizierte Datei zuerst

```
gkd12ca@cip50:~$ ls Desktop
Datenverarbeitung.pdf  Home.desktop
gkd12ca@cip50:~$ ls -a
.                .Xauthority    .bash_logout   .dmrc
..               .kderc         .mcop          Desktop
gkd12ca@cip50:~$ ls -l
drwx------ 2 gkd12ca gkd12 4096 Oct 26 08:37 Desktop
drwx------ 2 gkd12ca gkd12 4096 Oct 19 11:50 maildir
gkd12ca@cip50:~$ ls -al
drwxr-xr-x   15 gkd12ca gkd12  4096 Nov  2 08:04 .
drwxr-xr-x 1759 root    root  32768 Oct 29 16:26 ..
-rw-------    1 gkd12ca gkd12    51 Nov  2 08:04 .Xauthority
drwx------    3 gkd12ca gkd12  4096 Oct 26 08:36 .adobe
```

```
gkd12ca@cip50:~$ ls /
adm boot  chkpnt dev  etc    lib   mnt  opt  sbin sys usr
bin cdrom data   home initrd media proc root srv  tmp var
gkd12ca@cip50:~$ ls /usr/
X11R6 bin games include lib lib64 local man sbin share src
gkd12ca@cip50:~$ ls /usr/share/texmf-texlive/
bibtex doc ls-R makeindex tex
gkd12ca@cip50:~$ ls -l /usr/share/texmf-texlive/
drwxr-xr-x 4 root root 4096 Feb 25 2008 bibtex
lrwxrwxrwx 1 root root   18 Feb 25  2008 doc -> ../doc/texlive-do
lrwxrwxrwx 1 root root   27 Feb 25  2008 ls-R -> /var/lib/texmf/l
drwxr-xr-x 4 root root 4096 Feb 25 2008 makeindex
drwxr-xr-x 4 root root 4096 Feb 25 2008 tex
```

```
gkd12ca@cip50:~$ touch DateiB && sleep 20 && touch DateiC \
 && sleep 20 && touch DateiA && sleep 20 && touch DateiD
gkd12ca@cip50:~$  ls -l Datei?
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:14 DateiA
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:13 DateiB
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:13 DateiC
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:14 DateiD
gkd12ca@cip50:~$ ls -lt Datei?
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:14 DateiD
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:14 DateiA
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:13 DateiC
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:13 DateiB
gkd12ca@cip50:~$ ls -ltr Datei?
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:13 DateiB
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:13 DateiC
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:14 DateiA
-rw------- 1 gkd12ca gkd12 0 Nov  6 12:14 DateiD
```

- Handelt es sich um einen absoluten oder relativen Pfad und wohin zeigt dieser?

```
bin/checkLocations.sh
```

```
../../../sys/kernel/
```

```
/usr/share/gmt/../../lib/ruby/1.8/../../../local/bin/
```

```
/../../.././etc/default
```

- einige elementare Programme

```
echo [TEXT]
```

- [TEXT]
  - wird auf dem Terminal ausgegeben
  - nach Möglichkeit in "..." setzen

```
gkd12ca@cip50:~$ echo " Ein Ausgabetext von $USER. "
 Ein Ausgabetext von gkd12ca.
```

```
sleep [SEKUNDEN]
```

- [SEKUNDEN]
  - die Shell schläft für die angegebene Zahl an Sekunden

```
gkd12ca@cip50:~$ date ; sleep 30 ; date
Fri Nov  6 13:43:51 CET 2009
Fri Nov  6 13:44:21 CET 2009
gkd12ca@cip50:~$
```

```
cat [DATEIEN]
tac [DATEIEN]
```

- [DATEIEN]
  - Inhalt der Dateien wird nacheinander im Terminal ausgegeben oder in umgekehrter Reihenfolge

```
gkd12ca@cip50:~$  cat /etc/ntp.conf .dmrc
logfile /var/log/ntpd
driftfile /var/lib/ntp/ntp.drift
#broadcastclient yes
server 129.187.254.32
server 10.156.8.31
[Desktop]
Session=default
```

- Erstellen von Verzeichnissen

```
mkdir [PFAD]
```

- [PFAD]
  - wird durch einen absoluten oder relativen Pfad ersetzt

```
gkd12ca@cip50:~$ ls
Desktop
gkd12ca@cip50:~$ mkdir VerzeichnisA ; ls
Desktop  VerzeichnisA
gkd12ca@cip50:~$ mkdir $HOME/VerzeichnisB
gkd12ca@cip50:~$ mkdir /home/cip/$USER/VerzeichnisC
gkd12ca@cip50:~$ ls
Desktop  VerzeichnisA  VerzeichnisB  VerzeichnisC
```

- Löschen von Verzeichnissen

```
rmdir [PFAD]
```

- [PFAD]
  - wird durch einen absoluten oder relativen Pfad ersetzt
  - das zu löschende Verzeichnis muss leer sein

```
gkd12ca@cip50:~$ ls
Desktop  VerzeichnisA  VerzeichnisB  VerzeichnisC
gkd12ca@cip50:~$ rmdir VerzeichnisA Verzeichnis[BC]
gkd12ca@cip50:~$ ls
Desktop
gkd12ca@cip50:~$ rmdir Desktop/
rmdir: Desktop/: Directory not empty
```

- Kopieren von Dateien und Verzeichnissen

```
cp [OPTIONEN] QUELLE[N] ZIEL
```

- [OPTIONEN]
  - -a
    - alle Dateiattribute werden erhalten
  - -b
    - vorhandene Dateien werden umbenannt
  - -i
    - interaktive Nachfrage
  - -r
    - ermöglicht das Kopieren ganzer Verzeichnisse mit Inhalt
  - -u
    - nur Dateien kopieren für die keine neueren existieren

- `QUELLE[N] ZIEL`
  - werden ersetzt durch
    - Dateinamen oder
    - absoluten Pfad oder
    - relativen Pfad

```
gkd12ca@cip50:~$ cp .bashrc neueDatei
gkd12ca@cip50:~$ ls
Desktop  maildir  neueDatei
gkd12ca@cip50:~$ mkdir TEST
gkd12ca@cip50:~$ cp /etc/profile \
 /usr/share/doc/bash/README.abs-guide TEST/
gkd12ca@cip50:~$ ls -l TEST/
-rw------- 1 gkd12ca gkd12 1105 Nov  8 15:33 README.abs-guide
-rw------- 1 gkd12ca gkd12  486 Nov  8 15:33 profile
```

```
gkd12ca@cip50:~$ cp -i /etc/profile \
 /usr/share/doc/bash/README.abs-guide TEST/
cp: overwrite ‘TEST/profile’? y
cp: overwrite ‘TEST/README.abs-guide’? y
gkd12ca@cip50:~$ ls -l TEST/
-rw------- 1 gkd12ca gkd12 1105 Nov  8 15:34 README.abs-guide
-rw------- 1 gkd12ca gkd12  486 Nov  8 15:34 profile
gkd12ca@cip50:~$ cp -b /etc/profile \
 /usr/share/doc/bash/README.abs-guide TEST/
cp: overwrite ‘TEST/profile’? y
cp: overwrite ‘TEST/README.abs-guide’? y
gkd12ca@cip50:~$ ls -l TEST/
-rw------- 1 gkd12ca gkd12 1105 Nov  8 15:36 README.abs-guide
-rw------- 1 gkd12ca gkd12 1105 Nov  8 15:34 README.abs-guide~
-rw------- 1 gkd12ca gkd12  486 Nov  8 15:36 profile
-rw------- 1 gkd12ca gkd12  486 Nov  8 15:34 profile~
```

```
gkd12ca@cip50:~$ cp -r TEST test
gkd12ca@cip50:~$ ls
Desktop  TEST  maildir  test
gkd12ca@cip50:~$ ls -l TEST test
TEST/:
-rw------- 1 gkd12ca gkd12 1105 Nov  8 15:37 README.abs-guide
-rw------- 1 gkd12ca gkd12 1105 Nov  8 15:36 README.abs-guide~
-rw------- 1 gkd12ca gkd12  486 Nov  8 15:37 profile
-rw------- 1 gkd12ca gkd12  486 Nov  8 15:36 profile~

test/:
-rw------- 1 gkd12ca gkd12 1105 Nov  8 15:38 README.abs-guide
-rw------- 1 gkd12ca gkd12 1105 Nov  8 15:38 README.abs-guide~
-rw------- 1 gkd12ca gkd12  486 Nov  8 15:38 profile
-rw------- 1 gkd12ca gkd12  486 Nov  8 15:38 profile~
```

