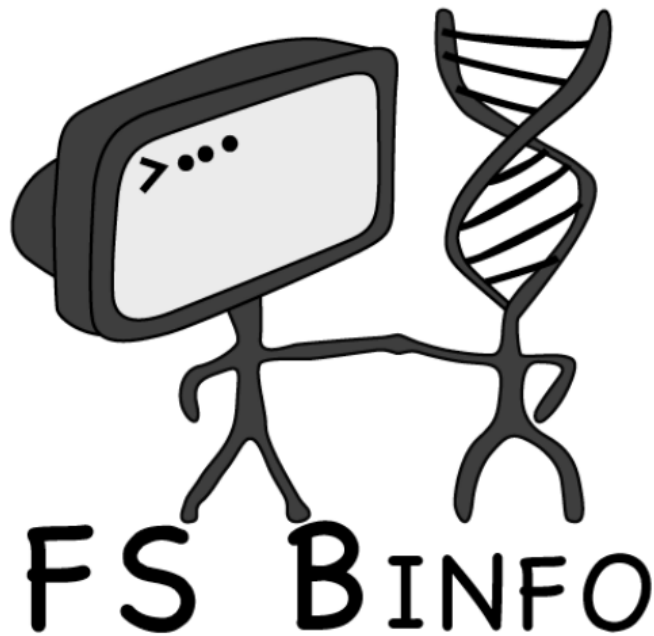


# Shell: Dateimanipulation, weitere Befehle



Hans Hawe

Fachschaft Bioinformatik

19.05.2011

(Folien basierend auf Tutorium  
von Bernhard Schauburger)

# Zeitplan

- 12.05. Einführung in Linux und die Shell
- 19.05. **Arbeiten mit Dateien auf der Shell**
- 26.05. Kommandozeileneditor: vim
- 09.06. Einführung in Java und Eclipse
- 16.06. Konditionen, Arrays und Schleifen
- 30.06. Rekursion und Iteration
- 07.07. Objektorientierte Programmierung
- 14.07. File-I/O und Wiederholung
- 21.07. Vererbung, Tipps und Fragen
- 25.01. Quiz, Evaluation, Fragen

# Kontakt

Website:

<http://www.bioinformatik-muenchen.com/bioinfocom/informatik-tutorium>

Feedback, Fragen etc. gerne an: [j.hawe@campus.lmu.de](mailto:j.hawe@campus.lmu.de)

# Letzte Woche, Übungen

## Fragen?

Übungen unklar? Lösungen gibt es auch im Netz, auf der Seite des Tutoriums.

## Praxis!

Übungen machen, selber ausprobieren! Z.B.

- hier im CIP-Pool, oder
- von zu Hause aus per **SSH** (Linux: *ssh* in Console, Windows: *putty*)
- SSH muss zuerst **freigeschalten** werden, siehe:  
<https://tools.rz.ifi.lmu.de/cipconf/> (Remote Enabler)

# Suchen von Dateien

- Befehl um nach Dateien zu suchen: *find*
- Synopsis: *find VERZEICHNIS -name MUSTER*

## Übung

- 1) Finde alle README Dateien in “/usr/share/doc”
- 2) Finde alle versteckten Verzeichnisse in deinem HOME-Verzeichnis
- 3) Finde alle leeren Verzeichnisse in deinem HOME-Verzeichnis

# Komprimierte Dateien I

- GZIP und BZIP2 sind Komprimierungsverfahren wie ZIP; viele Programme fuer Linux sind nur so verfuegbar
- (Ent)packen nur einzelne Dateien, mehrere Dateien oder Verzeichnisse müssen zu einer Datei (tar-Ball) zusammengefasst werden
- **GZIP:**
  - Entpacken: *gunzip ARCHIV.gz;*  
für tar-Balls: *tar xzvf ARCHIV.tar.gz*
  - Packen: *gzip ARCHIV.gz DATEI;*  
mehrere Dateien: *tar czvf ARCHIV.tar.gz DATEIEN*
- **BZIP2:**
  - Entpacken: *bunzip2 ARCHIV.bz2;*  
für tar-Balls: *tar xjvf ARCHIV.tar.bz2*
  - Packen: *bzip2 ARCHIV.bz2 DATEI;*  
mehrere Dateien: *tar cjvf ARCHIV.tar.bz2 DATEIEN*

# Texteditoren

- **nano**: Sehr einfacher Editor
- **kate/gedit**: Gute graphische Editoren
- **vim**: Sehr umfangreich und gewöhnungsbedürftig  
(Konsole, mehr im nächsten Tutorium)
- **emacs**: wie vim, nur graphisch

(alle Editoren gut für quick-n-dirty-Aktionen geeignet, bei guter Einarbeitung auch als einfache Entwicklungsumgebung geeignet)

## Übung

- 1) Erstelle mit einem Editor Deiner Wahl eine Datei *hallo* mit Inhalt  
“Hallo Welt”
- 2) Öffne die Datei erneut und ersetze den Inhalt durch etwas  
*sinnvolleres*

# Komplexe Probleme

## Frage

In der letzten Stunde habt ihr einige einfache Befehle kennen gelernt (ls, cd, cp, mv, rm, ...). Aber wie löst man damit komplexere Probleme?

