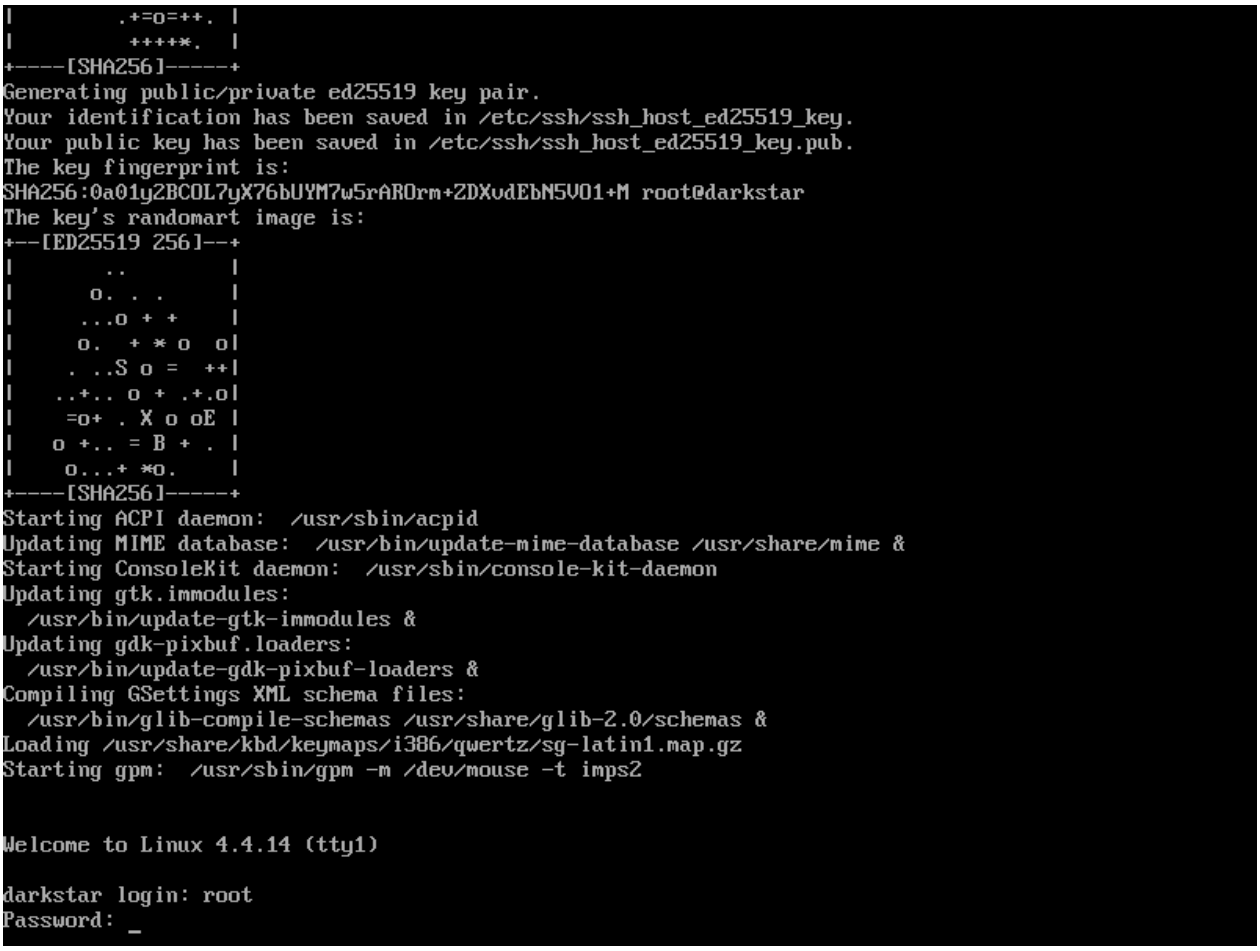
Linux Kurs - Erste Schritte

Herzlichen Glückwunsch. Du hast den ersten Teil des Kurses abgeschlossen und bereits ein lauffähiges Linux-System auf deinem Computer installiert. In diesem Kursteil geht es darum dein neues Betriebssystem besser kennenzulernen.

Der erste Start

Wenn Du dein System zum ersten Mal startest, sieht es wie folgt aus:



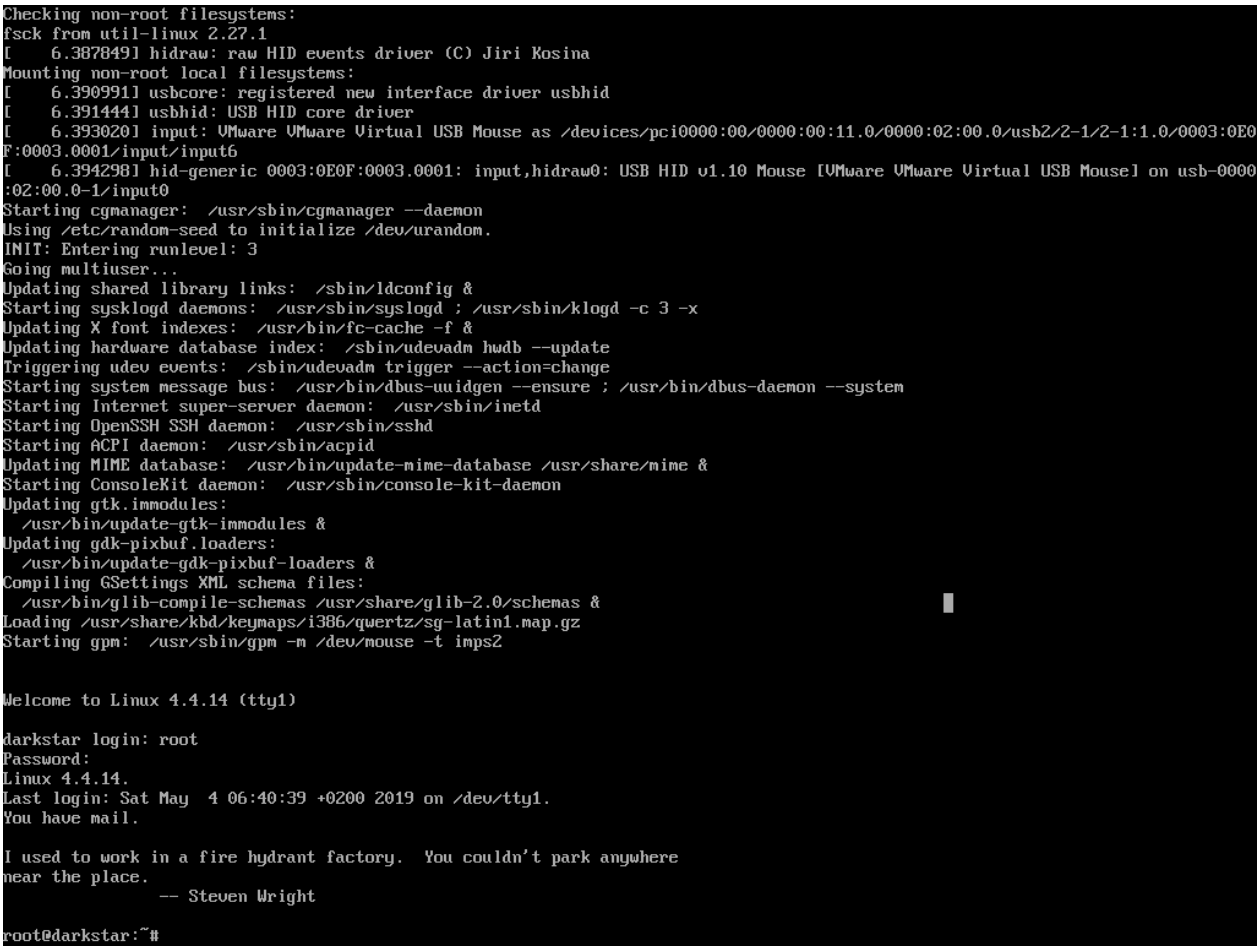
Die Zeichen die Du im Bereich *Generating public/private key pair* siehst wurden ausgegeben, da beim allerersten Systemstart ein sogenanntes SSH-Schlüsselpaar erzeugt wurde. Bei SSH handelt es sich um ein Protokoll mit dem Du von einem entfernten Rechner verschlüsselt auf deinen Linux Computer zugreifen kannst.

Du siehst ausserdem das der Dienst *gpm* gestartet worden ist. Dieser sollte dir bereits von der Installation her vertraut sein. Es ist der Dienst der für die Bereitstellung der Maus in der Konsole (*tty*) verantwortlich ist.

Melde dich nun bitte am sogenannten *Login-Prompt* an. Das ist die Zeile die mit *darkstar login:* beginnt. darkstar ist der *Hostname* des Computers.

Gebe als Benutzername bitte den Benutzer **root**, gefolgt von Enter ein. Das Passwort siehst Du während der Eingabe nicht, also wundere dich bitte nicht. Das Passwort hast Du während der Installation für den Benutzer root vergeben.

Nachdem Du dich angemeldet hast, bist Du in der Shell angekommen, einem wichtigen Herzstück des Linux-Systems. Bei Slackware ist die Standardshell *Bash*, es gibt allerdings auch viele Alternativen wie *zsh* oder *csh*. In diesem Kurs beschränken wir aus auf die *Bash* (Bourne-again shell).



Folgende Informationen werden ausgegeben:

- Die Linux-Kernelversion
- Last login (wann sich zuletzt jemand an dem System angemeldet hat)
- Eine Information ob neue Nachrichten vorhanden sind
- Die *Message of the Day* (MOTD)
- Der Prompt

Hinweis: Der rechteckige Klotz den Du auf dem Bildschirmfoto siehst ist übrigens der Mauscursor, den das Programm *gpm* bereitstellt.

Verzeichnisse und Kommandos

Der Bash-Prompt ist wie folgt aufgebaut: *Benutzername@Hostname:$ORT*, wobei *$ORT* das aktuelle Arbeitsverzeichnis ist. In diesem Fall ist es ~, was eine Abkürzung für das Homeverzeichnis darstellt.