- Verschieben/Umbenennen von Dateien und Verzeichnissen

```
mv [OPTIONEN] QUELLE[N] ZIEL
```

- `[OPTIONEN]`
  - -b
    - vorhandene Dateien werden umbenannt
  - -i
    - interaktive Nachfrage
  - -u
    - nur Dateien kopieren für die keine neueren existieren
- `QUELLE[N] ZIEL`
  - werden ersetzt durch
    - Dateinamen oder
    - absoluten Pfad oder
    - relativen Pfad

```
gkd12ca@cip50:~$ cd test/
gkd12ca@cip50:~/test$ ls
README.abs-guide  README.abs-guide~  profile  profile~
gkd12ca@cip50:~/test$ mv profile Daten-1.txt
gkd12ca@cip50:~/test$ ls
Daten-1.txt  README.abs-guide  README.abs-guide~  profile~
gkd12ca@cip50:~/test$ cd ..
gkd12ca@cip50:~$ mv test/ versuch
gkd12ca@cip50:~$ ls
Desktop  TEST  maildir  versuch
gkd12ca@cip50:~$ mv TEST/ versuch/
gkd12ca@cip50:~$ ls versuch/
Daten-1.txt  README.abs-guide  README.abs-guide~  TEST  profile~
```

- Löschen von Dateien und Verzeichnissen

```
rm [OPTIONEN] DATEIEN
```

- `[OPTIONEN]`
  - -i
    - interaktive Nachfrage
  - -r
    - rekursives Löschen von ganzen Verzeichnissen
  - -f
    - Ausführen ohne Nachfrage
- `DATEIEN`
  - wird ersetzt durch
    - Dateinamen oder
    - absoluten Pfad oder
    - relativen Pfad

```
gkd12ca@cip50:~$ rm versuch/Daten-1.txt
rm: remove regular file ‘versuch/Daten-1.txt’? y
gkd12ca@cip50:~$ ls versuch/
README.abs-guide  README.abs-guide~  TEST  profile~
gkd12ca@cip50:~$ rm -r versuch/TEST/
rm: descend into directory ‘versuch/TEST/’? y
rm: remove regular file ‘versuch/TEST//profile’? y
rm: remove regular file ‘versuch/TEST//README.abs-guide~’? y
rm: remove regular file ‘versuch/TEST//profile~’? y
rm: remove regular file ‘versuch/TEST//README.abs-guide’? y
rm: remove directory ‘versuch/TEST/’? y
gkd12ca@cip50:~$ ls versuch/
README.abs-guide  README.abs-guide~  profile~
gkd12ca@cip50:~$ rm -rf versuch/
gkd12ca@cip50:~$ ls
Desktop  maildir
```

- Verweise auf Dateien und Verzeichnisse

```
ln [OPTIONEN] QUELLE ZIEL
```

- [OPTIONEN]
  - -s
    - symbolischen Verweis/Link erzeugen

```
gkd12ca@cip50:~$ ln -s maildir/ E-MAIL
gkd12ca@cip50:~$ ln -s /etc/profile $HOME/
gkd12ca@cip50:~$ ls -l && rm -f E-MAIL profile
drwx------ 2 gkd12ca gkd12 4096 Nov  8 15:27 Desktop
lrwxrwxrwx 1 gkd12ca gkd12    8 Nov  8 16:14 E-MAIL -> maildir/
drwx------ 2 gkd12ca gkd12 4096 Oct 19 11:50 maildir
lrwxrwxrwx 1 gkd12ca gkd12   12 Nov  8 16:14 profile -> /etc/prof
```

Diverse Programme für Dateien und Verzeichnisse

```
touch DATEIEN
```

- Aktualisierung der Zugriffszeit für eine Datei

```
gkd12ca@cip50:~$ ls -l .bashrc ; echo "###" ; \
 sleep 80 ; echo "###" ; \
 touch .bashrc ; ls -l .bashrc
-rw------- 1 gkd12ca gkd12 2278 Nov  8 16:22 .bashrc
###
###
-rw------- 1 gkd12ca gkd12 2278 Nov  8 16:24 .bashrc
```

```
cut [OPTIONEN] DATEIEN
```

- zeilenweises Ausschneiden/Abschneiden von Dateiinhalten
- [OPTIONEN]
  - -d
    - verwende dieses Trennzeichen zwischen den Feldern anstatt von TAB
  - -f
    - nur die angegebenen Felder werden ausgegeben

```
gkd12ca@cip50:~$ cut -d\  -f 1,3 /etc/hosts
127.0.0.1
10.153.84.129
10.153.84.179 cip50
```

wc [OPTIONEN] DATEIEN

- Zählen der Zeichen, Wörter, Zeilen von Dateien
- [OPTIONEN]
  - -c
    - nur die Zeichen zählen
  - -w
    - nur die Wörter zählen
  - -l
    - nur die Zeilen zählen

```
gkd12ca@cip50:~$ wc /etc/hosts
  3   9 104 /etc/hosts
gkd12ca@cip50:~$ cat /etc/host* /etc/profile | wc -l
64
```

`less [OPTIONEN] DATEIEN`

- Anzeigen des Dateiinhalts
- `[OPTIONEN]`
  - -N
    - Anzeigen der Zeilennummer
  - -E
    - zurück zu Prompt bei Erreichen des Dateiendes
  - -p`[Suchpattern]`
    - Anzeige beginnt an Stelle, wo `[Suchpattern]` gefunden wird
- Navigation
  - Bewegen in der Datei – `Pfeiltasten`
  - Beenden der Anzeige – `q`
  - Suchen nach Pattern – `/[Suchpattern]`

```
gkd12ca@cip50:~$ less -N /etc/bash.bashrc
     1 # System-wide .bashrc file for interactive bash(1) shells.
     2
     3 # To enable the settings / commands in this file for login
     4 # this file has to be sourced in /etc/profile.
     5
...
gkd12ca@cip50:~$ cat /etc/default/* | less
# Specify options for acpid startup, Debian default is to enable
# use of sockets at a non-default position
OPTIONS=s /var/run/acpid.socket"

# Specify modules to load on acpid's startup
# MODULES may be uncommented to load "none", contain the string
# to load all acpi related modules or simply a space seperated
...
```

```
head [OPTIONEN] DATEIEN
tail [OPTIONEN] DATEIEN
```

- Anzeigen der ersten oder letzten Zeilen des Dateiinhalts
- `[OPTIONEN]`
  - `-n i`
    - Ausgeben der ersten oder letzten `i` Zeilen
  - `-f`
    - nur für `tail`
    - Programm verweilt in einer Endlosschleife
    - Anzeige aller Änderungen Dateiende
    - Datei bleibt geöffnet (`STRG-C`)

```
gkd12ca@cip50:~$ head -n 7 /etc/profile
# /etc/profile: system-wide .profile file for the Bourne shell
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ „id -u‘" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/games"
gkd12ca@cip50:~$ tail -n 4 -f /var/log/mail.log
Nov 13 04:11:24 cip50 postfix/cleanup[7544]: BEE3574477: message-
Nov 13 04:11:24 cip50 postfix/qmgr[3158]: BEE3574477: from=<root@
Nov 13 04:11:24 cip50 postfix/smtp[7550]: BEE3574477: to=<spann@c
Nov 13 04:11:24 cip50 postfix/qmgr[3158]: BEE3574477: removed
...
gkd12ca@cip50:~$ head -n 1 tomo.0??[05].dat
...
```

```
sort [OPTIONEN] DATEIEN
```

- Sortieren der Zeilen von Dateien
- [OPTIONEN]
  - -n
    - numerische Sortierung der Zeilen
  - -u
    - doppelte Einträge werden in Ausgabe entfernt

```
gkd12ca@cip50:~$ sort /etc/hosts
10.153.84.129   libby.cipmath.loc libby cip04
10.153.84.179 cip50.cipmath.loc cip50
127.0.0.1 localhost
```

```
gkd12ca@cip50:~$ wget \
 http://www.geophysik.uni-muenchen.de/~oeser/LV/Shell/sort_ex.txt
gkd12ca@cip50:~$ cat sort_ex.txt
...
gkd12ca@cip50:~$ sort sort_ex.txt
...
gkd12ca@cip50:~$ sort -u sort_ex.txt
000 IP
001 ICMP
002 IGMP
003 GGP
006 TCP
012 PUP
017 UDP
022 IDP
...
```

