Linux: Das Handwerkszeug für die tägliche Arbeit

Janis Streib
CC-BY-ND-4.0

08.04.-10.04.2019

https://creativecommons.org/licenses/by-nd/4.0/

Inhalt

Einführung
- Distributionen

Installation
- Bootmedium
- Installation (graphisch)

Arbeitsumgebungen

Grundlagen des Dateisystems

https://creativecommons.org/licenses/by-nd/4.0/

Hello World!

Über mich

- Janis Streib
- Bachelor of Science Informatik
- Student der Informatik (im Master)
- Freelancer
- Arbeit im Bereich Rechenzentren und Netzwerke
- Softwareentwickler
- `https://janis-streib.de`
- me@janis-streib.de

https://creativecommons.org/licenses/by-nd/4.0/

Linux

- 1991 von Linus Torvalds entwickelter Kernel (Betriebssystemkern)
- Quelloffen und unter freier Lizenz (GPL) veröffentlicht
  - ⇒ Änderungen am Quellcode müssen herausgegeben werden
  - ⇒ Große Unterstützung für verschiedenste Hardware und Architekturen
- Viele verschiedene Konfigurationen und Zusammensetzungen verfügbar (Distributionen)

Figure: Linux Torvalds.
`https://commons.wikimedia.org/wiki/File:Linus_Torvalds.jpeg`

`https://creativecommons.org/licenses/by-nd/4.0/`

Die Qual der Wahl

Distributionen im Desktop-Bereich – eine kleine Auswahl:

- Ubuntu
- Debian
- OpenSuSE
- Fedora
- Gentoo
- Arch Linux

https://creativecommons.org/licenses/by-nd/4.0/

Images beziehen

- Disk-Images (*.iso) stehen auf der Webseite der jeweiligen Distribution zum Download zur Verfügung
  - Images für CD/DVD
  - Disk-Images i.d.R. kompatibel für USB-Sticks

Sicherheitshinweis

Download nur von vertrauenswürdigen Quellen!

- Download möglichst nur über `https` und von Herstellerwebseite
- Alternativ: Prüfsumme über vertrauenswürdige Quelle beziehen (`https` über Herstellerseite) und Download prüfen, bzw. Signatur prüfen (mehr dazu später)

https://creativecommons.org/licenses/by-nd/4.0/

Erzeugen eines Boot-Mediums

- Auf Windows
  - USB-Stick: Externes Tool benötigt. Beispielsweise Rufus `https://rufus.ie/`
  - CD/DVD: Ab Windows 7 können Abbilddateien direkt gebrannt werden (Option im Kontextmenü)
- Auf Linux
  - USB-Stick: Z.B. mit `dd`

    ```
    dd if=<image> bs=1M of=/dev/<device>
    ```

  - CD/DVD: Z.B. mit dem Tool `brasero`

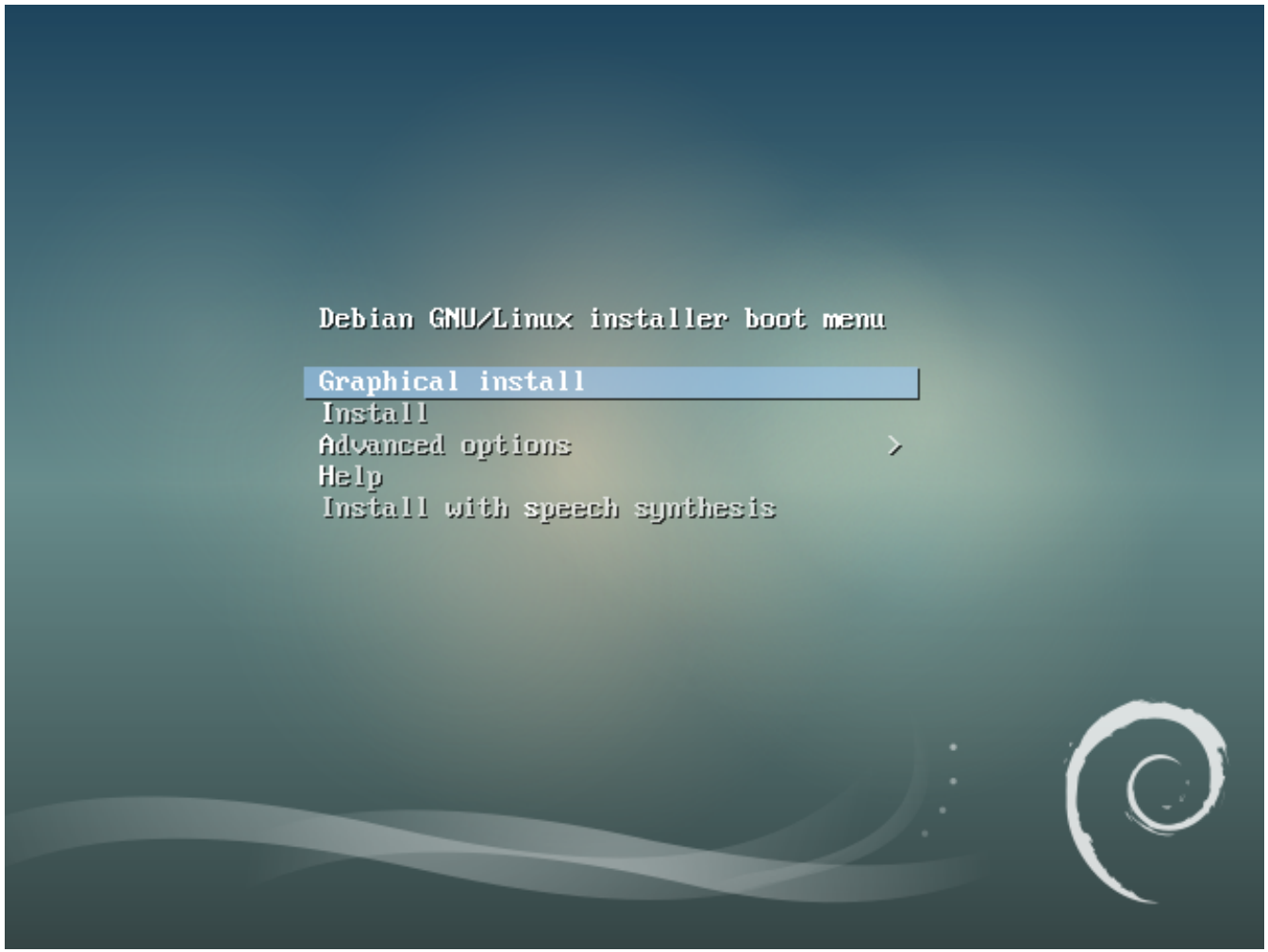
https://creativecommons.org/licenses/by-nd/4.0/

Von Bootmedium booten

1. Stick in Rechner stecken
2. In den BIOS/EFI-Einstellungen sicherstellen, dass zuerst von USB-Stick gebootet wird.
   Alternativ: In BIOS/EFI einmalig von USB-Stick booten ("Boot menu", je nach Hersteller unterschiedlich)
3. Menü der Distribution sollte auftauchen
4. Installationstyp (text-basiert oder grafisch) kann ausgewählt und gestartet werden