Mit dem Befehl:

```
pwd
```

kannst Du dir jederzeit den genauen Pfad ausgeben lassen, in dem Du dich befindest. **pwd** steht bezeichnenderweise für **p**rint **w**orking **d**irectory.

Wenn Du **pwd** im Homeverzeichnis des Benutzers root eingibst, sollte die Ausgabe wie folgt aussehen:

/root

ls

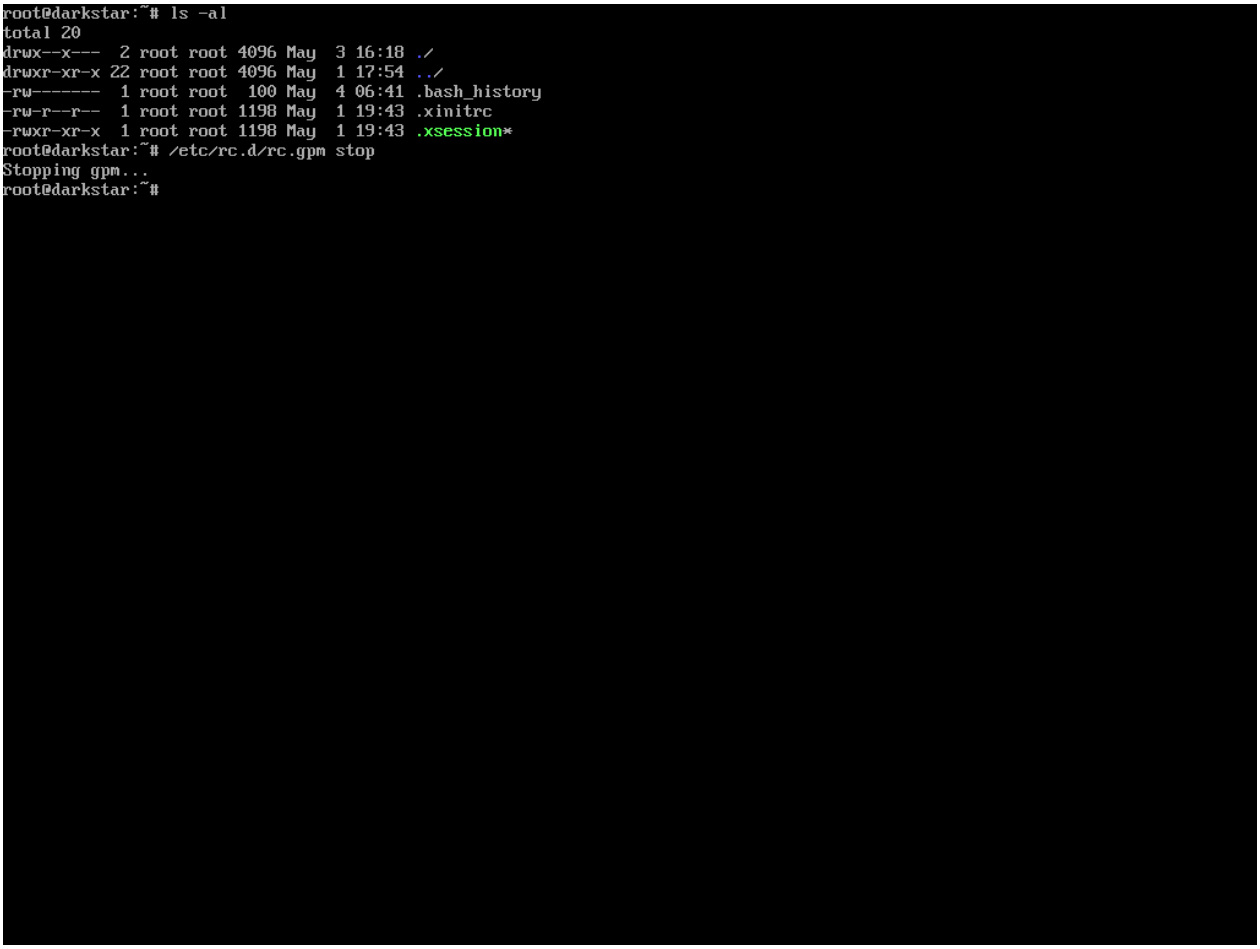
Mit dem Befehl:

```
ls
```

kannst Du dir den Verzeichnisinhalt ausgeben lassen. Da das Heimatverzeichnis des Benutzers **root** bisher leer ist, wird keine Ausgabe erscheinen. Doch aufgepasst: es gibt unter Linux sogenannte versteckte Dateien. Diese werden mit einem Punkt vorangestellt. Ein einfaches **ls** zeigt die versteckten Dateien nicht an. Versuche es noch einmal aber diesmal mit dem Befehl

```
ls -al
```

Die Ausgabe sollte dabei wie folgt aussehen:



Doch woher kommen diese Dateien und was bedeuten sie? Und was hat es mit den Zahlen und Zeichen auf sich?

Dateiberechtigungen

Schauen wir uns einmal eine der Dateien an:

> `-rw-------` `1 root root 100 May 4 06:41 .bash_history`

Der erste Bereich `-rw-------`, gibt die Rechte der Datei an. Diese werden von Linux in Oktalform verwaltet (4-2-1). Das hört sich erst einmal kompliziert an, ist aber ganz einfach, wenn Du es verstanden hast.

Das allererste `-` gibt den Typ an. In diesem Falle ist er nicht speziell definiert worden. Würde es sich um ein Verzeichnis handeln, würde dort *d* für Directory stehen.

Danach folgt ein `rw-`. Es gibt drei Blöcke, mit jeweils drei Werten. Die Blöcke stehen für **Owner** / **Group** und **Others**, also dem Dateieigentümer, der Gruppe und allen Anderen. Die drei Werte innerhalb der Blöcke stehen für:

- r = read (lesen) - Oktal: 4
- w = write (schreiben) - Oktal: 2
- x = execute (ausführen) - Oktal: 1

Für die Datei *.bash_history* sind nur die Werte `r` und `w` für den *Owner* definiert worden: `-rw-------` . Der Eigentümer der Datei, kann diese also lesen und schreiben:

Owner: `rw-` = 4+2+0 = 6 / **Group**: `---`: 0+0+0 = 0 / **Others**: `---`: 0+0+0 = 0 oder zusammengesetzt: **600**

Falls der Eigentümer die Datei auch ausführen können soll, sähe die Definition wie folgt aus:

> `-rwx------`

Oktal wäre dies **Owner**: `rwx` = 4+2+1 = 7 / **Group**: `---`: 0+0+0 = 0 / **Others**: `---`: 0+0+0 = 0 oder zusammengesetzt: **700**.

Kommen wir nochmals auf das Beispiel zurück:

> `-rw------- 1` `root root` `100 May 4 06:41 .bash_history`

Der Eigentümer und die Gruppe sind beide **root root**. Falls auch die Gruppe und alle Anderen lesend auf die Datei zugreifen können soll, würden die Rechte wie folgt dargestellt: `-rw-r--r--`

Oktal wäre dies **Owner**: `rw-` = 4+2+0 = 6 / **Group**: `r--`: 4+0+0 = 4 / **Others**: `r--`: 4+0+0 = 4 oder zusammengesetzt: **644**.

Diese Rechte könntest Du testweise mit folgendem Befehl ändern:

```
chmod 644 .bash_history
```

Wenn Du nun nochmals ein **ls -al** eingibst, könnte die Ausgabe jetzt wie folgt aussehen:

> `-rw-r--r--` `1 root root 264 May 4 06:48 .bash_history`

Ändere die Rechte der Datei daraufhin bitte wieder zurück auf den Ursprungswert:

```
chmod 600 .bash_history
```

Du hast wahrscheinlich beobachtet, dass sich der Zahlenwert hinter der *Owner* und *Group* Definition erhöht hat, ebenso die Änderungszeit:

> `-rw-r--r-- 1 root root` **`264`** `May 4` **`06:48`** `.bash_history`

Das hat damit zu tun, dass die History Datei in der Zwischenzeit angewachsen ist.

History

Die versteckte Datei **.bash_history** enthält eine Historie der vom Benutzer eingegebenen Befehle. Ausgeben lassen kannst Du sie dir einfach mit dem Befehl:

```
history
```

Dort sollte auch deine Eingabe von **pwd** mit einer Nummer davor aufgeführt werden. Möchtest Du dieses Kommando erneut ausführen, kannst Du einfach **!$NUMMER** eingeben, also das Ausrufezeichen, direkt gefolgt von der angegebenen Nummer, zum Beispiel **!3**, wobei Du die 3 natürlich durch die entsprechende Nummer ersetzen musst.