Suchen nach Dateien und Verzeichnissen

```
find [OPTIONEN] [PFAD] [SUCHKRITERIUM] [AKTION]
```

- [OPTIONEN]
  - -P
    - niemals symbolischen Verweisen folgen
  - -L
    - immer symbolischen Verweisen folgen
- [PFAD]
  - wird ersetzt durch
    - absoluten Pfad oder
    - relativen Pfad
- [SUCHKRITERIUM]
  - -amin/atime [n]
    - sucht nach Dateien, auf die in den letzten [n] Minuten/Tage ein Zugriff erfolgte

- `[SUCHKRITERIUM]`
  - `-cmin/ctime [n]`
    - sucht nach Dateien, die in den letzten `[n]` Minuten/Tage neu erstellt wurden
  - `-empty`
    - sucht nach leeren, regulären Dateien und Verzeichnissen
  - `-gid [n]`
    - sucht nach Dateien, deren Gruppen-ID `[n]` ist
  - `-group [NAME]`
    - sucht nach Dateien, die der Gruppe `[NAME]` zugeordnet sind
  - `-mmin/mtime [n]`
    - sucht nach Dateien, deren Inhalt innerhalb der letzten `[n]` Minuten/Tage modifiziert wurde
  - `-name [NAME]`
    - sucht nach Dateien mit dem Namen `[NAME]`
  - `-perm [RECHT]`
    - sucht nach Dateien mit dem Zugriffsrecht `[RECHT]`

- `[SUCHKRITERIUM]`
  - `-size [n]`
    - sucht nach Dateien mit der Größe n, wobei die Einheit festgelegt werden kann (2k $\equiv$ 2 Kilobyte)
  - `-type [TYP]`
    - sucht nach Dateien des Typs `[TYP]` (b (Blockdatei), c (Characterdatei), d (Verzeichnis), p (FIFO), f (reguläre Datei), l (Softlink) und s (Socket))
  - `-uid [UID]`
    - sucht nach Dateien, die dem Benutzer mit der ID UID gehören
  - `-user [NAME]`
    - sucht nach Dateien, die dem Benutzer mit dem Namen `[NAME]` gehören
  - `! [SUCHKRITERIUM]`
    - logische Verneinung
  - `[SUCHKRITERIUM] -a/and / -o/or [SUCHKRITERIUM]`
    - logisches `UND` beziehungsweise `ODER`

- `[AKTION]`
  - `-exec [CMD] {} \;`
    - führt das Kommando `[CMD]` mit den gefundenen Dateien `{} \;` aus
  - `-print`
    - gibt jeden gefundenen Dateinamen in einer separaten Zeile aus
  - `-print0`
    - gibt den Dateinamen mit anschließendem `'\0'`-Zeichen aus
  - `-ls`
    - listet die gefundenen Dateien mit dem Kommando `ls -lisa` auf
  - `-fls/fprint [DATEI]`
    - äquivalente Ausgabe wie `-ls` oder `-print` - mit Unterschied, dass Ausgabe in die Datei `[DATEI]` geschrieben wird

```
gkd12ca@cip50:~$ find ~ -name "*.pdf"
/home/cip/gkd12ca/Desktop/Datenverarbeitung.pdf
```

```
gkd12ca@cip50:~$ find /usr/ -type d -name examples
/usr/share/apps/kwordquiz/examples
/usr/share/apps/kvoctrain/examples
/usr/share/apps/kturtle/examples
/usr/share/apps/artsbuilder/examples
...
gkd12ca@cip50:~$ find /usr/ -type d -name examples -ls
1329753 4 drwxr-xr-x 2 root root 4096 Oct 10 2007 /usr/share/apps
1318914 4 drwxr-xr-x 2 root root 4096 Oct 10 2007 /usr/share/apps
1318710 4 drwxr-xr-x 4 root root 4096 Oct 10 2007 /usr/share/apps
1249525 4 drwxr-xr-x 3 root root 4096 Oct 10 2007 /usr/share/apps
...
gkd12ca@cip50:~$ cp /etc/profile . ; cp -b /etc/profile .
gkd12ca@cip50:~$ find . -name "*~" -print -exec rm -f {} \;
./profile~
```

Suchen in Dateien

```
grep [OPTIONEN] [MUSTER] [DATEIEN]
```

- [OPTIONEN]
  - -c
    - zählen der Treffer
  - -v
    - Negation
  - -H
    - Ausgabe von Dateinamen und Treffer
- [MUSTER]
  - wird ersetzt durch einen regulären Ausdruck
- [DATEIEN]
  - wird ersetzt durch Dateinamen

```
gkd12ca@cip50:~$ wget \
 http://www.geophysik.uni-muenchen.de/~oeser/LV/Shell/ort_ex.txt\
 http://www.geophysik.uni-muenchen.de/~oeser/LV/Shell/ex-1.xyz\
 http://www.geophysik.uni-muenchen.de/~oeser/LV/Shell/ex-2.xyz
gkd12ca@cip50:~$ less -e *ex*
gkd12ca@cip50:~$ grep a ort_ex.txt
Friedrichshafen
Halle
Furtwangen
...
gkd12ca@cip50:~$ grep -i "[fn]$" *ex*
ex-1.xyz:# Hainichen
ex-2.xyz:# Hainichen
ort_ex.txt:Bremen
ort_ex.txt:Friedrichshafen
...
```

Besitzer und Gruppe von Dateien und Verzeichnissen ändern

```
chown [OPTIONEN] [USER]  [DATEIEN]
chgrp [OPTIONEN] [GROUP] [DATEIEN]
```

- [OPTIONEN]
  - -R
    - rekursive alle Unterverzeichnisse und Dateien bearbeiten
- [DATEIEN]
  - wird ersetzt durch
    - Dateinamen oder
    - absoluten Pfad oder
    - relativen Pfad

```
gkd12ca@cip50:~$ finger $USER
Login: gkd12ca          Name: DV-GeophysikI
Directory: /home/cip/gkd12ca          Shell: /bin/bash
On since Sun Nov 29 10:13 (CET) on pts/0 from munich.geophysik
No mail.
No Plan.
gkd12ca@cip50:~$ id
uid=24053(gkd12ca) gid=24000(gkd12) groups=24000(gkd12)
gkd12ca@cip50:~$ chown gkd12ca sort_ex.txt
gkd12ca@cip50:~$ chgrp -R gkd12 Desktop/
gkd12ca@cip50:~$ ls -l Desktop/
total 1240
-rw------- 1 gkd12ca gkd12 1242345 Nov 23 08:31 Datenverarbeit
-rw------- 1 gkd12ca gkd12    4485 Oct 26 08:28 Home.desktop
-rw------- 1 gkd12ca gkd12    3819 Oct 26 08:28 System.desktop
-rw------- 1 gkd12ca gkd12    5066 Oct 26 08:28 trash.desktop
```

Zugriffsrechte von Dateien und Verzeichnissen ändern

```
chmod [OPTIONEN] [ZUGRIFFSRECHTE] [DATEIEN]
```

- [OPTIONEN]
  - -R
    - rekursive alle Unterverzeichnisse und Dateien bearbeiten
- [DATEIEN]
  - wird ersetzt durch
    - Dateinamen oder
    - absoluten Pfad oder
    - relativen Pfad

```
gkd12ca@cip50:~$ ls -l ex*
-rw------- 1 gkd12ca gkd12 565 Nov 29 10:17 ex-1.xyz
-rw------- 1 gkd12ca gkd12 797 Nov 29 10:17 ex-2.xyz
```

- [ZUGRIFFSRECHTE]
  - Zugriffsrechte werden ganz links in der Ausgabe angezeigt, das erste Zeichen gibt den Dateityp an und die weiteren 3 Dreiergruppen die Zugriffsrechte
  - Zugriffsrechte werden geordnet angegeben nach:
    - Besitzer (`u` – `user`)
    - Gruppe (`g` – `group`)
    - Andere (`o` – `other`)
  - jede Dreiergruppe zeigt ob folgende Zugriffsrechte gesetzt sind:
    - lesen (`r` – `read`)
    - schreiben (`w` – `write`)
    - ausführen (`x` – `execute`)

- wenn Zugriffsrecht gesetzt ist, wird der Buchstabe (`r,w,x`) angezeigt
- wenn Zugriffsrecht nicht gesetzt ist, wird der Bindestrich angezeigt
- dies entspricht der binären Darstellung durch 0 und 1
- für 3 Rechten wird genau eine Oktalzahl (Zahl zur Basis 8 – 2^3) benötigt:

oktal	0	1	2	3	4	5	6	7
binär	000	001	010	011	100	101	110	111

Oktal	Übersetzung	Interpretation
777	`rwxrwxrwx`	alle dürfen alles
664	`rw-rw-r--`	u,g dürfen r,w – o darf r
440	`r--r-----`	u,g dürfen r – o darf nichts