https://creativecommons.org/licenses/by-nd/4.0/

1. Wahl der Sprache, Region und Tastaturlayout I

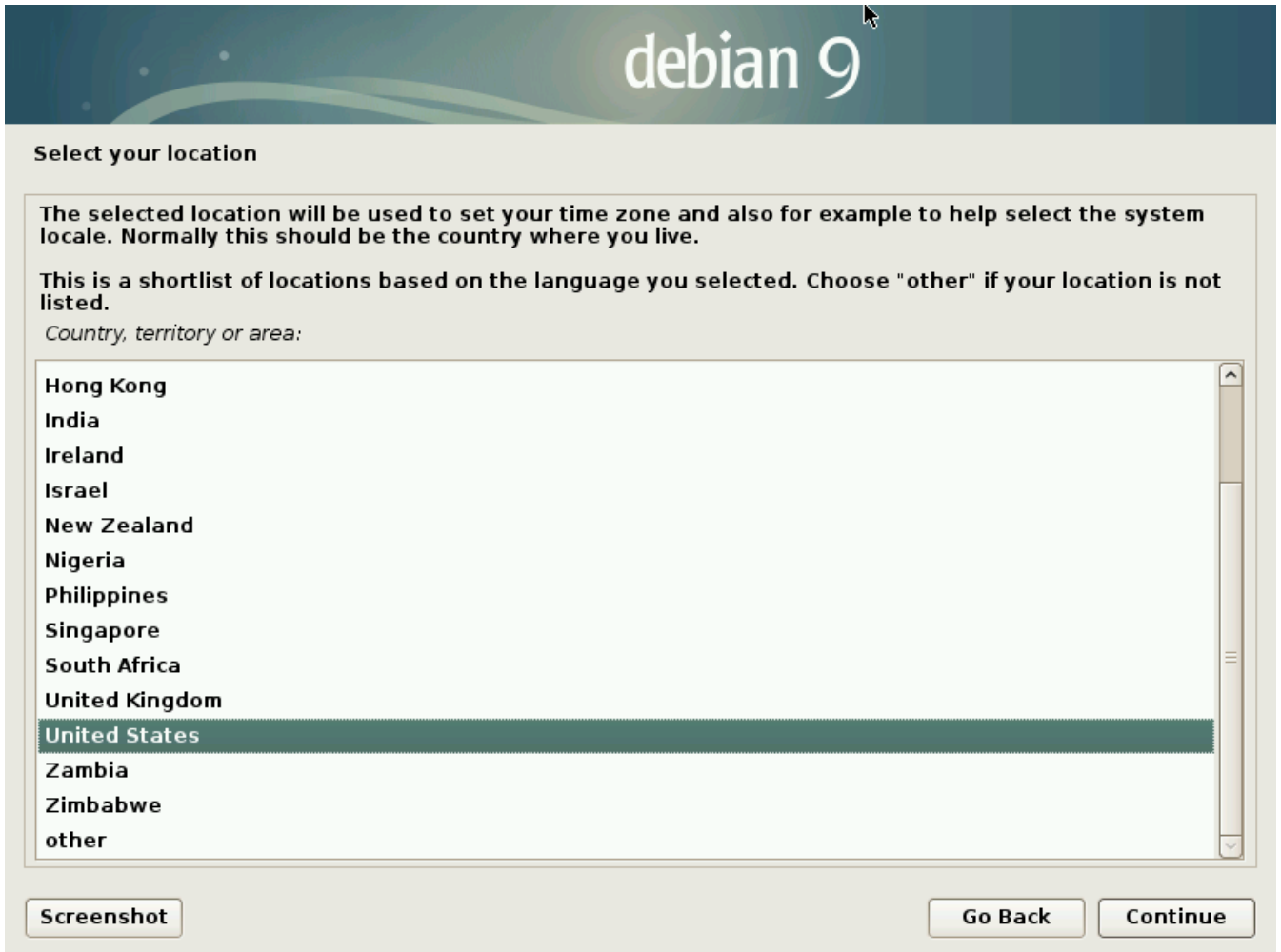


https://creativecommons.org/licenses/by-nd/4.0/

1. Wahl der Sprache, Region und Tastaturlayout II

https://creativecommons.org/licenses/by-nd/4.0/

1. Wahl der Sprache, Region und Tastaturlayout III

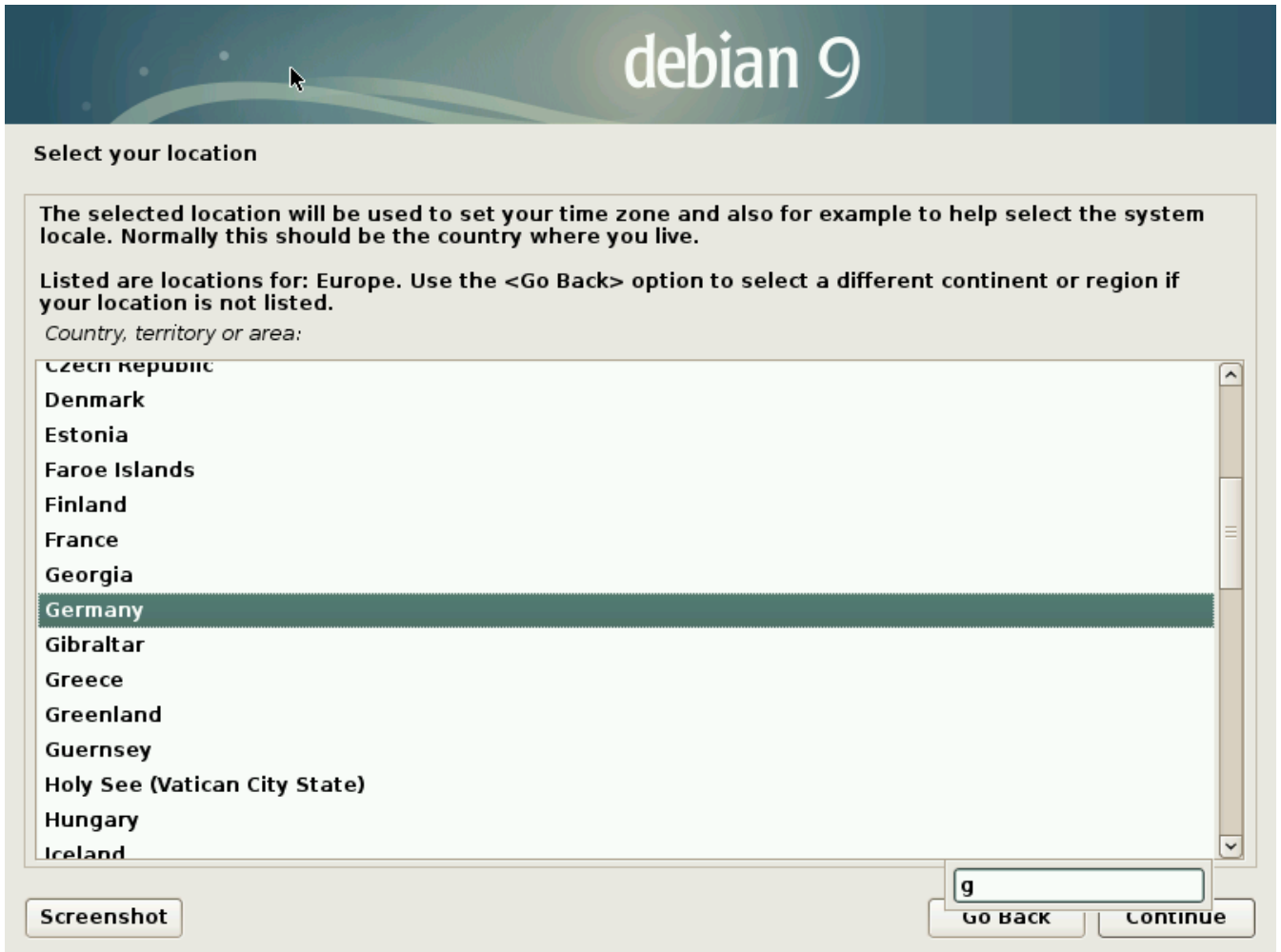


https://creativecommons.org/licenses/by-nd/4.0/

1. Wahl der Sprache, Region und Tastaturlayout IV

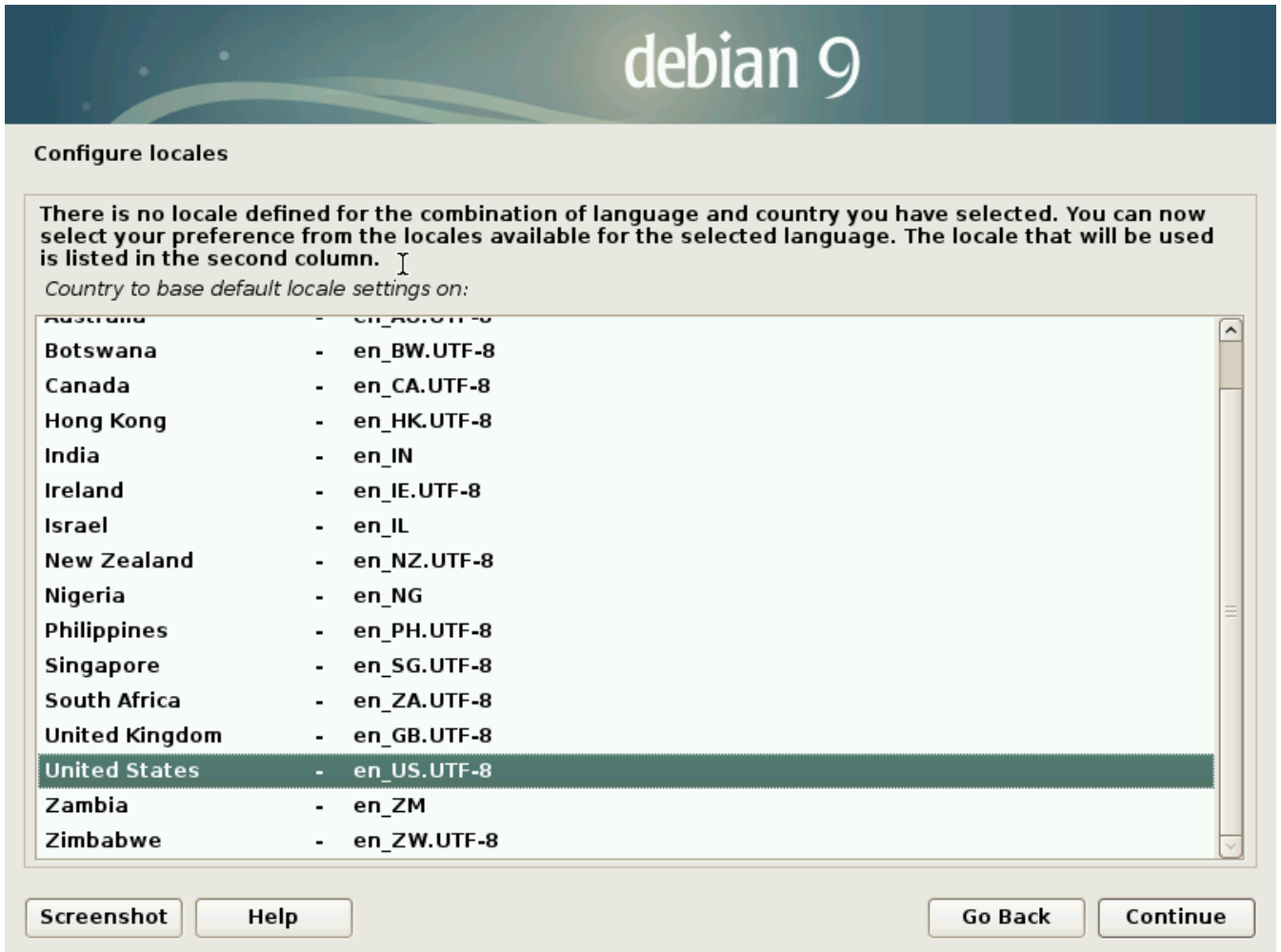
https://creativecommons.org/licenses/by-nd/4.0/

