Linux Einführung

Fachschaft Elektrotechnik und Informationstechnik
Technische Universität München

Tagesordnung

1. Was ist Linux?

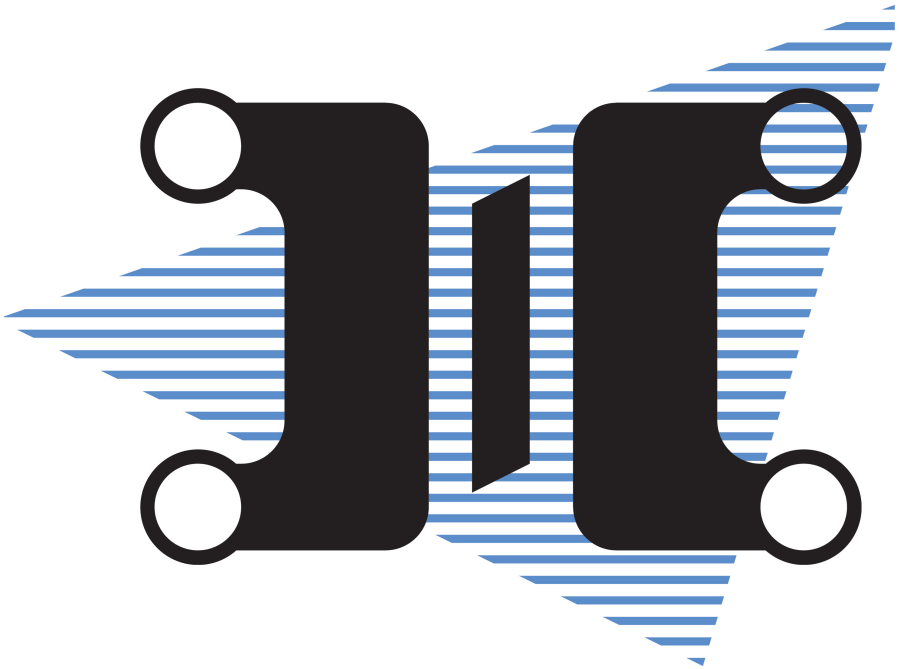
2. Linux - Der größte Lego-Bausatz der Welt

- Distributionen
- Graphische Oberflächen

3. Linux 101 - Shell und Bash Befehle

- Wie verbinde ich mich mit dem Eikon?
- Orientierung
- Navigation auf der Shell
- Nützliches
- Manipulation von Dateien
- Tipps, Tricks und mehr Befehle

4. Bash-scripting



Was ist Linux?

- Ein Betriebssystem (OS) wie MS Windows oder macOS
- Von Linus Torvalds entwickelt in 1991
- Wird von seinen Nutzern weiterentwickelt $\Longrightarrow$ sowohl große Firmen als auch die Linux-Community
- Beliebtes OS für Server, Embedded Systems **und** Supercomputer
- Das *Linux-Terminal* ist ein mächtiges Werkzeug für Admins aller Art
- Verkörpert die Philosophie von Unix und FOSS
- Software wird über einem **Packet-Manager** installiert (ähnlich wie ein App-Store)

