Tag_1_neu

February 25, 2025

1 Linux

2 Tag 1

3 1

Linux ist eigentlich ein Kernel (das Gehirn des Computers) und die Distribution ist dann das Betriebssystem (operating system OS).

Die Distributionen enthalten den Kernel und weitere Anwendungen. Welche Anwendungen vorinstalliert sind, hängt davon ab wofür diese Distribution gedacht ist. Eine Distribution, die rein für den Betrieb von Servern gedacht ist, braucht nicht dieselben Programme wie eine für den Heimgebrauch.

Es gibt sogenannte Familien von Distributionen, die verschiedene Paketmanager benutzen:

Debian/Ubuntu/Mint benutzen `dpkg`, `apt` und `apt-get` um `deb`-Pakete zu installieren.

Red Hat/Fedora/CentOS benutzen `yum` und `dnf` um `rpm`-Pakete zu installieren.

4 2

Es geht jetzt erstmal darum die Kommandozeile zu zeigen und wie man mit ihr arbeiten kann.

Wenn man einen Filemanager wie Nautilus, Nemo oder sogar den Windows Explorer hat, dann kann man mit diesem gewisse Sachen machen, wie Dateien kopieren, löschen, erstellen und so weiter. All das kann man mit der Kommandozeile aber auch und sogar effizienter. Denn im GUI Dateimanager kann man nur EIN Verzeichnis nach dem nächsten Erstellen aber in der Kommandozeile kann man mit `mkdir` mehrere Verzeichnisse und sogar Unterverzeichnisse mit EINEM Befehl erstellen.

Zuerst werden einfach die häufigsten Datei- und Verzeichnisoperationen vorgestellt und beigebracht:

4.0.1 Shell

Shell: Das ist die Kommandozeile

Ubuntu nutzt die Bash Shell, es gibt aber noch z.B. zShell, cShell und so weiter. Wir benutzen aber für alles die Bash Shell.

4.0.2 Bash

Bash ist auch eine Programmiersprache, konzipiert um möglichst effizient mit Dateien und Verzeichnissen zu arbeiten.

Jeder Befehl wird in der Regel sofort ausgeführt, wenn man Enter drückt. Genau wie in Python gibt es builtin Befehle.

Buildin Befehle:

Steuerungsbefehle

```
break - Beendet eine Schleife vorzeitig.
continue - Springt zur nächsten Iteration einer Schleife.
exit - Beendet die Shell oder ein Skript.
return - Beendet eine Funktion und gibt einen Wert zurück.
exec - Ersetzt den aktuellen Shell-Prozess durch den angegebenen Befehl.
```

Variablen und Parameter

```
declare - Deklariert Variablen und legt ihre Attribute fest.
export - Markiert Variablen und Funktionen, die an Subshells weitergegeben werden sollen.
local - Deklariert eine Variable innerhalb einer Funktion als lokal.
readonly - Markiert Variablen oder Funktionen als schreibgeschützt.
unset - Entfernt Variablen oder Funktionen.
```

Eingabe/Ausgabe

```
echo - Gibt Text auf der Konsole aus.
read - Liest eine Zeile von der Standardeingabe und weist sie einer Variablen zu.
printf - Gibt formatierten Text aus, ähnlich wie printf in C.
mapfile (oder readarray) - Liest eine Datei oder Eingabe in ein Array ein.
```

Prozesssteuerung

```
jobs - Listet aktive Jobs auf.
kill - Sendet ein Signal an einen Prozess.
wait - Wartet auf das Ende eines Hintergrundprozesses.
```

Verzeichnisse

```
cd - Wechselt das aktuelle Arbeitsverzeichnis.
dirs - Listet den Verzeichnis-Stack auf.
pushd - Fügt ein Verzeichnis zum Verzeichnis-Stack hinzu und wechselt dorthin.
popd - Entfernt das aktuelle Verzeichnis aus dem Verzeichnis-Stack und wechselt zum vorherigen.
```

Bedingungen und Vergleiche

```
[[ ... ]] - Führt erweiterte bedingte Ausdrücke aus.
test oder [ - Bewertet bedingte Ausdrücke (wird oft mit if verwendet).
```

Shell-Steuerung

```
source (oder .) - Führt eine Datei in der aktuellen Shell aus.
type - Zeigt an, wie ein Befehl interpretiert wird (built-in, Funktion, Alias, etc.).
command - Führt einen Befehl ohne Shell-Funktionen oder Aliase aus.
builtin - Führt einen eingebauten Befehl explizit aus.
hash - Verwalten und Anzeigen der Hash-Tabelle der Befehle.
history - Zeigt die Befehlshistorie an.
```

Shell-Optionen und Funktionen

```
set - Setzt oder entfernt Shell-Optionen und Positionsparameter.
shopt - Setzt oder zeigt Shell-Optionen für bash an.
alias - Erstellt Aliase für Befehle.
unalias - Entfernt Aliase.
function - Definiert eine Shell-Funktion.
```

Spezielle

```
help - Zeigt Hilfe für eingebaute Befehle an.
trap - Setzt Befehlsketten, die auf Signale reagieren.
true - Gibt immer einen erfolgreichen Exit-Status zurück.
false - Gibt immer einen fehlerhaften Exit-Status zurück.
: - Führt einen Null-Befehl aus (hat keine Wirkung, gibt immer 0 zurück).
```

prompt

Der prompt ist alles was ganz links in der Shell zu sehen ist. Er enthält einige Informationen über den momentanen Nutzer und das momentane Arbeitsverzeichnis.

```
NUTZERNAME@HOSTNAME: MOMENTANES VERZEICHNIS $
Für einen normalen Benutzer

ROOT@HOSTNAME: MOMENTANES VERZEICHNIS #
Für den Superuser/root
```

Das $ und Nutzername zeigt einen normalen Benutzer an und die # und root einen Superuser

5 Befehle

Typischer Befehlsaufbau:

```
Befehl OPTIONEN ARGUMENTE
```

Optionen verändern das Verhalten des Befehls und Argumente sind Dateien oder Verzeichnisse mit denen der Befehl arbeiten soll. Oft sind Optionen und/oder Argumente nicht notwendig.

5.0.1 cd - Change Directory

```
cd ..
```

```
Geht einen Schritt runter als von /home/username/ zu /home/
```

`cd ~` oder `cd /home/NUTZERNAME`

```
Gehe ins Homeverzeichnis des Nutzers EGAL wo du gerade bist: /home/NUTZERNAME/
WINDOWS c:\Benutzer\NUTZERNAME
```

```
cd /
```

```
Ganz runter auf root egal wo du bist
Identisch zu WINDOWS c:\
```

```
cd /user/bin
```

```
Gehe ins Verzeichnis /usr/bin
```

`cd ../Music`

`Gehe aus dem momentanen Verzeichnis heraus und in das Verzeichnis Music`

`../Music` ist ein RELATIVER Pfad.

`/user/bin` ist ein ABSOLUTER Pfad. Diese beginnen IMMER mit /

ls - list

ls listet inhalt des verzeichnises auf

`ls /expliziter Pfad`

Der explizite/absolute Pfad beginnt IMMER mit /!

`ls ~` oder `ls /home/USERNAME`

`Gibt den Inhalt des Homeverzeichnises aus EGAL in welchem Verzeichnis man gerade ist`

`ls -a`

`Versteckte directorys/dateien (alles was einen Punkt davor hat ist versteckt (können trotzdem geöffnet werden)) -a ist eine Option von ls`

`ls -l`

`Gibt den Inhalt des Verzeichnisses im long format an:`

```
drwxr-xr-x 1 user1 user1 4096 Aug 11 06:40  Music
-rw-r--r-- 1 user1 user1 1748 Aug 13 11:41  Datei1
lrwxrwxrwx 1 user1 user1   23 Aug 11 06:22  Softlink

d ganz Links zeigt ein VERZEICHNIS an.

- ganz Links zeigt eine Datei an

l  ganz Links zeigt einen Softlink an

s deutet auf einen Socket hin

rwxr-xr-x  sind die Rechte für den Besitzer, die Gruppe und die restlichen
Benutzer

1 Anzahl an Links
user1 Besitzer
user1 Gruppe
4096 Größe in Bytes
Aug 11 6:40 Änderungsdatum
Music   Datei- oder Verzeichnisname
```

Rechte:

`r` - Lesen

`w` - Schreiben

`x` - Ausführen

`-` - Rechte nicht erhalten

Beispiel: rwxr-xr-x

Besitzer: darf lesen, schreiben und ausführen (die ersten 3 Zeichen)

Gruppe: darf lesen und ausführen, kein Schreiben, weil statt w hat man dort - (die Mittleren)

Rest: darf lesen und ausführen, kein Schreiben, weil statt w hat man dort - (die letzten Drei)

`ls -lt` sortieren der long Liste nach der Zeit, NEUSTE Datei zuoberst

`ls -ltr` sortieren der long Liste nach der Zeit, NEUSTE Datei ganz unten wegen r = reverse

`ls | less` Der Output von ls wird an less weitergegeben und kann dann seitenweise gelesen werden. Das ist hilfreich, wenn man ein Verzeichnis mit hunderten an Datein auflistet

```
ls -lhs
```

```
4.0K -rw-r--r-- 1 user1 user1 1.7K Aug 13 11:41  Datei1
```

4.0K ist der Festplattenspeicherverbrauch der Datei 1.7K ist die tatsächliche Dateigröße

Der Rest ist identisch zu oben.

```
ls -ld /etc/security/
```

```
drwxr-xr-x 4 root root 4096 Aug 13 14:26 /etc/security/
```

Mit der Option "-d" kann man sich die Informationen über das Verzeichnis anzeigen lassen, statt dem Inhalt.

`exit` (schließt Linux)

`which`

```
sucht den Speicherort der ausführbaren Datei zu einem Befehl
Jeder Befehl ist quasi eine ausfürbare datei
```

`sudo` (superuser DO (durch # gekenzeichnet)) fragt nach passwort und logt dich als root (admin) ein kannst dann buchstäblich alles machen.

Hoch und runter Pfeiltasten gehen die Bashhistory (Befehle, die du zuvor eingegeben hast.) durch

`clear` oder `Strg + L` löscht den Bildschirm

5.1 Hilfe mit man oder –help

Befehl –help

Ruft eine "kurze" Hilfe auf mit den häufgsten Optionen

`man Befehl`

man ruft intern den less Befehl auf welcher eine Navigation der Hilfedatei ermöglicht! Ruft die Hilfedokumentation von `Befehl` auf. Man kann dort mit `G` ganz nach unten ans Ende der Datei und mit `g` an den Anfang gehen. `e` oder `Pfeil runter` geht eine Zeile weiter runter, `y` oder `Pfeil rauf` geht eine Zeile rauf. `f` oder `Bild runter` geht eine Seite runter und `b` oder `Bild rauf` geht eine Seite rauf.