1. Wahl der Sprache, Region und Tastaturlayout V



https://creativecommons.org/licenses/by-nd/4.0/

1. Wahl der Sprache, Region und Tastaturlayout VI



https://creativecommons.org/licenses/by-nd/4.0/

1. Wahl der Sprache, Region und Tastaturlayout VII

https://creativecommons.org/licenses/by-nd/4.0/

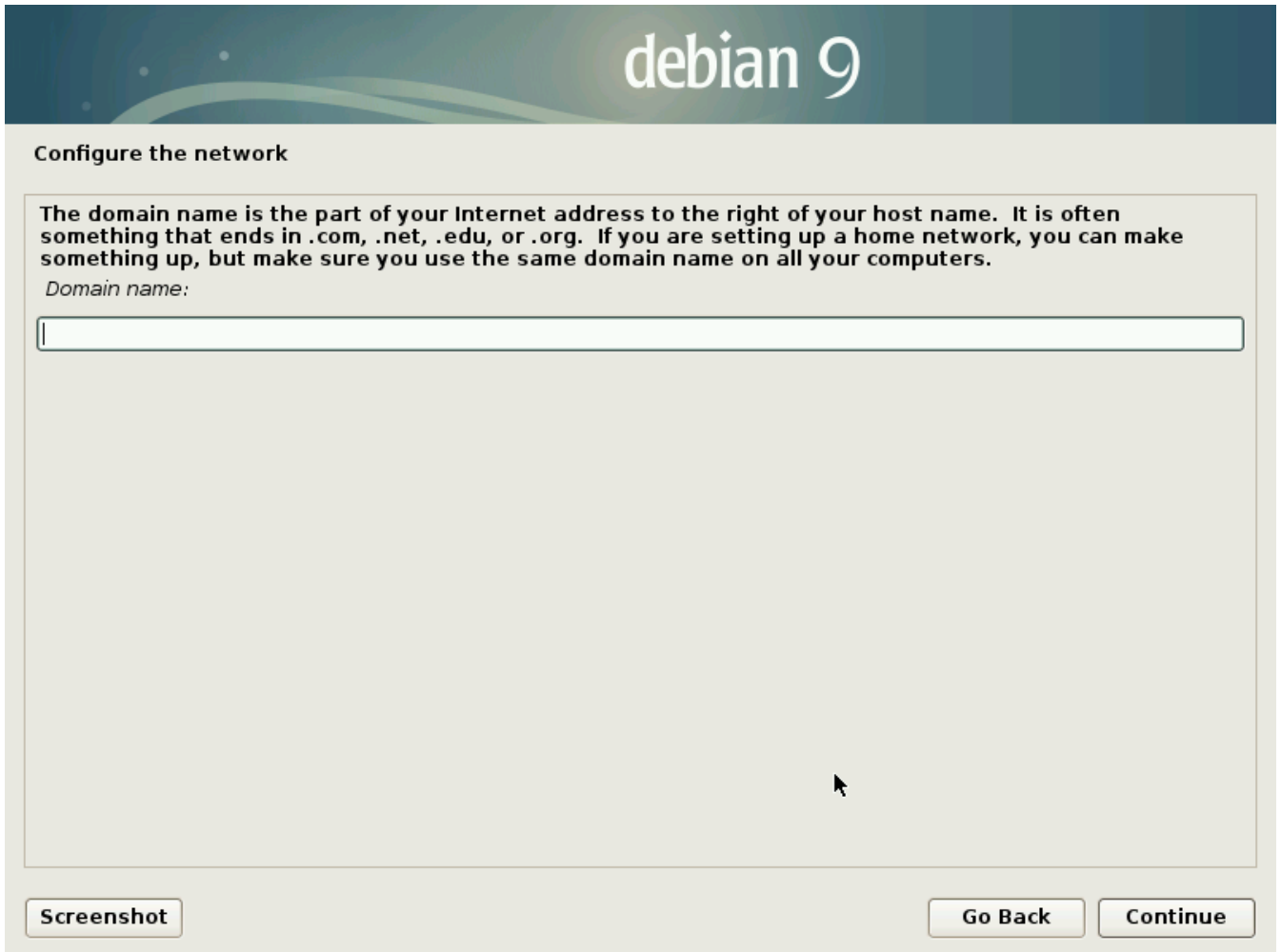
2. Herstellung der Netzwerkverbindung

https://creativecommons.org/licenses/by-nd/4.0/

3. Name und Domain der Maschine festlegen I

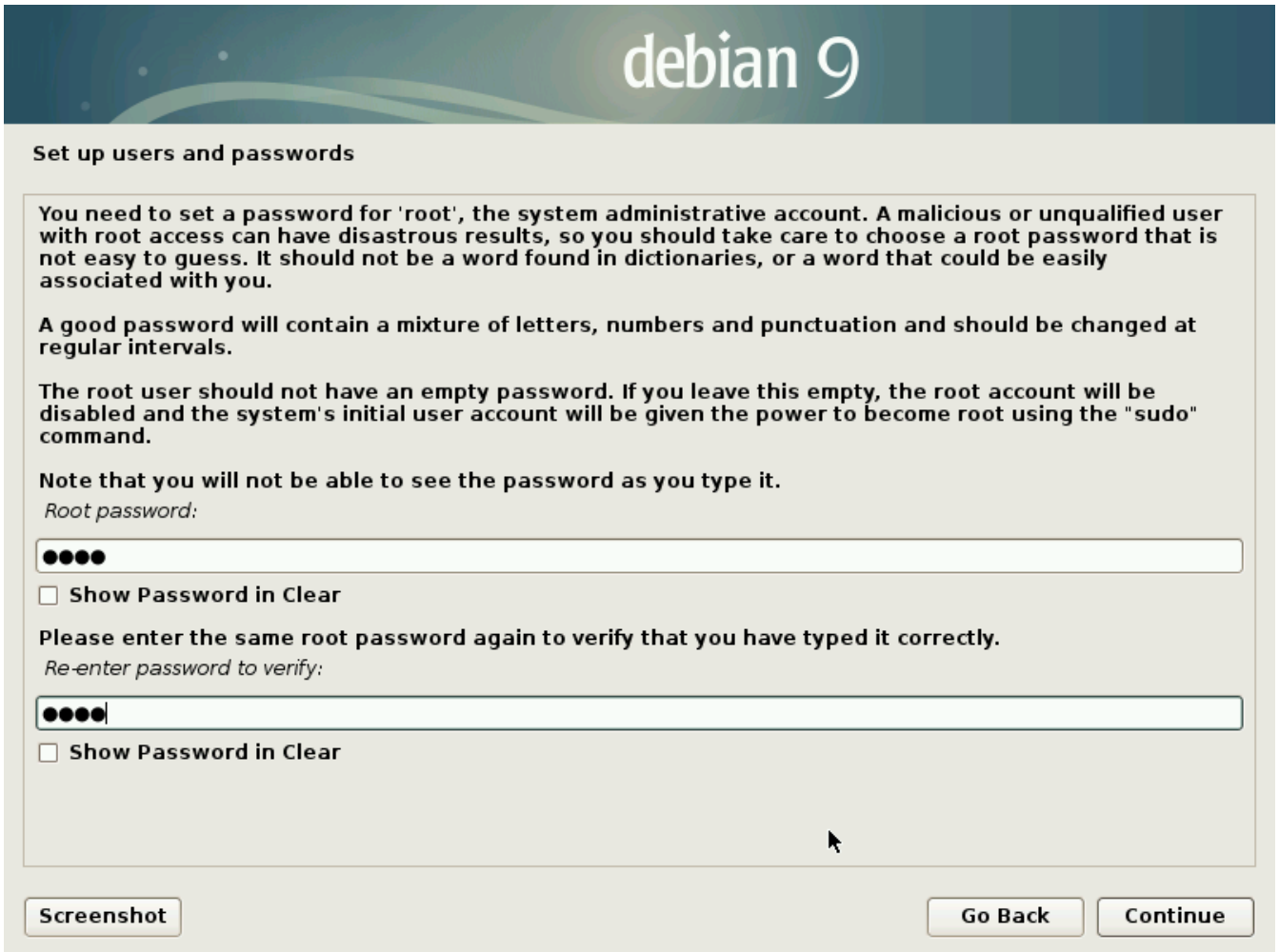
https://creativecommons.org/licenses/by-nd/4.0/

3. Name und Domain der Maschine festlegen II



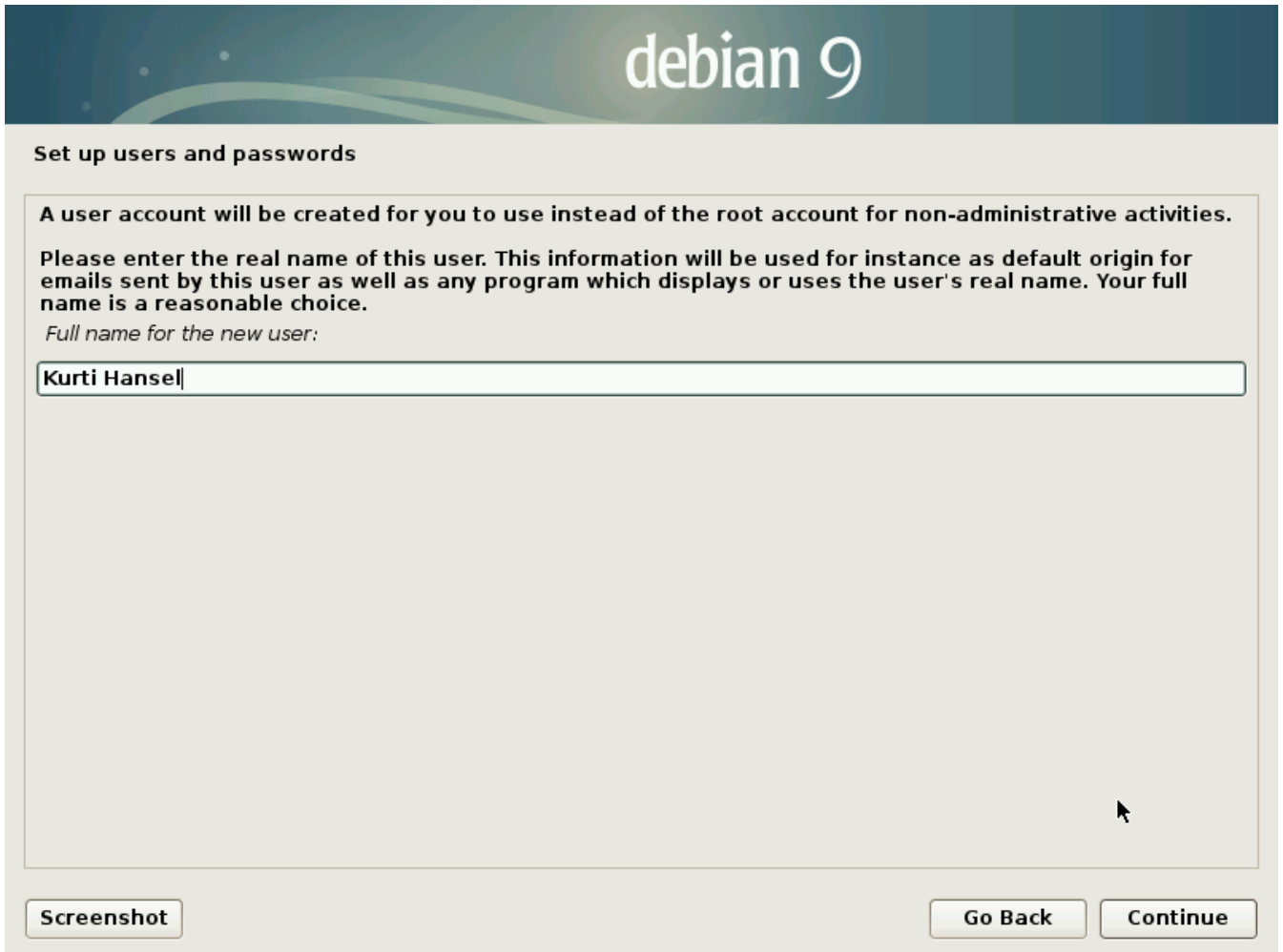
https://creativecommons.org/licenses/by-nd/4.0/