Linus Torvalds

Tux der Penguin, das Linux Maskottchen

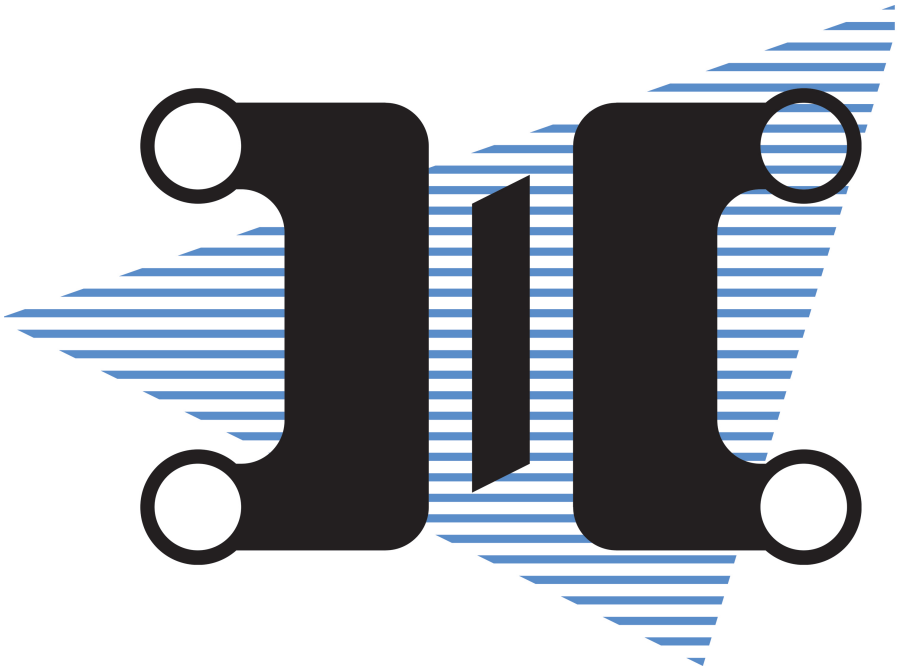
Tagesordnung

1. Was ist Linux?

2. **Linux - Der größte Lego-Bausatz der Welt**
   - Distributionen
   - Graphische Oberflächen

3. Linux 101 - Shell und Bash Befehle
   - Wie verbinde ich mich mit dem Eikon?
   - Orientierung
   - Navigation auf der Shell
   - Nützliches
   - Manipulation von Dateien
   - Tipps, Tricks und mehr Befehle

4. Bash-scripting



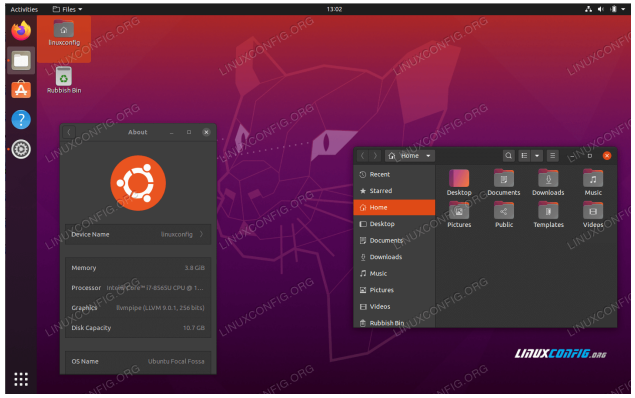
Distributionen

- Umgangssprachlich: *Distros*
- Distributionen sind Ansammlungen von Applikationen, die auf dem Linux-Kernel aufbauen
- Distros haben unterschiedliche Ziele und Packet-Manager
- Beliebte Distros:
  - Ubuntu
  - Debian
  - Arch
  - Fedora
- Distro-Watch listet sämtliche Distros auf: `https://distrowatch.com/`
- Man kann Distros als *Live*-System oder auf `https://distrotest.net` ausprobieren

debian ubuntu

Graphische Oberflächen

- Linux hat eine graphische Oberfläche, ähnlich wie MS Windows oder MacOS
- Es gibt verschiedene FOSS- (*Free and Open-Source Software*) Projekte für die graphische Oberfläche
- Einige beliebte graphische Oberflächen sind, z. B.
  - Gnome
  - KDE-Plasma
  - XFCE4
  - Cinnamon