Mit `/` oder `?` startet man die vorwärts/rückwärts Suche innerhalb der Datei. Gibt man z.B. `/Linux` ein, dann sucht less nach dem Wort Linux in der Datei. Mit `n` kann man zum nächsten Treffer springen und mit `N` springt man wieder zurück.

Es gibt ein Hilfemenu in den durch man angezeigtem Bildschirm erreichbar durch `h`.

Abschnitt	Beschreibung
NAME	Name des Befehls und kurze Beschreibung
SYNOPSIS	Beschreibung der Befehlssyntax
DESCRIPTION	Beschreibung der Wirkung des Befehls
OPTIONS	Verfügbare Optionen
ARGUMENTS	Verfügbare Argumente
FILES	Hilfsdateien
EXAMPLES	Ein Beispiel für den Einsatz des Befehls
SEE ALSO	Querverweise zu verwandten Themen
DIAGNOSTICS	Warn- und Fehlermeldungen
COPYRIGHT	Autor(en) des Befehls
BUGS	Bekannte Fehler und Beschränkungen des Befehls

In der Praxis enthalten die meisten Manpages nicht alle diese Teile.

Kategorie/Sektion	Beschreibung
1	Benutzerbefehle
2	Systemaufrufe
3	Funktionen der C-Bibliothek
4	Treiber und Gerätedateien
5	Konfigurationsdateien und Dateiformate
6	Spiele
7	Verschiedenes
8	Systemadministrator-Befehle
9	Kernel-Funktionen (nicht Standard

Die Kategorien sind wichtig, wenn es Dateien mit gleichem Namen gibt, die eigene man pages haben. Z.B. gibt es passwd als Benutzerbefehl und auch als Konfigurationsdatei. Der Benutzerbefehl ist Kategorie 1 und die Konfigurationsdatei ist Kategorie 5. Jede Datei ist in GENAU einer Kategorie enthalten.

Will man die Hilfe für die Konfigurationsdatei öffnen, dann muß man man 5 passwd eintippen.

5.1.1 touch

touch Dateiname(Dateiname2 Dateiname3)

Ändert es das "letzte Änderungsdatum" der Datei(en), ABER wenn die Datei(en) NICHT existiert(en), dann wird eine(oder mehrere) leere neue Datei(en) erstellt!

5.1.2 echo

echo STRING

Gibt den String auf dem Screen zurück

5.1.3 Redirection > und »

echo STRING > Dateiname

> erstellt eine Datei und fügt den String ein, wenn die Datei NICHT existiert, ansonsten wird die Datei ÜBERSCHRIEBEN

echo STRING » Dateiname

>> erstellt eine Datei, wenn Dateiname nicht existiert, ODER der String wird hinten an die Datei angehängt!

Mit > und » wird Kanal 1 der Befehlszeile in eine Datei umgeleitet. Kanal 2 sind die Fehlermeldungen. Um diese umzuleiten, kann man 2> oder 2» benutzen.

cat /etc/* 2> error.txt

2> erstellt die error.txt, falls nicht vorhanden, ansonsten überschreibt es sie. Dort werden alle Fehler reingeschrieben, z.B. Permission denied

cat /etc/* 2» error.txt

2> erstellt die error.txt, falls nicht vorhanden, ansonsten wird hinten drangehängt

Desweiteren existierst noch &> und &». Damit werden BEIDE Kanäle in eine Datei umgeleitet. Leitet man in "/dev/null" um, dann werden alle Meldungen weggeworfen.

Kanal 0 ist die Standard Eingabe. Auch hier gibt es eine Umleitung: < und «

Mit < kann man Dateien an Befehle weitergeben, was nützlich ist, wenn diese Befehle normalerweise KEINE Dateien annehmen.

tr ist so ein Befehl:

tr -d "l" < Datei

Damit wird im Text aus der Datei l gelöscht. Die Datei ändert sich NICHT.

tr "l" "o" < Datei

Alle l werden durch o ersetzt im Text der Datei.

Will man das Ergebnis speichern, dann muß man mit > und » die Ausgabe umleiten.

« Damit kann man mehere Zeilen Text oder Code an einen Befehl weiterleiten. Dadurch erspart man sich die Nutzung von Dateien.

cat « EOF Zeilen an Text oder Code die man ausführen will EOF

Das Wort HINTER « gibt den Start des Text- oder Codeblocks an. Man muß dasselbe Wort nochmal schreiben, um die Eingabe abzusenden.

Das nennt man "Here Document". Es erlaubt ganze Codeblöcke an Befehle weiterzuleiten.

```
Dateiinhalt:
1.2.3.4.
a.b.c.d.
pi.alpha.beta.gamma.
```

cut -f 3 -d "." Dateiinhalt

Die Ausgabe besteht aus dem Inhalt des 3ten FELDES. Das ist der Satz NACH dem Zweiten "." und vor dem Dritten ".". -d gibt den FELDTRENNER (Delimiter) an.

```
Ausgabe:
3
c
beta
```

5.1.4 Pipes

Man kann die Ausgabe eines Befehls mit Hilfe der Pipes | als Eingabe eines anderen Befehls übergeben.

cut -f 1,3 -d "." Datei

```
Ausgabe:
1.3
a.c
pi.beta
```

Will man nun den Punkt durch ein Leerzeichen, kann man die Ausgabe als Datei speichern und dann mit < an tr übergeben oder man macht es direkt mit einer Pipe.

cut -f 1,3 -d "." Datei | tr "." " " > Spalten13

Zuerst werden die 1 und 3 Spalte ausgeschnitten und dann dan tr übergeben, welches den Punkt durch ein Leerzeichen ersetzt und dann in der Datei Spalten13 speichert.

Oder

ls -l /etc/* | head -n 5 | wc -w

`Damit wird die Dateien aus /etc/ im Long-format aufgelistet. Diese Ausgabe geht dann an head, welches standardmäßig nur die ersten 10 Zeilen anzeigt, weitergeleitet. Man würde also nur die ersten 5 Zeilen sehen können wegen dem -n 5 bei head. wc steht für word count und die Option -w zählt die Wörter.`

Die Ausgabe ist 45. 9 Wörter pro Zeile

Es gibt auch `tail` welches die letzten 10 Zeilen einer Datei anzeigt. Mit der Option "-f" werden später hinzugefügte Zeilen auch ausgegeben. Damit kann man sich live den Inhalt einer Log-Datei anzeigen lassen.

5.1.5 Multiline String

```
echo "Hallo 'Du Gummischuh'
 nächste Zeile
 und noch eine Zeile"

AUSGABE:
Hallo 'Du Gummischuh'
nächste Zeile
```

Benutzt man ein Anführungszeichen, dann kann man eine mehrzeiligen String durch die Eingabetaste erzeugen. Erst wenn das zweite Mal dasselbe Anführungszeichen erreicht wird, wird der String ausgegeben.

5.1.6 Anführungszeichen

Will man mit touch eine Datei mit einem Leerzeichen im Namen erstellen, dann gibt es dazu drei mögliche Wege. " ", ' ' um den Dateinamen herum oder das Leerzeichen mit "escapen".

Normalerweise hat das Leerzeichen die Bedeutung, daß Argumente und Optionen dadurch getrennt werden.

```
touch Hallo Welt
```

wird als der Befehl touch mit ZWEI Argumenten angesehen. Es werden dann diese Dateien erstellt oder ihr Datum geändert.

```
touch "Hallo Welt"
touch 'Hallo Welt'
touch Hallo\ Welt
```

In allen drei Fällen wird EINE Datei mit Namen 'Hallo Welt' erstellt.

Hier gibt es zwischen " und ' KEINEN Unterschied.

Bei der Verwendung von Variablen allerdings gibt es einen wichtigen Unterschied.

```
echo "Hallo $USER"
```

Hier wird einfach "Hallo user1" ausgegeben, wenn die Variable USER den Wert 'user1' enthält. Das $ weist Bash an die Variable auszuwerten und ihren Wert auszugeben.

```
echo 'Hallo $USER'
```

gibt "Hallo $USER" zurück.

Beim " verlieren bis auf $, und ` alle Sonderzeichen ihre Bedeutung. Beim ' verlieren ALLE Sonderzeichen ihre Bedeutung. Deswegen wird hier die Variable NICHT durch ihren Wert ersetzt.

6 Tag 2

6.0.1 cat

Damit fügt man mehrere Datein zusammen indem sie hintereinander gehängt werden. Das kann man dann in eine neue Datei umlenken. Am häufigsten wird cat aber zum Anzeigen des Inhalts

von Dateien verwendet.

`-n  Zeilennummern anzeigen`

6.0.2 pwd

pwd - print working directory Gibt den ABSOLUTEN Pfad zum Verzeichnis in dem man sich gerade befindet an.

6.0.3 mkdir

mkdir - make directory Erstellt neue Verzeichnisse. Wenn man ein Verzeichnis erstellen will bei dem das ParentVerzeichnis NICHT existiert, dann muß man -p oder –parents benutzen.

`mkdir Bilder Dokumente etc` erstellt die Verzeichnisse Bilder, ...

`mkdir Musik/Klassik Musik/Techno` erzeugt eine Fehlermeldung, wenn das Verzeichnis Musik NICHT existiert.

`mkdir -p Musik/Klassik Musik/Techno` erzeugt die Verzeichnisse Musik und darin dann Klassik und Techno

6.0.4 rmdir

rmdir - remove directory löscht LEERE Verzeichnisse

`--ignore-....  Unterdrückt die Fehlermeldung, wenn das Verzeichnis nicht leer ist, aber nichts wird gelöscht`

`-p  Löscht das Verzeichnis und die Parent-Verzeichnisse, sofern ALLES leer ist.`

`rmdir -p Filme/A` entfernt A und entfernt Filme, wenn A der einzige Inhalt in Filme war!!! D.h. nach dem Löschen des leeren Verzeichnisses A MUSS Filme LEER sein!!

