```
pdftk file1.pdf file2.pdf anhaenge*.pdf ouput ergebnis.pdf
pdftk original.pdf cat 17 155-158 2 320-end ouput auswahl.pdf
```

pdftotext kann den text aus einem PDF heraushohlen. (allerdings nur wenn der text im PDF vorhanden ist. Mit einem PDF das nur aus einem Bild besteht geht das nicht. Hier kann aber ein OCR tool helfen: **tesseract**

15.5 LaTeX

Wer eine professionele LaTeX- Umgebung benötigt ist mit Linux gut bedient. Da ist eine komplette latex-Installation mit allen tools enthalen.

15.6 Mathematik

octave ein Klon der kommerziellen MatLab umgebung. Ideal zum für Berechnungen aller Art.

sympy symbolische Mathematik

R "das" Statistikpaket

15.7 Programmierung

gcc, perl, python, ...- Die meisten Programmiersprachen sind gut im Linux integriert.

16 CLI basics

16.1 Der BASH Prompt

```
karli@meinlaptop:~$
```

In einer Text-Console wird man/frau üblicherweise mit einem ''Prompt" begrüst. Der könnte z.b. So aussehen:

Mit der Eingabetaste bekommt man einen neuen Prompt. Der Prompt lässt sich konfigurieren. Üblicherweise ist er so eingestellt dass wir sehen unter welchem User wir arbeiten, auf welchem System wir arbeiten (könnte auch eine sein, dass am anderen Ende der Welt steht) und das aktuelle Verzeicnis. ein $ sagt uns dass wir "normaler User" sind. Ein # ist i.a. für den root User.

Es gibt verschiedene Shell Programme die uns die Kommandoeingabe ermöglichen. Default im Linux ist die **bash**. Alternativen sind **tcsh** (default bei Mac OS/X). Beliebt ist auch die **zsh**.

16.2 Bewegen im Verzeichnisbau

Mit den Befehlen `pwd, cd, df, ls ls -l, ls -ltr` (siehe voriges Kapital) können wir uns im Verzeichnisbaum bewegen und uns Datein und Verzeichnisse ansehen.

16.3 keyboard shortcuts für die BASH

ctrl-p ctrl-n cursor up/down

history der history befehl zeigt die zuletzt eingegen Befehle.

ctrl-c Abbruch des aktuellen befehls.

ctrl-a zum Zeilenanfang

ctrl-k löschen bis zum Ende der Zeile

ctrl-r "**r**everse incremental search" - suche in der history durch Eingabe von Buchstaben.

ctrl-s stopped die ausgabe am Terminel.

ctrl-q ausgabe wieder erlauben. (Continue).

ctrl-z stopped den gerade laufenden Befehl. Duch Eingabe von **bg** wird dieser in Hintergrund geschickt und läuft dort weiter. Damit ist die shell wieder frei zur Eingabe weiterer Befehle. Will man/frau einen Befehl gleich im Hintergrund starten geht das mit einem & am Ende des Befehls. z.B.: `sleep 30 &`. Mit **fg** kann ein im Hintergrund laufener Prozess wieder in den Vordergrund geholt werden. **jobs** zeigt alle im Hintergrund laufenden jobs der aktuellen shell. (Siehe: Kapitel 16.10, Seite 18 - für das Verwalten aller jobs, nicht nur derer aus der aktuellebn shell).

ctrl-d ist das "Datei-Ende" und wird auch von vielen Programmen bentutzt die Eingabe lesen, um diese Eingabe zu Beenden. Mit dem Befehl **exit** oder mit einem **Ctrl-D** kann eine shell geschlossen werden.

TAB die Tabulator Taste dient zur Verfollständigung von Eingaben. Einmal gedrückt macht sie, falls möglich, eine Verfollständigung: Zweimal gedrückt macht sie Vorschläge. Normalerweise werden Datei- und Verzeichnisnamen und die Namen eigegebener Programme vervollständigt. Inzwischen sind aber auch Bash-Plugins gängig die, z.B. die gängigen Befehlsoptionen von Befehlen kennen und auch Vervollständigen können. Damit ist die Eingabe von Befehlen sehr komfortabel und schnell.

16.4 Wer ist eingelogt?

who und **w** zeigen dir wer gerade aller am selben Computer eingelogt ist. **last** zeigt dir wann du zuletzt eingeloggt warst.

