Zusammenfassung Tag 8

Mit Pfaden arbeiten

- Der Befehl `cd` nimmt Pfadangaben als Parameter entgegen. Diese können wir absolut oder relativ angeben
    - Absolute Pfade
      Starten mit einem `/` und beschreiben den Pfad zum gewünschten Verzeichnis ausgehend vom obersten Punkt im Dateisystem
    - Relative Pfade
      Beschreiben den Weg zum Ziel ausgehend vom aktuellen Standort
- Die Variable `$OLDPWD` enthält das vorige Verzeichnis, indem man sich befunden hat
- Um ein Programm auszuführen, haben wir zwei Möglichkeiten:
    - Wir geben den absoluten Pfad vorneweg mit an
    - Wir schreiben `./` davor, das steht für das aktuelle Verzeichnis
- Wenn wir Dateien öffnen wollen, zBsp. mit einem Editor, können wir direkt die Datei angeben, wenn sie sich im selben Verzeichnis befindet

Verzeichnisse erstellen und löschen

- Unter Ubuntu wird in der Datei `.profile` im Homeverzeichnis festgelegt, welche Pfade in der `$PATH` Variable aufgenommen werden
- Unter CentOS wird in der Datei `.bash_profile` im Homeverzeichnis festgelegt welche Pfade in der `$PATH` Variable aufgenommen werden
- Mit `mkdir` werden Verzeichnisse erstellt, mit `rmdir` gelöscht (nur leere Verzeichnisse ohne Unterordner). Mit der Option `-p` auch mit Unterordner
- Der Befehl `rm` löscht sowohl Dateien als auch Verzeichnisse.
  Vorsicht beim Einsatz dieses Befehls!

Verzeichnis-Listings verstehen

- Ausgabe von des Befehls `ls -l` (langes Format)
    - Jede Datei bzw. jedes Unterverzeichnis stehen in einer eigenen Zeile
    - Der Punkt steht für das aktuelle Verzeichnis und der Doppelpunkt für das übergeordnete Verzeichnis
    - In der ersten Spalte steht der Dateityp
        - `d` steht für Directory, also ein Verzeichnis
        - `-` steht für normale Dateien
        - `l` wie Link, das sind Symbolische oder Softlinks
        - `c` für Character Device, das sind zeichenorientierte Geräte (Modems)
        - `b` für Block Device, das sind blockorientierte Geräte (Speichermedien)
    - Ausführbare Dateien werden in grün dargestellt (anhand der Rechte)
    - Die Rechte werden entsprechend angezeigt
    - Die Anzahl der Links auf diesen Eintrag
    - Der Eigentümer des Eintrags und die Gruppe, der der Eintrag zugewiesen wurde
    - Die Größe der Datei wird in Bytes angegeben
    - Datum der letzten Änderung
- Der Befehl `ls -F` kann durch angehängt Zeichen kennzeichnen, um welche Art von Eintrag es sich handelt:
    - / Verzeichnisse
    - * ausführbare Dateien
    - @ symbolische Links
- Der Befehl `ls` unterstützt sehr viele Optionen die flexibel miteinander kombiniert werden können. Siehe dazu Man-Page von `ls` und Befehlsübersicht dieses Kurses
- Das Programm `tree` zeigt sehr übersichtlich den Verzeichnisbaum ab einem bestimmten Punkt an

Dateien erstellen, kopieren, verschieben und löschen

- Es gibt drei gängige Möglichkeiten eine Datei zu erstellen:
    - Datei anlegen mit `touch`
    - Mit einem Editor eine nicht existierende Datei öffnen und speichern
    - Mit `echo` und dem Umleitungszeichen eine Datei erstellen und beschreiben
- Mit dem Befehl `cp` (copy) können Dateien kopiert werden (auch unter anderem Namen) mit der Option `-r` sogar komplette Verzeichnisse
- Mit dem Befehl `mv` (move) können Dateien verschoben und umbenannt werden.
  Vorsicht: Ist das Ziel kein Verzeichnis, sondern eine existierende Datei, wird diese Datei mit dem Inhalt der Quelldatei überschrieben