- um in Verzeichnisse zu wechseln, benötigt man immer das Ausführungsrecht

- damit gibt es zwei Möglichkeiten, die Rechte zu ändern

```
gkd12ca@cip50:~$ ls -l DATEI
-r--r--r-- 1 gkd12ca gkd12 90 Nov 13 15:07 DATEI
gkd12ca@cip50:~$ chmod u=rw,g+w,o-r DATEI
gkd12ca@cip50:~$ chmod 660 DATEI
```

```
gkd12ca@cip50:~$ chmod 755 Desktop/
gkd12ca@cip50:~$ ls -l
drwxr-xr-x 2 gkd12ca gkd12 4096 Nov  8 15:27 Desktop
gkd12ca@cip50:~$ chmod u=rw,g=r,o= *ex*
gkd12ca@cip50:~$ ls -l *ex*
-rw-r----- 1 gkd12ca gkd12 565 Nov 29 10:17 ex-1.xyz
-rw-r----- 1 gkd12ca gkd12 797 Nov 29 10:17 ex-2.xyz
-rw-r----- 1 gkd12ca gkd12 119 Nov 23 07:55 ort_ex.txt
-rw-r----- 1 gkd12ca gkd12  90 Nov 13 15:07 sort_ex.txt
```

Datei- und Verzeichnisarchive erstellen – mit/ohne Komprimierung

```
tar [OPTIONEN] [DATEIEN]
```

- [OPTIONEN]
  - -c
    - ein Archiv erzeugen
  - -x
    - ein Archiv entpacken
  - -t
    - ein Archiv betrachten
  - -f [DATEINAME]
    - Archiv mit entsprechenden Namen verwenden
  - -z
    - gzip-Komprimierung einsetzen
  - -j
    - bzip2-Komprimierung einsetzen

- [DATEIEN]
  - wird ersetzt durch
    - Dateinamen oder
    - absoluten Pfad oder
    - relativen Pfad

```
gkd12ca@cip50:~$ tar -c -f Beispiele-1.tar -- *ex*
gkd12ca@cip50:~$ ls -l Beispiele-1.tar
-rw------- 1 gkd12ca gkd12 10240 Nov 29 11:58 Beispiele-1.tar
gkd12ca@cip50:~$ tar -t -f Beispiele-1.tar
ex-1.xyz
ex-2.xyz
ort_ex.txt
sort_ex.txt
gkd12ca@cip50:~$ tar -c -f Beispiele-2.tar.gz -z  -- *ex*
gkd12ca@cip50:~$ ls -l Beispiele-2.tar.gz
-rw------- 1 gkd12ca gkd12 872 Nov 29 11:59 Beispiele-2.tar.gz
```

Dateien komprimieren/dekomprimieren

```
gzip [DATEIEN]
gunzip [DATEIEN]

bzip2 [DATEIEN]
bunzip2 [DATEIEN]

zip ZIPDATEI [DATEIEN]
unzip ZIPDATEI
```

- [DATEIEN]
  - wird ersetzt durch
    - Dateinamen oder
    - absoluten Pfad oder
    - relativen Pfad

```
gkd12ca@cip50:~$ bzip2 Beispiele-1.tar
gkd12ca@cip50:~$ ls -l Beispiele-1.tar.bz2
-rw------- 1 gkd12ca gkd12 841 Nov 29 11:58 Beispiele-1.tar.bz2
gkd12ca@cip50:~$ bunzip2 Beispiele-1.tar.bz2
gkd12ca@cip50:~$ gzip *.txt
gkd12ca@cip50:~$ ls -l *.gz
...
gkd12ca@cip50:~$ gunzip *.txt.gz
gkd12ca@cip50:~$ zip  xyz.zip ex-*
  adding: ex-1.xyz (deflated 58%)
  adding: ex-2.xyz (deflated 60%)
gkd12ca@cip50:~$ unzip xyz.zip
replace ex-1.xyz? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: ex-1.xyz
replace ex-2.xyz? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: ex-2.xyz
```

Befehle für das Drucken von Dateien

```
lp [DATEIEN]
a2ps [DATEI]
```

- [DATEIEN]
  - wird ersetzt durch
    - Dateinamen oder
    - absoluten Pfad oder
    - relativen Pfad

```
gkd12ca@cip50:~$ a2ps ort_ex.txt
gkd12ca@cip50:~$ cp /usr/share/doc/gs-gpl/examples/tiger.eps.gz .
gkd12ca@cip50:~$ gunzip tiger.eps.gz
gkd12ca@cip50:~$ gv tiger.eps
gkd12ca@cip50:~$ lp tiger.eps
```

Befehle für das Druckmanagement

```
lpstat
cancel
lpoptions
```

- `lpstat`
  - Statusinformationen zu Druckern und Druckaufträgen abfragen
- `cancel`
  - Druckaufträge löschen
- `lpoptions`
  - Druckoptionen anpassen

weitere Programme

- `logout` und `exit`
  - aktuelle Terminalsitzung beenden
- `passwd`
  - eigenes Passwort ändern
- `id`
  - eigene Benutzer- und Gruppenkennung (UID und GID) abfragen
- `logname`
  - eigenen Loginnamens abfragen
- `who`
  - momentan angemeldeten Benutzer abfragen
- `finger [USERNAME]`
  - Informationen zu anderen Benutzern abfragen
- `last`
  - An- und Abmeldezeiten von Benutzern erfragen

Prozess- und Resourcenmanagement

`top`

- Auflistung der Prozesse geordnet nach CPU-Belastung
- Tastaturabkürzungen
  - `h` – Hilfe
  - `q` – Programm `top` beenden
  - `u` – Prozesse eines Nutzers anzeigen
  - `k` – Beenden eines Prozesses mit der `PID`

```
top - 14:36:39 up 22:53,  2 users,  load average: 1.00, 1.05, 1.03
Tasks: 113 total,   2 running, 111 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy, 50.1%ni, 49.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3107484k total,  2735880k used,   371604k free,   217456k buffers
Swap:  2104472k total,        0k used,  2104472k free,   352780k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+ COMMAND
 9892 mdd73ak   30  10 1991m 1.9g  344 R  100 65.6   2:22.35 ldms
    1 root      20   0  1940  636  540 S    0  0.0   0:01.28 init
```

`ps [OPTIONEN]`

- Informationen zu Prozessen auflisten
- `[OPTIONEN]`
  - `-e -A`
    - Informationen zu allen Prozessen anzeigen
  - `-f`
    - vollständige Informationen anzeigen
  - `-l`
    - viele Informationen anzeigen
  - `-p [PID]`
    - Informationen zu Prozess mit der Nummer `PID`
  - `-u [USERNAME]`
    - Informationen zu Prozessen des Benutzers `USERNAME`

```
gkd12ca@cip50:~$ ps -u mdd73ak -f
UID        PID PPID  C STIME TTY          TIME CMD
mdd73ak   8010    1  0 09:52 ?        00:00:00 /bin/bash ./task.sh
mdd73ak   9892 8010 99 14:34 ?        00:20:52 .././ldms 328292300
gkd12ca@cip50:~$ ps -e -l
F S UID  PID PPID C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   0    1    0 0  80   0 -   485 -      ?        00:00:01 init
1 S   0    2    0 0  75  -5 -     0 -      ?        00:00:00 kthreadd
1 S   0    3    2 0 -40   - -     0 -      ?        00:00:00 migration/0
1 S   0    4    2 0  75  -5 -     0 -      ?        00:00:00 ksoftirqd/0
5 S   0    5    2 0 -40   - -     0 -      ?        00:00:00 watchdog/0
1 S   0    6    2 0 -40   - -     0 -      ?        00:00:00 migration/1
1 S   0    7    2 0  75  -5 -     0 -      ?        00:00:00 ksoftirqd/1
gkd12ca@cip50:~$ ps -p 1 -f
UID  PID PPID C STIME TTY          TIME CMD
root   1    0 0 Dec08 ?        00:00:01 init [2]
```

```
kill -s [SIGNAL] PID
```