4. Einrichtung des root-Passworts (administrativer User)



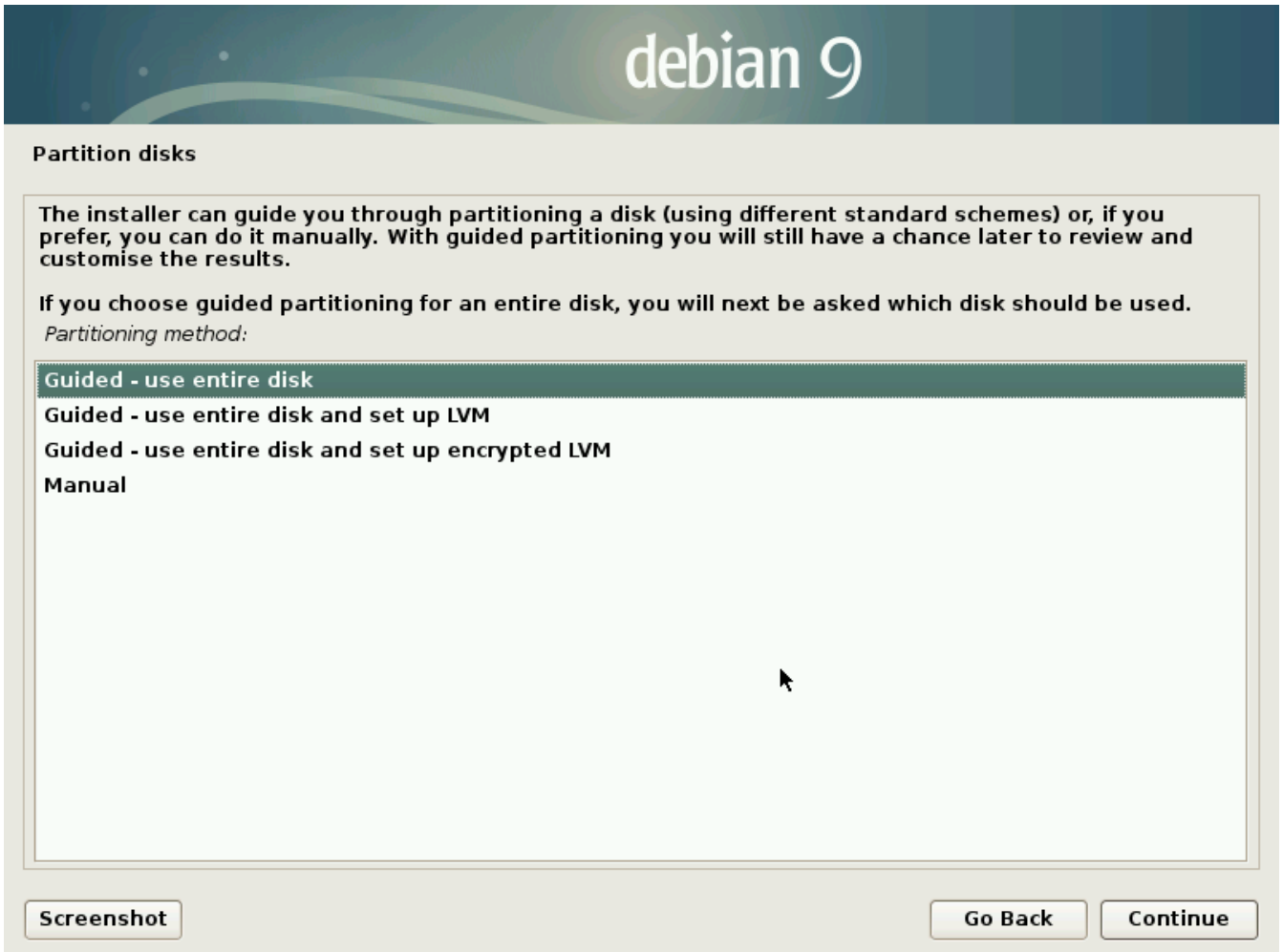
https://creativecommons.org/licenses/by-nd/4.0/

5. Einrichtung des ersten Benutzers I



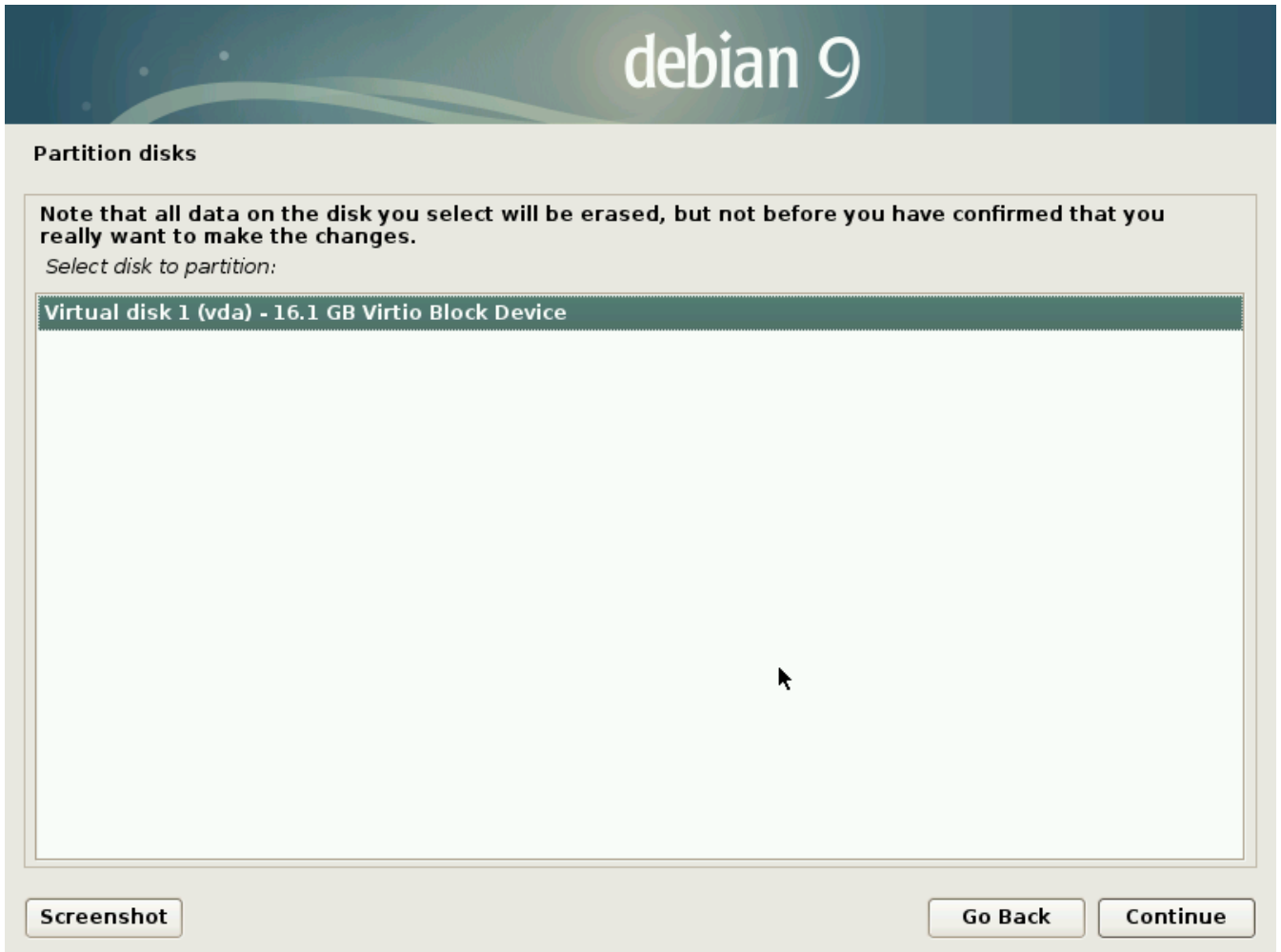
https://creativecommons.org/licenses/by-nd/4.0/

6. Einrichtung der Partitionen I



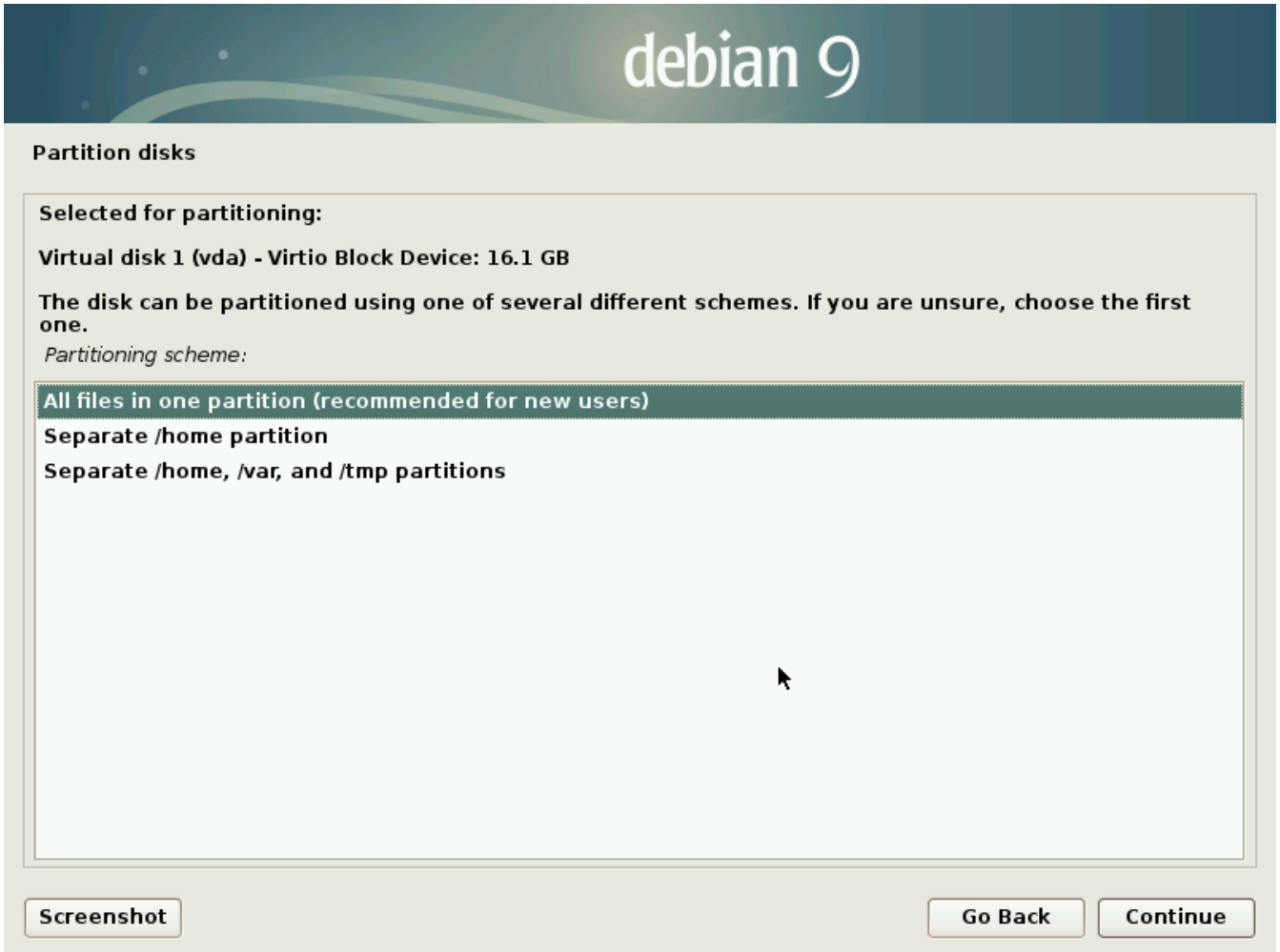
https://creativecommons.org/licenses/by-nd/4.0/