Gnome auf Ubuntu

KDE-Plasma

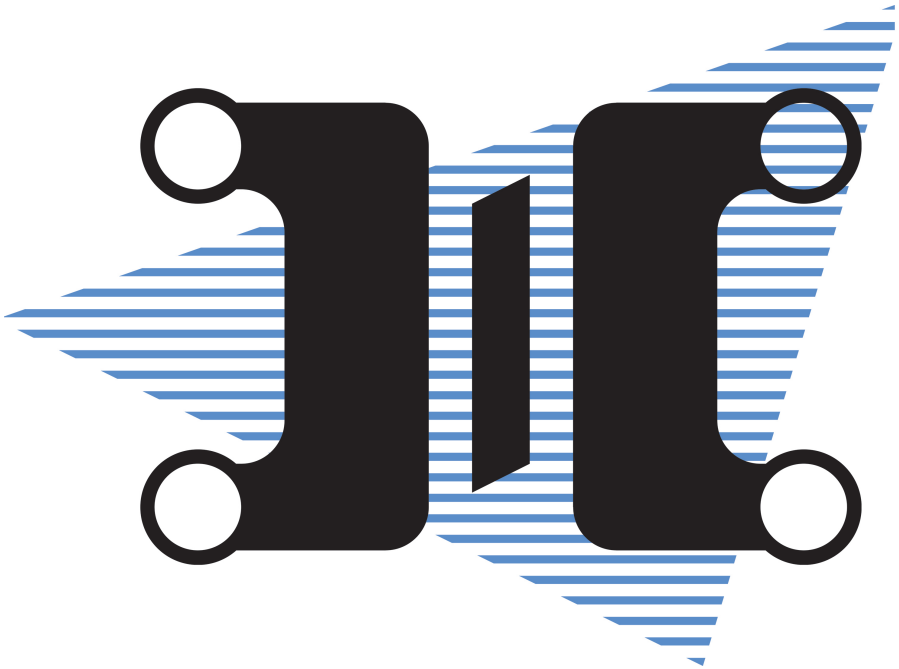
Tagesordnung

1. Was ist Linux?

2. Linux - Der größte Lego-Bausatz der Welt
   - Distributionen
   - Graphische Oberflächen

3. Linux 101 - Shell und Bash Befehle
   - Wie verbinde ich mich mit dem Eikon?
   - Orientierung
   - Navigation auf der Shell
   - Nützliches
   - Manipulation von Dateien
   - Tipps, Tricks und mehr Befehle

4. Bash-scripting



Wie verbinde ich mich mit dem Eikon?

- Der Eikon-Turm stellt 90 Linux-Rechner zur Verfügung (nummeriert von 00 bis 89)
- Man kann sich mit dem `ssh`-Befehl (*secure shell*) von Zuhause aus verbinden
- **Per SSH verbinden:**
  1. Terminal App öffnen: *PowerShell* (Windows) oder *Terminal* (Linux, macOS)
  2. Folgenden Befehl eingeben:
     `ssh <tum-Kennung>@linux[0-8][0-9].clients.eikon.tum.de`
  3. ECDSA Fingerprint des Eikon Rechners akzeptieren: 'yes' eingeben
  4. Passwort eingeben
  5. Fertig!
- **Fun Fact:** Windows konnte bis 2018 nicht mit seinen Standardprogrammen ssh-Verbindungen zu anderen Computern aufbauen. Sollte euch das betreffen, installiert und nutzt PuTTY:
  `https://www.putty.org/`

  Alternativ könntet ihr auch ein Smartphone nutzen (z. B. mit JuiceSSH auf Android)

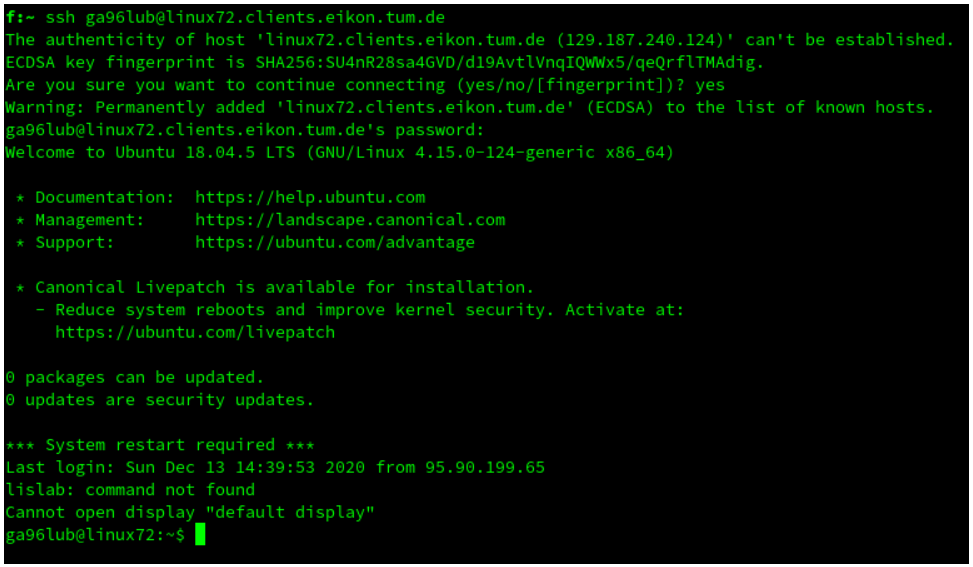
Orientierung - Ankunft

Was seht ihr?