- Signale an Prozesse (über PID) senden
- [SIGNAL]
  - QUIT
    - Prozess sauber beenden
  - KILL
    - Prozess wird beendet
  - STOP
    - Prozess wird angehalten
  - CONT
    - Prozess wird fortgesetzt

```
gkd12ca@cip50:~$ wget \
 http://www.geophysik.uni-muenchen.de/~oeser/LV/Shell/rechne
gkd12ca@cip50:~$ chmod 700 rechne
gkd12ca@cip50:~$ ./rechne &
gkd12ca@cip50:~$ top

gkd12ca@cip50:~$ kill -s STOP 18148
[1]+  Stopped                 ./rechne
gkd12ca@cip50:~$ ps -l -C rechne
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN TTY          TIME CMD
0 T 24053 18148 17982 77  80   0 -    23 -     pts/1    00:00:42 rech
gkd12ca@cip50:~$ kill -s CONT 18148
gkd12ca@cip50:~$ ps -l -C rechne
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN TTY          TIME CMD
0 R 24053 18148 17982 66  80   0 -    23 -     pts/1    00:00:46 rech
```

Hintergrundprozesse und Prozesspriorität

- `BEFEHL &`
  - Befehl im Hintergrund starten
- `nohup BEFEHL`
  - Befehl bei Sitzungsende weiter laufen lassen
- `nice -n WERT BEFEHL`
  - Befehl mit anderer Priorität starten

```
gkd12ca@cip50:~$ nohup xclock &
[1] 10354
gkd12ca@cip50:~$ nohup: appending output to ‘nohup.out’
[STRG-D]
...
gkd12ca@cip50:~$ nice -n 19 iceweasel &
gkd12ca@cip50:~$ ps -f -C iceweasel
```

Remotezugriff auf Rechner

```
ssh [OPTIONEN] RECHNERNAME [BEFEHL]
```

- gesichert mit entferntem Rechner verbinden und da arbeiten
- `[OPTIONEN]`
  - `-x -X`
    - Deaktivierung/Aktivierung der Weiterleitung der grafischen Ausgabe
  - `-l USERNAME`
    - Anmeldung als Nutzer mit dem Nutzernamen `USERNAME`
- `RECHNERNAME`
  - wird ersetzt durch den Namen des entfernten Rechners (`FQDN`)
- `[BEFEHL]`
  - wird ersetzt durch einen Befehl der auf dem entfernten Rechner ausgeführt werden soll

```
gkd12ca@cip50:~$ ssh -X cip76
Last login: Fri Dec 11 14:19:43 2009 from cip50.cipmath.loc
gkd12ca@cip76:~$ xclock
[STRG-C]
gkd12ca@cip76:~$ ssh -x cip52 xeyes
Error: Can't open display:
gkd12ca@cip76:~$ ssh -l $USER cip30 'ps -f -u $LOGNAME'
UID        PID PPID C STIME TTY          TIME CMD
gkd12ca 5390 5387 0 15:37 ?    00:00:00 sshd: gkd12ca@notty
gkd12ca 5393 5390 0 15:37 ?    00:00:00 ps -f -u gkd12ca
gkd12ca@cip76:~$ ssh cip52
gkd12ca@cip52:~$ echo $HOSTNAME
cip52
gkd12ca@cip52:~$ exit
logout
Connection to cip52 closed.
```

```
sftp [USERNAME@]RECHNERNAME[:VERZEICHNIS]
scp [USERNAME@][RECHNERNAME:]QUELLDATEIEN] \
    [USERNAME@][RECHNERNAME:]ZIELDATEIEN
```

- gesichert zu/von entferntem Rechner Dateien übertragen
- `[USERNAME]`
  - wird ersetzt durch den Nutzernamen mit dem die Anmeldung am Rechner erfolgen soll
- `RECHNERNAME`
  - wird ersetzt durch den Namen des entfernten Rechners (`FQDN`)
- `[VERZEICHNIS]`
  - wird ersetzt durch ein Verzeichnis auf dem entfernten Rechner
- `[DATEIEN]`
  - wird ersetzt durch die zu übertragenden Dateien

```
gkd12ca@cip50:~$ sftp cip34
Connecting to cip34...
sftp> put ex-1.xyz ex1.dat
Uploading ex-1.xyz to /home/cip/gkd12ca/ex1.dat
ex-1.xyz                      100% 565 0.6KB/s 00:00
sftp> get ex-2.xyz ex2.dat
Fetching /home/cip/gkd12ca/ex-2.xyz to ex2.dat
/home/cip/gkd12ca/ex-2.xyz    100% 797 0.8KB/s 00:00
sftp> ls
Desktop ex-1.xyz ex-2.xyz ex1.dat ex2.dat kopie.txt maildir ort_e
sftp> exit
gkd12ca@cip50:~$ scp $USER@cip76:~/ex-1.xyz Desktop/
ex-1.xyz  100% 565 0.6KB/s 00:00
gkd12ca@cip50:~$ scp kopie.txt $USER@cip76:~/neu.dat
kopie.txt 100%  90 0.1KB/s 00:00
```

Umleitung der Ein- und Ausgabe

```
BEFEHL <  DATEI
BEFEHL >  DATEI
BEFEHL >> DATEI
BEFEHL 2> DATEI
BEFEHL > DATEI 2>&1
```

- Umleitung der Standardeingabe, -ausgabe und -fehlerausgabe
- ermöglicht flexiblen Umgang mit Dateiinhalten
  - `BEFEHL`
    - wird durch jeden beliebigen Befehl ersetzt
  - `DATEI`
    - wird durch einen Dateinamen ersetzt

```
gkd12ca@cip50:~$ env > output.dat
gkd12ca@cip50:~$ ps -u $USER -f >> output.dat
gkd12ca@cip50:~$ grep gkd < output.dat
USER=gkd12ca
MAIL=/var/mail/gkd12ca
PWD=/home/cip/gkd12ca
HOME=/home/cip/gkd12ca
LOGNAME=gkd12ca
gkd12ca 15969 15963 0 09:30 ?     00:00:00 sshd: gkd12ca@pts/0
gkd12ca 15972 15969 0 09:30 pts/0 00:00:00 -bash
gkd12ca 15994 15972 0 09:31 pts/0 00:00:00 ps -u gkd12ca -f
gkd12ca@cip50:~$ grep gkd < output.dat > treffer.dat
gkd12ca@cip50:~$ cat treffer.dat
...
gkd12ca@cip50:~$ grep gkd output.dar > treffer.dat 2> fehler.dat
gkd12ca@cip50:~$ cat fehler.dat
grep: output.dar: No such file or directory
```

Zusammenfassung

- Welche wesentlichen Punkte sollten Sie für den Umgang mit einem Linux-Terminal verinnerlichen?
  - Wie sieht der allgemeine Kommandosyntax unter Linux aus?
  - Wie erhalten Sie die Hilfe/Dokumentation zu Befehlen?
  - Was versteht man unter dem relativen und absoluten Pfad?
  - Welche Zeichen haben eine Sonderbedeutung für die `SHELL`?
  - Was sind Variablen und wie werden diese gesetzt/ausgelesen?
- Gibt es Fragen oder Unklarheiten zu diesem Kapitel?
- Der Link DV-Uebung-Linux.txt führt Sie zu einer Textdatei, welche verschiedene Beispielbefehle enthält. Wenn Sie in der Lage sind, die aufgeführten Befehle zu erklären, so können Sie beruhigt den zweiten Teil der Klausur bearbeiten.

awk-Programmierung

- die Anfangsbuchstaben der Schöpfer ergeben den Namen `awk`
  - **A**ho, Alfred V.
  - **W**einberger, Peter J.
  - **K**ernighan, Brian W.
- Ziele des Programms `awk`
  - einfaches und mächtiges Programmierwerkzeug
  - leichte und zugleich elegante Analyse und Manipulation der tagtäglich anfallenden Datensätze
- Merkmale des Programms `awk`
  - Suche nach Textmustern in beliebig vielen Dateien
  - Textmuster können bestimmte Aktionen veranlassen (Text ersetzen, Rechnungen ausführen und . . .)
  - `awk`-Programme sind meist nicht länger als ein oder zwei Zeilen

Literatur

- Offizielle Webseite
- Eintrag in Wikipedia
- awk & sed – Die Profitools zur Dateibearbeitung und -edition

- drei erste Anwendungsbeispiele zu `awk`
  - `echo Hallo Welt | awk '{print $1}'`
    - erzeugt als Ausgabe: `Hallo`
  - `echo Hallo Welt | awk '{print $2}'`
    - erzeugt als Ausgabe: `Welt`
  - `echo Hallo Welt  | awk '{printf "%s %s!\n",$1,$2}'`
    - erzeugt als Ausgabe: `Hallo Welt!`