6. Einrichtung der Partitionen II



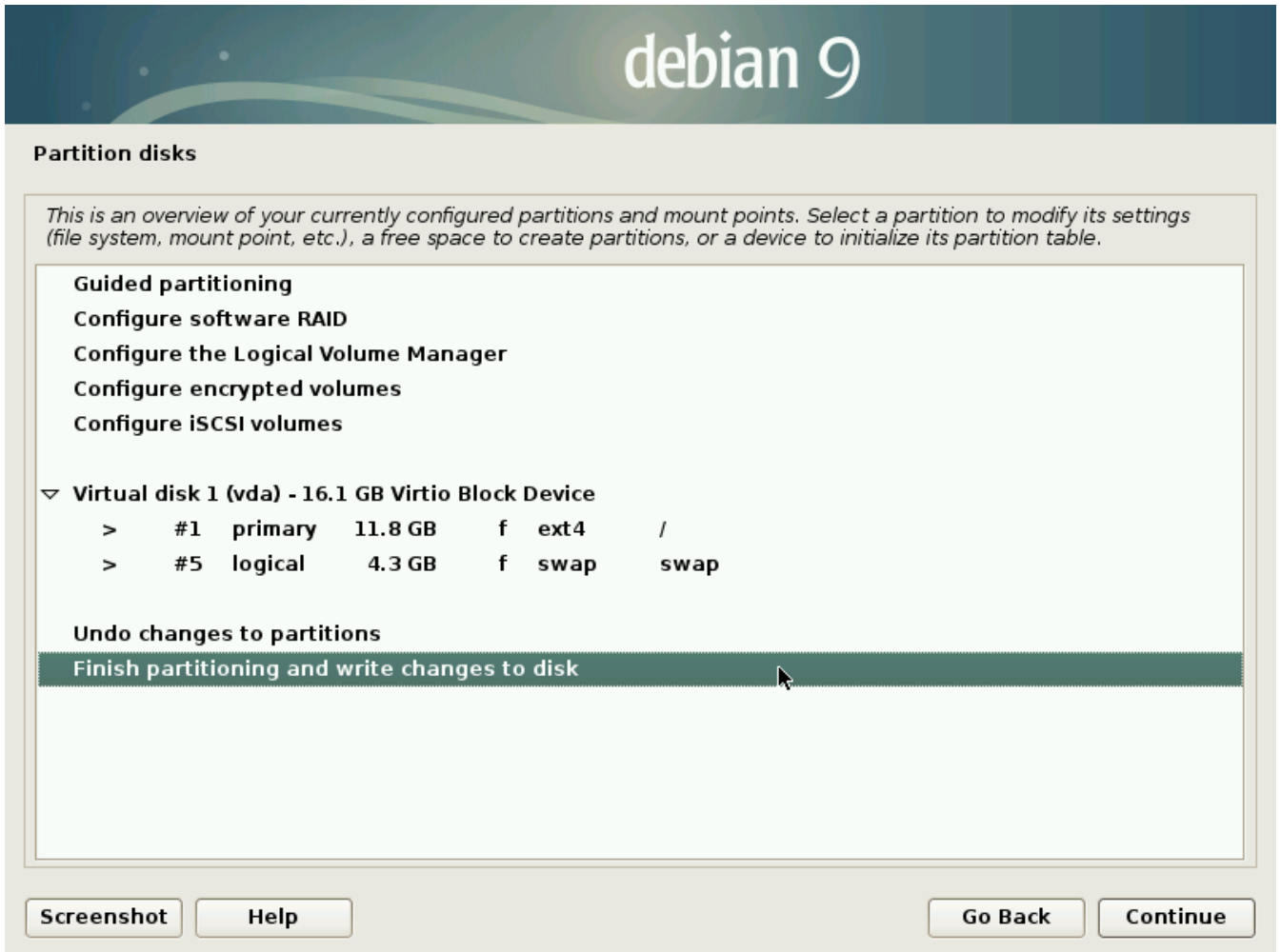
https://creativecommons.org/licenses/by-nd/4.0/

6. Einrichtung der Partitionen III



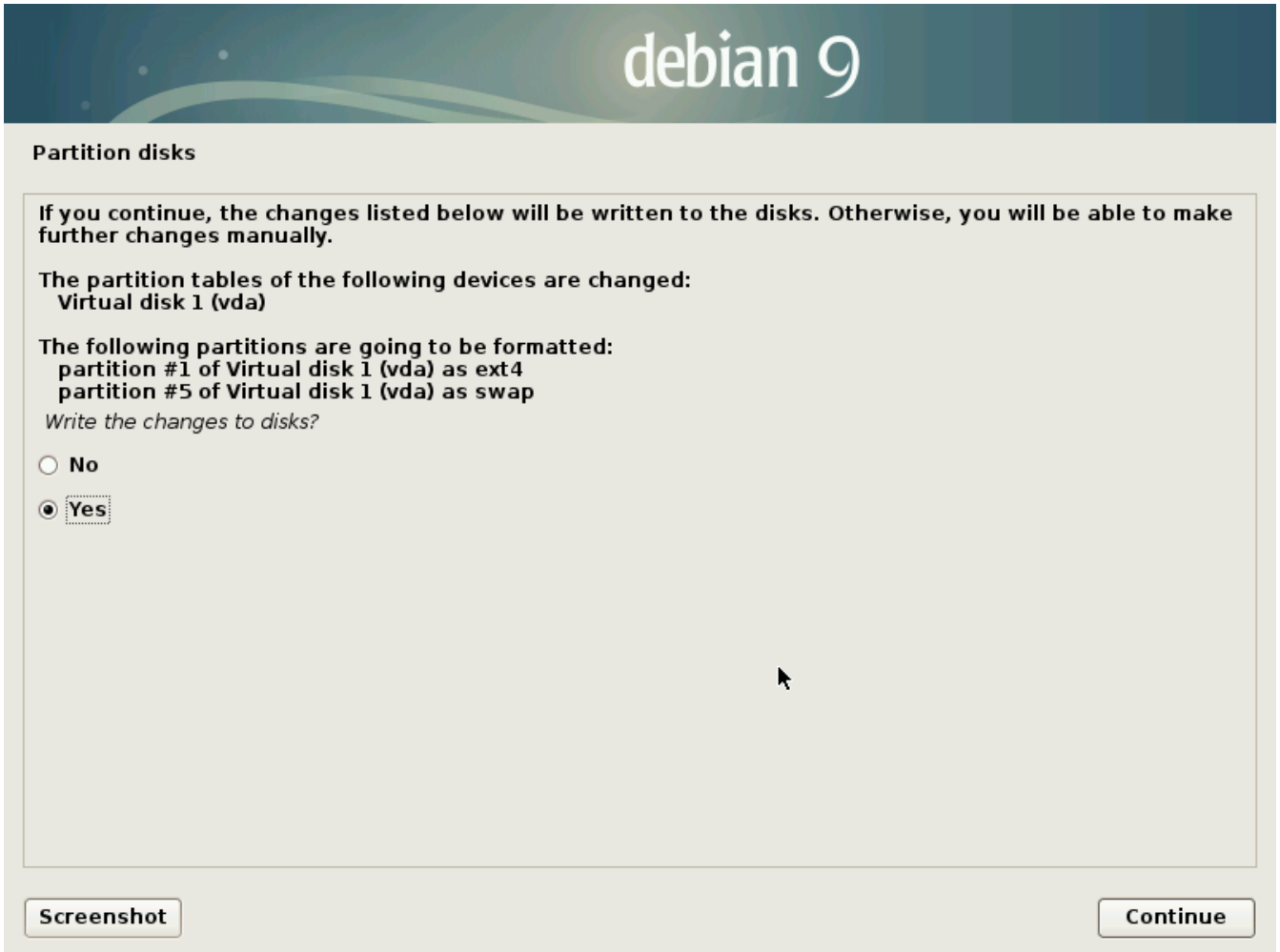
https://creativecommons.org/licenses/by-nd/4.0/

6. Einrichtung der Partitionen IV



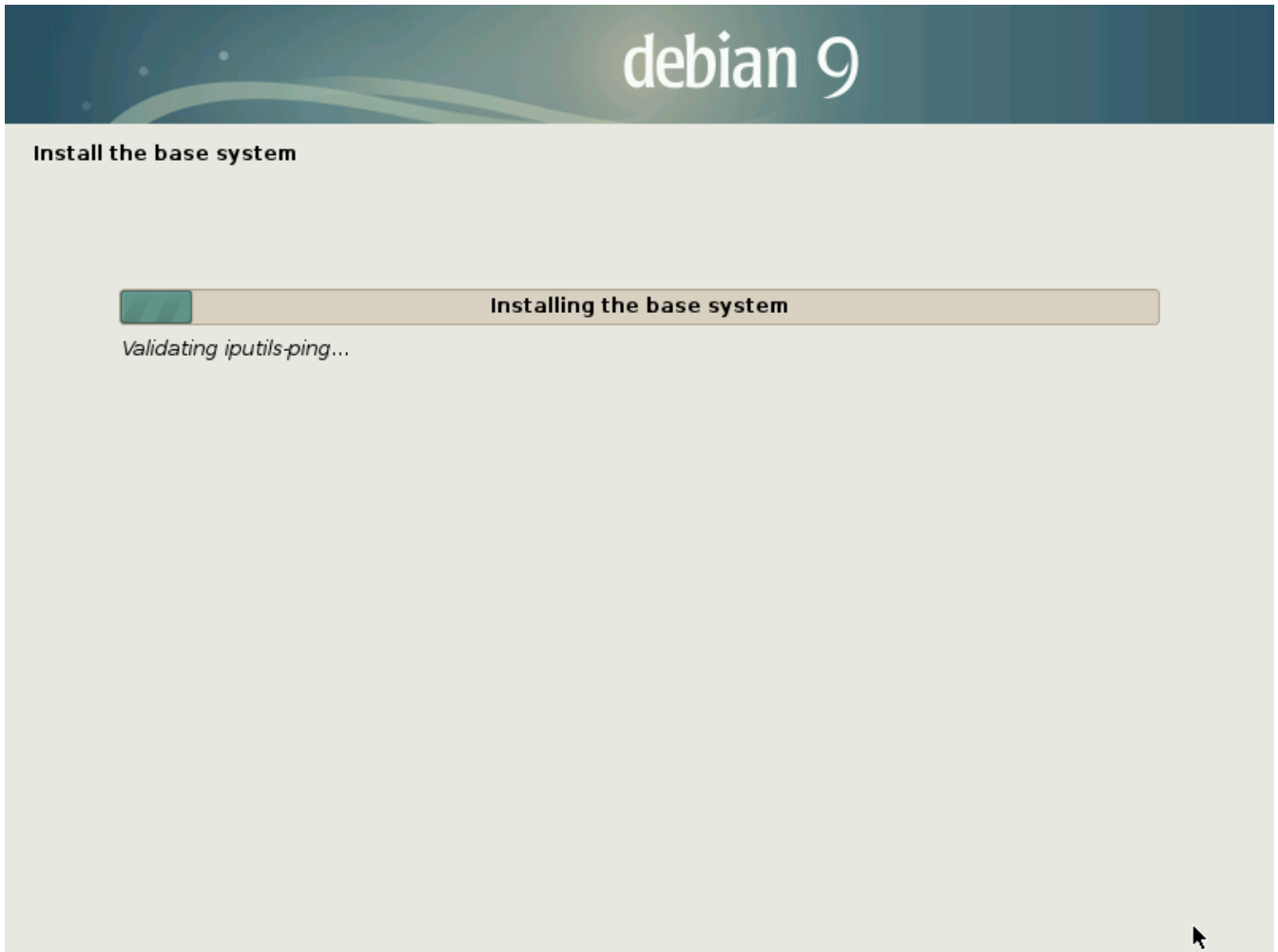
https://creativecommons.org/licenses/by-nd/4.0/