- Meldungen eures Loginprozesses
- Statusmeldung des Systems, auf das ihr euch eingeloggt habt
- „Prompt“ - die Eingabeaufforderung:
  `[example@linux42 ~]$`

  Bedeutung:
  `[<user>@<host> <current directory>]$`

  Der Prompt ist immer am unteren Ende der Konsole.
  Wenn der Prompt sichtbar ist, ist das System bereit für eure Befehle.



Orientierung - Was ist die Shell?

- Textbasiertes Interface des Betriebssystems
- Ermöglicht Eingabe von Befehlen/Kommandos
- Eingaben werden mit `Enter` bestätigt
- Verschiedene Versionen möglich, meist `bash`, beliebte Alternativen sind zsh, sh, fish, ...

Erste Interaktionen
Haut beliebig auf eure Tastatur und drückt dann die `Enter`-Taste.
Was sagt euch das Terminal?

Erste Schritte - Wie sehe ich meine Umgebung?

Wir sind irgendwo in einem fremden PC. Was sehen wir um uns?

Probiert die folgenden Befehle aus:

`ls`	**ls**t directories and files
`ls -l`	option `-l`: langes Format, mehr Infos
`ls -a`	option `-a`: alle Files
`ls -lh`	option `-h`: human readable
`ll`	alias für `ls -alF`

Probiert, den Inhalt eines Ordners anzuschauen:

```
ls .config
```

Erste Schritte - Was tut ein Befehl?

Wenn ihr mehr über einen Befehl wissen wollt, probiert:

`man <Befehl>` - wie „manual“

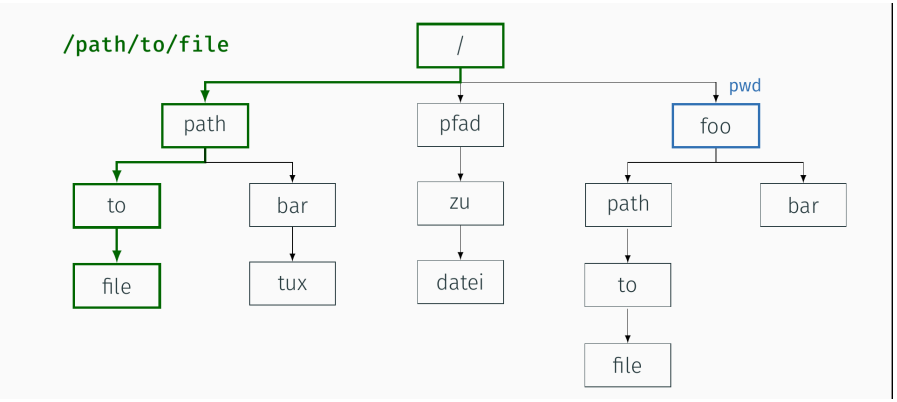
Meist wird hier folgendes erklärt:

1. Was tut der Befehl?
2. Wie ist die Syntax, um ihn zu verwenden?
3. Welche Optionen gibt es?

Navigation - Wo bin ich?

Ihr habt jetzt Dateien und Ordner in eurem momentanen Verzeichnis gesehen. Doch wo steckt ihr eigentlich gerade?
Um euren momentanen Pfad - das „working directory“ - zu sehen, nutzt:

`pwd` **p**rint **w**orking **d**irectory
$\Longrightarrow$ `/home/<user>`



Beispiel einer Ordnerhierarchie

Pfade:

`Downloads` relativ zur momentanen Position

`..`

`/usr/share` absolut, startet mit dem Wurzelverzeichnis `/` (auch „root“ genannt)

`/boot`

Navigation - Wie bewege ich mich?