Ein erstes einfaches Beispiel

```
gkd12ca@cip50:~$ wget \
 http://www.geophysik.uni-muenchen.de/~oeser/LV/AWK/bundliga.txt
gkd12ca@cip50:~$ head -n 2 bundliga.txt
Muenchen - Nuernberg                    3 : 2 34000 Zuschauer
Kaiserslautern - Moenchengladbach       2 : 1 28260 Zuschauer
```

- Liste der siegreichen Heim-Mannschaften

```
gkd12ca@cip50:~$ awk '$4 > $6   { print $1 }'   bundliga.txt
```

- Liste der unentschiedenen Partien

```
gkd12ca@cip50:~$ awk '$4 == $6 { print $1, " - " , $3, \
 $4 " : " $6 }' bundliga.txt
```

Struktur von awk-Programmen

- alle awk-Programme sind eine Abfolge von

```
pattern { aktion }
```

- in einem solchen awk-Programm werden die vorgegebenen Daten Zeile für Zeile verarbeitet
- in jeder Zeile wird geprüft, ob das Pattern (Muster) vorhanden ist
- für jede erfolgreiche Prüfung wird die zugehörige Aktion ausgeführt
- eine Aktion muss immer in `{...}` angegeben werden
- es ist möglich entweder `pattern` oder `{ aktion }` wegzulassen, aber niemals beide in einer awk-Anweisung

- wird ein Pattern ohne `{ aktion }` angegeben, wie in

```
awk '$4 > 2' bundliga.txt
```

- so wird jede Zeile, auf die das Pattern zutrifft vollständig ausgegeben

```
Muenchen - Nuernberg 3 : 2 34000 Zuschauer
Uerdingen - Homburg 3 : 0 10000 Zuschauer
Frankfurt - Waldhof 3 : 1 20000 Zuschauer
```

- wird ein Aktion ohne `pattern` angegeben, wie in

```
awk '{ print $1 " schiesst zuhause " $4 " Tore " }' bundliga.txt
```

- so wird die Aktion für jede Zeile ausgeführt

```
Muenchen schiesst zuhause 3 Tore
Kaiserslautern schiesst zuhause 2 Tore
Uerdingen schiesst zuhause 3 Tore
St.Pauli schiesst zuhause 0 Tore
Leverkusen schiesst zuhause 1 Tore
Stuttgart schiesst zuhause 2 Tore
Bochum schiesst zuhause 0 Tore
Frankfurt schiesst zuhause 3 Tore
Duesseldorf schiesst zuhause 1 Tore
```

Aufrufsyntax eines awk-Programms

Ein awk-Programm kann auf vier verschiedene Arten aufgerufen werden.

- 1. Art:

```
awk ' awk-Programm ' Datei(en)
```

- das angegebene Programm verarbeitet alle Datei(en)
- Beispiel:

```
gkd12ca@cip50:~$ awk ' $6 > $4 ' bundliga-1.txt bundliga-2.txt
```

- gibt alle Spielergebnisse aus der 1. und 2. Bundesliga aus, bei denen die Auswärtsmannschaft gewonnen hat

- 2. Art:

```
awk ' awk-Programm '
```

- werden keine Dateien angegeben, verarbeitet das Programm alle nachfolgenden Bildschirmeingaben (bis EOF – STRG-D)
- Beispiel:

```
gkd12ca@cip50:~$ awk '$1 > 0 { print $1 " ;mit MWST: " $1*1.19 }'
123
123 ;mit MWST: 146.37
-90
1002
1002 ;mit MWST: 1192.38
```

- für jede eingegebene positive Zahl wird die Zahl ohne und mit MWST ausgegeben

- 3. Art:

```
awk -f Programmdatei
```

- awk-Programm wird direkt in einer Programmdatei gespeichert (bei längeren Programmen von Vorteil)
- Beispiel:

```
gkd12ca@cip50:~$ cat mwst.awk
$1 > 0 { print $1 " ;mit MWST: " $1*1.19 }
gkd12ca@cip50:~$ awk -f mwst.awk
123
123 ;mit MWST: 146.37
-90
```

- für jede eingegebene positive Zahl wird die Zahl ohne und mit MWST ausgegeben

- 4. Art:

```
awk -f Programmdatei Datei(en)
```

- awk-Programm wird direkt in einer Programmdatei gespeichert (bei längeren Programmen von Vorteil) und die Daten werden aus `Datei(en)` gelesen
- Beispiel:

```
gkd12ca@cip50:~$ cat betraege
123
-90
gkd12ca@cip50:~$ awk -f mwst.awk betraege
123 ;mit MWST: 146.37
```

- für jede positive Zahl in der Datei `betraege` wird die Zahl ohne und mit MWST ausgegeben

Einfache Ausgaben

- jede Zeile ausgeben

```
gkd12ca@cip50:~$ awk '{ print }' bundliga.txt
Muenchen - Nuernberg                    3 : 2 34000 Zuschauer
Kaiserslautern - Moenchengladbach       2 : 1 28260 Zuschauer
...
gkd12ca@cip50:~$ awk '{ print $0 }' bundliga.txt
Muenchen - Nuernberg                    3 : 2 34000 Zuschauer
Kaiserslautern - Moenchengladbach       2 : 1 28260 Zuschauer
...
```

- $0 entspricht der ganzen Zeile
- da kein Pattern angegeben wurde, wird die Aktion auf alle Zeilen angewendet

- nur bestimmte Felder ausgeben

```
gkd12ca@cip50:~$ awk '{ print $1, $3, $4 $5 $6 }' bundliga.txt
Muenchen Nuernberg 3:2
Kaiserslautern Moenchengladbach 2:1
...
```

- $1 entspricht dem ersten Feld
- $2 entspricht dem zweiten Feld
- usw.
- ein Komma in einer `print` Anweisung sorgt für ein Leerzeichen in der Ausgabe

- Text ausgeben

```
gkd12ca@cip50:~$ awk ' $6 > $4 { print $3 " \
 gewinnt in " $1 }' bundliga.txt
Koeln gewinnt in Bochum
```

- einfacher Text muss durch "..." geklammert werden

- Anzahl von Feldern

```
gkd12ca@cip50:~$ awk '{ print NF, $1, $(NF-1), \
 $NF }' bundliga.txt
8 Muenchen 34000 Zuschauer
8 Kaiserslautern 28260 Zuschauer
8 Uerdingen 10000 Zuschauer
...
```

- `awk` zählt die Felder einer Zeile und legt die Anzahl in der Variablen NF ab
- `$` kann im Zusammenhang mit jedem möglichen Ausdruck angegeben werden
- im Beispiel besteht eine Zeile aus 8 Feldern, sodass `$NF` in `$8` und `$(NF-1)` in `$7` resultiert

- aktuelle Zeilennummer

```
gkd12ca@cip50:~$ awk '{ print " Spiel " \
 , NR, $1 $2 $3 }' bundliga.txt
 Spiel 1 Muenchen-Nuernberg
 Spiel 2 Kaiserslautern-Moenchengladbach
 Spiel 3 Uerdingen-Homburg
...
```

- awk zählt die bisher gelesenen Zeilen und legt die Anzahl in der Variablen NR ab

Formatierte Ausgaben

- `print` ist für schnelle und einfache Ausgaben gedacht
- `printf` ist für formatierte Ausgaben ausgelegt
- die allgemeine Form der `printf`-Anweisung ist:

```
printf(Format,Wert1,Wert2,...,WertN)
```

- `Format` ist als Zeichenkette anzugeben
- `Format` legt fest, wie die die einzelnen Werte (`Wert1`, `Wert2`, ...) ausgegeben werden
- `Format` kann einfache Zeichen und Formatiervorgaben enthalten
- Formatiervorgaben beginnen immer mit % gefolgt von einigen Zeichen
- erste Formatiervorgaben legt fest wie `Wert1` auszugeben ist usw.

```
gkd12ca@cip50:~$ awk '{ printf( " %s spielt gegen %s %3d : %2d \
 \n" , $1, $3, $4, $6 ) }' bundliga.txt
 Muenchen spielt gegen Nuernberg   3 :  2
 Kaiserslautern spielt gegen Moenchengladbach   2 :  1
 Uerdingen spielt gegen Homburg   3 :  0
```