16.5 Diverse kleine, nüztliche Tools

cal zeigt dir einen aktuellen Kalender. `cal 1 2025` zeigt dir den Kalender von Jänner 2025.

gcal ist ein erweiterter Kalender. `gcal -n -qAT` zeigt dir eine Liste aller österreichischen Feiertage im aktuellen Jahr.

date - zeigt Datum, Uhrzeit und Zeitzone an.

factor - Primfaktoren Zerlegung. Das geht erstaunlicherweise bis hin zu relativ großen Zahlen: `factor 380708481735525870694501390857711137 93`

ping schickt ein ping Packet zum angegeben Server.

echo gibt die angegeben Argumente wieder aus. z.B.: `echo hallo da`

wc zählt Zeichen, Wörter und Zeilen in einer Datei

seq macht eine Zahlensequenz: z.b: [seq 1 10] die Zahlen 1 bis 10.

md5sum rechnet eine Prüfsumme über den Inhalt der angegebenen Dateien. (praktisch zum sehen ob die gleich sind wie wo anders).

16.6 Ausgabeumleitung in und von Dateien

Das großer-Zeichen: > leitet die Ausgabe eines Befehls, die ansonsten am Bildschrim landen würde (auch STDOUT genannt) in ein File um.
z.B.:

```
date > heute.txt
```

Würde die Ausgabe vom `date` Befehl in die Datei namens "heute.txt" schreiben. **Achtung:** Falls heute.txt schon existiert wird es kommentarlos überschrieben!

Den Inhalt der neuerstellten Datei können wir uns mit einem Editor oder mit dem `cat` Befehl ansehen: z.B.: `cat heute.txt`

Cat gibt ein oder mehrere Datein am Bildschirm aus. Natürlich lässt sich auch die Ausgabe von cat wieder mit > umleiten. Das Minuszeichen − wird sehr oft an Stellen benutzt an denen ein Filename verwendet werdekn kann um statt dessen die Eingabe (also die Zeichen die wir gerade eintippen, auch STDIN genann) zu lesen.

`cat - > test.txt` liest die Eingaben die wir tippen und schreibt sie in eine Datei namens text.txt. (Beenden der Eingabe mit **Ctrl-D**).

Analog dazu macht das Kleiner-Zeichen < eine Umleitung einer Eingabe. Ein Programm das normalerweise eine Eingabe lesen würde bekommt nun die daten aus einer Datei. z.B:

```
seq 1 1000000 > mille.txt
wc < mille.txt
```

Die Zeile mit dem `seq 1 1000000 > mille.txt` erzeugt eine Datei namens mille.txt mit den Zahlen 1 bis 1000000 als Inhalt. (Eine Zahl pro Zeile). Diese Dateil leiten wir indas `wc` Tool um zu zählen wieviele Zeilen, Wörter und Zeichen dort drinnen sind.

Verwendet man statt einem > zwei >> so bedeutet das bei der Ausgabeumleitung, dass man eine eventuell vorhandene Datei nicht überschreiben will sondern den Inhalt am Ende **anfügen** will.

```
echo noch eine zeile >> mille.txt
```

macht unsere mille.txt Datei noch um eine Zeile länger.

Fehlermeldungen werden von den meisten Tools nicht über die STDOUT Ausgabe ausgegeben sondern über einen einen Kanal "gesndet". Der wird STDERR genannt und hat im unix die nummer 2. Will man dies Ausgabe auch umleiten dann Verwendet man $\boxed{2>}$. Soll die STDERR Ausgabe wie eine normale Ausgabe behandelt werden so können wir das mit dem Kürzel $\boxed{2>\&1}$ machen.

z.B.:

```
ls /etc /gibtsnicht  >bla.txt   2>&1
```

16.7 wildcards in der shell

* und ? haben in der shell eine spezielle Bedeuting. Sie sind "Wildcards" in Datei- und Verzeichnisnamen. Ein Stern * wird durch ein oder mehrere Beliebige Zeichen ersetzt. Ein Fragezeichen ? wird durch genau ein beliebiges Zeichen ersetzt:

cat *.txt gibt alle files aus dem aktuellen Verzeichnis, die auf .txt enden aus.

gimp bild?.jpg ruft das Programm **gimp** auf und übergibt alle Dateien die mit bild beginnen und dann genau ein zeichen haben. Also bild1.jpg und bildx.jpg aber **nicht** bild22.jpg.

16.8 Dateien kopieren, verschieben und verlinken

cp steht für copy. `cp quelle.txt ziel.txt` würde die Dateie quelle.txt auf die datei zeil.txt kopieren. (**Achtung:** falls die Datei ziel.txt schon existiert wird sie überschrieben). Ist das letzte Argument ein Verzeichnis so werden die Files (es können in diesem Falle mehrerer sein) mit dem selben Namen ins Zielverzeichnis kopiert. (Wenn mehr als 2 Argumente angegeben werden, so muss das letzte Argument ein Verzeichnis sein). cp hat viele Optionen u.a. die Option -r für "rekursiv", falls Verzeichnisse samt aller Unterordner kopiert werden sollen.

mv funktiniert ähnlich wie cp, aber es wird nicht kopiert sondern verschoben (move).

ln -s erstellt einen, so genannten "sym-link". Das schaut aus wie eine Datei ist aber nur ein Verweis auf eine Datei oder ein Verzeichnis. Das ist praktisch um sich z.b. "abkürzungen" zu erstellen. z.B.: `ln -s /usr/share/sounds Desktop/klaenge` würde einen Link namens "klaenge" im Verzeichnis Desktop erstellen das auf den Systemordner zeigt in dem Linux viele Audiofiles abgelegt hat.