6.0.5 rm

rm - remove file or directory löscht Dateien oder LEERE Verzeichnisse

`-r/-R  lösche rekursiv ALLE Verzeichnisse UND ihren Inhalt`

`-i  frage bei jeder Datei/Verzeichnis nach, ob es wirklich gelöscht werden soll`

`-d/--dir  lösche leeres Verzeichnis`

`-f/--force  ignoriert ALLE Fehlermeldung und frage NIEMALS nach!`

`rm -rf /` LÖSCHT ALLES AUF DEM RECHNER!!!!! NICHT MACHEN !!!!

`rm -r` Mehrere/Verz/hintereinander/ entfernt die Verzeichnisse und alle Dateien in ihnen

`rm Dateiname` löscht Dateiname

`rm -d Mehrere` löscht das LEERE Verzeichnis Mehrere

`rm *` löscht ALLE Dateien in momentanen Ordner

`rm -d *` löscht alle LEEREN Verzeichnisse und ALLE Dateien

6.0.6 GLOBBING

Verschiedene Platzhalter (Wild card) für ein oder mehrere Zeichen zur Verwendung mit Dateinamen. Damit kann man zum Beispiel Dateien mit gemeinsamen Namensfragmenten anzeigen lassen (ls) oder löschen (rm).

Dateien im HOME Verzeichnis:

```
Datei  Datei20  Datei23  Datei24  Datei7  DateiHallo  DateiWas  Detei8  Detei9
DateiA Datei2A  DateiWas Datei789
```

∗ Beliebige Anzahl an Zeichen

? Genau EIN beliebiges Zeichen

[] Klasse/Bereich von Zeichen

```
ls Dat*
Datei  Datei20  Datei23  Datei24  Datei7  DateiHallo  DateiWas DateiA Datei2A
```

Ausgabe sind ALLE Dateien und Verzeichnisse die mit `Dat` beginnen (Linux unterscheidet Groß- und Kleinschreibung)!

```
ls Datei?
Datei7 DateiA
```

Ausgabe sind ALLE Dateien und Verzeichnisse die mit `Datei` beginnen und danach **GENAU EIN WEITERES** Zeichen haben!

```
ls Datei[7-9]
Datei7

ls Datei[A-Z]
DateiA

ls Datei[1-9A-Z]
Datei7 DateiA

ls Datei[1-9][1-9]
Datei23 Datei24

ls Datei[1-9][1-9][1-9]
Datei789

ls Datei[a-zA-Z]*
DateiWas DateiHallo DateiA
```

[] ersetzt auch EIN Zeichen, aber hier kann man angeben, ob es ein Buchstabe oder eine Zahl ist.

```
[a-z] irgendein Kleinbuchstabe

[A-Z] irgendein Großbuchstabe
```

```
[0-9] irgendeine Ziffer
```

Man kann auch mehrere Bereiche vermischen:

```
[a-zA-Z] irgendein Buchstabe

[0-9A-Z] irgendeine Zahl oder irgendeine Ziffer

[:$] ist auch möglich
```

6.0.7 mv

mv - move (rename) files and directories

Mit mv werden Dateien oder Verzeichnisse verschoben, d.h. ans Ziel kopiert und dann wird das Original gelöscht. Es wird in der Regel NICHT gefragt, ob Dateien oder Verzeichnisse die schon existieren überschrieben werden sollen oder nicht.

Option:

-i, –interactive fragt nach, ob überschrieben werden soll oder nicht

```
mv Datei1 Datei2
    Hier Wird Datei1 in Datei2 umbenannt.

mv Datei1 Verzeichnis/
    Datei1 wird in Verzeichnis bewegt

mv Datei Dateien/DateiNeu
    Bewegt die Datei ins Verzeichnis Dateien und erstellt dort die neue DateiNeu oder
    überschriebt falls vorhanden

mv Dateien/ DateienAlt/
    Verzeichnis Dateien wird in DateienAlt umbenannt

mv Dateien/ ~/DateienAlt/
    Bewegt das Verzeichnis Dateien in das Verzeichnis DateinAlt
```

6.0.8 cp

cp - copy files and directories

Damit kopiert man Dateien oder Verzeichnisse.

Option

-i
Nachfragen beim Überschreiben

```
cp Datei Datei2
    kopiert Datei auf die Datei2, wenn Datei2 existiert wird sie überschrieben, wenn nicht wird
```

```
    sie erstellt

cp Datei Dateien/
    Mit dem / am Ende wird cp gesagt, daß Dateien ein Verzeichnis ist und die Datei wird in das
    Verzeichnis kopiert sofern es existiert. Existiert es nicht gibt es eine Fehlermeldung.
    Man kann / weglassen, wenn man sicher ist, daß das Verzeichnis existiert.

cp Datei Dateien/DateiNeu
    Kopiert die Datei ins Verzeichnis Dateien und erstellt dort die neue DateiNeu oder
    überschriebt falls vorhanden
```

6.0.9 find

find ohne Optionen gibt den Inhalt des momentanen Verzeichnisses und ALLER Unterverzeichnisse an.

6.0.10 locate

```
locate Dateinamenteil
```

Sucht nach ALLEN Dateien die Dateinamenteil in ihrem Namen stehen haben. locate fügt automatisch ein $*$ vor und hinter dem Argument Dateinamenteil ein.

7 3

7.0.1 Kompressionstools

Beim Komprimieren wird die Dateigröße verringert. Das kann man sich so vorstellen: Häufig auftauchende Muster in der Datei werden mit einem kleinerem Muster ersetzt.

Da gibt es auf den meisten Systemen gzip, bzip2 und xz. Auf älteren Systemen oder sehr abgespeckten Systemen ist evtl. nur eines dieser Programme installiert.

Alle benutzen unterschiedliche Algorithmen, die NICHT zueinander kompatibel sind. D.h. man braucht zum Entkomprimieren die korrekten Werkzeuge. Diese sind natürlich bei der Installation mit installiert worden.

Bei allen wird der Befehl mit dem Dateinamen, der zu komprimierenden Datei, aufgerufen.

```
gzip Bigfile1
bzip2 Bigfile2
xz Bigfile3
```

Die Datei wird komprimiert und dann wird sie umbenannt. Es wird ans Ende eine Endung hinzugefügt, die angibt mit welchem Programm die Datei komprimiert wurde.

```
gunzip Bigfile1.gz
bunzip2 Bigfile2.bz2
xzunzip Bigfile3.xz
```

Damit werden die Dateien entkomprimiert.

Alle drei Kompressionswerkzeuge arbeiten verlustfrei. D.h. nach dem Entkomprimrieren ist der Originalzustand wiederhergestellt.

Verlustbehaftete Kompression kommt sehr häufig bei Multimediadateien vor. Dort ist der Qualitätsverlust entweder verschmerzbar im Vergleich zum Platzgewinn oder er ist unbemerkbar.

Es gibt weitere Hilfsprogramm wie zcat, bzcat und xycat mit denen man komprimierte Dateien verarbeiten kann OHNE diese zu entkompromieren.

7.0.2 Archivierungsprogramm

Will man Ordner (directory) mit den enthaltenen Dateien komprimieren, dann braucht man ein Archivierungsprogramm, weil Verzeichnisse nicht komprimiert werden können. Auf Linux ist das tar. Auf Windows sind WinZip, WinRar und 7zip.

```
tar -cf Archivname QUELLE
    Hier wird aus den Datei(en) oder Verzeichnis(sen) EINE Archivdatei erstellt mit Namen
    Archivname
```

Man darf hier das - weglassen. Die Archivdatei wird im Ordner in dem der Befehl aufgerufen wurde erstellt. Man kann natürlich wie bei den meisten anderen Befehlen auch diesen Ort wählen.

```
tar -cf ~/Archives/A/alt.tar Dateien/
    Das Archiv alt.tar wird im Ordner /home/BENUTZERNAME/Archives/A/ erstellt.
```

Zum entpacken nimmt man die Option x

```
tar -xf Archivname
    Entpackt den Inhalt in dem Verzeichnis in dem dieser Befehl aufgerufen wurde.
```

Beim Erstellen des Archivs kann man auch sofort dieses komprimieren. Dazu gibt es die Optionen z, j und J.

z wird zum Komprimieren mit gzip verwendet. j ist für bzip2 und J ist für xz.

```
tar -czf Archivname.tar.gz Quelle
tar -cjf Archivname.tar.bz2 Quelle
tar -cJf Archivname.tar.xz Quelle
```

WICHTIG: f MUSS am Ende der Optionen stehen. Die Endungen werden NICHT automatisch erstellt, sondern müssen vom Nutzer selber hinzugefügt werden. Sie sind dazu da anzugeben mit welchem Programm archiviert wurde und mit welchem Programm komprimiert wurde.

```
tar -tf Archivname
    zeigt den Inhalt des Archivs an


tar -uf Archivname Quelle
    Hängt die Quelle hinten an das Archiv an. Ältere Versionen einer Datei werden durch neuere
    ersetzt. Man kann hier NICHT gleichzeitig komprimieren.


tar -uzf erzeugt eine Fehlermeldung.
```

Wenn man mit Windows zip Dateien arbeiten muss, dann muß man das zip Paket installieren.

```
zip -r zipfile.zip Quelle
    hiermit erstellt man eine Windows zip Datei

unzip entpackt man eine Windows zip Datei
```

8 Tag 3

8.0.1 grep - global regular expression print

```
grep Muster Suchort
```

Suchen nach Mustern in Dateien. grep ist kurz für "global regular expression print". Mit der Option -E kann man erweitere Suchen durchführen.

Beispiel: ls -l | grep "st"

Alle Dateien im Verzeichnis mit "st" im Namen werden ausgegeben, alle anderen nicht!

-i

```
Groß- und Kleinschreibung wird ignoriert
```

-r

```
sucht zusätzlich in allen Unterverzeichnissen
```

-v

```
negiert das Muster, man sucht nun nach Mustern die NICHT der Angabe entsprechen
```

-c

```
zählt die Anzahl an Zeilen mit einem oder mehreren Treffern
```

-E

```
erweiterte reguläre Ausdrücke sind möglich
```

grep a *

```
Sucht in allen Dateien, wegen *, in diesem Verzeichnis nach dem Buchstaben "a". Es
werden ALLE ZEILEN mit Treffern ausgegeben und die Treffer werden markiert.
```

Ausgabe:

Datei: hier ist ein a in dem Satz

8.0.2 RegEx - reguläre Ausdrücke

.

```
Sucht nach einem einzelnen Zeichen (siehe ? bei Globbing)
```

[abcABC]

```
sucht nach einem beliebigen Zeichen in der Klammer, also hier sucht es nach
a, b, c, A, B oder C JEDER der 6 Buchstaben zählt als Treffer und die Zeile
```

wird ausgegeben.