- `Format` enthält vier Formatiervorgaben
- erstes %s $\longrightarrow$ $1 als Zeichenkette ausgeben
- zweites %s $\longrightarrow$ $3 als Zeichenkette ausgeben
- %3d $\longrightarrow$ $4 als ganze Zahl in einem 3 Stellen breiten Feld ausgeben
- %2d $\longrightarrow$ $6 als ganze Zahl in einem 2 Stellen breiten Feld ausgeben
- der Rest von `format` wird unverändert ausgegeben
- \n sorgt für einen Zeilenumbruch (\t entspricht TAB)

```
gkd12ca@cip50:~$ awk '{ printf( " %-20s: %6d Zuschauer; %3d \
 Tore gefallen\n" , $1, $7, $4+$6 ) }' bundliga.txt
 Muenchen             :  34000 Zuschauer;   5 Tore gefallen
 Kaiserslautern       :  28260 Zuschauer;   3 Tore gefallen
 Uerdingen            :  10000 Zuschauer;   3 Tore gefallen
```

- %-20s $\longrightarrow$ $1 linksbündig mit mindestens 20 Stellen ausgeben
- %6d $\longrightarrow$ $7 als ganze Zahl in einem 6 Stellen breiten Feld ausgeben
- %3d $\longrightarrow$ Summe aus $4+$6 als ganze Zahl in einem 3 Stellen breiten Feld ausgeben
- %8.2f $\longrightarrow$ wird für Gleitpunktzahlen verwendet, wobei die Zahl vor dem Punkt die Anzahl an Stellen und die Zahl nach dem Punkt die Anzahl an Nachkommastellen vorgibt

Vergleichsmöglichkeiten in Pattern

- einfache Vergleiche

```
gkd12ca@cip50:~$ awk '$4 > $6 { print $1 }' bundliga.txt
Muenchen
Kaiserslautern
...
```

- Vergleiche mit Berechnungen

```
gkd12ca@cip50:~$ awk '$4+$6 > 3 { printf(" In %s fielen mehr \
 als 3 Tore.\n" , $1) }' bundliga.txt
 In Muenchen fielen mehr als 3 Tore.
 In Frankfurt fielen mehr als 3 Tore.
```

- Textvergleiche

```
gkd12ca@cip50:~$ awk '$1 == "Muenchen" { printf(" Im \
 Olympiastadion fielen %d Tore.\n" , $4+$6) }' bundliga.txt
 Im Olympiastadion fielen 5 Tore.
```

- mehrere Vergleiche verknüpfen

```
gkd12ca@cip50:~$ awk '$4 > $6 && $7 > 20000 { printf(" %s siegt \
 vor mehr als 20000 Zuschauern.\n" , $1) }' bundliga.txt
 Muenchen siegt vor mehr als 20000 Zuschauern.
 Kaiserslautern siegt vor mehr als 20000 Zuschauern.
 Leverkusen siegt vor mehr als 20000 Zuschauern.
 Stuttgart siegt vor mehr als 20000 Zuschauern.
```

- im Unterschied dazu:

```
gkd12ca@cip50:~$ awk ' \
  $4 > $6    { printf(" %s siegt; ", $1) } \
  $7 > 20000 { printf(" mehr als 20000 Zuschauer in %s" , $1) } \
             { printf(" \n") }' bundliga.txt
 Muenchen siegt; mehr als 20000 Zuschauer in Muenchen
 Kaiserslautern siegt; mehr als 20000 Zuschauer in Kaiserslautern
 Uerdingen siegt;
 mehr als 20000 Zuschauer in St.Pauli
 Leverkusen siegt; mehr als 20000 Zuschauer in Leverkusen
 Stuttgart siegt; mehr als 20000 Zuschauer in Stuttgart
 mehr als 20000 Zuschauer in Bochum
 Frankfurt siegt;
 mehr als 20000 Zuschauer in Duesseldorf
```

- logische Operatoren (`&&`(AND), `||`(OR), `!`(NOT)) zur Musterverknüpfung

BEGIN und END

```
BEGIN { aktion }
```

- wird einmal ausgeführt, bevor die erste Eingabezeile bearbeitet wird

```
END { aktion }
```

- wird einmal ausgeführt, nachdem die letzte Eingabezeile bearbeitet wurde
- das Programm in

```
gkd12ca@cip50:~$ wget \
 http://www.geophysik.uni-muenchen.de/~oeser/LV/AWK/ergebnis.awk
gkd12ca@cip50:~$ awk -f ergebnis.awk bundliga.txt
```

- erzeugt folgende Ausgabe

```
gkd12ca@cip50:~$ awk -f ergebnis.awk bundliga.txt
      Heimmannschaft - Gastmannschaft        | Ergebnis
-----------------------------------------------+---------
            Muenchen - Nuernberg               |  3 :  2
      Kaiserslautern - Moenchengladbach        |  2 :  1
           Uerdingen - Homburg                 |  3 :  0
            St.Pauli - Bremen                  |  0 :  0
          Leverkusen - Dortmund                |  1 :  0
           Stuttgart - Karlruhe                |  2 :  0
              Bochum - Koeln                   |  0 :  1
           Frankfurt - Waldhof                 |  3 :  1
         Duesseldorf - HSV                     |  1 :  1
-----------------------------------------------------------
```

Variablen einsetzen

- eigene Variablen müssen nicht extra deklariert werden, sondern werden automatisch angelegt und initialisiert:
  - `0` bei numerischen Variablen
  - `""` bei Zeichenkettenvariablen (leerer String)
- 1. Beispiel:

```
gkd12ca@cip50:~$ awk '$6 > $4  { ausgew = ausgew + 1 } \
 END { print ausgew, " Auswaertsmannschaft(en) haben/hat \
 gewonnen" }' bundliga.txt
1  Auswaertsmannschaft(en) haben/hat gewonnen
```

- `ausgew` ist eine numerische Variable

- 2. Beispiel:

```
gkd12ca@cip50:~$ awk ' { sum = sum + $7 } \
 END { print NR, " Spiele" ; \
       print "Gesamtzuschauer:", sum ; \
       print "Durchschnitt pro Spiel:", sum/NR ; \
     }' bundliga.txt
9 Spiele
Gesamtzuschauer: 224860
Durchschnitt pro Spiel: 24984.4
```

- `sum` ist eine numerische Variable
- `NR` ist eine vordefinierte Variable, die die Anzahl an Zeilen enthält

- 3. Beispiel:

```
gkd12ca@cip50:~$ awk '$7 > max { max =$7; team =$1 } \
 END { print "Die meisten Zuschauer waren in", team, \
 "(" max ")" }' bundliga.txt
Die meisten Zuschauer waren in Duesseldorf (35000)
```

- `max` ist eine numerische Variable
- `team` ist eine Zeichenkettenvariable
- falls mehrere Aktionen zusammen als eine Aktion ausgeführt werden sollen, so müssen diese durch `;` voneinannder getrennt werden

Zeichenketten zusammenhängen

```
gkd12ca@cip50:~$ awk ' \
BEGIN { print "Folgende Mannschaften erlitten \
                Auswaertsniederlage:" } \
       $4 > $6 { verein = verein $3 " " } \
END    { print verein } ' bundliga.txt
Folgende Mannschaften erlitten Auswaertsniederlage:
Nuernberg Moenchengladbach Homburg Dortmund Karlruhe Waldhof
```

- falls die Auswärtsmannschaft verloren hat, so wird deren Name zusammen mit einem Leerzeichen an die Variable `verein` angehängt

Builtin-Funktionen

- viele fest eingebaute Funktionen
  - numerische Funktionen (`sin`, `sqrt`)
  - Zeichenketten-Funktionen (`length`, `substr`)
  - Ein-/Ausgabe-Funktionen (`printf`)
- `length` bestimmt die Länge einer Zeichenkette, im folgenden Beispiel sollen die Anzahl der Zeichen, Wörter und Zeilen bestimmt werden:

```
gkd12ca@cip50:~$ awk ' \
{ nz = nz + length($0)+1 ; \
  nw = nw + NF ; } \
END \
{ print NR, " Zeilen, " , nw, " Woerter, " , nz, " Zeichen " }' \
 bundliga.txt
9 Zeilen, 72 Woerter, 404 Zeichen
```

Auswahlanweisung if-else

```
if ( if-pattern ) { if-aktion }
else if ( else-if-pattern ) { else-if-aktion }
else { else-aktion }
```

- Bedingung in `if`-Anweisung muss geklammert werden
- wenn `if-pattern` erfüllt, so wird `if-aktion` ausgeführt
- ansonsten wenn `else-if-pattern` erfüllt, so wird `else-if-aktion` ausgeführt
- ansonsten wird `else-aktion` ausgeführt
- `else if` und/oder `else` sind optional