Hardlinks und Softlinks erstellen

- Ein Link ist ein Verweis auf eine andere Datei oder ein Verzeichnis
- **Hardlinks**
    - Es handelt es sich um ein- und dieselbe Datei auf die mittels eines Inodes referenziert wird. Dies ist ein Verweis auf eine Datei, allerdings in der Dateisystem-Tabelle – quasi die ID der Datei.
    - Hardlinks können nur innerhalb einer Partition existieren
    - Hardlinks können nur auf Dateien angewendet werden, nicht auf Verzeichnisse
    - Wird ein Hardlink gelöscht, so besteht die Originaldatei noch weiter
- **Softlinks / Symbolic Links, kurz: Symlinks**
    - Mit Symlinks können Verweise, also Links partitionsübergreifend erstellt werden
    - Das gilt gleichermaßen für Dateien und Verzeichnisse
    - Sie sind deutlich einfacher zu erkennen, da sie mit Pfaden arbeiten
    - Wird die Originaldatei gelöscht, existiert keine Kopie mehr

Dateien archivieren und komprimieren

- Um Daten und Dateien sichern oder bereit zu stellen, bietet es sich an, diese in einer Archivdatei als Ganzes einzupacken und zu komprimieren
- Spart Platz auf dem Speichermedium bzw. Bandbreite bei der Übertragung
- Das am häufigsten verwendete Programm zum Archivieren ist `tar`
- Ein *Tarball* ist eine mit `tar` erstellte und `gzip` oder `bzip2` komprimierte Archivdatei
- `Bzip2(.bz2)` und `gzip(.gz)` sind Kompressionsverfahren
- `Bzip2` ist die neuere und effektivere Variante
- Dateien mit der Endung `tar` sind unkomprimierte Archive die mit `tar -xf` entpackt werden können. Nach dem Entpacken bleibt die Archiv-datei weiterhin vorhanden
- Archive beinhalten meistens mehrere Dateien und oft auch Verzeichnisse. Diese können mit `tar -tf` angezeigt werden
- Mehrere Dateien und Verzeichnisse können mit `tar -cf` zu einem Archiv zusammengefasst werden
- Alternative zu `tar` ist `cpio`
- Alternative zu `gzip` und `bzip2` ist `xz`

Zugriffsrechte auf Dateien und Verzeichnisse verstehen

- Zugriffsrechte auf Dateien und Verzeichnisse werden mit `ls -l` dargestellt:

Beispiel: Verzeichnis

```
drwxr-xr-x 2 eric users          4096 Mai  5 19:25 Dokumente
```

Beispiel: normale Datei

```
-rw-r--r-- 1 asterix gallier 3794421 Mai  7 19:31 grundkurs-linux.pdf
```

Beispiel: ausführbare Datei (Programm, Skript):

```
-rwxr-xr-x 1 root projekt4711  33242 Juni 1 12:01 myscript
```

User (Eigentümer) Gruppe Gruppe Welt (alle anderen)

r: read, lesen

w: write, schreiben

x: execute, ausführen

- Das bedeuten die Zugriffsrechte:

Zugriffsrecht	Datei	Verzeichnis
Read (r)	Darf lesend geöffnet werden	Datei- und Verzeichnisnamen dürfen gelesen werden, nicht jedoch deren weitere Attribute, wie Berechtigungen, Besitzer, etc.
Write (w)	Darf schreibend geöffnet werden (impliziert ändern und löschen)	Einträge im Verzeichnis dürfen modifiziert werden (impliziert erstellen, ändern und löschen)
Execute (x)	Darf ausgeführt werden (nur sinnvoll bei Programmen und Skripts)	Erlaubt den Wechsel in das Verzeichnis und weitere Attribute zu den enthaltenen Dateien abzurufen (bei bekanntem Dateinamen unabhängig vom Leserecht)

- Zugriffsrechte setzen mit `chmod`:

u – User, Eigentümer
g – Group
o – Others, Welt bzw. alle anderen
a – all, Alle oben genannten