[^abcABC]

sucht NICHT nach einem beliebigen Zeichen in der Klammer, also wird nach ALLEM gesucht AUSSER den 6 Buchstaben.

[a-z]

sucht nach einem Kleinbuchstaben

[^a-z]

sucht NICHT nach einem Kleinbuchstaben

sun|moon

Sucht nach sun oder moon

^

Zeilenanfang

$

Zeilenende

Klassen Die gleichen Klassen gibt es auch beim Globbing. In Python findet man die meisten als string-Methoden. Sie beginnen dort mit einem is: isalnum(), isalpha(), ...

[:alnum:]

Buchstaben und Zahlen.

[:alpha:]

Groß- oder Kleinbuchstaben.

[:blank:]

Leerzeichen und Tabs.

[:cntrl:]

Steuerzeichen, z.B. Backspace, Glocke, NAK, Escape.

[:digit:]

Zahlen (0123456789).

[:graph:]

Alle graphischen Zeichen (alle Zeichen außer ctrl und Leerzeichen)

[:lower:]

Kleinbuchstaben

[:print:]

Druckbare Zeichen (alnum, punct und das Leerzeichen).

[:punct:]

Interpunktionszeichen, d.h. !, &, ".

[:space:]

Whitespace-Zeichen, z.B. Tabs, Leerzeichen, Zeilenumbrüche.

[:upper:]

Großbuchstaben

[:xdigit:]

Hexadezimale Zahlen (normalerweise 0123456789abcdefABCDEF)

Quantifier $*$

Null oder mehr Vorkommen des Musters vor $*$

Hierfür braucht man die Option -E:

+

Ein oder mehr Vorkommen des Musters vor $+$

?

Null oder ein Vorkommen des Musters vor ?

Dateiinhalt:
aaa

aa bb cc
bbb
aaabbb
aaabbb22
aabbcc1
Das hier ist eine Emailaddresse: f.e@gibtnicht.com
Webpage: https://www.gibtnicht.com
Webpage: https://de.gibtsnicht.net
www.gibtsnicht.de
z 123 ist eine Zahl
Noch eine Zeile mit ß und :?
a

grep ^$ Dateiinhalt

gibt die leeren Zeilen aus zwischen dem Zeilenanfang und -ende ist NICHTS im Muster

grep ^.$ Dateiinhalt

gibt die Zeile mit GENAU EINEM Zeichen aus.

grep a Dateiinhalt

`alle Zeilen mit dem Buchstaben a, egal ob er einmal oder mehrmals auftaucht.`

grep ab Dateiinhalt

`hier werden nur die Zeilen, die a und b in GENAU DER REIHENFOLGE enthalten ausgegeben`

grep [ab] Dateiinhalt

`alle Zeilen mit a oder b werden ausgegeben`

grep [[:alnum:][:blank:]] Dateiinhalt

`Zeilen mit Sonderzeichen wie :, /, ?, ... werden ausgegeben`

grep b[0-9]* Dateiinhalt

`gibt alle Zeilen die b enthalten aus und markiert auch die Ziffern sofern sie DIREKT hinter b stehen`

grep -E b[0-9]+ Dateiinhalt

`gibt alle Zeilen mit b direkt gefolgt von mindestens einer Ziffer aus`

grep -E "[1]+$" Dateiinhalt

`gibt alle Zeilen mit NUR Buchstaben aus`

grep -E "[2]*$" Dateiinhalt

`gibt alle Zeilen mit NUR Buchstaben aus und Leerzeilen`

grep -E "page|.com" Datei

`gibt alle Zeilen mit page oder .com zurück`

9 Skripte

Ein Skript ist eine ausführbare Datei mit zeilenweise Code. Wenn man sie aufruft werden die Befehle zeilenweise ausgeführt.

Dazu muß man die Rechte der Datei erweitern. Das geht mit dem Befehl `chmod`:

chmod +x skript gibt der Datei skript die Execute-Rechte

Jetzt kann man sie mit `./skript` ausführen. Man benötigt "./", weil die Bash-Shell NICHT im Homeverzeichnis nach ausführbaren Dateien sucht. Man könnte auch die Pfadvariable PATH ändern oder die Datei in eins der Verzeichnisse, die im PATH stehen, verschieben.

[1] [:alpha:]
[2] [:alpha:]

9.0.1 Variablen

Zahlen, Buchstaben und Unterstriche sind als Variablennamen erlaubt. Linux unterscheidet Groß- und Kleinschreibung. "user" und "User" sind unterschiedliche Variablen. Zahlen sind GANZ am Anfang von Variablennamen NICHT erlaubt.

Will man den Wert der Variablen ausgeben, dann muß man `$Variablenname` oder `${Variablenname}` verwenden.

Alle Variablen werden als Strings angesehen, was Berechnung unmöglich macht. Wie man da Zahlen draus macht wird hier nicht durchgenommen.

Will man in den Variablen Sonderzeichen, z.B. Leerzeichen, speichern, dann muß man entweder diese mit escapen oder " " benutzen. Bei der Auflösung der Variablen ist die Art der Anführungszeichen wichtig.

'$username'gibt‘username‘ aus

"$username" gibt den Wert gespeichert in username aus

shebang In der ersten Zeile von vielen Linuxskripten findet man den shebang. Das ist eine Zeile, die den Interpreter benennt und den absoluten Pfad zur Shell, hier Bash, enthält, der für das Skript benutzt werden soll.

```
#! /usr/bin/sh
```

Sollte der Pfad nicht stimmen, gibt es eine Fehlermeldung!

9.0.2 Argumente

Will man beim Ausführen des Skriptes Argumente übergeben, dann muß man diese im Skript in Variablen speichern. Dazu gibt es das Sonderzeichen "$ + Zahl". Für das erste Argument nimmt man also "$1", $frdasZweiteistes$"$2" und so weiter.

9.0.3 if

Man kann die Anzahl der übergebenen Argumente mit dem Sonderzeichen $# auslesen. Dann kann man damit über if-Statements den Ablauf des Skripts ändern.

```
skript.sh
#! /bin/bash

username=$1
username2=$2

if [ $# -eq 1 ]


then username=\(1
        echo "\)username''
elif {[} \(# -eq 2 ]
then
    echo "\)username und \$username2''
```

```
else echo ``Falsche Anzahl an Argumenten: mehr als 2''
fi
```

WICHTIG: Nach der [und vor der] im if Statement MUSS mindestens EIN Leerzeichen sein. Sonst funktioniert das Skript nicht.

Mit `fi` wird das if Statement beendet! Das Einrücken ist nur für die Lesbarkeit und hat keine Funktion. Und man kann auch nicht einfach {} nehmen, um die Befehle im if zu markieren.

Aufruf: `./skript.sh Dennis` Druckt einfach Dennis aus!

`./skript.sh Dennis Florian Sarah` Druckt "Falsche Anzahl an Argumenten: mehr als 2" aus

-eq

`gleich`

-ne

`Nicht gleich`

-gt

`Größer als`

-ge

`Größer oder gleich`

-lt

`Kleiner als`

-le

`Kleiner oder gleich`

9.0.4 exit codes

Mit der Variablen $? kann man sich den `exit code` des letzten gelaufenen Programms anschauen. Ist es ohne Fehler durchgelaufen, ist er 0. Gab es Fehler ist er 1 oder 2.

Mit dem Befehl `exit` kann man den exit code setzen. Das Skript wird beim Erreichen eines exit Befehls beendet. Hinter exit muß eine positive ganze Zahl stehen.

Man kann dann die Variable in Skripten benutzen, um automatisiert auf Fehler zu reagieren.

9.0.5 mehrere Argumente

In $∗ oder $@ werden alle Argumente gespeichert. Das ist wie *args bei Funktionen in Python. $∗ und $@ erstellen das folgende Array, wenn man das Skript mit "Dennis Florian Sarah" aufruft:

0	1	2
Dennis	Florian	Sarah

Namen="Florian Sarah Dennis" <- Hiermit wird ein Array erzeugt! Die Elemente werden durch Leerzeichen getrennt.

Der Unterschied zwischen $* und $@ wird erst sichtbar, wenn man Anführungszeichen benutzt. "$@" verhält sich wie OHNE Anführungszeichen, aber "$*" wandelt die Einträge des Arrays in EINEN String um. In einer for-Schleife würde also diese NUR EINMAL durchlaufen werden.

9.0.6 for Schleifen

```
#! /bin/bash

names="Dennis Florian Sarah"

for name in $names
do
echo "$name"
done

for name in "$@"
do
echo "$name mit @"
done

for name in "$*"
do
echo "$name mit *"
done
```

Ruft man das Skript nun mit den Argumenten "Dennis Florian Sarah" auf, erhält man:

```
Dennis
Florian
Sarah
Dennis mit @
Florian mit @
Sarah mit @
Dennis Florian Sarah mit *
```

Die ersten drei Zeilen kommen von der names Variablen, die ganz oben im Skript erstellt wurde. Dann kommen die Namen aus dem Argument gespeichert in $@. Wegen dem Anführungszeichen um $* wird die letzte for-Schleife nur einmal durchlaufen und der Wert von "$*" ist der String "Dennis Florian Sarah"

Würde man die Anführungszeichen weglassen in den for-Schleifen, dann gäbe es KEINEN Unterschied zwischen $@ und $*.

```
#! /bin/bash

if [ $# -eq 0]
then
    echo "Mindestens einen Namen angeben!"
```

```
    exit 1
else
    echo -n "Hallo $1"
    shift
    for name in $@
        do
        echo -n ", und $name"
    done
    echo "!"
    exit 0
 fi
```

`shift` entfernt den ersten Eintrag in der Argumentenliste. `-n` bei echo unterdrückt den Zeilenumbruch.