- folgendes Beispiel gibt die Totozahlen aus:

```
gkd12ca@cip50:~$ awk ' \
BEGIN { print "Die Totozahlen sind:" } \
      { if ($4 > $6) \
          printf("1  ") ; \
        else if ($4 == $6) \
          printf("0  ") ; \
        else \
          printf("2  ") ; } \
END   { printf("\n") } ' bundliga.txt
Die Totozahlen sind:
1  1  1  0  1  1  2  1  0
```

Wiederholungsanweisungen

- für die wiederholte Ausführung von `aktion` existieren die:
  - `while`-Anweisung
  - `for`-Anweisung
  - `do-while`-Anweisung

```
while ( pattern ) { aktion }

for ( pattern1; pattern2; pattern3 ) { aktion }

do { aktion } while ( pattern )
```

- Ausführung von `aktion` solange wie Auswertung von `pattern` `WAHR` ergibt

- Beispiel 1:
  - Berechnung der Quersumme:

```
gkd12ca@cip50:~$ cat while.awk
{  i=1
   q_summe = 0
   while ( (zahl=substr($1,i++,1)) != "" )
      q_summe += zahl
   print " Quersumme( " $1 " )= " q_summe
}
gkd12ca@cip50:~$ awk -f while.awk
125
Quersumme(125)=8
```

- Beispiel 2:
  - Berechnung der Summe aus mehreren Zahlen und deren Prozentanteil an der Summe:

```
gkd12ca@cip50:~$ cat for.awk
    { zahl[NR] = $1
      summe += $1 }
END { if (summe)
        for (i=1 ; i<=NR ; i++)
           print zahl[i], "(" 100*zahl[i]/summe " %) "
      print " Summe: " summe }
gkd12ca@cip50:~$ echo -e "12\n45" | awk -f for.awk
12 (21.0526 %)
45 (78.9474 %)
 Summe: 57
```

Anwendungsbeispiel 1 aus der Geophysik

- 3 Zeilen Skript für Extrahierung und Darstellung von Messdaten
- das tar-Archiv GMT-Magnetik.tar.gz enthält alles notwendige

```
gkd12ca@cip50:~$ tar xzf GMT-Magnetik.tar.gz
gkd12ca@cip50:~$ cat Magnetik/Script-Magnetik.sh
#!/bin/bash
awk ' if ( NR > 2 && $9 > 98 ) print $1, $2, $7 ' \
 FUR_2006-12-01.csv > FUR_2006-12-01.xyz

GMT nearneighbor -R11.273/11.276/48.162/48.165 -I0.5c \
 -S5c -GFUR_2006-12-01.nc -V FUR_2006-12-01.xyz

GMT grdimage FUR_2006-12-01.nc -R -JM6i -P -B0.001 \
 -CFUR.cpt | gv -
```

Anwendungsbeispiel 2 aus der Geophysik

- Skript zur Darstellung von tektonischen Messdaten
- das `tar`-Archiv GMT-Tectonic.tar.gz enthält alles notwendige
- Vorsicht: 91 MB Dateigröße
- `Script-tectonic.sh` enthält unter anderem folgenden `awk`-Befehl:

```
awk '
 function acos(x) { return atan2((1.-x^2)^0.5,x) }
 function asin(x) { return atan2(x,(1.-x^2)^0.5) }
 {
 pi = atan2(0,-1)
 lat = $2; lon = $1
 alpha = $4*pi/180
 a = $3*'${crosssize}'
```

```
lat_right = 90 - acos(cos(a)*cos((90 - lat)*pi/180) \
 + sin(a)*sin((90 - lat)*pi/180)*cos(alpha))*180/pi

lon_right = lon + asin(sin(a)/sin((90-lat_right)*pi/180) \
 * sin(alpha)) * 180/pi

lat_left = 90 - acos(cos(a)*cos((90 - lat)*pi/180) \
 + sin(a)*sin((90 - lat)*pi/180)*cos(alpha-pi))*180/pi

lon_left = lon - asin(sin(a)/sin((90-lat_right)*pi/180) \
 * sin(alpha)) * 180/pi
}
{
printf(" > -Z%.2f\n %9.5f %9.5f\n %9.5f %9.5f\n %9.5f %9.5f\n" \
 , a, lon_left, lat_left, lon, lat, lon_right, lat_right) \
} ' phi_shear.xysp > dir1
```

Generic Mapping Tools

- GMT-Projekt begann 1987 durch die Arbeit der Masterstudenten Paul Wessel und Walter H. F. Smith am "Lamont-Doherty Earth Observatory" (Version 4.5.9 vom 01.01.2013)
- GMT folgt der Unix-Philosophie der kleinen, auf eine bestimmte Aufgabe spezialisierten Programme (kommandozeilenbasiert, hardwareunabhängig, plattformunabhängiges PS-Dateiformat)
- GMT-Befehle können leicht in Skripten verwendet werden

Dokumentation zu GMT

- „Technical Reference and Cookbook“
  - im Internet
  - auf Festplatte
- „A Map-making Tutorial“
  - im Internet
  - auf Festplatte
- „GMT Online Man Pages“
  - im Internet
  - auf Festplatte
- „GMT Supplemental Online Man Pages“
  - im Internet
  - auf Festplatte

Erste Schritte mit GMT

- im folgenden werden ausgewählte Abschnitte aus dem Dokument „A Map-making Tutorial“ bearbeitet werden:
  - Abschnitt 1: Grundlagen und Kartenprojektionen
  - Abschnitt 2: `psxy`, `pscoast` und `pstext`
  - Abschnitt 3: „Gridding“ mit `grdcontour`, `xyz2grd`, `nearneighbor` und `surface`
  - Abschnitt 4: Farbe und Perspektive
- eine gute Übersicht liefert ebenfalls der Artikel zu GMT im Linux-Magazin 05/2010 „Auf Mercators Spuren“ (siehe Kopie)
- einige Beispiele aus Geophysik und Geowissenschaften sind zusammengetragen unter:
  - http://www.geophysik.uni-muenchen.de/~oeser/LV/GMT/

Tutorialvorbereitungen

- öffnen Sie im Browser die Datei `/usr/share/doc/gmt/html/GMT_Tutorial.html`
- geben Sie im Terminal folgende Befehle ein:

```
gkd12ca@cip50:~$ cp -r /usr/share/doc/gmt-tutorial/tutorial/ .
gkd12ca@cip50:~$ cd tutorial/
gkd12ca@cip50:~/tutorial$ gunzip *.gz
gkd12ca@cip50:~/tutorial$ ls
bermuda.grd data quakes.cpt quakes.ngdc ship.xyz topo.cpt us.grd
```

- Achtung: immer die GMT-Befehl mit dem Wrapper-Skript `GMT` aufrufen

```
gkd12ca@cip50:~$ GMT pscoast ...
gkd12ca@cip50:~$ GMT psxy ...
```

Python

- https://de.wikipedia.org/wiki/Python_(Programmiersprache)
- Material zu Python ist zusammengetragen unter:
  - http://www.geophysik.uni-muenchen.de/~oeser/LV/Python/
  - bitte `.tar.gz`-Dateien lokal speichern
- Anaconda ist einefür die wissenschaftliche Nutzung vorbereitete Python Distribution
  - Installationsanleitung: https://www.continuum.io/downloads
  - Python 2.7 Version für 64-bit GNU/Linux im CIP Pool notwendig

Vorbereitungen für Python

```
gkd12ca@cip50:~$ mkdir /data/$USER && cd /data/$USER
gkd12ca@cip50:~$ wget \
 https://repo.continuum.io/archive/Anaconda2-4.2.0-Linux-x86_64.sh
gkd12ca@cip50:~$ chmod u+x Anaconda2-4.2.0-Linux-x86_64.sh
gkd12ca@cip50:~$ ./Anaconda2-4.2.0-Linux-x86_64.sh
Installationspfad im CIP Pool:
 /data/gkd12ba/anaconda2
gkd12ca@cip50:~$ conda config --add channels conda-forge
gkd12ca@cip50:~$ conda config --add channels obspy
gkd12ca@cip50:~$ conda install basemap basemap-data-hires obspy visvis
gkd12ca@cip50:~$ tar -xzf Python*.tar.gz
gkd12ca@cip50:~$ cd PythonIntro
gkd12ca@cip50:~/PythonIntro$ jupyter-notebook
gkd12ca@cip50:~$ rm -rf /data/$USER # AUFRÄUMEN am ENDE
```