Ihr wisst jetzt, wo ihr seid, und was für andere Ordner es gibt. Wie könnt ihr zu diesen wechseln? Um sich zwischen Verzeichnissen zu bewegen nutzt man:

`cd <pfad>`	**c**hange **d**irectory

Beispiele:

`cd .config`	wechselt in Ordner `.config`
`cd ..`	wechselt in übergeordneten Ordner
`cd` oder `cd ~`	wechselt in euer Homeverzeichnis
`cd -`	wechselt zu vorigem Verzeichnis

Wechselt mit `cd` in verschiedene Verzeichnisse und untersucht ihren Inhalt mit `ls`!

Navigation - Eine Tour durch Linux

An neuen Orten sollte man meist eine geführte Tour nehmen. Hier gibt es eine Reihe wichtiger Verzeichnisse, die in den meisten Distros existiern.
Besucht sie mit `cd` und begutachtet sie mit `ls`

`/`	Wurzelverzeichnis „root“
`/home`	Verzeichnisse der Nutzer
`/bin`	Standardprogramme
`/boot`	
`/media`	
`/mnt`	

Nützliches - Autocomplete oder geht das nicht schneller?

Um mühsames Tippen zu sparen und die Arbeit zu verringern, haben fast alle Terminals Autovervollständigung auf der Taste `<Tab>`.

1. Kehrt mit `cd<Enter>` in euer Homeverzeichnis zurück.
2. Tippt „`cd `“ ein. (Leerzeichen nicht vergessen!)
3. Drückt zweimal `<Tab>`: Eine Liste von möglichen Verzeichnissen erscheint!
4. Tippt `.c` ein, sodass in eurem Befehl „`cd .c`“ steht.
5. Drückt zweimal `<Tab>`: Die Liste von möglichen Verzeichnissen enthält nun nur noch die passenden Einträge!
6. Tippt `o` ein, sodass in eurem Befehl „`cd .co`“ steht.
7. Drückt einmal `<Tab>`: Da nur noch `.config` in Frage kommt, wird euer Befehl zu `cd .config/` ergänzt!
8. Drückt zweimal `<Tab>`: Die Liste der möglichen Unterverzeichnisse erscheint!

Auf diese Weise könnt ihr direkt aus dem `cd` Befehl nachschauen, in welches Verzeichnis ihr wollt, und schnell dorthin navigieren.

Nützliches - Was habe ich getan?!?

Das Terminal speichert jede eurer Eingaben. Mit dem Befehl `history` könnt ihr diese ansehen.

`history`	Zeigt eure zuvor ausgeführten Befehle
`!<nummer>`	Führt den Befehl in Zeile <nummer> der history aus
↑↓	Mit den Pfeiltasten könnt ihr in der history suchen.
`<Strg>+C`	Bricht einen Befehl ab

Probiert einmal ohne Eingabe ↑ zu drücken und den Befehl auszuführen.
Vergleicht mit nach der Eingabe von `ls` ↑ zu drücken.

Manipulation - Ordner und Dateien erstellen

Wir wissen jetzt, wie man durch Ordner navigiert und darin enthaltene Dateien auflistet. Doch wie ändern wir selbst etwas an dem System?
Um das zu testen, erstellen wir einen Ordner in unserem Heimverzeichnis und füllen ihn mit Dateien:

1. Wechselt in euer Heimverzeichnis
2. Erstellt einen Ordner: `mkdir test`
3. Wechselt in diesen Ordner
4. Erstellt Dateien: `touch file1 file2`
5. Nutzt die euch bekannten Befehle, um den Ordner zu untersuchen

Ordner erstellen:
`mkdir` **m**a**k**e **dir**ectory

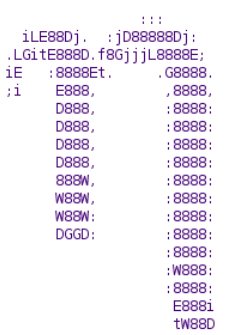
Dateien erstellen, falls sie nicht bereits existieren:
`touch`

Manipulation - Texteditoren

Wir haben jetzt Dateien, doch wie können wir sie füllen?