9.0.7 Überprüfung der Eingaben

Man kann im Skript mit grep und regulären Ausdrücken die übergebenen Argumente überprüfen. Z.B. schauen, ob sie alle nur aus Buchstaben bestehen.

```
#! /bin/bash

if [ $# -eq 0]
then
    echo "Mindestens einen Namen angeben!"
    exit 1
else
    echo $1 | grep "^[A-Za-z]*$" > /dev/null #Checkt ob alles Buchstaben sind
if [ $? -eq 1 ]
    then
        echo "ERROR: Zahl im Namen gefunden!"
        exit 2
    else
        echo -n "Hallo $1"
fi
    shift  #Entfernt das erste Element des Argumentenarrays
    for name in $@
        do
        echo $name | grep "^[A-Za-z]*$" > /dev/null #Checkt ob alles Buchstaben sind
        if [ $? -eq 1 ]
        then
            echo "ERROR: Zahl im Namen gefunden!"
            exit 2
        else
            echo -n ", und $name"
        fi
        done
```

```
    echo "!"
    exit 0
fi
```

10 4

Beim Starten des OS wird zuerst der Kernel geladen und dann der Prozess mit der Process ID (PID) 1 gestartet. Das ist `systemd`. Er startet dann ein Kette an weiteren Prozessen. Aktive Prozesse bekommen vom OS Ressourcen zugeordnet (Tastatur, Maus, Speicher (RAM), ...).

Im Verzeichnis /boot/ befinden sich die Kerneldateien:

config-4.9.0-9-amd64 Konfigurationsdatei für Einstellungen wie Optionen und Module, die zusammen mit Kernel kompiliert wurden

initrd-img-4.9.0-9-amd64 Hilft dem Bootvorgang indem ein temporäres Dateisystem geladen wird

System-map-4.9.0-9-amd64 Speicheraddressorte für Kernelsymbolnamen, ändert sich bei jedem Booten

vmlinuz-4.9.0-9-amd64 komprimierter, selbstextrahierender Kernel vm = virtual memory (virtueller Speicher) z = komprimiert

grub Konfigurationsverzeichnis für den grub2 Bootloader

10.0.1 /proc/

Ist ein virtuelles Verzeichnis im Arbeitsspeicher. Jedes Verzeichnis mit einer Zahl als Namen enthält Informationen über den Prozess mit derselben Zahl als PID. Z.B. enthält der Ordner /proc/1/ alle Informationen über den Prozess, der das OS ganz am Anfang startet.

Die Datei cpuinfo im Verzeichnis /proc/ enthält Informationen über die CPU.

Mit `cat /proc/cpuinfo` kann man sie anschauen.

/proc/cmdline Strings, die beim Booten an den Kernel übergeben werden

/proc/modules Liste der geladenen Module

10.0.2 /proc/sys

Enthält Verzeichnisse in denen Konfigurationsdateien zu finden sind. Häufig enthalten diese Dateien nur eine 0 oder 1. Diese Dateien wirken dann wie ein Schalter.

/proc/sys/net/ipv4/ip_forward enthält eine 0. D.h. Pakete werden NICHT weitergeleitet, unser PC ist kein Router

Die Verzeichnisse sind nach Kategorien eingeteilt.

dev Geräte

fs Dateisystem

kernel Kernel

net Internet

. . . . .

10.0.3 /dev

Alle Geräte sind hier als Dateien gespeichert.

Der Inhalt der `sda` Datei ist der Inhalt der ersten Festplatte!

sd - storage device

Die Festplatten werden dann mit Buchstaben durchnummeriert: a, b, c

Hat die Festplatte Partitionen, dann wird noch eine Zahl hintendrangehängt.

sda2 ist die 2te Partition der ersten Festplatte.

Festplatten schreiben und lesen Daten in Blöcken, deswegen nennt man sie block devices. Gibt man den Befehl `ls -l` in /dev/ ein, dann sieht man ganz links ein "b" vor den Rechten stehen. Das zeigt eben das Blockdevice an.

Bei tty steht dort ein "c". Das ist ein Characterdevice. Dort werden die Daten seriell, Zeichen für Zeichen, eingelesen und geschrieben. Weitere Charactergeräte sind Tastatur, Konsole, ...

/dev/zero man kann damit eine Reihe an Nullzeichen erstellen

/dev/null Mülleimer, verwirft alles was dorthin geschickt wird

/dev/urandom erzeugt Pseudozufallszahlen

10.0.4 /sys

/sys ist ein virtuelles Dateisystem.

/sys enthält Informationen über Hardware-Geräte, damit Prozesse mit ihnen interagieren können. Z.B. findet man in /sys/block/sda/removable entweder eine 1 oder eine 0, je nachdem ob man die Blockdevice (Festplatte) sda entfernen kann oder nicht. Informationen über die Internetverbindung findet man in /sys/class/net/eth0/. Die MAC-Adresse ist in der Datei "address".

10.0.5 Speicher

Mit dem Befehl `free` kann man sich Informationen über den Arbeitsspeicher (RAM) anzeigen lassen. RAM besteht in der Regel aus einer Reihe an Mikrochips, die auf einer oder beiden Seiten einer Bank befestigt sind. Jede Speicherzelle in dem Speichermodul hat eine Addresse.

SWAP: Ist ein Bereich auf der Festplatte der benutzt wird, wenn der RAM zu voll wird. Festplatten sind aber SEHR VIEL LANGSAMER als RAM Speicher.

Der virtuelle Speicher ist die Summe aus RAM und SWAP.

```
free --si -h
               total        used        free      shared  buff/cache   available
Mem:            15G        628M         14G        3.0M        411M         15G
Swap:          4.1G          0B        4.1G
```

Mit `free --si -h` werden K (Kilo), M (Mega), G (Giga) benutzt. D.h. die Bytes werden in Tausender, Millioner, Milliardener Packungen angegeben. Mit `free -m` würden sie in "mebi bytes" angegeben. Das sind "1024 * 1024" Bytes.

```
free -m
               total        used        free      shared  buff/cache   available
Mem:           15962         626       14924           3         411       15069
Swap:           4096           0        4096
```

total Gesamtmenge

used benutzt

free benutzbarer freier Speicher

shared vom tmpfs (temporäres Dateisystem im RAM) benutzter Speicher

buff/cache Von Kernelpuffern, Seitencache und Slabs benutzter Speicher

available geschätzte benutzbare Menge an Speicher für neue Prozesse

11 Prozesse

Prozesse haben eine EINDEUTIGE Prozess ID (PID) und eine ID für den Elternprozess (Parent Process, PPID).

Mit dem Befehl `top` kann man sich die momentan laufenden Prozesse anschauen.

```
 top
top - 15:33:55 up  7:35,  1 user,  load average: 0.00, 0.02, 0.00
Tasks:  37 total,   1 running,  36 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.0 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  5.6/15962.3  [                                                ]
MiB Swap:  0.0/4096.0   [                                                ]

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  416 root      20   0  795400  84168  24236 S   0.0   0.5   0:16.39 python3.10
  232 root      20   0 2133576  48276  20284 S   0.0   0.3   0:03.05 snapd
  558 root      20   0   44184  37656  10180 S   0.3   0.2   1:33.45 python3
17554 root      20   0  293008  22092  17136 S   0.0   0.1   0:00.12 packagekitd
  312 root      20   0  107224  20976  12796 S   0.0   0.1   0:00.07 unattended-upgr
  229 root      20   0   30104  19232  10440 S   0.0   0.1   0:00.07 networkd-dispat
   72 root      19  -1   47752  15040  14008 S   0.0   0.1   0:00.17 systemd-journal
  130 root      20   0  377284  14292    332 S   0.0   0.1   0:00.65 snapfuse
  151 root      20   0  302520  14072    356 S   0.0   0.1   0:01.06 snapfuse
    1 root      20   0  167188  12596   8304 S   0.7   0.1   3:10.12 systemd
  235 root      20   0  392688  12476  10324 S   0.0   0.1   0:00.08 udisksd
  172 systemd+  20   0   25540  12456   8264 S   0.0   0.1   0:00.16 systemd-resolve
```

Man sieht ganz oben einige Informationen über das Gesamtsystem: CPU Auslastung und Speicherverbrauch

Mit “M” sortiert man nach Speicherverbrauch, mit “N” nach der Nummer des Prozesses, “T” nach Laufzeit und “P” nach %CPU. Mit “R” kann man die Reihenfolge umdrehen.

Für eine statische Momentaufnahme der Prozesse in der aktuellen Shell kann man `ps` benutzen.

```
 ps
    PID TTY          TIME CMD
    478 pts/0    00:00:00 bash
 109808 pts/0    00:00:00 ps
```

Angegeben werden die PID, das Terminal (TTY), in dem der Prozess läuft, CPU-Zeit, die der Prozess benötigt (TIME) und welcher Befehl ihn gestartet hat (CMD)

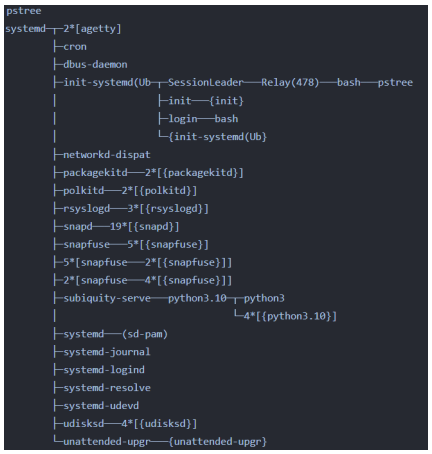
Mit der Option -f werden mehr Informationen gedruckt:

```
 ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
user1        478     477  0 07:58 pts/0    00:00:00 -bash
user1     110093     478  0 15:39 pts/0    00:00:00 ps -f
```

Zusätzlich die PPID, User ID (UID), Startzeit (STIME) und C ist %CPU. Man sieht, daß der Prozess 478 den Prozess 110093 gestartet hat. Eine gewisse graphische Darstellung dieser Tatsache erhält man mit `ps -uf`:

```
ps -uf
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
user1        546  0.0  0.0   6124  4760 pts/1    S+   07:58   0:00 -bash
user1        478  0.0  0.0   6344  5332 pts/0    Ss   07:58   0:00 -bash
user1     110140  0.0  0.0   7912  3508 pts/0    R+   15:39   0:00  \_ ps -uf
```

Graphische Darstellung mit pstree:



312

12 Netzwerk

Layers eines Netzwerks:

Link Layer (Verbindungsschicht): Geräte sind direkt miteinander verbunden (Heimnetzwerk) Die Addressen sind hier die MAC (Media Access Control) Adressen. Diese sind NICHT global eindeutig, weswegen man mit ihnen NICHT ins Internet kann.

Network Layer (Netzwerkschicht) Will man über den Link Layer mit Geräten kommunizieren, dann braucht man einen Router mit dem man dann aus dem Link Layer rauskommt.

IPv4 oder/und IPv6 für die Addressen im Internet.