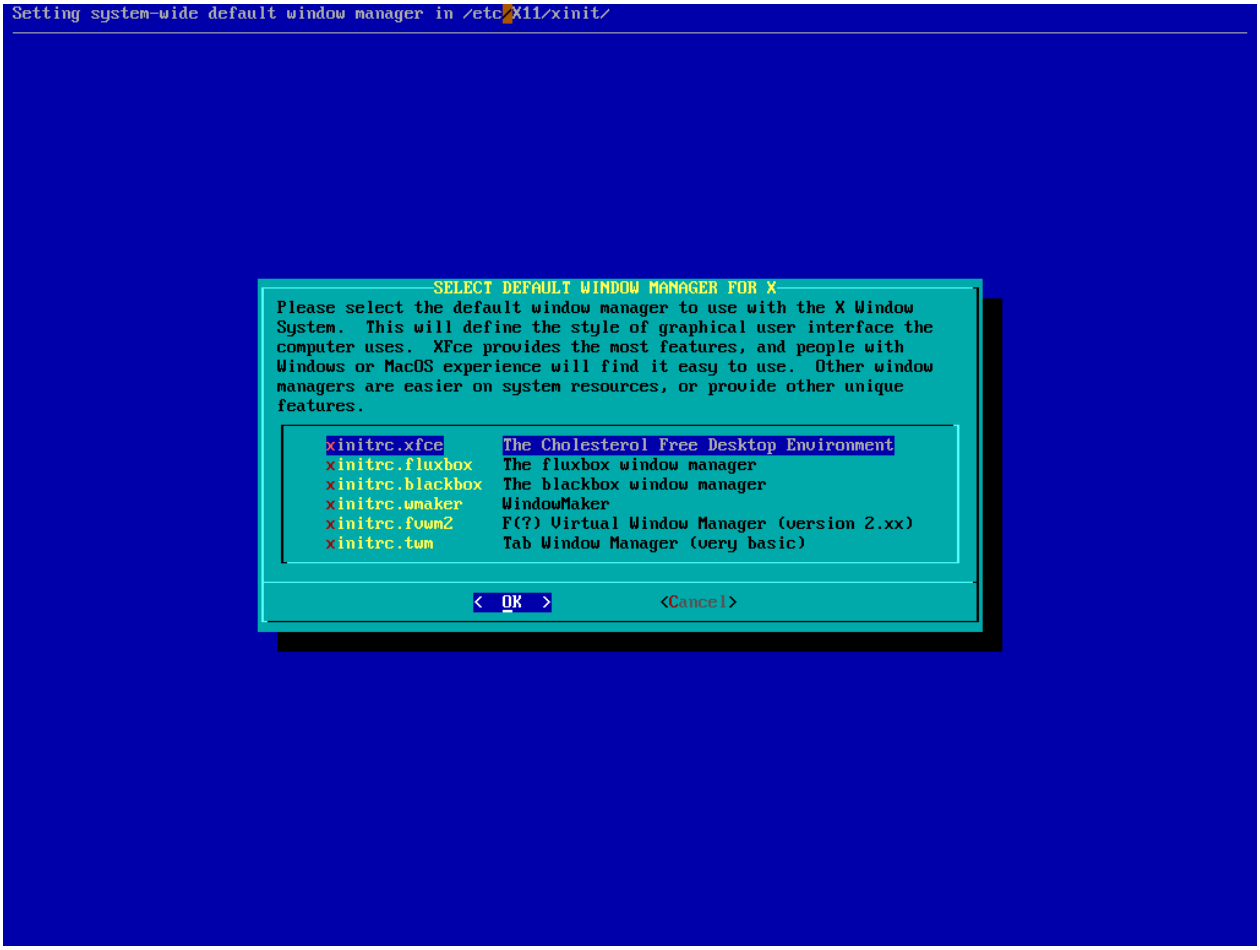
Du kannst in der Bash auch ganz einfach mit den Pfeil-hoch, Pfeil-runter Tasten die History durchblättern. Enter löst dann jeweils den gefundenen Befehl aus.

Mit Hilfe von **Ctrl+R** kannst du die History durchsuchen. Gebe einfach **Ctrl+R** gefolgt von dem Suchbegriff ein, zum Beispiel *pwd*. Wenn Du **Ctrl+R** erneut eingibst wird der nächste Treffer angezeigt (falls es einen weiteren Treffer gibt). Enter löst auch hier den Befehl aus.

Der Befehl **history -c** löscht den gesamten Inhalt der History.

xwmconfig

Die beiden weiteren versteckten Dateien die Du auf dem Bildschirmfoto gesehen hast (*.xinitrc* und *.xsession*), wurden während der Installation vom **setup** Kommando erstellt, welches wiederum den Befehl **xwmconfig** ausgeführt hat. Rufe **xwmconfig** erneut auf. Die Anzeige sollte dir von der Installation vertraut sein:



Belasse die Einstellung bitte auf xinitrc.xfce.

Systemdienste

Im Bildschirmfoto das die *ls -al* Ausgabe anzeigt, hast Du vielleicht bemerkt, dass ein weiterer Befehl ausgeführt worden ist:

```
/etc/rc.d/rc.gpm stop
```

/etc/rc.d/rc.gpm ist der Befehl und **stop** der Parameter.

Damit wurde der Dienst gpm der für die Maus in der Konsole zuständig ist beendet. Dieser Befehl wurde nur für den Kurs ausgeführt, damit auf den folgenden Anzeigen der Mauscursor nicht mehr erscheint. Du kannst es dennoch gerne einmal testen. Der Mauszeiger sollte nachdem Du den Befehl ausgeführt hast verschwinden. Beim nächsten Systemstart würde er allerdings wieder angezeigt. Mit Hilfe des folgendem Befehls kannst Du den Dienst wieder starten:

```
/etc/rc.d/rc.gpm start
```

Falls Du nicht möchtest, dass ein Dienst beim Systemstart gestartet wird, kannst Du unter Slackware einfach die Startdatei als nicht-ausführbar markieren.

Schaue dir zuvor einmal die Datei mit dem **ls -al** Kommando an.

```
ls -al /etc/rc.d/rc.gpm
```

```
root@darkstar:~# ls -al /etc/rc.d/rc.gpm
-rwxr-xr-x 1 root root 1156 May  1 18:48 /etc/rc.d/rc.gpm*
root@darkstar:~#
```

Du siehst, dass die Datei für jeden ausführbar ist beschreiben darf allerdings nur der *Owner*, in diesem Falle root.:

> `-rwxr-xr-x`

Oktal wäre dies **Owner**: `rwx` = 4+2+1 = 7 / **Group**: `r-x`: 4+0+1 = 5 / **Others**: `r-x`: 4+0+1 = 5 oder zusammengesetzt: **755**.

Der **chmod** Befehl bietet einen einfachen Parameter um eine Datei nicht mehr als ausführbar zu markieren:

```
chmod -x /etc/rc.d/rc.gpm
```

Dies entfernt alle *executable* Rechte von der Datei. Gebe erneut **ls -al** ein um die Änderungen zu prüfen:

```
ls -al /etc/rc.d/rc.gpm
```

> `-rw-r--r--`

Somit wird der Dienst beim Systemstart nicht mehr gestartet. Möchtest Du, dass er wieder gestartet werden soll, führe den **chmod** Befehl einfach mit **+x** aus:

```
chmod +x /etc/rc.d/rc.gpm
```

Zeitzone und Uhrzeit

Bevor wir mit den Grundsätzen des Linux-Dateisystems fortfahren, stelle bitte zunächst sicher, dass die Zeitzone und Uhrzeit des Systems richtig eingestellt sind.

Wenn Du dich erinnerst, so hast Du während der Installation die Zeitzone bereits angegeben. Wir werden nun überprüfen, ob diese korrekt gesetzt worden ist. Die Zeitzonendefinitionen liegen im Verzeichnis */usr/share/zoneinfo/* und dort in nach Kontinenten benannten Unterverzeichnissen. Schaue dich dort ein wenig um. Dazu kannst du zunächst mit dem Befehl **cd** in das Verzeichnis wechseln:

```
cd /usr/share/zoneinfo/
```

Gebe dort `ls` ein

```
root@darkstar:/usr/share/zoneinfo# ls
Africa/      Brazil/  EST      GB         HST       Japan      NZ         Portugal  UTC          posix/
America/     CET      EST5EDT  GB-Eire    Hongkong  Kwajalein  NZ-CHAT    ROC       Universal    posixrules
Antarctica/  CST6CDT  Egypt    GMT        Iceland   Libya      Navajo     ROK       W-SU         right/
Arctic/      Canada/  Eire     GMT+0      Indian/   MET        PRC        Singapore WET          timeconfig@
Asia/        Chile/   Etc/     GMT-0      Iran      MST        PST8PDT    Turkey    Zulu         zone.tab
Atlantic/    Cuba     Europe/  GMT0       Israel    MST7MDT    Pacific/   UCT       iso3166.tab  zone1970.tab
Australia/   EET      Factory  Greenwich  Jamaica   Mexico/    Poland     US/       localtime@
root@darkstar:/usr/share/zoneinfo# _
```

In unserem Beispiel gehen wir von der Zeitzone *Europe/Zurich* aus.

Sollte die Zeitzone in deinem Falle abweichen, passe die folgenden Befehle bitte an deine Gegebenheiten an.

```
cd Europe
ls
```

```
root@darkstar:/usr/share/zoneinfo/Europe# ls
Amsterdam  Berlin      Chisinau    Isle_of_Man  Lisbon      Mariehamn  Paris      San_Marino  Tallinn    Vatican    Zaporozhye
Andorra    Bratislava  Copenhagen  Istanbul     Ljubljana   Minsk      Podgorica  Sarajevo    Tirane     Vienna     Zurich
Astrakhan  Brussels    Dublin      Jersey       London      Monaco     Prague     Simferopol  Tiraspol   Vilnius
Athens     Bucharest   Gibraltar   Kaliningrad  Luxembourg  Moscow     Riga       Skopje      Ulyanovsk  Volgograd
Belfast    Budapest    Guernsey    Kiev         Madrid      Nicosia    Rome       Sofia       Uzhgorod   Warsaw
Belgrade   Busingen    Helsinki    Kirov        Malta       Oslo       Samara     Stockholm   Vaduz      Zagreb
root@darkstar:/usr/share/zoneinfo/Europe#
```

Dort siehst Du die Zeitzonendatei *Zurich*. Die aktive Zeitzone wird unter */etc/localtime* gesetzt. Als nächstes vergleichen wir die beiden Dateien mit dem Befehl **diff**

`diff /usr/share/zoneinfo/Europe/Zurich /etc/localtime`

Es sollte keine Meldung erscheinen. In diesem Falle stimmen die Dateien überein und die aktive Zeitzone ist korrekt auf *Europe/Zurich* eingestellt worden. Du kannst es testweise mit einer anderen Datei versuchen, zum Beispiel:

`diff Oslo /etc/localtime`

Die Ausgabe sollte wie folgt aussehen:

> Binary files Oslo and /etc/localtime differ

Du hast wahrscheinlich bemerkt, das wir beim zweiten Aufruf nicht den kompletten Pfad zur *Oslo* Datei angegeben haben. Das war in diesem Falle möglich, da wir uns bereits im richtigen Verzeichnis befinden (wir sind zuvor mit *cd /usr/share/zoneinfo/* und *cd Europe* dort hinein gewechselt. Du kannst es mit **pwd** überprüfen).