16.9 Hilfe!

Neben Doktor google, gibt es auch direkt im Linux viel eingebaute Hilfe.

Viele Befehle haben direkt Hilfe eingebaut. Mit den Optionen **-h** –**help** oder **-help** (leider nicht einheitlich) wird eine kurze Hilfe angezeigt. Diese Hilfe ist meist nur kurz und listet meist nur die Optionen auf.

Der **help** Befehl zeigt Hilfe zur aktuellen shell (BASH) an. **help echo** würde z.B. zum (intern in der shell eingebauten) echo Befehl anzeigen. Das funktioniert aber nur bei Befehlen die direkt, intern in der shell eingebaut sind. Für die meisten Befehle brauchen wir daher eine andere Hilfe:

man ist der Befehl zum aufruf der **man**ual Seiten. Die meisten Befehle haben solche man-pages.

`man echo` würde z.b. die man page zum echo Befehl ausgeben (der neben dem eingeboten echo kommando ebenfalls existiert).

man verwendet zur Anzeige der Seiten einen so genannten “Pager”. Da viele man-pages sehr lang sind kann man/frau damit in den Pages blättern.

Hier die wichtigsten Keyboard-Shortcuts für den standard Pager. Der standard-Pager im Linux ist “less” und ist eine weiterentwicklung von “more”.

Leertaste zum weiterblättern.

q beenden

Pfeiltasten oder j k um jeweils eine Zeile nach unten oder oben zu verschieben.

/suchstring sucht nacht “suchstring”. (Wobei “suchstring” eine so genannte “regular expression” ist - mehr dazu später).

n die vorgehende Suche wiederhohlen (weitersuchen -**n**ext).

N weitersuchen nach oben (also in die andere Richtung).

Auf einer grafischen oberfläche kann auch das Gnome-Help tool “yelp” verwendet werden: z.b.: `yelp man:ls`

Viele GNU Tools haben neben einer man-page auch eine “info” page. **info** ist etwas mühsam zu bediene, aber kann dafür Links zu anderen info Seiten enthalten. Eine frühe Form von “hypertext”.[10]

16.10 Prozessmanagment

Um zu sehen was auf deinem Linux gerade so läuft:

```
ps uax
```

Das **ps** Komamndo zeigt dir die aktuell laufenden Prozesse im System. Die Optionen “uax” und “elf” werden oft verwendet um mehr info zu sehen als ps alleine annzeigt. Das **pstree** Tool zeigt eine Baumstruktur der Prozesse an (wer welchen gestartet hat).

Oft ist interessant zu sehen wer gerade die meiste CPU verbraucht. Dafür ist **top** nützlich. top ist ein interaktives tool und listet die Prozesse nach ihrer CPU Auslastung. (Alternativ wird nach Memory bedarf sortiert: nach dem Starten von top M (groß) drücken). Ein kleines q (quit) oder ein Ctrl-c beendet das Tool.

Alle Prozesse haben eine Nummer. Diese kann bentutz werden um die Prozesse zu steuern. z.B. um ihnen zu sagen sie sollen sich beenden:

`kill 1234` würde dem Prozess mit der nummer 1234 sagen, er soll sich bitte beenden. Notfalls muss das Betriebsystm mit dem Holzhammer nachhelfen: `kill -9 1234` hilft dann.

16.11 CLI für Vortgeschrittene

Die bisher gelernten Befehle reichen für einfache Aufgaben. Wer etwas mehr in der Shell machen will benötigt aber noch einige extrams. Hier die wichtisten davon:

16.12 Mit ssh auf einem anderen Rechner arbeiten

Mit

```
ssh username@hostname
```

kann eine verschlüsselte Konsole Verbindung zu einem anderen Linux Rechner aufgebaut werden. Dort kann man/frau dann so arbeiten wie am lokalen Rechner. Mehr dazu im eigenen SSH Kapitel.

16.13 Dateien suchen mit “find”

Find haben wir schon kurz erwähnt. Ohne Argumente listet es alle Datei und Verzeichnisnamen ab dem aktuellen Verzeichnis auf. Ein Argument wird als Verzeichnis gelesen, ab dem gesucht werden soll. z.B.:

```
find /etc
```

Die Liste ist dabei meist recht lang. Um spezifischer suchen zu können gibt es sehr sehr viele Argumente im find. Hier einige der wichtigsten um die Suche einzuschränken:

[10] Der KDE Eigene Webbrowser konqueror erlaut die Eingabe von `man:` und `info:` URLs im Browser die direkt zu info und man Seiten führen. z.B.: `info:seq`