Application Layer (Anwendungsschicht) Programme verbinden sich hier

Der Router hat quasi die Hausadresse und die Geräte sind dann die Mietparteien. Die Verbindungen zwischen den Routern sind dann die Straßen.

Daten werden als Pakete verschickte. Dazu werden sie in kleine Blöcke zerlegst und bekommen einen Header, der besagt, woher das Paket kommt und wohin es will. Trifft das Pakete auf einen Router, prüft dieser ob das Pakete zu seinem Netzwerk will, wenn ja wird es reingelassen, wenn nein wird es weitergeschickt (Routing Tabelle).

Broadcast Pakete werden an alle Empfänger in einem Netzwerk geschickt und deswegen immer angenommen.

`ip link show` zeigt die Link Adressen aller Netzwerkgeräte an.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1492 qdisc mq state UP mode DEFAULT
group default qlen 1000
link/ether 00:15:5d:b7:e4:f2 brd ff:ff:ff:ff:ff:ff
```

Die “00:00:00:00:00:00” ist die MAC Adresse vom Loopback Device. eth0 ist der Ethernet-Gerät mit MAC “00:15:5d:b7:e4:f2”.

12.1 IPv4

Die Adressen und das Routing werden im Internet Protocol (IP) definiert. Es gibt etwa 4 Milliarden IPv4 Adressen, welche 32 Bit lang sind. Diese sind global eindeutig. Mittlerweile reichen diese 4 Milliarden aber nicht mehr aus.

IP Adresse als 32 Bit Folge:

```
11000000.10101000.00000010.00000001
```

Menschlich lesbarer ist:

```
192.168.2.1
```

Man hat hier bis zu 256 Addressen, weil die letzte Stelle von 0 bis 255 gehen kann.

12.1.1 Subnetze

Mit den Subnetzen wird die IP Adresse in die Adresse des Netzwerks und die Adresse des Zielgeräts zerlegt. Die Subnetzmaske besteht aus 1 und 0rn.

Häufig ist die Subnetzmaske einfach:

```
11111111.11111111.11111111.00000000
```

oder

```
255.255.255.0
```

```
IP      11000000.10101000.00000010.00000001
Subnet  11111111.11111111.11111111.00000000

IP     192.168.002.1
Subnet 255.255.255.0

IP      11000000.10101000.00000010.00000001
Subnet  11111111.11111111.11110000.00000000

IP     192.168.002.1
Subnet 255.255.240.0
```

Dort wo die Subnetzmaske > 0 ist, befindet sich die Netzwerkadresse. Dort wo sie 0 ist wird das Gerät in diesem Netzwerk adressiert.

Bei der menschlich lesbarer Variante wird's problematisch, wenn man NICHT 255 oder 0 hat.

Eine weitere Art die Subnetzmaske anzugeben ist via Classless Inter Domain Routing (CIDR). Dabei gibt man an wieviele 1 die Maske enthält. Diese beginnen IMMER ganz links und gehen ununterbrochen nach rechts.

```
IP     192.168.002.1
Subnet 255.255.240.0
Subnet 11111111.11111111.11110000.00000000  <= 20mal die 1
MIT CIDR:
IP     192.168.002.1/20
```

Bei einer Subnetzmaske der Länge 20 hat man 4096 Adressen für die Geräte im eigenem Netzwerk zur Verfügung.

Von den 4 Milliarden Adressen sind einige NICHT benutzbar: alle Broadcast, Multicast (Daten werden nicht an ALLE in einem Netzwerk geschickt, sondern nur an bestimme IP-Adressen: 224.0.0.0 bis 239.255.255.255, verwendet für Streaming, Online-Gaming, Video-Konferenzen) und private Adressen.

10.0.0.0/8 - 10.255.255.255/8

172.16.0.0/12 - 172.31.255.255/12

192.168.0.0/16 - 192.168.255.255/16

Der Router verbirgt die ganzen internen Adressen und vertritt das Link Layer mit einer externen IP Adresse. Das wird über die Network Address Translation (NAT) gemacht. Diese Masquerading hat aber keine Sicherheitsfunktion, weil die Pakete NICHT überprüft werden. Stimmt die Zieladresse im Header des Paketes zu diesem Netzwerk, dann läßt der Router es ins Netzwerk rein.

12.1.2 DHCP - IP Adressvergabe

Windows und Linux verteilen die IP Adressen in der Regel automatisch mit dem "Dynamic Host Configuration Protocol" (DHCP). Dabei steuert ein zentraler Server die Vergabe der Adressen und die Weitergabe von einigen wichtigen Informationen, wie z.B. Adresse von DNS-Servern, Standard-Router oder IP-Adressen zum Starten eines OS vom Netzwerk aus.

Manuell kann man IP Adressen über `ip addr add` vergeben:

```
sudo ip addr add 192.168.0.5/255.255.255.0 dev eth0
```

und löschen:

```
sudo ip addr delete 192.168.0.5/255.255.255.0 dev eth0
```

Struktur des Befehls:

```
ip addr add/delete ADRESSE/SUBNETMASK dev GERÄT
```

Anzeigen kann man die Adresse mit `ip addr show`!

Mit `ping` kann man Verbindungen prüfen. ping sendet ein Echo Request aus und wenn die Verbindung besteht, dann kommt eine Antwort zurück. Sollten Pakete verloren gehen, dann ist die Verbindung schlecht.

Wenn man ping aufruft OHNE "-c ZAHL", dann läuft es ununterbrochen weiter bis man es über einen Keyboard Interrupt (STRG + C) beendet. Mit "-c ZAHL" werden nur ZAHL Pakete gesendet.

12.1.3 Routing Tabelle

Jedes Gerät hat eine und sie besagt, wohin man DIREKT gehen kann und welche Netzwerke man über einen Router erreichen kann.

Wird ein Paket verschickt, wird zuerst in der Routingtabelle nachgeschaut, ob man das Ziel direkt erreichen kann oder nicht. Ist es erreichbar wird es einfach dahin geschickt. Wenn nicht wird es an an einen Router weitergeschickt. Der schaut in seiner Tabelle nach und dann wiederholt sich alles bis das Paket ankommt oder verworfen wird.

Sollte das Paket nicht zu einer der Routen in der Tabelle passen, wird es auf die Default Route geschickt. Beim Heimnetzwerk ist der Weg vom einzigen Router zum Router des Internet Service Providers (ISP) die Default Route.

```
ip route show
default via 172.18.160.1 dev eth0 proto kernel
172.18.160.0/20 dev eth0 proto kernel scope link src 172.18.162.24
```

Default Route geht über den "Router" mit der Adresse 172.18.160.1.

Die IP Adresse des WLS Containers ist 172.18.160.0 und die Geräteadresse ist 172.18.162.24.

Will man eine Default Route hinzufügen, dann kann man `sudo ip route add default via 192.168.0.1` benutzen.

Weitere Informationen zur Ausgabe von ip route show

In der von dir angegebenen Ausgabe von `ip route show` in einer WSL (Windows Subsystem for Linux) Umgebung, sind einige wichtige Netzwerkdetails zu sehen. Lass uns die einzelnen Zeilen und die darin enthaltenen Adressen aufschlüsseln:

12.1.4 1. Default Route

```
default via 172.18.160.1 dev eth0 proto kernel
```

- `default`: Dies ist die Standardroute. Wenn das System ein Paket senden möchte, dessen Ziel-IP-Adresse nicht spezifisch in der Routing-Tabelle enthalten ist, wird es über diese Route geschickt.
- `via 172.18.160.1`: Das ist die IP-Adresse des Standard-Gateways. Das Gateway ist der Router, der als Ausgangspunkt für den Verkehr ins Internet oder in andere Netzwerke dient.
- `dev eth0`: Das Netzwerkinterface `eth0` wird für diese Route verwendet. `eth0` ist in diesem Fall das erste Ethernet-Interface in deinem System.
- `proto kernel`: Dies gibt an, dass diese Route automatisch vom Kernel hinzugefügt wurde.

Zusammenfassung: Wenn dein WSL-Container (oder Linux-Subsystem) Daten an eine IP-Adresse senden muss, die sich nicht im lokalen Netzwerk befindet, werden die Daten an das Gateway `172.18.160.1` gesendet, das sie dann weiterleitet.

12.1.5 2. Lokales Netzwerk (Subnetz)

```
172.18.160.0/20 dev eth0 proto kernel scope link src 172.18.162.24
```

- `172.18.160.0/20`: Dies ist die Netzwerkadresse des Subnetzes, in dem sich dein WSL-Container befindet. Das `/20` bedeutet, dass die ersten 20 Bits der IP-Adresse das Netzwerk definieren. Dies ergibt eine Subnetzmaske von `255.255.240.0`. Damit reicht das Subnetz von `172.18.160.0` bis `172.18.175.255`.
- `dev eth0`: Wieder wird das `eth0` Interface für dieses Netzwerk verwendet.
- `proto kernel`: Diese Route wurde vom Kernel hinzugefügt.
- `scope link`: Dies bedeutet, dass die Route auf diesem Link (dem Interface `eth0`) gültig ist. Es wird also nur für Geräte innerhalb des gleichen Netzwerks verwendet.
- `src 172.18.162.24`: Dies ist die IP-Adresse deines WSL-Containers innerhalb dieses Subnetzes.

Zusammenfassung: Dein WSL-Container hat die IP-Adresse `172.18.162.24` im Subnetz `172.18.160.0/20`. Dieses Subnetz umfasst IP-Adressen von `172.18.160.0` bis `172.18.175.255`. Pakete, die an Ziele innerhalb dieses Subnetzes gesendet werden, bleiben im lokalen Netzwerk und verwenden das Interface `eth0`.

12.1.6 Gesamteinschätzung

In der WSL-Umgebung hat dein Container eine IP-Adresse `172.18.162.24` und ist Teil des Subnetzes `172.18.160.0/20`. Alle Verbindungen zu Zielen innerhalb dieses Subnetzes bleiben lokal, während alle anderen über das Gateway `172.18.160.1` ins weitere Netzwerk oder Internet gesendet werden.

Diese Konfiguration könnte darauf hindeuten, dass WSL ein virtuelles Netzwerk zwischen deinem Host-System (Windows) und dem WSL-Container eingerichtet hat, wobei das Subnetz

`172.18.160.0/20` und das Gateway `172.18.160.1` verwendet werden, um den Netzwerkverkehr zwischen ihnen zu verwalten.

12.1.7 IPv6

Bei IPv6 hat man 128 Bit für die Adressen und das sind definitiv genug für alle Geräte, so daß Masquerading nicht mehr nötig ist.

Die Adresse bestehen aus 8 Gruppen aus 4 Hexadezimalzahlen getrennt durch ":".