Tipp: Sollte die Zeitzone nicht stimmen oder Du möchtest eine andere Zeitzone einstellen, kannst Du einfach das entsprechende Zonenfile nach */etc/localtime* kopieren: `cp /usr/share/zoneinfo/Europe/Zurich /etc/localtime`. Da die Zeitzone richtig eingestellt sein sollte, musst du diesen Schritt nicht durchführen.

Du hast in diesem Abschnitt bereits viele wichtige Linux-Befehle kennengelernt, wie zum Beispiel **cd** um Verzeichnisse zu wechseln oder das Powerkommando **diff** um Dateiinhalte zu vergleichen.

Lass uns als nächstes die Uhrzeit des Systems anschauen.

Gebe dazu bitte den Befehl **date** ein.

```
root@darkstar:~# date
Sun May  5 11:50:20 CEST 2019
root@darkstar:~#
```

In diesem Beispiel stimmt die Uhrzeit nicht und wir möchten sie korrigieren. Dies erfolgt auch mit Hilfe des **date** Kommandos. Die einzustellende Zeit kann in unterschiedlichen Formaten angegeben werden. Wir haben uns für die Eingabe im Format **$JAHR-$MONAT-$TAG $STUNDE:$MINUTE** entschieden:

`date --set "2019-5-15 09:30"`

Bei Bedarf kannst Du die **:$SEKUNDE** ergänzen. Zu einem späteren Zeitpunkt werden wir die Uhrzeit allerdings über das Internet mit einem Timeserver abgleichen, daher ist eine minutengenaue Eingabe vorerst durchaus ausreichend.

Diese Uhrzeit muss nun noch in das BIOS des Computers geschrieben werden.

Dazu wird folgender Befehl ausgeführt:

```
hwclock --systohc --utc
```

Damit wird die aktuell auf dem System eingestellte Uhrzeit im UTC-Format in das BIOS übertragen.

Sprache

Die Standardsprache bei Slackware ist Englisch. Diese lässt sich sehr leicht umstellen. Eine Liste mit allen verfügbaren Sprachen kannst Du dir mit folgendem Befehl ausgeben lassen:

```
locale -av
```

Du wirst feststellen, dass es sich um eine sehr lange Liste handelt, die in der Konsole vorbei rauscht. Du kannst die Ausgabe mit Hilfe des **more** Kommandos so anpassen, dass sie zeilenweise oder seitenweise blätterbar ist:

```
locale -av | more
```

Das Pipe Zeichen | erreichst Du auf der schweizerdeutschen Tastatur mit Hilfe von "AltGr + 7"

Nun kannst Du mit Enter zeilenweise in der Ausgabe fortfahren und mit Space seitenweise blättern. **Ctrl+c** bricht die Ausgabe ab und bringt dich wieder zurück auf die Shell.

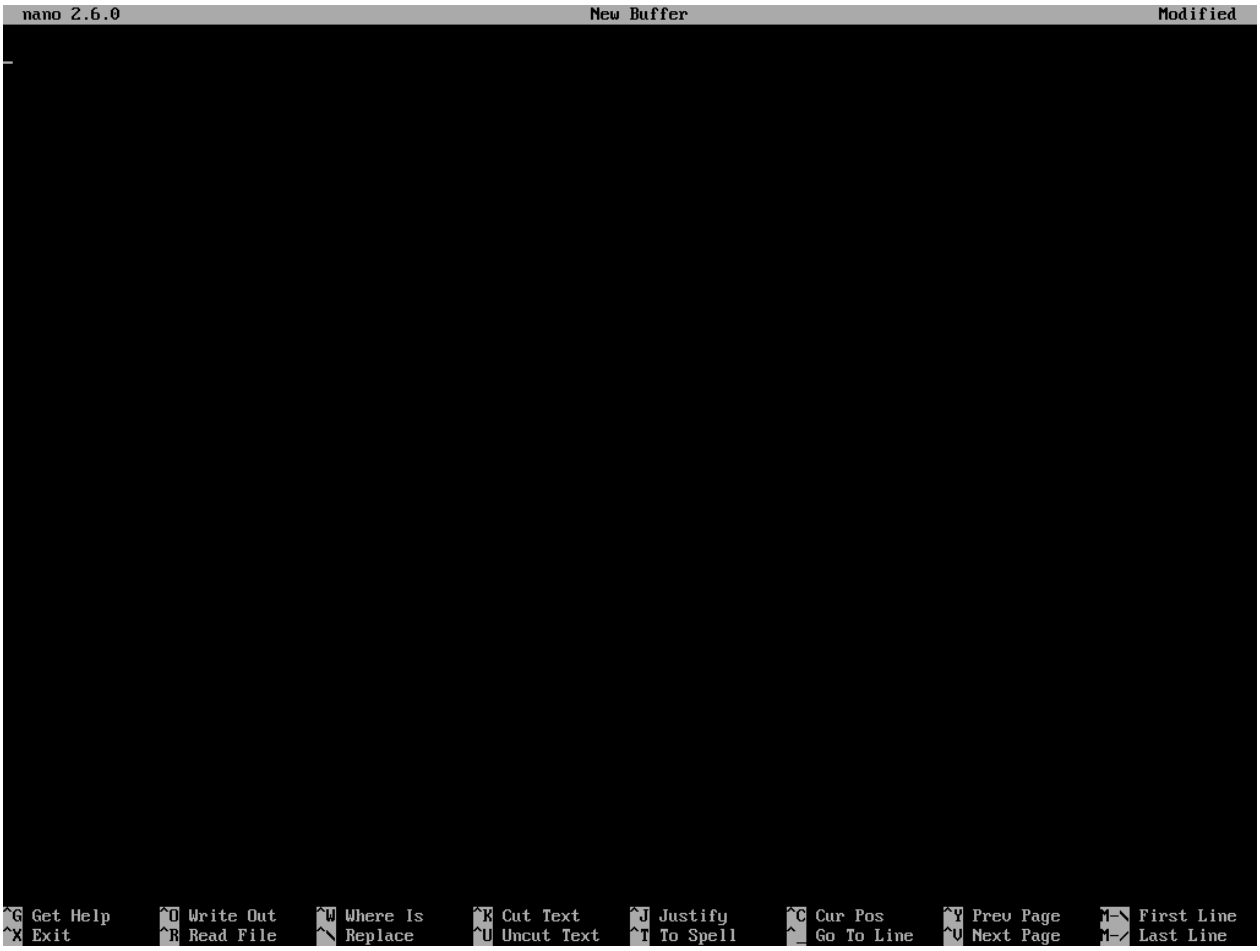
Wichtig für uns ist die Bezeichnung der jeweiligen *locale*. Im Falle der Schweiz ist dies: **de_CH.utf8**, für Deutschland **de_DE.utf8** und für Österreich **de_AT.utf8**. Überprüfen kannst Du dies mit folgendem Befehl:

```
locale -av | grep -A 11 'de_CH.utf8'
```

Mit Hilfe des | Zeichens leiten wir die Ausgabe an einen weiteren Befehl namens **grep** weiter. Dieser Befehl durchsucht den Text nach Vorkommnissen von *de_CH.utf8* und gibt alle Treffer gefolgt von den nächsten 11 Zeichen aus (Parameter: *-A 11*).

Texteditor nano

Da Du nun herausgefunden hast, wie die passende Locale heisst, kannst Du diese entsprechend konfigurieren. Dazu benötigst Du einen Texteditor. Ein einfacher Editor unter Linux ist **nano**. Wenn Du das Programm das erste mal startest, präsentiert es sich wie folgt:



Im oberen Bereich kannst Du einfach los tippen wie es von einem gewöhnlichen Editor bekannt ist.

Unten findest Du eine Übersicht von Befehlskürzeln. Das ^ Steht für *Ctrl*. Das *M* für *Alt*. Um zum Beispiel die Hilfe aufzurufen, reicht ein Druck auf *Ctrl+g*. Die Hilfe verlassen kannst Du mit *Ctrl+x*. Die Kürzel werden zwar mit grossen Buchstraben angegeben, also zum Beispiel *Ctrl+G*, aufgerufen werden sie allerdings mit Kleinbuchstaben. Konkret heisst das, Du musst nicht *Ctrl+Shift+G* eingeben sondern nur *Ctrl+G*. An dieser Stelle widerspricht nano etwas dem Linux-Konzept, da unter Linux in der Regel immer zwischen Gross- und Kleinschreibung unterschieden wird.

Wenn Du nano ohne Parameter startest, wird eine leere Datei geöffnet. Mit *Ctrl+o* würde nano nachfragen wo diese gespeichert werden soll. Dort muss der gesamte Pfad angegeben werden, andernfalls würde nano die Datei in dem Verzeichnis abspeichern in dem Du dich gerade befindest.

Machen wir einen kleinen Test. Tippe in nano folgenden Text ein: Dies ist nur ein Test.