## Grundlegende Idee

Jedes Programm hat genau eine Funktion (z.B. sortieren, filtern oder anzeigen).

Komplexe Probleme werden durch **Verkettung** einfacher Befehle gelöst.

# Verkettung von Befehlen

Jedes Programm hat einen Eingabekanal und zwei Ausgabekanäle (STDIN, STDOUT, STDERR). Diese kann man **umleiten** – und damit genau die Verkettung von Befehlen erreichen.

## Die Pipe “|”

“|” verbindet **STDOUT** eines Programms mit **STDIN** des nächsten.

Beispiel: `ls | less` (zeigt Ausgabe von ls mit less an)

## > und <

- “>” verbindet **STDOUT** mit einer (beliebigen) **Datei**
- “<” verbindet **STDIN** mit einer **Datei**

Beispiel: `find / -name "*.mp3" > ergebnis.txt`

Fehler werden aber angezeigt, da **STDERR nicht** umgeleitet wird.

# Weitere Programme

- **sort** [-n]: Sortiert Daten alphabetisch (numerisch)
- **grep** *MUSTER*: Filtert die Daten nach *MUSTER*
- **head**: Zeigt den Anfang einer Datei an
- **tail**: Zeigt das Ende einer Datei an
- **wc**: Zählt Buchstaben, Wörter Zeilen

Die Daten für diese Programm kommen dabei in der Regel aus der STDIN, also per Pipe

## Übung

- 1) Finde in der Datei “/etc/passwd” deine eigene UserID.
- 2) Wieviele Benutzer haben “bash” als Standardshell?
- 3) Gib die ersten 20 Dateien in “/usr/lib” alphabetisch aus!

# Berechtigungen I

## Idee

Für ein Mehrbenutzersystem wie Linux muss es natürlich auch geregelte Dateizugriffe geben; realisiert durch Berechtigungen.

Dazu gibt es 3x3 Stufen der Berechtigungen:

wer [u(ser), g(roups), und a(lle)] und

was [lesen:4,schreiben:2, ausführen 1].

## Berechtigungen ändern

Dazu gibt es Befehl `chmod BERECHTIGUNG DATEI`; man muss dabei angeben, für wen welche Rechte gelten sollen (rel./abs.).

- `chmod g+w DATEI`: für die Gruppe zum Schreiben freigeben
- `chmod +r DATEI`: für alle Benutzer zum lesen freigeben
- `chmod 644 DATEI`: für alle zum Lesen und für den User zum Schreiben freigeben

# Berechtigungen II

## Weitere Befehle

- `chgrp GRUPPE DATEI` Ändert die Gruppenzugehörigkeit einer Date
- `chown BENUTZER DATEI` Ändert den Besitzer der Datei
- `ls -la DATEI` Zeigt die Berechtigungen einer Datei an

## Übung

- 1) Erstelle eine Datei `test` und schränke die Rechte so ein, dass nur du vollen Zugriff hast.
- 2) Erstelle einen Ordner und lasse nur dich und die Gruppe `bioinfo` darauf lesen und schreiben

**Fragen?**

# Übungen für Zuhause

## Übung

- 1) Versuche dich in vim oder emacs einzuarbeiten (wird auch im nächsten Tutorium behandelt), Links:
  - <http://www.informatikserver.at/selflinux/html/vim.html>
  - bzw.
  - <http://www.linuxhaven.de/dlhp/HOWTO/DE-Emacs-Einsteiger-HOWTO-2.html>
- 2) Erstelle einen Ordner “gruppenarbeit” und gib ihn nur für Deine Gruppe (stud) frei
- 3) Kopiere den Ordner “/usr/share/doc/zsh” in den neu erstellten Ordner
- 4) Komprimiere den Ordner zu zsh.tar.gz und entpacke ihn wieder nach zsh-neu
- 5) Durchsuche den entpackten Ordner nach allen .gz Dateien und lass die Anzahl ausgeben.