- Das Terminal ist ausschließlich textbasiert, deshalb sind Texteditoren sehr wichtig
- Die klassischen Editoren im Terminal:
  - `nano`
  - `emacs`
  - `vim`
- Heute werden wir uns auf den Editor `nano` konzentrieren

  `$ nano <Textdatei>`
- `<STRG>+s` speichert die Textdatei
- `<STRG>+x` schließt die Datei



Füllt ein paar Dateien mit Text!

Manipulation - Texte anzeigen

Manchmal will man Dateien nicht gleich editieren, sondern nur lesen. Besonders bei großen Dateien sind die folgenden Programme meist schneller:

`head`	zeigt die ersten Zeilen einer Datei
`tail`	zeigt die letzten Zeilen einer Datei
`less`	lädt die komplette Datei, ermöglicht scrollen und suchen
`cat`	eigentlich zum Concatenaten von strings, kann auch Dateien ausgeben

Tipp: `tail -f` gerne verwendet, um logs zu beobachten, da es dann kontinuierlich die letzten Zeilen anzeigt, und neue ausgibt, sobald sie geschrieben werden.

Stellt sicher, dass eure Dateien auch das enthalten, was ihr erwartet!

Manipulation - verschieben, kopieren und löschen

Nun wissen wir, wie man Dateien erstellt und ändert. Doch was, wenn wir sie an einen anderen Ort bewegen oder kopieren möchten?

`mv <datei> <ziel>`	**mov**e file to target
`cp <datei> <ziel>`	**cop**y file to target

Tipp: Umbennenen einer Datei ist eigentlich auch nur verschieben unter einen anderen Namen!

Und wie können wir Dateien und Ordner löschen?

`rm <datei>`	**rem**ove file
`rmdir <ordner>`	**rem**ove **dir**ectory

Achtung: `rm` fragt nicht nach, ob man sich sicher ist. Linux hat keinen Mülleimer, aus dem man versehentlich gelöschte Dateien wieder herstellen kann. Nutzt es also mit Vorsicht!

Probiert diese Befehle an euren Testdateien:

- Verschiebt und kopiert Dateien und Ordner.
- Was passiert, wenn die Zieldatei bereits existiert?
- Was passiert, wenn man `rm <ordner>` macht?

Tipps, Tricks und mehr Befehle

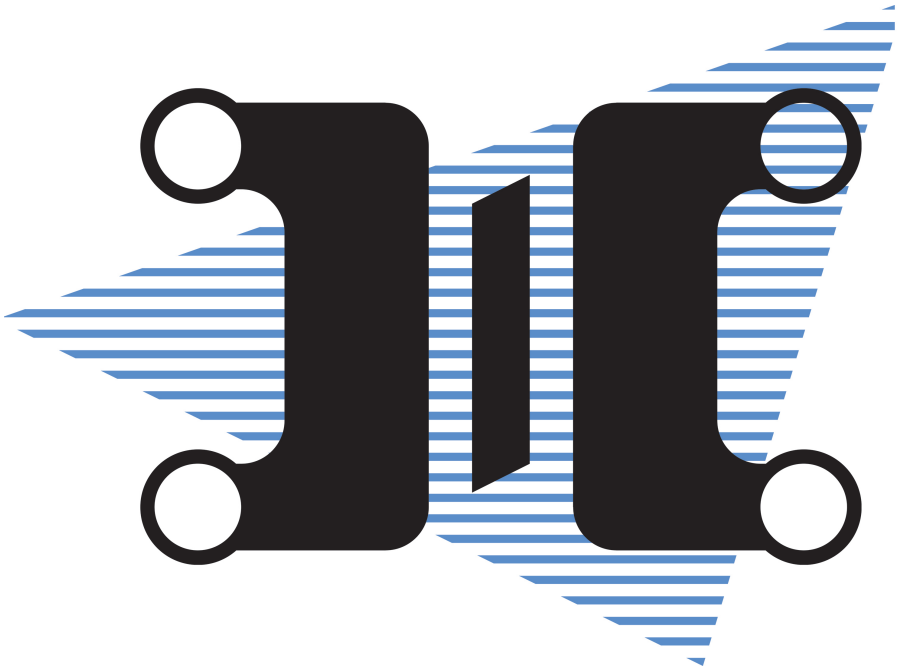
- `alias` macht neue `bash`-Befehle, z. B.:
  `$ alias ll="ls -la"`
- **Probiert den `which`-Befehl aus:**
  `$ which ls`
- `grep` findet das angegebene Wort/Muster in der angegebenen Datei

```
$ grep  Hello  test.txt
        ↑ Muster     ↑ Datei
```