Gib dann *Ctrl+o* ein und Du wirst nach einem Speicherort gefragt. Du kannst die Datei beispielsweise im Heimatverzeichnis des Benutzer root (*/root*) speichern. Gebe bei **File Name to Write:**

> /root/test.txt

ein. Mit *Ctrl+x* verlässt Du den Editor wieder. Nun kannst Du prüfen, ob die Datei tatsächlich erstellt worden ist:

```
cat /root/test.txt
```

Der Befehl **cat** gibt den Inhalt einer Datei aus. Wenn an dieser Stelle *Dies ist nur ein Test.* ausgegeben wird, hast Du es geschafft und Du bist bereit deine erste Konfigurationsdatei zu bearbeiten.

locales konfigurieren

Die systemweite Sprachkonfiguration kannst du in der Datei */etc/profile.d/lang.sh* konfigurieren. Öffne diese datei mit *nano*.

```
nano /etc/profile.d/lang.sh
```

> **Hinweis**: In der BASH gibt es eine Wortvervollständigung. Du kannst zum Beispiel anfangen zu tippen: `nano /etc/pr` und dann die Tab-Taste drücken. In diesem Falle passiert noch nichts, da der Pfadname noch nicht eindeutig ist. Gebe also `nano /etc/pr` gefolgt von Tab-Tab ein und es werden dir alle Dateien und Verzeichnisse angezeigt die mit *pr* beginnen:

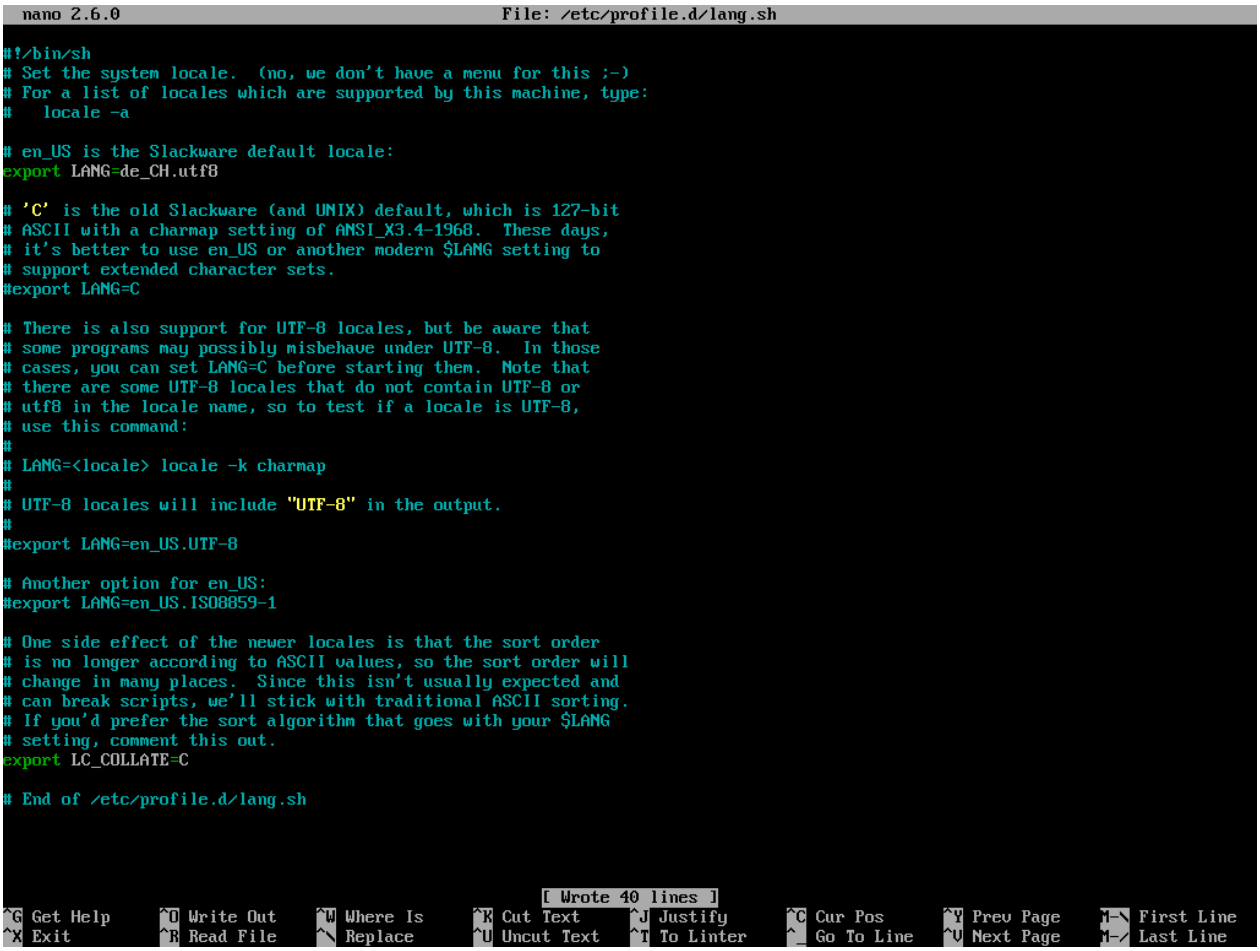
```
root@darkstar:~# nano /etc/pr
printcap      profile       profile.d/    proftpd.conf  protocols
root@darkstar:~# nano /etc/pr_
```

Probiere ein wenig die Tab-Autocompletion aus. Diese Funktion mag im Moment noch ungewohnt erscheinen, wird dir in Zukunft sehr viel Arbeit einsparen.

Nachdem Du die Datei */etc/profile.d/lang.sh* im Editor *nano* geöffnet hast, bearbeite bitte die Zeile *export LANG=...*, so dass dort

> export LANG=de_CH.utf8

oder die zuvor ermittelte Locale steht. Speichere die Datei mit *Ctrl+o* und verlasse den Editor mit *Ctrl+x*.



Hinweis: um im Editor nano an ein Zeilenende zu springen, kannst Du die *End* Taste auf deiner Tastatur verwenden. *Home* springt an den Anfang einer Zeile. In der Bash kannst Du eine ähnliche Funktion verwenden. Wenn Du in der Bash in einer Zeile stehst, springst Du mit *Home* an den Anfang der Zeile und mit *End* an das Ende. Alternativ funktioniert in der Bash auch *Ctrl+A* für den Zeilenanfang und *Ctrl+E* für das Zeilenende.

Gehe bitte analog für die Datei */etc/profile.d/lang.csh* vor. Dabei handelt es sich um die entsprechende Profildatei für die *C Shell*, einer Alternative zur BASH.

lilo Konfiguration

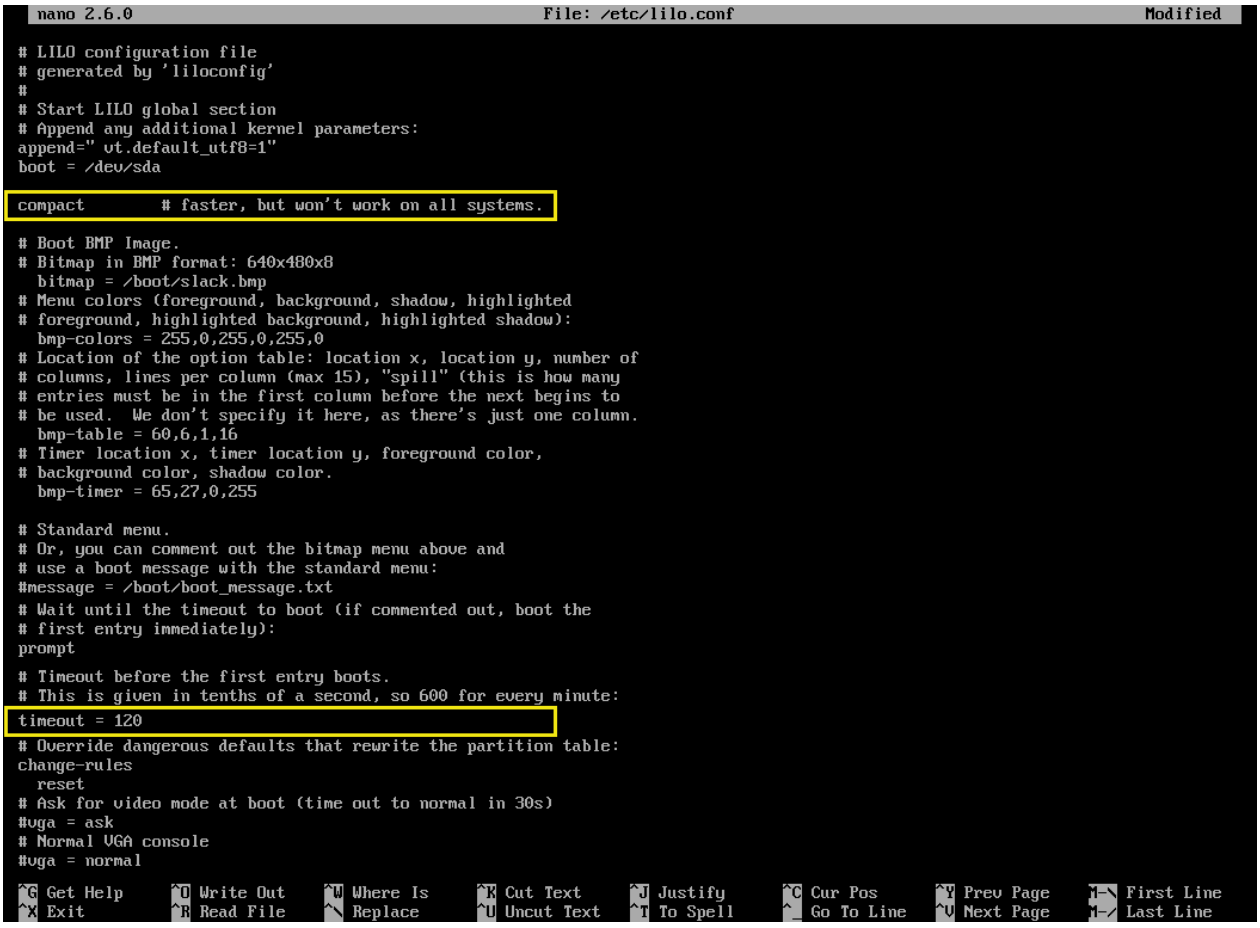
Abschliessend prüfen wir noch die Konfiguration des Bootloaders lilo. Diese sollte bereits während der Installation korrekt geschrieben worden sein, da wir dort UTF-8 ausgewählt haben.

Öffne dazu bitte mit nano die Datei /etc/lilo.conf

```
nano /etc/lilo.conf
```

Wichtig ist, dass dort am Anfang *append=" vt.default_utf8=1"* steht. Wenn Du gerade in der Datei bist, kannst Du noch zwei Anpassungen vornehmen. Möglicherweise ist dir aufgefallen, dass lilo beim Systemstart 2 Minuten wartet, bevor der Defaulteintrag gebootet wird. Ausserdem lädt der Kernel am Anfang recht langsam. Dies kannst Du durch die zwei Parameter *compact* und *timeout* beeinflussen.