= – Rechte genau so setzen
+ – Recht hinzufügen
- – Recht entziehen

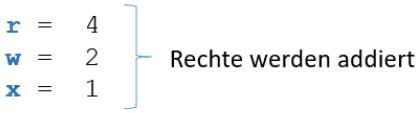
User erhält alle Rechte, Gruppe erhält zusätzlich Schreiben und Welt wird Lesen und Ausführen entzogen:
```
chmod u=rwx,g+w,o-rx /projekte/projekt_zaubertrank
```

User erhält zusätzlich Schreibrecht, Gruppe kein Ausführen-Recht mehr und Welt erhält gar keine Rechte:
```
chmod u+w,g-x,o= /projekte/projekt_zaubertrank
```

Welt werden alle Rechte entzogen
```
chmod o= /projekte/projekt_zaubertrank
```

- User und Gruppen können mit `chown` gesetzt werden, der Befehl nimmt durch Doppelpunkt getrennt Benutzer und Gruppe an:
  - `chown <benutzer>:<gruppe> <datei>`
- Die zweite Möglichkeit besteht darin, mit `chown` den User und mit `chgrp` die Gruppe festzulegen:
  - `chown <benutzer> <datei>`
  - `chgrp <gruppe> <datei>`

- Zugriffsrechte setzen mit der Octal-Methode:



User erhält alle Rechte, Gruppe erhält Lesen und Ausführen, Welt erhält keinen Zugriff:

```
chmod 750 /projekte/projekt_zaubertrank
```

User erhält Lesen und Schreiben, Gruppe und Welt erhalten nur Lesezugriff:

```
chmod 644 mydata.txt
```

Sonderrechte – SUID-Bit, SGID-Bit und Sticky-Bit

- Wird bei den Benutzerrechten einer Datei ein **s** statt einem **x** angezeigt, handelt es sich um das Set UID-Bit oder Kurz: SUID-Bit. Es sorgt dafür, dass das Programm immer mit den Rechten des Dateibesitzers läuft
- Das SUID-Bit können wir setzen mit **`chmod u=rwxs`** `<Datei>` oder **`chmod 4755`** `<Datei>`
- Das SGID-Bit sorgt bei einer Datei mit Ausführungsrechten dafür, dass sie immer im Kontext der Gruppe läuft. Bei einem Verzeichnis sorgt das Set GID-Bit dafür, dass die für das Verzeichnis festgelegte Gruppe auf alle neu angelegten Unterverzeichnisse und Dateien vererbt wird
- Das SGID-Bit können wir setzen mit **`chmod g=rwxs`** `<Datei>` oder **`chmod 2755`** `<Datei>`
- Das Sticky-Bit wird durch ein **t** anstatt des **x** für Others gesetzt, also ganz am Ende der Rechteliste und kommt z.B. beim /tmp-Verzeichnis zum Einsatz
- Wird das Sticky-Bit auf einen Ordner angewendet – und das ist der einzige Einsatzzweck – so können darin erstellte Dateien und Verzeichnisse nur vom Dateibesitzer gelöscht oder umbenannt werden
- Es wird über den Buchstaben t gesetzt bzw. Octal über die 1 in der vierten, vorangestellten Ziffer

Umask und die Standardrechte

- Wird ein Verzeichnis erstellt, dann werden standardmäßig bestimmte Rechte gesetzt:
  - **rwx** für den User,
  - **rx** für die Gruppe
  - **rx** für Others
- Wird eine Datei erstellt, dann werden standardmäßig bestimmte Rechte gesetzt:
  - **rw** für den User,
  - **r** für die Gruppe
  - **r** für Others
- Diese Standard-Rechte werden mit `umask` festgelegt
- Da es sich um eine Maske handelt, wird das angezeigt, was verdeckt wird, also nicht gesetzt ist:
  - Bei Verzeichnissen ziehen wir die `umask` von jeweils 7 ab, um die gesetzten Werte zu erhalten. Dabei wird die erste Stelle ignoriert, da sie Sonderrechte betrifft, die in der `umask` ohnehin nicht gesetzt werden
  - Bei Dateien ist die 6 der Ausgangswert
- `umask` festlegen:
  - Bei CentOS, systemweit: `/etc/profile`, für Benutzer: `.bash_profile`
  - Bei Ubuntu, systemweit: `/etc/login.defs`, für Benutzer: `.profile`