6. Einrichtung der Partitionen V



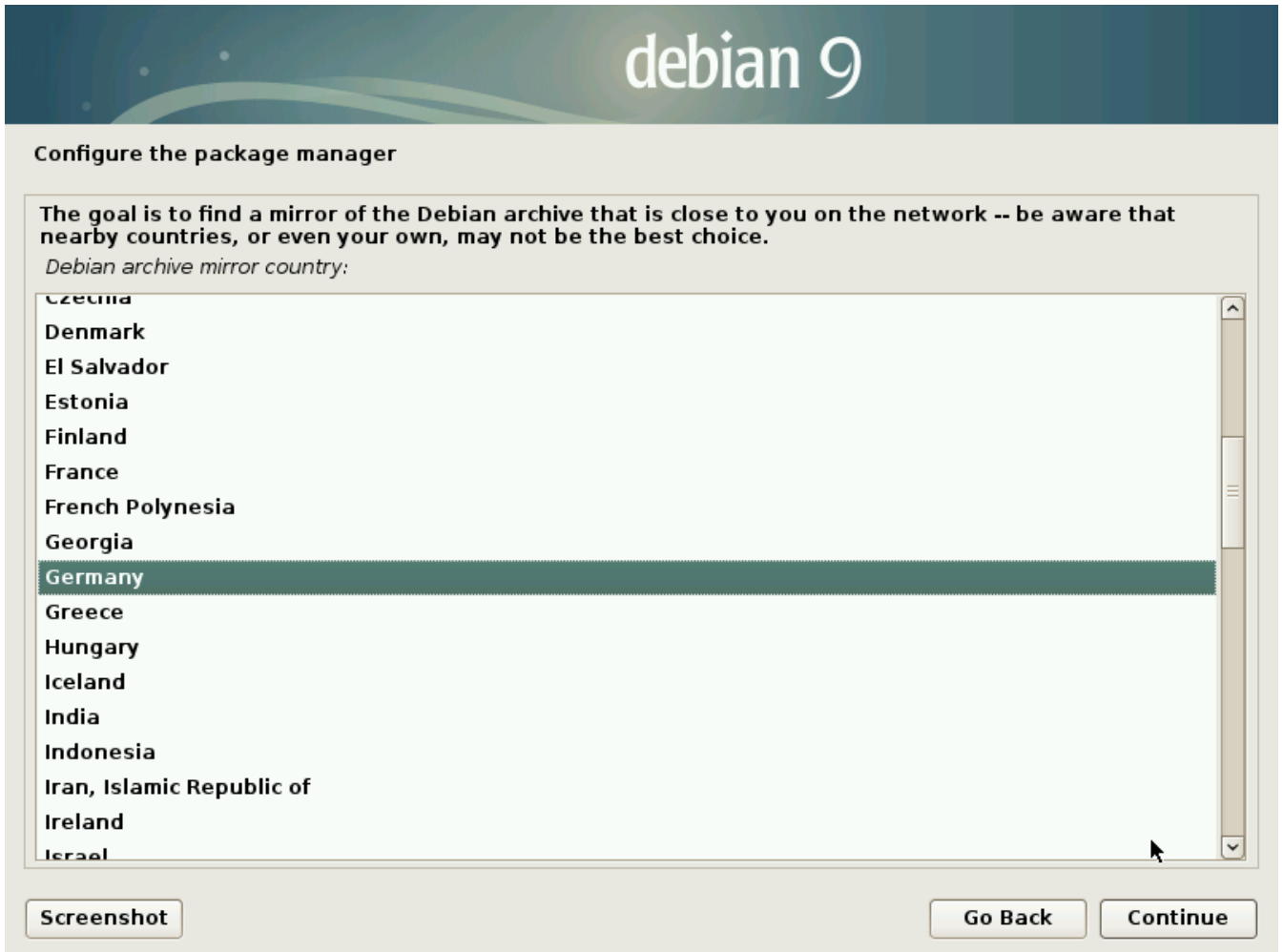
https://creativecommons.org/licenses/by-nd/4.0/