```
2001:0db8:0000:abcd:0000:0000:0000:7334     inet6 Adresse
```

oder

```
2001:0db8:0:abcd:0:0:0:7334
```

oder

```
2001:0db8:0000:abcd::7334
```

Man darf mehrere Blöcke mit Nullern EINMAL in so einer Adresse einfach weglassen.

Link Adresse:

```
fe80:0000:0000:0000:0215:5dff:feb7:e4f2/64  Link Adresse
fe80::0215:5dff:feb7:e4f2/64  Link Adresse
```

Insgesamt hat man 128 Bit davon werden die ersten 64 Bit als Routing-Präfix genommen und benutzt die Adressierung der Netzwerke. Subnetting bezieht NUR auf diese ersten 64 Bit. In der Regel werden da /48 oder /56 von ISP genommen.

Global Unique Address Die eindeutige Adresse für das Netzwerk

Unique Local Address Adressen für interne Netzwerke, damit die Geräte eine Adresse, selbst wenn es kein Internet gibt. Die ersten 8 Bit sind immer fc oder fd gefolgt von 40 zufällig erzeugten Bits.

Link Local Address Beginnt immer mit fe80 und ist immer nur für einen Link gültig. Sie werden verwendet um weitere Computer im Netzwerk zu finden oder um automatische Konfiguration anzufordern.

Die letzten 64 Bit sind dann für die Identifizierung der Geräte vorhanden. Sie werden "Schnitstellenkennung" genannt.

12.1.8 Adressvergabe

Es ist wieder automatisch oder manuell machbar. Für die automatische Verteilung gibt es aber zwei Möglichkeiten: DHCPv6 oder SLAAC (Stateless Address Autoconfiguration).

SLAAC: Benutzt das Neighbor Discovery Protocol. Damit können Geräte sich gegenseitig finden und Informationen abfragen. Informationen über das Netzwerk kommen vom Router und enthalten IPv6-Präfixe. Die Geräte hängen dann ihre Schnittstellenkennungen an. Der Router wird nicht informiert über die fertigen Adressen.

DHCPv6: DHCP erstellt um mit IPv6 zu funktionieren.

Erstellen einer IPv6 Adresse:

```
sudo ip addr add 2001:db8:0:abcd:0:0:0:7334/64 dev eth0
```

Anpingen:

```
ping -c 3 2001:db8:0:abcd:0:0:0:7334
```

Pingt man eine lokale LINK Adresse an, dann muß man ganz hinten den Interfacename (Gerät) mit Prozentzeichen "%eth0" anhängen:

```
ping -c 3 fe80::215:5dff:feb7:e4f2%eth0
```

12.1.9 DNS

Der DNS Server ist ein Telefonbuch in dem die Zuweisung des Namens zu seiner IP Adresse drinsteht.

DNS Hostnamen sind die ersten Zeichen der URL.

de.wikipedia.com

de ist der DNS-Hostname. wikipedia.com ist der Domainname.

Wenn man die URL in einen Browser eingibt, dann schickt dieser die URL an einen DNS-Resolver. Dieser findet dann die IP-Adresse und schickt den Browser dann dahin.

/etc/resolv.conf enthält den DNS-Resolver.

und in

/etc/hosts befinden sich DNS Einträge. Sollte also eine URL in der hosts Datei gefunden werden, dann wird der DNS-Server NICHT kontaktiert, sondern die IP aus der hosts genommen.

Mit dem Befehl `host` kann man die IP-Adresse von URL finden.

```
host www.example.com
```

`dig` liefert mehr Informationen.

```
dig www.example.com
```

12.1.10 Sockets

Programme verwenden Sockets um miteinander zu kommunizieren. Sind die Sockets auf verschiedenen Geräten, dann muß es ein Netzwerk zwischen den Geräten geben. Der Browser redet über Sockets mit dem Webserver.

Arten von Sockets:

UNIX - für Kommunikation von Prozessen auf demselben Gerät

TCP - sorgt dafür, daß die Pakete in der Reihenfolge ankommen in der sie geschickt werden.

UDP - fire and forget

TCP und UDP haben mehrere Ports zur Verfügung mit denen sie mehrere Sockets auf derselben IP Adresse ansprechen können. Ein Socket besteht aus der IP Adresse und der Portnummer.

Der Ausgangsport ist zufällig aber der Eingangsport ist häufig zweckgebunden. Port 80 ist HTTP, 443 HTTPS, SSH ist Port 22, FTP ist 21.

Mit `ss -t` kann man tcp ports anzeigen lassen, mit "-u" UDP und mit "-l" die Ports die gerade auf Action warten.

13 5

13.0.1 Benutzerkonten

Jeder Benutzer hat eine User ID (UID) und eine primäre Group ID (GID). Diese sind in der Regel dieselbe Zahl und sie beginnen für normale Benutzer bei 1000. Außerdem hat ein Nutzer ein Homeverzeichnis und eine Standard-Shell.

13.0.2 Systemkonten

UID/GID ist unter 1000, das Homeverzeichnis ist in der Regel NICHT in /home/ angelegt und es gibt keine gültige Standard-Shell. Das wird mit /sbin/nologin in /etc/passwd eingetragen.

Diese Konten sind dazu da Programme und Dienste laufen zu lassen, die keine Root-Rechte benötigen.

13.0.3 Servicekonten

Sie werden bei der Installation und Konfiguration von Diensten angelegt und haben eine UID/GID von > 1000. Das Homeverzeichnis ist in der Regel NICHT in /home/ angelegt.

13.0.4 Wechsel der login shell

Mit `chsh` kann man die login shell ändern.

Entweder über den interaktiven Modus, der startet wenn man NUR "chsh" aufruft, oder über "chsh -s PFAD_ZUR_SHELL".

```
id
uid=1000(user1) gid=1000(user1)
groups=1000(user1),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),
29(audio),30(dip),44(video),46(plugdev),116(netdev)
```

Mit dem Befehl `id` kann man sich Informationen über den jetzigen Nutzer anzeigen lassen. Die UID, primäre GID und weitere existierende Gruppen denen der Nutzer angehört.

13.0.5 Erfolgreiche Einlogvorgänge

Der Befehl `last` listet die erfolgreichen Einlogvorgänge auf und `lastb` listet die nicht erfolgreichen Vorgänge.

```
last
user1    pts/1                         Fri Aug 23 07:53   still logged in
reboot   system boot  5.15.153.1-micro Fri Aug 23 07:53   still running
user1    pts/1                         Fri Aug 23 06:58 - crash  (00:54)
reboot   system boot  5.15.153.1-micro Fri Aug 23 06:58   still running
user1    pts/1                         Thu Aug 22 07:58 - crash  (22:59)
reboot   system boot  5.15.153.1-micro Thu Aug 22 07:58   still running
```

Der Nutzer, das Terminal in dem er eingeloggt ist, der Zeitpunkt des Einloggens werden aufgelistet. Wenn man das System runterfährt wird der Auslogzeitpunkt eingetragen. Hier passiert das nicht, weil Ubuntu in WSL anscheinend nicht mitbekommt, daß man Ubuntu geschloßen hat. Erst beim nächsten Login, wird dann das Ausloggen als CRASH eingetragen.

(00:54) zeigt an wie lange die Session angedauert hat.

```
sudo lastb
[sudo] password for user1:
btmp begins Sun Aug 11 06:36:33 2024
```

Es gibt also nur einen fehlgeschlagenen Einlog-Versuch.

13.0.6 Neue Nutzer erstellen

Es gibt zwei Befehle, um neue Nutzer zu erstellen: `useradd` oder `adduser`

Mit `useradd` wird der Nutzer erstellt zusammen mit seiner primären Gruppe. Die GID ist identisch zur UID und der Gruppenname ist identisch zum Nutzernamen. Man kann hier problemlos Großschreiben.

Es wird KEIN Heimverzeichnis und auch KEIN Passwort erstellt. Der Nutzer hat zwar ein Heimverzeichnis es existiert aber nicht. Die Standard-Shell wird aus "/bin/sh" gessetzt!

Will man die Erstellung des Heimverzeichnisses erzwingen, dann muß man die Option "-m" benutzen und will man "bash" als Standard-Shell einsetzen, dann braucht man "-s /bin/bash"!

Mit "adduser" werden sehr viel mehr Informationen ausgedruckt und abgefragt. Das Heimverzeichnis wird angelegt und es wird nach einem Password und weiteren Nutzerinformationen gefragt. Der Nutzername muß kleingeschrieben sein, oder man muß die Option "–force-badname" genommen werden.

Da bei `useradd` kein Passwort abgefragt wird, muß man dieses nachträglich hinzufügen. Dazu gibt es den Befehl `passwd` mit dem man ein Passwort ändern kann. Ruft man es ohne Argumente auf, dann ändert man sein eigenes Passwort.

Mit `userdel NUTZERNAMEN` wird der Nutzer und die primäre Gruppe gelöscht. Aber das Heimverzeichnis bleibt. Erst mit der Option "-r" wird es beim Löschen des Nutzers AUCH gelöscht.

Optionen für `useradd`:

-c Für Vollen Namen

-d Setzt den Wert für das Heimverzeichnis in der passwd Datei. Das Verzeichnis wird NICHT erstellt, wenn es nicht existieren sollte.

-e Deaktivierungsdatum als YYYY-MM-DD, Das Datum wird in Tage nach 1.1.1970 umgerechnet und dann in EXPDATE eingesetzt

-f Setzt die INACTIVE in /etc/shadow

-g setzt die primäre GID in passwd, sofern diese Gruppe existiert

-G fügt den Nutzer in diese sekundären Gruppen ein, sofern diese Gruppe existiert

-m erstellt das Heimverzeichnis, wenn es nicht existiert

-M erstellt neuen Nutzer OHNE Heimverzeichnis, selbst wenn dieses normalererweise automatisch erstellt werden würde

-s setzt die Standard-Shell in "/etc/passwd"

-u setzt die UID in "/etc/passwd/"

Optionen für `passwd`:

-d löscht das Passwort

-e erzwingt eine Passwortänderung

-l sperrt das Konto, dazu wird vor das Passwort in "/etc/shadow" ein "!" gesetzt

-u entsperrt das Konto

-S gibt Informationen über den Status des Passworts

13.0.7 Gruppen erstellen und löschen

Mit dem Befehl `groupadd GRUPPENNAME` kann man Gruppen erstellen. Die Option "-g" erlaubt die Eingabe der Gruppenid.

Zum Löschen ist der Befehl `groupdel GRUPPENNAME`.

Man kann die Primäre Gruppe eines Nutzers NICHT mit "groupdel" löschen, sofern der Nutzer noch existiert.

13.0.8 su und sudo

Mit `sudo` (Superuser do) ruft man den danach folgenden Befehl als Superuser, also root, auf und kann dann im Grunde alles damit machen. z.B. könnte man mit `sudo passwd USERNAME` das Passwort des Nutzers ändern, OHNE daß man das jetzige Passwort kennen muß. Effektiv kann man damit Nutzer ausschließen.

Mit `su` wechselt man den Nutzer. "su" bedeutet "switch user". `su root` logt einen als Superuser ein.

Mit `sudo -i` kann man sich als root auch einloggen oder mit `su -`.

"/etc/sudoers" enthält Informationen darüber wer "sudo" ausführen darf und wer nicht

"/etc/sudoers.d" hier können weitere Einstellungen für sudo drinnen stehen

Access-Control-Dateien