- **Tipp 1:** *Ihr müsst nicht immer alles eintippen!* Die `<TAB>`-Taste macht eine automatische Vervollständigung!
- **Tipp 2:** Man bekommt häufig Hilfe mit `<Befehl> -help` *oder* `man <Befehl>`.
- **Tipp 3:** Das Terminal achtet auf Groß-/Kleinschreibung, d. h. `nano` ist ein Befehl während `Nano` oder `NANO` keine Befehle sind.

Tagesordnung

1. Was ist Linux?
2. Linux - Der größte Lego-Bausatz der Welt
   - Distributionen
   - Graphische Oberflächen
3. Linux 101 - Shell und Bash Befehle
   - Wie verbinde ich mich mit dem Eikon?
   - Orientierung
   - Navigation auf der Shell
   - Nützliches
   - Manipulation von Dateien
   - Tipps, Tricks und mehr Befehle
4. **Bash-scripting**



Bash scripting - Pipelines

Pipelining mit |: lenkt Textausgabe des linken Befehls in Eingabe des rechten Befehls
Beispiel:
Wir wollen alle verfügbaren Programme zu sichten, doch der Output ist höher als der Monitor:

```
$ ls /bin /usr/bin
```

Wir könnten diesen Output in less lesen:

```
$ ls /bin /usr/bin | less
```

Oder ihn direkt mit einem Programm wie `grep <suchwort>` durchsuchen:

```
$ ls /bin /usr/bin | grep rm
```

Dann stellen wir fest, dass die Liste nicht sinnvoll sortiert ist, da wir ja zwei Ordner lesen. Jemand verrät uns, dass `sort` uns da helfen kann, und wir fügen es in unsere Pipeline ein:

```
$ ls /bin /usr/bin | sort | less
```

Doch manchmal existieren mehrere Versionen des selben Programms auf einer Maschine. Uns interessieren also nur einzigartige Namen, die wir mit `uniq` filtern könnten:

```
$ ls /bin /usr/bin | sort | uniq | less
```

Bash scripting - Redirection

Mit <, > und >> kann man Textstreams in und aus Dateien leiten:

`cmd < file`	leitet Datei als Textinput in das Kommando
`cmd > file`	schreibt Textoutput in Datei
`cmd >> file`	schreibt Textoutput an das Ende der Datei

Testet folgende Befehle und untersucht die Ergebnisse:

- `ls /bin > programme.txt`
- `ls /bin » programme.txt`
- `cat programme.txt | grep rm`
- `grep rm < programme.txt`
- `grep rm < programme.txt > suche.txt`
- `grep rm < programme.txt | uniq > suche.txt` (man kann pipes und redirection kombinieren)

Bash scripting - .sh-Dateien

- `bash`-Skripte enden mit `.sh`
- **Idee:** Applikation ruft Applikation
- Man kann beispielsweise folgende `bash`-Skript erzeugen:
  `$ echo "ls /bin /usr/bin | sort | uniq | grep zip">> search-programs.sh`
- Bevor man `search-programs.sh` aufrufen kann, muss man die Rechte der Datei anpassen:
  `$ chmod 700 search-programs.sh`
- Die neuen Rechte der Datei machen sie *ausführbar*
- Schließlich kann man das `bash`-Skript wie folgt aufrufen:
  `$ ./search-programs.sh`

Wie geht es weiter?

- Inhalt dieser Einführung stützt sich auf „*The Linux Command Line*“ von William Shotts
- The Linux Documentation Project: `https://tldp.org`
- Generell gute Dokumentationen, z. B.: `https://wiki.ubuntu.com/`, `https://ubuntuusers.de/`, `https://wiki.archlinux.org/`
- „*Beginning Linux Programming*“ von Neil Matthew, Richard Stones