7. Installation des Basissystems



https://creativecommons.org/licenses/by-nd/4.0/

8. Auswahl des Paketquellen-Servers I



https://creativecommons.org/licenses/by-nd/4.0/

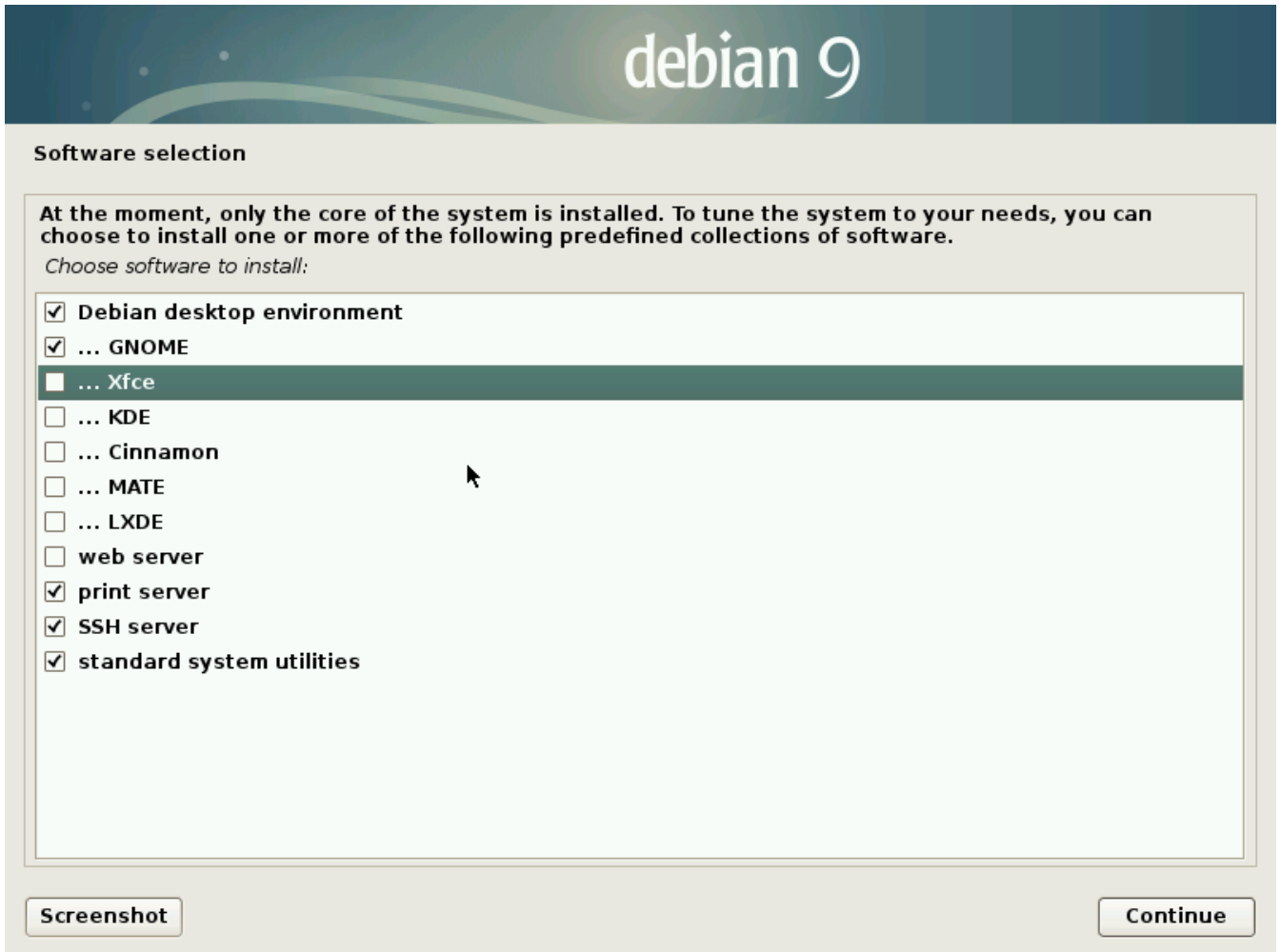
8. Auswahl des Paketquellen-Servers II

https://creativecommons.org/licenses/by-nd/4.0/

8. Auswahl des Paketquellen-Servers III

https://creativecommons.org/licenses/by-nd/4.0/

9. Auswahl zusätzlicher Pakete



https://creativecommons.org/licenses/by-nd/4.0/

10. Installation des Bootloaders I

https://creativecommons.org/licenses/by-nd/4.0/

10. Installation des Bootloaders II

https://creativecommons.org/licenses/by-nd/4.0/

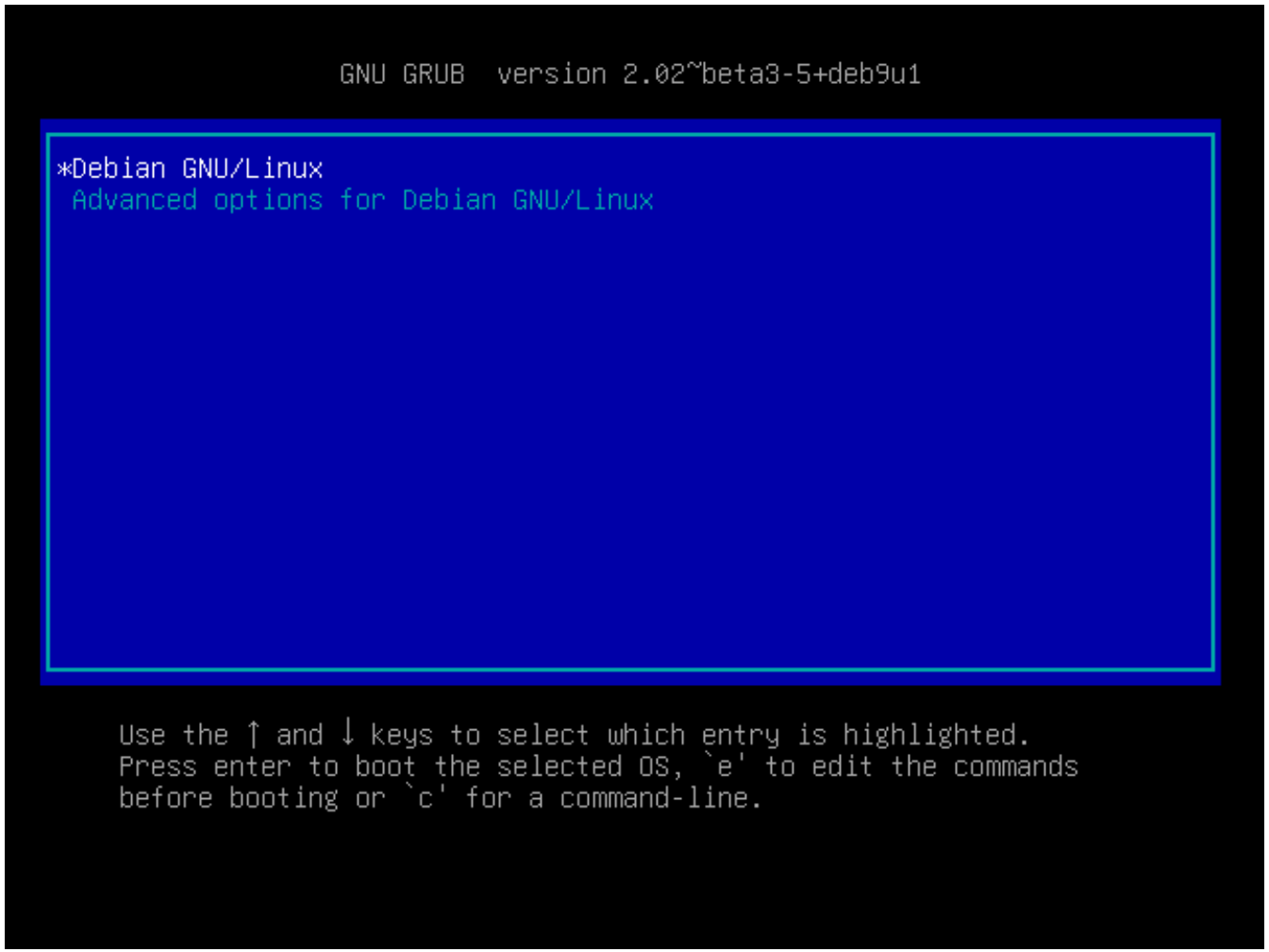
10. Installation des Bootloaders III

https://creativecommons.org/licenses/by-nd/4.0/

10. Installation des Bootloaders IV

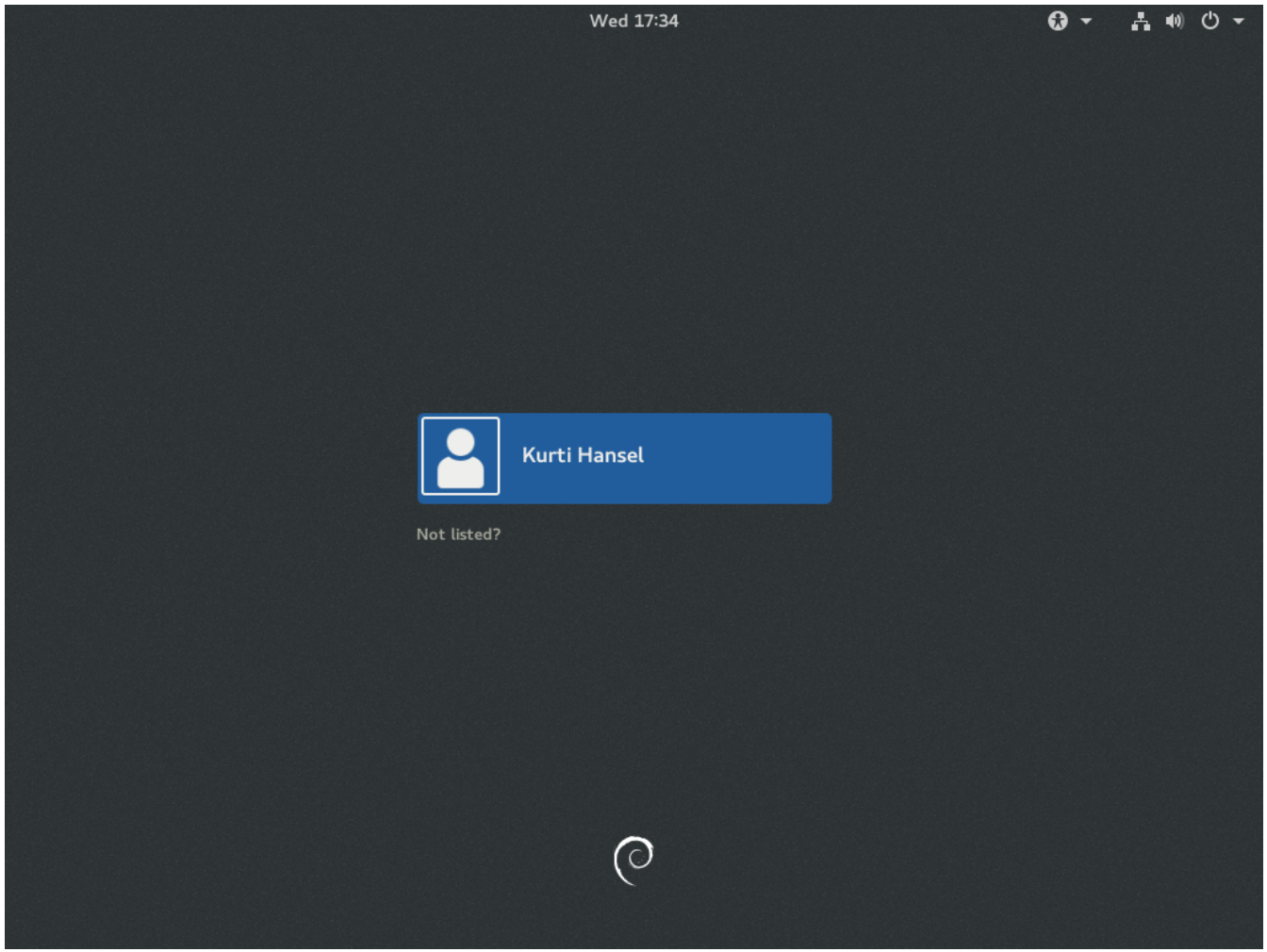
https://creativecommons.org/licenses/by-nd/4.0/

Erster Boot I



https://creativecommons.org/licenses/by-nd/4.0/

Erster Boot II



https://creativecommons.org/licenses/by-nd/4.0/

Arbeitsumgebungen

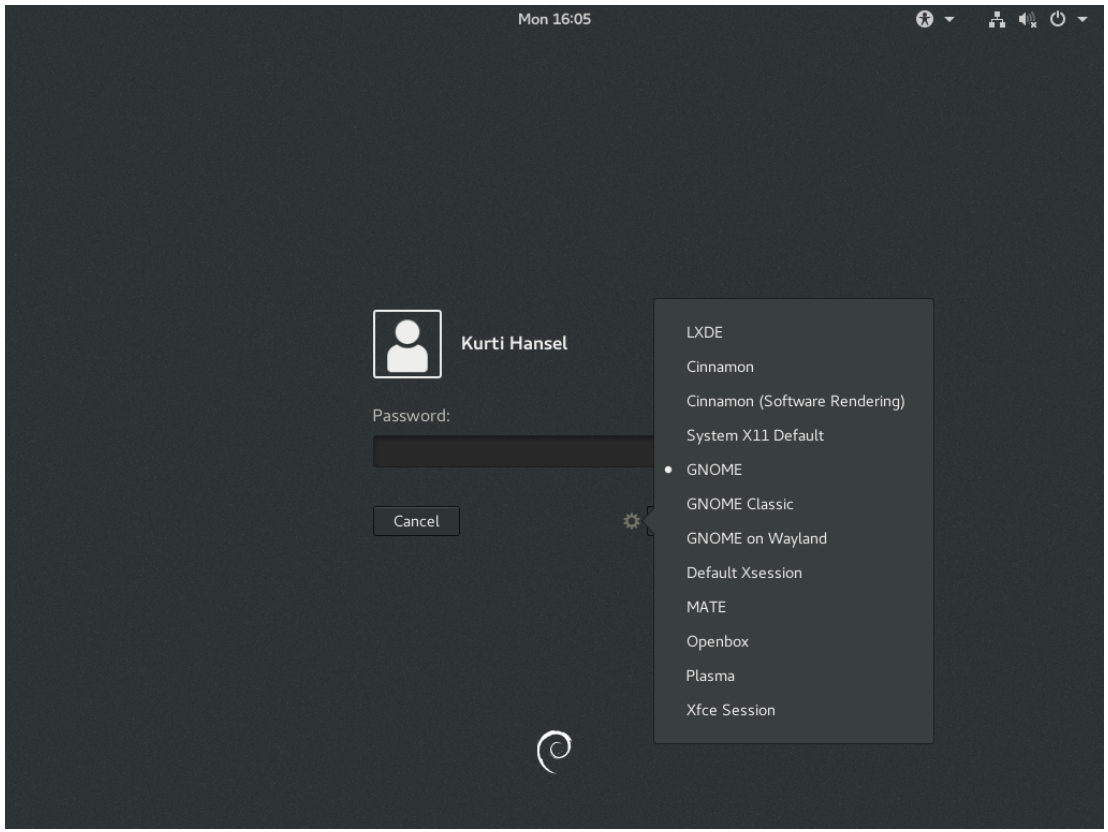
- Die "blanke" Konsole (TTY, **T**ele**TY**pewriter)
- GNOME
- KDE
- LXDE
- Cinnamon
- Mate