Entferne als erstes die # vor *compact*, dadurch sollte der Kernel schneller starten. Ändere daraufhin den timeout von *1200* auf *120*. Die Datei sollte nun wie folgt aussehen:



Speichere die Datei ab und verlasse nano. Die Änderungen werden erst aktiv, wenn du lilo neu in den MBR schreibst. Gebe dazu in der Shell den Befehl **lilo** ein:

```
lilo
```

```
root@darkstar:~# lilo
Warning: LBA32 addressing assumed
Added Linux  *
One warning was issued.
root@darkstar:~#
```

Starte dein Linux mit Hilfe des Befehls **reboot** neu:

```
reboot
```

Der lilo Timeout sollte deutlich kürzer sein und auch der Kernel sollte schneller starten. Logge dich erneut als Benutzer *root* ein und prüfe ob die Spracheinstellungen übernommen worden sind. Am einfachsten geht das mit dem Kommando **man** das Hilfeseiten zu fast allen Linux-Kommandos ausgibt. Schaue dir beispielsweise einmal die Manpage für das Kommando *man* an:

```
man man
```

Die Hilfe sollte in deutscher Sprache ausgegeben werden. Blättern kannst du mit Pfeil-rauf und Pfeil-runter, Space wechselt die Ansicht seitenweise. Verlassen kannst du die Manpage durch Eingabe von *q*.

Leider sind nicht viele Manpages in die deutsche Sprache übersetzt. Die meisten Kommandos bieten aber zusätzlich einen Parameter *--help* an.

Versuche es mit dem *cp* Kommando:

```
cp --help | more
```

Auch hier ist eine Weiterleitung an das Kommando **more** sinnvoll, da sonst die Ausgabe sehr schnell durchläuft.

Benutzer anlegen

Bisher warst Du immer als Benutzer *root* angemeldet. Dieser Benutzer hat alle Rechte auf einem Linux-System. Insbesondere wenn Du später mit einer grafischen Benutzeroberfläche arbeitest, solltest Du dich immer mit einem normalen Benutzerkonto anmelden und nur mir root-Rechten arbeiten, wenn dies wirklich notwendig ist. Mit folgendem Befehl kannst Du als Benutzer root ein neues Konto für einen normalen Benutzer anlegen:

```
useradd -m -s /bin/bash -g users -c 'Linux User' -G audio,cdrom,floppy,plugdev,video linuxuser
```

Hinweis: Das Single-Tick Zeichen findest Du auf der Schweizerdeutschen Tastatur wenn Du die Taste mit dem ? drückst.

useradd ist ein Kommando zum Anlegen eines Benutzerkontos. Der Parameter **-m** stellt sicher, dass auch das Heimatverzeichnis für den Benutzer erstellt wird, falls dieses noch nicht existiert.

Wie Du bereits gelernt hast, ist das Heimatverzeichnis für den Benutzer *root* `/root`. Die Homeverzeichnisse für normale Benutzer befinden sich in der Regel im Verzeichnis /home. In diesem Verzeichnis sollte sich nun ein neues Unterverzeichnis mit dem Namen des Benutzers befinden. Prüfen kannst Du dies mit:

```
ls -al /home
```

```
root@darkstar:/# ls -al /home/
insgesamt 16
drwxr-xr-x  4 root      root  4096 Mai 20 10:01 ./
drwxr-xr-x 22 root      root  4096 Mai  1 17:54 ../
drwxr-xr-x  2 root      root  4096 Jun 13  2016 ftp/
drwxr-xr-x  2 linuxuser users 4096 Mai 20 10:01 linuxuser/
root@darkstar:/# _
```

Damit man in einen Ordner wechseln kann, müssen *read* (**r**) und *execute* (**x**) vergeben sein. Die Standardberechtigungen für neue Verzeichnisse sind:

> `-rwxr-xr-x`

Oktal wäre dies **Owner**: `rwx` = 4+2+1 = 7 / **Group**: `r-x`: 4+0+1 = 5 / **Others**: `r-x`: 4+0+1 = 5 oder zusammengesetzt: **755**.

Der Eigentümer *linuxuser* kann lesend und schreibend auf das Homeverzeichnis zugreifen. Die Gruppe *users* und alle anderen haben ebenfalls die Möglichkeit in das Verzeichnis zu wechseln.

Damit nur der Benutzer selbst auf sein Homeverzeichnis zugreifen kann, empfiehlt es sich diese Rechte anzupassen:

```
chmod 700 /home/linuxuser
```

Prüfe die Ausgabe erneut mit:

```
ls -al /home
```

```
root@darkstar:/# chmod 700 /home/linuxuser/
root@darkstar:/# ls -al /home/
insgesamt 16
drwxr-xr-x  4 root      root  4096 Mai 20 10:01 ./
drwxr-xr-x 22 root      root  4096 Mai  1 17:54 ../
drwxr-xr-x  2 root      root  4096 Jun 13  2016 ftp/
drwx------  2 linuxuser users 4096 Mai 20 10:01 linuxuser/
root@darkstar:/# _
```

Schauen wir uns noch einmal das **useradd** Kommando an:

```
useradd -m -s /bin/bash -g users -c 'Linux User' -G audio,cdrom,floppy,plugdev,video linuxuser
```

Der Parameter **-s /bin/bash** besagt, dass die BASH als Login Shell verwendet werden soll. **-g users** definiert die primäre Gruppe des Benutzers. Alle Dateien die von dem Benutzer angelegt werden, erhalten standardmässig diese Gruppenberechtigungen, sofern nichts anderes difiniert wurde.

Über **-c 'Linux User'** wird der Klarname des Benutzers angegeben. Falls dieser Leerzeichen oder Sonderzeichen enthält muss er in *Single Ticks* Anführungszeichen angegeben werden, da sonst die BASH damit nicht umgehen kann.

Mit Hilfe von **-G audio,cdrom,floppy,plugdev,video** kannst Du zusätzliche Gruppen angeben, zu denen der Benutzer hinzugefügt werden soll. Die Gruppennamen sind oftmals selbsterklärend. Durch die Angabe der obigen Gruppen erhält der Benutzer Zugriff auf Audiogeräte, CD- und DVD-Laufwerke, Floppydrives, angeschlossene externe Datenträger und auf die Videoausgabe.

Es folgt der Loginname **linuxuser**. Dieser darf ausschliesslich aus Kleinbuchstaben bestehen und sollte keine Leerzeichen enthalten.

Überprüfen kannst Du die Gruppenzugehörigkeit mit dem Befehl **id**. Als Parameter ist der Loginname des Benutzers anzugeben, in diesem Beispiel *linuxuser*.

`id linuxuser`

```
root@darkstar:/# id linuxuser
uid=1000(linuxuser) gid=100(users) Gruppen=100(users),11(floppy),17(audio),18(video),19(cdrom),83(pl
ugdev)
root@darkstar:/#
```

Die Ausgabe von **id** enthält immer zuerst den numerischen Bezeichner, gefolgt vom wörtlichen Bezeichner. *uid*= ist die id des Benutzers (hier 1000) gefolgt vom Loginnamen. *gid* gibt die primäre Gruppe an, in diesem Fall *users* mit der id *100*. Darauf folgen alle weiteren zugeordneten Gruppen.

Der Befehl **ls** bietet mit dem **-n** Parameter die Möglichkeit an, numerische Bezeichner anzeigen zu lassen. Dabei ersetzt der Parameter **-n** den zuvor verwendeten Parameter **-l**:

`ls -an /home`

```
root@darkstar:/# ls -an /home/
insgesamt 16
drwxr-xr-x  4    0   0 4096 Mai 20 10:01 ./
drwxr-xr-x 22    0   0 4096 Mai  1 17:54 ../
drwxr-xr-x  2    0   0 4096 Jun 13  2016 ftp/
drwx------  2 1000 100 4096 Mai 20 10:01 linuxuser/
root@darkstar:/#
```

Du siehst das anstatt wie zuvor der Benutzer *linuxuser* und die Gruppe *users* ausgegeben wird, nun die numerischen Bezeichner an der gleichen Stelle angezeigt wird. Im Normalfall wirst Du allerdings die wörtlichen Bezeichner verwenden.

vipw und vigr

Linux speichert die Informationen über Benutzer in der Datei */etc/passwd*. Gruppeninformationen werden in der Datei */etc/groups* hinterlegt.

Diese Dateien sollten nicht direkt bearbeitet werden, sondern mit den Kommandos **vipw** (für */etc/passwd*) und **vigr** (für /etc/groups).

Die Kommandos **vipw** und **vigr** starten den Editor *elvis* und öffnen die entsprechenden Benutzer respektive Gruppendateien. Bei *elvis* handelt es sich um einen Clone des Editors *vi*.

Elvis wird aufgerufen wenn man unter Slackware **vi** eingibt, da **vi** ein sogenannter symbolischer Link ist, der auf das Programm *elvis* zeigt.

Symlinks

Gebe folgenden Befehl ein, um dir den Link anzeigen zu lassen:

`ls -al /usr/bin/vi`

```
root@darkstar:~# ls -al /usr/bin/vi
lrwxrwxrwx 1 root root 5 Mai  1 17:48 /usr/bin/vi -> elvis*
root@darkstar:~#
```

Du siehst das */usr/bin/vi* auf eine Datei Namens *elvis* zeigt. Wird beim Linkziel kein Pfad angegeben, wird davon ausgegangen, dass sich das Ziel im gleichen Verzeichnis wie der Link befindet (*/usr/bin*)

Symbolische Links werden in der Ausgabe von **ls -al** wie folgt dargestellt:

> `lrwxrwxrwx`

Das erste **l** gibt an, dass es sich um einen Link handelt. Die Permissions des Linkfiles sind immer 777. Das heisst, der Link selbst darf von allen aufgelöst werden, bedeutet aber nicht, dass man Zugriff auf das Linkziel hat. Denn es gelten letztendlich die Berechtigungen, die auf dem Linkziel gesetzt sind.

Wir möchten statt *elvis/vi* die erweiterte Version mit dem Namen *vim* verwenden (*vi impoved*). Dazu müssen wir dem System zunächst mitteilen, dass bei einem Aufruf des Kommandos **vi** nicht **elvis** sondern **vim** gestartet wird.

Das können wir lösen, indem wir einen Symlink erstellen, der auf *vim* statt auf *elvis* zeigt.