```
/etc/passwd
....
user1:x:1000:1000:,,,:/home/user1:/bin/bash
....
```

Hier hat man Informationen über die Nutzer drinnen. Ganz Links kommt der Nutzername, dann das Password, hier ein x, die UID und die primäre GID folgt darauf.

Das nächste Feld enthält weitere Informationen über den Nutzer, wie den vollen Namen, Ort, und zwei Telefonnummern. Mit `chfn` kann man diese Informationen ändern. Das Feld nennt man GECOS.

Danach kommt dann das Heimverzeichnis und die Standard-Shell am Ende.

Das x zeigt an, daß hier das Passwort NICHT gespeichert wird. Man findet es in /etc/shadow

```
/etc/shadow
....
user1:$y$j9T$4DzgvRGCaOgO8MQu4O8p1.$jIjyaOw.....eOS3IX32:19946:0:99999:7:INACTIVE:EXPDATE:
...
```

Auch hier hat man den Nutzernamen ganz links. Danach kommt das verschlüsselte Password. `$y$j9T$` beschreibt den yescrypt Algorithmus. Die j9T legen bestimmte Parameter fest. Danach kommt der Hash-Salt und dann das Passwort als Hash.

Die 19946 ist das Datum der letzten Änderung des Passwortes. Es ist die Anzahl an Tagen seit dem 1.1.1970. Dann kommt das Mindestalter des Passworts in Tagen. Hier ist das 0, man darf also das Passwort ändern wann man will.

Die 99999 sind das maximale Alter in Tagen und das sind mehr als 273 Jahre.

WARNZEIT: 7 Tage vor Ablauf der 99999 Tage beginnt die Erinnerung daran, daß man es ändern muß. Das wird mit der 7 ausgedrückt.

INACTIVE Nach Ablauf des Maximalalters des Passworts erhält man eine weitere Frist in der man das Passwort ändern kann. Diese steht im INACTIVE Feld. Läuft diese auch ab, dann wird Konto deaktiviert.

EXPDATE Datum zu dem das Konto deaktiviert wird in Tagen seit 1.1.1970.

Reservierte Feld Für zukünftige Verwendung reserviert

INACTIVE und EXPDATE sind nicht gesetzt. Die Felder sind leer.

```
/etc/group
user1:x:1000:
```

Nutzername, Gruppenpasswort, GID, MItglieder

Ist die Gruppe die PRIMÄRE Gruppe des Nutzers, dann wird er in der Regel NICHT bei den Mitglieder aufgelistet.

```
/etc/gshadow
user1:!::
```

Die Passwort-Datei für Gruppen.

Nutzername, verschlüsseltes Passwort, Administrator der Gruppe, Mitglieder

Die Adminstratoren sind durch Komma voneinander getrennt, genauso wie die Mitglieder

! an der Stelle des Passworts bedeutet, daß man sich NICHT mit dem Befehl `newgrp` zu einem temporären Mitglied dieser Gruppe machen kann.

13.0.9 Skeleton Verzeichnis

Im "/etc/skel" Verzeichnis stehen alle Dateien und Verzeichnisse, die beim Erstellen eines neuen Nutzers in das Heimverzeichnis dieses Nutzers kopiert werden sollen.

Das sollten mindestens die versteckten Dateien für die Konfiguration der Standard-Shell sein.

13.0.10 Dateiberechtigungen ändern

-rwxrwxrwx 1 BESITZER GRUPPE GRÖSSE ÄNDERUNGSDATUM DATEINAME

Das erste "rwx" sind die Rechte des Besitzers, dann folgen die der Gruppe und danach die der restlichen Benutzer.

Mit `chmod` kann man die Rechte ändern.

Symbolisch u - user = Besitzer

g - group = Gruppe

o - other = Rest

```
chmod u-x Dateiname
```

Damit wird der Besitzer das Recht zum Ausführen dieser Datei verlieren.

```
chmod u+x,go-rw Dateiname
```

Nach diesem Befehl hat der Nutzer zusätzlich zu schon existierenden Rechten das Recht die Datei auszuführen (u+x), während die Gruppe und der Rest die Rechte zum Lesen und Schreiben verlieren (go-rw). Dazwischen steht ein Komma, weil es unterschiedliche Rechte verteilt werden.

Will man die Rechte dem Besitzer, der Gruppe oder dem Rest geben, dann muß man u, g oder o gefolgt von einem "+" gefolgt vom Recht (r,w,x) angeben. Danach folgt der Dateiname oder Verzeichnisname.

Will man für mehrere Gruppen die Rechte ändern, dann muß das mit , trennen, es sei denn sie sollen dieselben Rechte kriegen.

Die Vergabe der Rechte über den symbolischen Weg ist hilfreich, wenn man nur ein oder zwei Rechte ändern will. Will man die Rechte einfach auf einen bestimmten Wert setzen, dann ist der NUMERISCHE Weg besser.

13.0.11 numerische

Das Lese-Recht hat den Wert 4, Schreiben hat den Wert 2 und Ausführen hat den Wert 1.

D.h.

r–	rw-	rwx	r-x	-w-	-wx	–x	—
4	6=4+2	7=4+2+1	5=4+1	2	3=2+1	1	0

```
chmod 645 Dateiname
```

Damit wird dem Nutzer die Lese- und Schreibrechte gegeben, der Gruppe nur die Leserechte und dem Rest nur Lesen und Ausführen.

```
chmod 666 Verzeichnisname
```

Die Rechte von ALLEN werden auf Lesen und Schreiben gesetzt für das Verzeichnis. D.h. man kann das Verzeichnis NICHT betreten und z.B. ls -l Verzeichnisname/ zeigt nur ? an, weil das Ausführen nicht erlaubt ist. Die Dateien und Verzeichnisse IN diesem Ordner behalten ihre Rechte. Erst wenn man die Option "-R" benutzt werden die Rechte des Verzeichnisses und aller Dateien und Verzeichnisse in ihm geändert.

13.0.12 Ändern Dateibesitz

```
chown Besitzer:Gruppe DATEINAME oder VERZEICHNISNAME
```

Will man z.B. den Besitzer NICHT ändern, dann kann man den Wert einfach weglassen oder umgekehrt.

```
chown :Gruppe DATEINAME oder VERZEICHNISNAME
```

oder

```
chown NeuerBesitzer DATEINAME oder VERZEICHNISNAME
```

Evtl. muß man `sudo` benutzen.

Beim Verzeichnisnamen kann man wieder mit "-R" den Besitzer oder die Gruppe rekursiv ändern.

13.0.13 groups und groupmems

Mit `groups` kann man die Gruppen denen ein Nutzer angehört auflisten. Läst man den Namen weg wird der momentane Nutzer benutzt:

```
groups
user1 adm dialout cdrom floppy sudo audio dip video plugdev netdev

groups sarah
sarah : sarah
```

user1 gehört einer Menge an Gruppen an, sarah aber nur ihrer eigenen primären Gruppe.

`sudo groupmems -g cdrom -l` listet alle Mitglieder dieser Gruppe auf, dabei wird die Datei gshadows gelesen, weswegen man "sudo" benötigt.

-g Gruppenname

-l Liste der Mitglieder

13.0.14 Sticky Bit

Verzeichnis kann man spezielle Rechte zuordnen, die verhindern, daß Dateien überschrieben oder gelöscht werden, wenn NICHT der Besitzer auf sie zugreift.

```
chmod 1644 VERZEICHNISNAME
```

Dazu übergibt man VIER Zahlen an `chmod`. Die erste ist eine 1, wenn man das Bit setzen will und eine 0 wenn man es entfernen will.

```
chmod 1777 user1/
ls -l
drwxrwxrwt 7 user1   user1   4096 Aug 23 13:36 user1
```

Das “t” am Ende der Rechteliste, dort wo normalerweise das x für die Ausführrechte der restlichen Nutzer ist, zeigt an, daß das Sticky Bit gesetzt wurde.

Beim Versuch von sarah eine Datei im user1/ Ordner zu löschen, nachdem das Sticky Bit gesetzt wurde, tritt der folgende Fehler auf:

```
rm Test
rm: remove write-protected regular empty file 'Test'? y
rm: cannot remove 'Test': Operation not permitted
```

13.0.15 Set GID

Übergibt man nicht 1, sondern 2 im “chmod” Befehl, dann wird die GID gesetzt. Hat zum Beispiel das Verzeichnis dann die Gruppe sarah, dann werden alle neu erstellten Dateien und Verzeichnisse dieser Gruppe angehören.

`chmod 2777 user1/` nachdem man `sudo chown :sarah user1/` gemacht hat:

```
drwxrwsrwx 7 user1   sarah   4096 Aug 23 13:52 user1
```

Erstellt man nun eine neue Datei im Verzeichnis user1/ dann hat sie die Gruppe sarah:

```
-rw-r--r-- 1 user1 sarah    0 Aug 23 13:50 Test2
```

13.0.16 Set UID

Übergibt man eine 4 im “chmod” Befehl, dann wird die Datei mit den Rechten des Benutzers ausgeführt dem die Datei gehört, nicht den eigenen.

```
-rwsr--r-- 1 user1 user1    0 Aug 23 13:50 Test2
```

Gehört die Datei z.B. dem root, dann wird diese Datei mit root-Rechten ausgeführt.

[1]:
```
bin(168)
```

[1]:
```
'0b10101000'
```