PATH

Linux sucht ausführbare Programme (Binaries) anhand der sogenannten PATH Variable. Diese wird linear abgearbeitet. Du kannst dir den Inhalt der Pfad Variable mit folgendem Befehl ausgeben lassen:

```
echo $PATH
```

```
root@darkstar:~# echo $PATH
/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/games:/usr/lib64/qt/bin:/usr/share
/texmf/bin
root@darkstar:~#
```

Das System sucht in diesem Fall Programme zunächst im Verzeichnis */usr/local/sbin* dann in */usr/sbin*, */sbin*, */usr/local/bin /usr/bin* und so weiter.

Die *sbin* Verzeichnisse stehen nur dem Benutzer root zur Verfügung, normale Benutzer haben diese Verzeichnisse nicht in der $PATH Variable.

Du siehst, dass immer erst in den */usr/local* Verzeichnissen gesucht wird. Diese Verzeichnisse dienen lokalen Systemanpassungen. Gleichnamige Programme in diesen Verzeichnissen werden bevorzugt. Das heisst, wenn in /usr/local/bin eine Datei oder ein Link mit dem Namen *vi* existiert, wird dieser vor einer gleichnamigen Datei oder einem Link in */usr/bin* bevorzugt.

Diesen Mechanismus können wir uns zu nutze machen, indem wir einen symbolischen Link für *vi* in */usr/local/bin* anlegen, der auf *vim* zeigt:

```
ln -s /usr/bin/vim /usr/local/bin/vi
```

```
root@darkstar:~# ln -s /usr/bin/vim /usr/local/bin/vi
root@darkstar:~# _
```

Der Befehl **ln** erstellt einen Link. Der Parameter **-s** gibt an, dass es sich um einen symbolischen Link handelt. Darauf folgt das Linkziel (*/usr/bin/vim*) und der Name des Links (*/usr/local/bin/vi*).

Damit die Änderungen appliziert werden musst Du dich einmal abmelden und erneut als Benutzer root anmelden. Eine aktive Shellsitzung kannst Du mit **Strg d** oder durch Eingabe von **exit** beenden.

Wenn Du nun *vi* eingibst, öffnet sich der Editor *vim*.

Ein wichtiger Befehl vorab: `Esc` + `:` + `q` + `!`, also **Escape** gefolgt von einem **:**, einem **q** und einem **!**. Dieser Befehl verlässt den Editor ohne die Änderungen die Du gemacht hast abzuspeichern. Dies ist besonders jetzt in der Übungsphase hilfreich, da Du so nicht unbeabsichtig etwas an wichtigen Systemdateien verstellen kannst.

Hinweis: im Folgenden verzichten wir auf die Verwendung des **+** Zeichens zur Erläuterung der Befehlsverkettungen.

Testweise kannst Du *elvis* aufrufen, indem Du den kompletten Pfad zu dem Systemsymlink angibst:

```
/usr/bin/vi
```

Du siehst das dich *elvis* nicht mit dem ausführlichen Willkommenstext von *vim* begrüsst.

Die Grundkommandos in *elvis/vi* und *vim* sind allerdings identisch. Auch hier lannst du `Esc :q!` eingeben um den Editor zu verlassen.

vim

Nun kannst Du deinen Ausflug in die Welt des Editors *vim* beginnen.

Der Editor **vim** ist ganz anders als alle Editoren die du wahrscheinlich kennst. Er ist sehr mächtig und wird von vielen Linux-Anwendern gerne verwendet. Alle Funktionen des Editors zu erklären würde den Rahmen dieses Kurses sprengen.

Bevor Du **vim** das allererste mal startest, bietet es sich an, einige Vorkonfigurationen vorzunehmen. Diese Einstellungen kannst Du mit dem Editor **nano** vornehmen:

`nano ~/.vimrc`

Das *~/* steht für das Homeverzeichnis des aktuell angemeldeten Benutzers, in diesem Falle */root*. Wir bearbeiten die vim Konfigurationsdatei *.vimrc* im Verzeichnis /root.

Folgende Konfiguration hat sich bewährt:

```
set mouse=
set nobackup
set noswapfile
set noundofile
set noai
set paste
set noincsearch
set nohlsearch
```

Speichere die Datei in nano mit **Strg o** ab und verlasse nano mit **Strg x**.

Wir schauen uns als nächstes die Grundfunktion von **vim** am Beispiel des Kommandos **vipw** an. Doch zuvor sichern wir die wichtige Systemdatei */etc/passwd* mit folgendem Befehl:

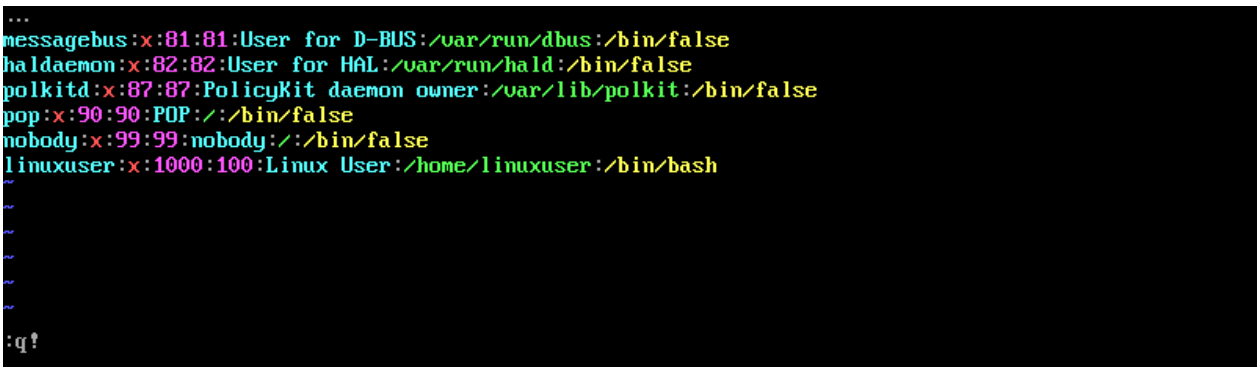
`cp /etc/passwd /etc/passwd.orig`

Daraufhin kannst Du **vipw** starten:

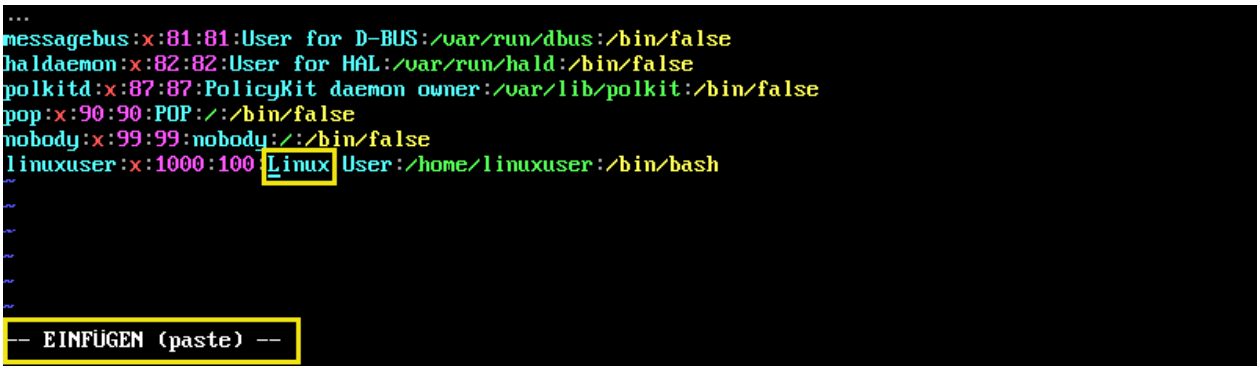
```
vipw
```

Der grösste Unterschied von *vim* im Vergleich zu anderen Editoren ist, dass es einen Befehlsmodus und einen Eingabemodus gibt. Der Editor startet standardmässig immer in den Befehlsmodus. Dort kannst Du zwar mit den Cursortasten navigieren, allerdings nicht einfach lostippen. Du kannst jederzeit wieder mit der Escape-Taste in den Befehlsmodus zurückkehren.

Befehle werden in vi am unteren Bildschrimrand angezeigt und mit der Entertaste ausgelöst.



Versuche nun einmal den Anzeigenamen deines Benutzers zu ändern. Navigiere dazu mit den Pfeiltasten auf den ersten Buchstaben des Anzeigenamens deines Benutzers; in unserem Beispiel wäre dies *Linux User*. Drücke die **i** Taste um in den Eingabemodus zu gelangen. Dieser wird unten links signalisiert.

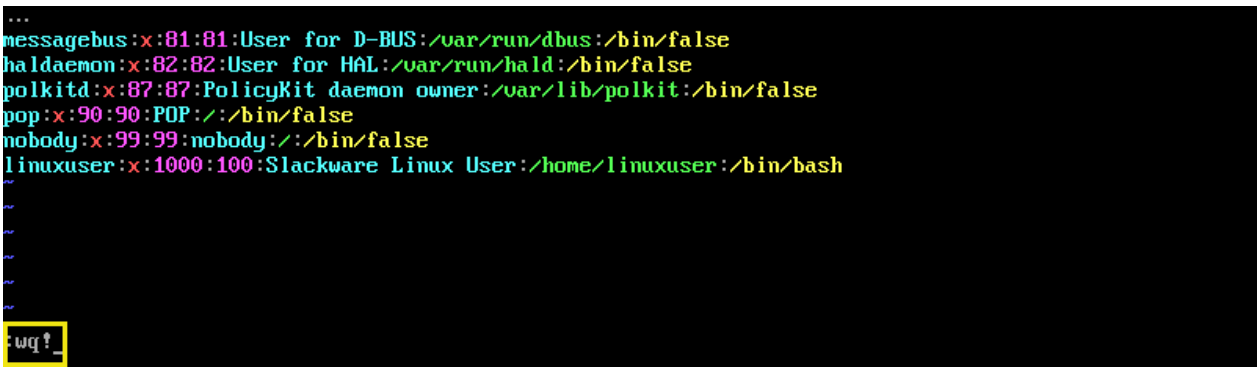


Erst jetzt hast Du die Möglichkeit Text einzufügen. Schreibe nun etwas anderes dort hinein, in diesem Beispiel *Slackware Linux User*.

Wenn Du einen Fehler gemacht hast, kannst du mit `Esc u`, also **Escape** gefolgt von **u** deine letzte Änderung rückgängig machen. *vim* kann im Gegensatz zu *elvis/vi* beliebig viele Befehle rückgängig machen.

Beachte bitte, dass du nach der Nutzung der *undo* Funktion erst wieder mit *i* in den Eingabemodus wechseln musst um erneut Text einzufügen.

Abspeichern kannst Du deine Änderung jederzeit mit `Esc :w!`. Mit Hilfe von `Esc :wq!` speicherst Du die Änderungen ab und verlässt den Editor.



Wie du vielleicht festgestellt hast, beginnt die Eingabe genau an der Stelle, an der sich der Cursor befindet, wenn Du den Eingabemodus mit **i** gestartet hast. Alternativ kannst Du **a** verwenden um die Eingabe rechts neben dem Cursor zu starten.

Nachdem Du ein wenig herumexperimentiert hast, wirst Du wahrscheinlich feststellen, dass insbesondere das Löschen von Zeichen oder Wörtern anders funktioniert als in anderen Editoren.

Am einfachsten geht das Löschen von Zeichen im Eingabemodus mit Hilfe der **Delete** Taste. Diese löscht das Zeichen unter dem Cursor.

Noch komfortabler geht es im Befehlsmodus. Gehe dazu mit dem Cursor auf den ersten Buchstaben des Wortes das Du löschen möchtest. In unserem Beispiel möchten wir versuchen, den Benutzernamen von *Slackware Linux User* auf *Slackware User* abzuändern. Wenn Du auf dem *L* von *Linux* stehst, drücke `Esc dw`, also Escape gefolt von *d* und *w* und das gesamte Wort unter dem Cursor wird gelöscht.

Alternativ kannst Du mit `Esc d Pfeil-rechts oder Pfeil-links` ein Zeichen rechts oder links neben dem Cursor entfernen. Mit Hilfe von `Esc dd` löscht du eine ganze Zeile.

Noch ein paar nützliche Kommandos:

- **`Esc Shift_A`**, Escape gefolgt von Shift und a (gleichzeitig) springt an das Ende einer Zeile und startet den Eingabemodus
- **`Esc Shifg_g`** Escape gefolgt von Shift und g (gleichzeitig) springt in die letzte Zeile der Datei, bleibt allerdings im Befehlsmodus
- **`Esc /Suchwort`** Escape gefolgt von / und einem Suchwort (z.B. **`Esc /User`**) durchsucht die Datei nach dem Suchwort und springt mit dem Cursor zum ersten Treffer. Weitere Treffer können angezeigt werden, wenn Du nach der Suche direkt **n** für *next* angibst.

Ein wirklich praktisches Features ist die integrierte Suchen-und-Ersetzen Funktion von *vim*. Sie wird wie folgt aufgerufen:

`Esc :%s/Suchwort/Ersatz/g`

Dieser Befehl ersetzt alle Vorkomnisse von dem Begriff *Suchwort* durch *Ersatz*.

Damit hast Du die wichtigsten Kommandos zum Umgang mit dem Editor *vim* gelernt, den wir auch in diesem Kurs in Zukunft zum Bearbeiten von Dateien verwenden werden.

Solltest Du dir während der Tests die wichtige Systemdatei */etc/passwd* verstellt haben, kannst Du die Sicherung mit folgendem Befehl zurückspielen:

```
cp /etc/passwd.orig /etc/passwd
```

rm

Mit Hilfe des **rm** Befehls kannst Du die *.orig* Datei löschen:

```
rm /etc/passwd.orig
```

rm hat die Datei ohne Nachfrage entfernt. Das möchten wir gerne ändern, damit nicht versehentlich wichtige Dateien gelöscht werden.

Erstelle dazu bitte die Datei *.bash_profile*. Dabei handelt es sich um eine benutzerspezifische Konfigurationsdatei der BASH, die bei jeder interaktiven Anmeldung an einem Terminal ausgeführt wird.

```
vi /root/.bash_profile
```

Trage dort folgende Zeile ein:

```
alias rm='rm -i'
```

Mit dieser Konfigurationseinstellung (*alias*) legst Du fest, dass jedes Mal wenn Du den Befehl **rm** ausführst, stattdessen **rm -i** gestartet wird.

Der Parameter **-i** sorgt dafür, dass vor dem Entfernen einer Datei eine Nachfrage erscheint.

Damit die Einstellung auch in der aktuell laufenden Shell appliziert wird, kannst Du die Konfigurationsdatei einlesen. Man spricht bei diesem Vorgang auch von *sourcen*:

```
. /root/.bash_profile
```

Achte bitte hierbei auf das Leerzeichen zwischen dem Punkt und dem /. Alternativ könntest Du dich abmelden und als Benutzer *root* erneut anmelden.

Teste ob die Anpassungen aktiv sind, indem Du zunächst eine leere Datei anlegst und daraufhin versuchst diese zu löschen. Leere Dateien lassen sich einfach mit Hilfe des Befehls **touch** erstellen:

```
touch /root/test
rm /root/test
```

Die Nachfrage die erscheint kannst du mit **y** bestätigen.

```
root@darkstar:~# touch /root/test
root@darkstar:~# rm /root/test
rm: reguläre leere Datei '/root/test' entfernen? y
```

Benutzer ändern

Du hast bereits gelernt wie man mit **useradd** einen unpriviligierten Benutzer mit dem Loginnamen *linuxuser* und dem Klarnamen *Slackware Linux User* erstellt. Diesen kannst Du mit dem **usermod** Kommando auf Deinen eigenen Namen anpassen.

Ein Beispiel für Maria Muster:

`usermod -c 'Maria Muster' -d /home/mmuster -m -l mmuster linuxuser`

Hier wurde der Loginname in der Form $ERSTER_BUCHSTABE_VOM_VORNAMEN gefolgt von $NACHNAME angegeben (*mmuster*). Du kannst natürlich auch einfach deinen Vornamen oder einen Fantasienamen angeben.

Der Parameter **-c 'Maria Muster'** legt den neuen Klarnamen des Benutzers fest. **-d /home/mmuster** ist das neue Homeverzeichnis. Es wird empfohlen, dass der Name des Homverzeichnisses dem des Loginnamens entspricht (in diesem Fall *mmuster*). Mit **-m** legst Du fest, dass der Inhalt des vorhandenen Homeverzeichnisses übernommen werden soll. **-l mmuster** gibt den neuen Loginnamen an. Der Parameter am Ende des Befehls ist der Loginnname des Benutzers den du ändern möchtest (hier: *linususer*).

Überprüfen kannst Du die Änderungen wie gelernt mit `ls -al /home` und dem Befehl **vipw** sowie dem Befehl **id** gefolgt vom Loginnamen des Benutzers. Alternativ zu **id** kannst Du das Kommando **groups** verwenden, welches ebenfalls mit dem Loginnamen als Parameter aufgerufen werden kann.

Hinweis: Im Verlauf des Kurses wird weiterhin der Benutzer *linuxuser* als Beispiel verwendet. Bei entsprechenden Befehlsaufrufen musst Du diesen durch deinen neuen Loginnamen ersetzen:

`groups linuxuser`

```
root@darkstar:/home# groups linuxuser
linuxuser : users floppy audio video cdrom plugdev
root@darkstar:/home#
```

Passwort festlegen

Dem neu erstellten Benutzer wurde bisher noch kein Passwort zugewiesen. Dies kannst Du nun mit dem Befehl **passwd** nachholen.

`passwd linuxuser`

Den Loginnamen des Benutzer kannst Du durch deinen neuen Loginnamen ersetzen. Das Passwort wird während der Eingabe nicht angezeigt.

```
root@darkstar:/home# passwd linuxuser
Passwort für linuxuser wird geändert.
Geben Sie das neue Passwort ein (mindestens 5 Zeichen).
Bitte benutzen Sie eine Kombination aus Groß- und Kleinbuchstaben
sowie Ziffern.
Neues Passwort:
Passwort wiederholen:
passwd: Passwort geändert.
root@darkstar:/home# _
```

Abschliessend kannst Du dich mit deinem neuen Benutzerkonto am System anmelden. Öffne dazu eine weitere *tty* durch Eingabe von `Strg Alt F2`. Die Tasten werden gleichzeitig gedrückt. Bei einer *tty* handelt es sich um eine virtuelle Konsole. Üblicherweise gibt es unter Linux sechs virtuelle Konsolen die du mit `Strg Alt F1 - F6` erreichst. Die grafische Oberfläche wird später mit `Strg Alt F7` anwählbar sein.

Melde dich dort mit deinem Benutzernamen und Passwort des normalen Benutzerkontos an.

```
Welcome to Linux 4.4.14 (tty2)

darkstar login: linuxuser
Password:
Linux 4.4.14.
Last login: Tue May 21 04:06:02 +0200 2019 on /dev/tty1.
No mail.

Toto, I don't think we're in Kansas anymore.
                -- Judy Garland, "Wizard of Oz"

linuxuser@darkstar:~$
```

Zu deiner Rootshell kannst Du durch mit Hilfe von **Strg Alt F1** zurückwechseln.

Somit hast Du die ersten Schritte in deinem Linux-System getan, erfolgreich die Sprache umgestellt, Systemdienste kennengelernt und erste Erfahrungen mit vielen wichtigen Kommandos machen können. Du hast dein eigenes Benutzerkonto erstellt, mit dem du dich an deinem Linux-System anmelden kannst. Im nächsten Kursteil geht es um die Einrichtung der grafischen Benutzeroberfläche.

Grafische Oberfläche < PDF >
↪ https://faircomputer.ch/linuxkurs/Grafische_Benutzeroberflaeche.pdf

Grafische Oberfläche < HTML >
↪ https://faircomputer.ch/linuxkurs/Grafische_Benutzeroberflaeche.html

---