... und viele weitere

https://creativecommons.org/licenses/by-nd/4.0/

Auswahl der Arbeitsumgebung

- TTY: Strg + F1-12
- Graphisch: Über Optionen im Anmeldebildschirm



https://creativecommons.org/licenses/by-nd/4.0/

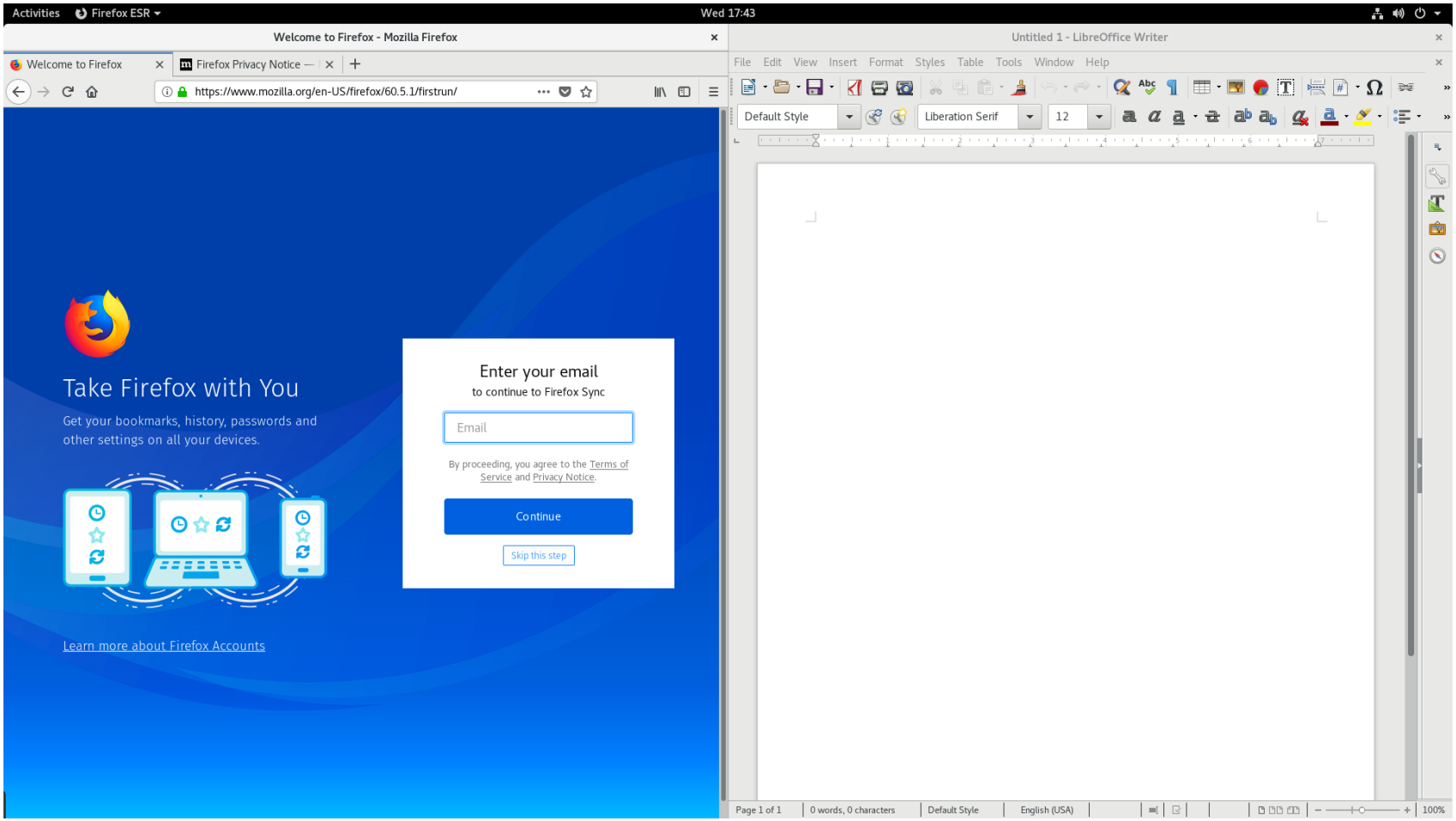
GNOME I

https://creativecommons.org/licenses/by-nd/4.0/

GNOME II

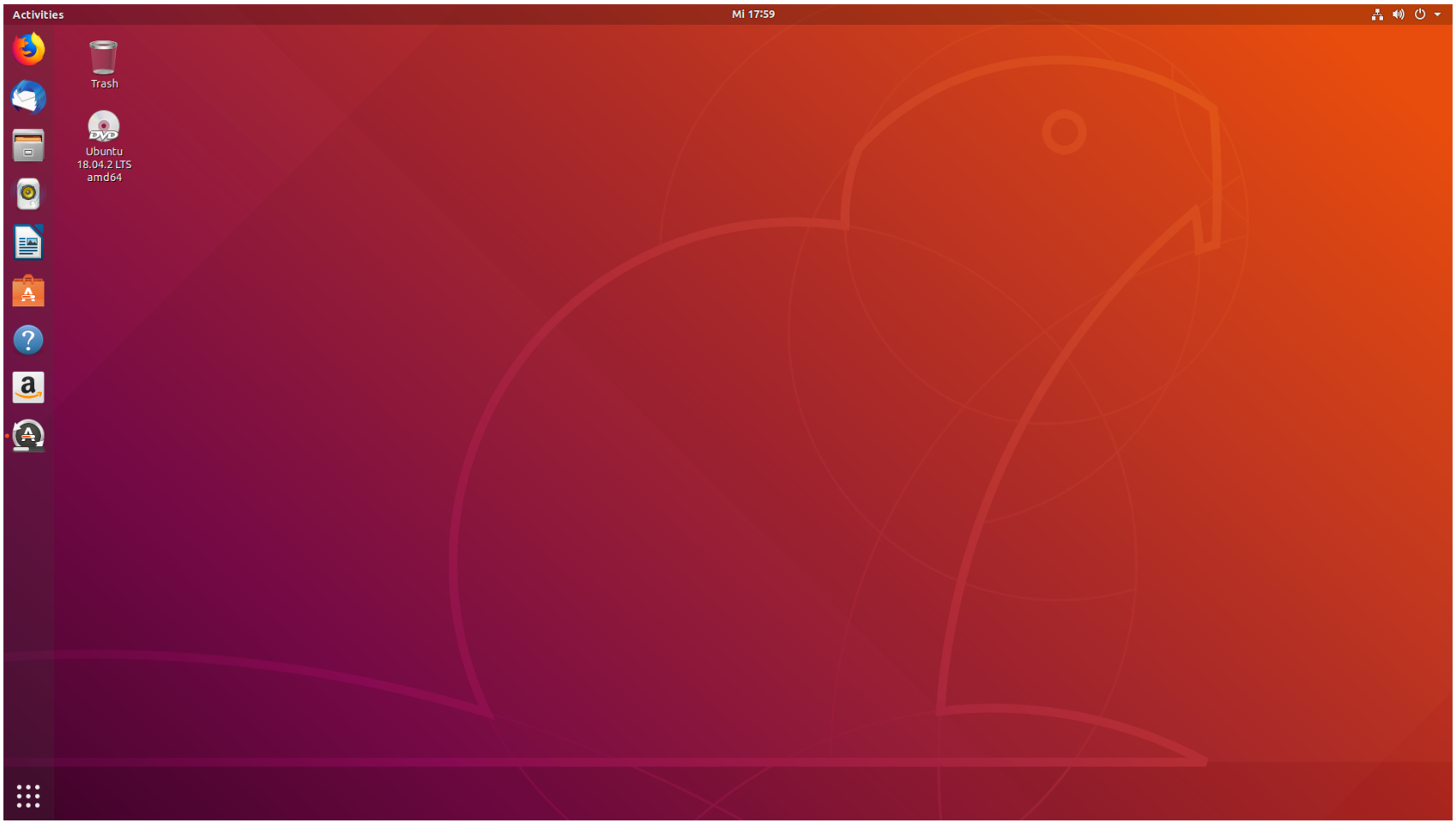
https://creativecommons.org/licenses/by-nd/4.0/

GNOME III



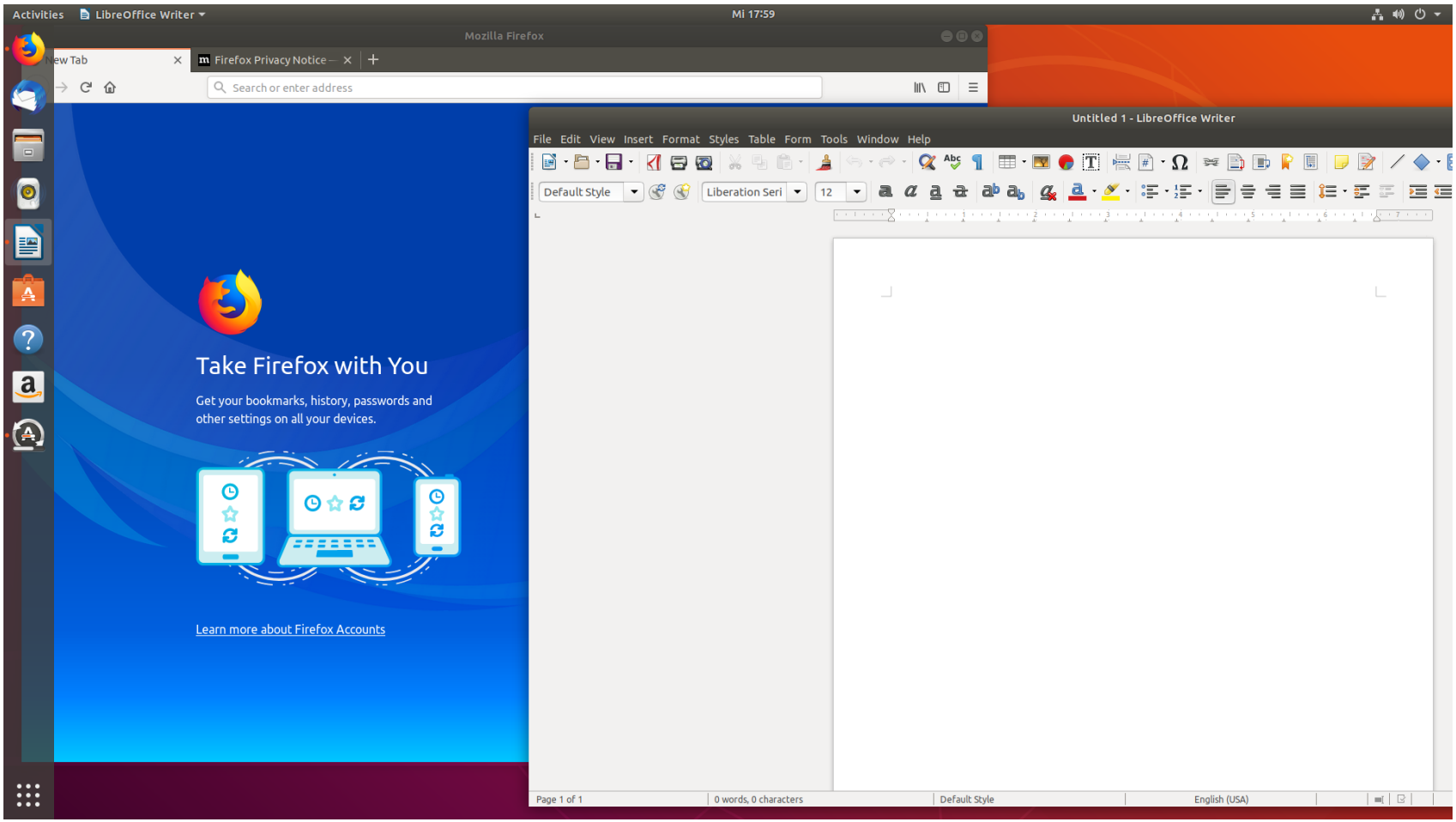
https://creativecommons.org/licenses/by-nd/4.0/

GNOME (Ubuntu) I



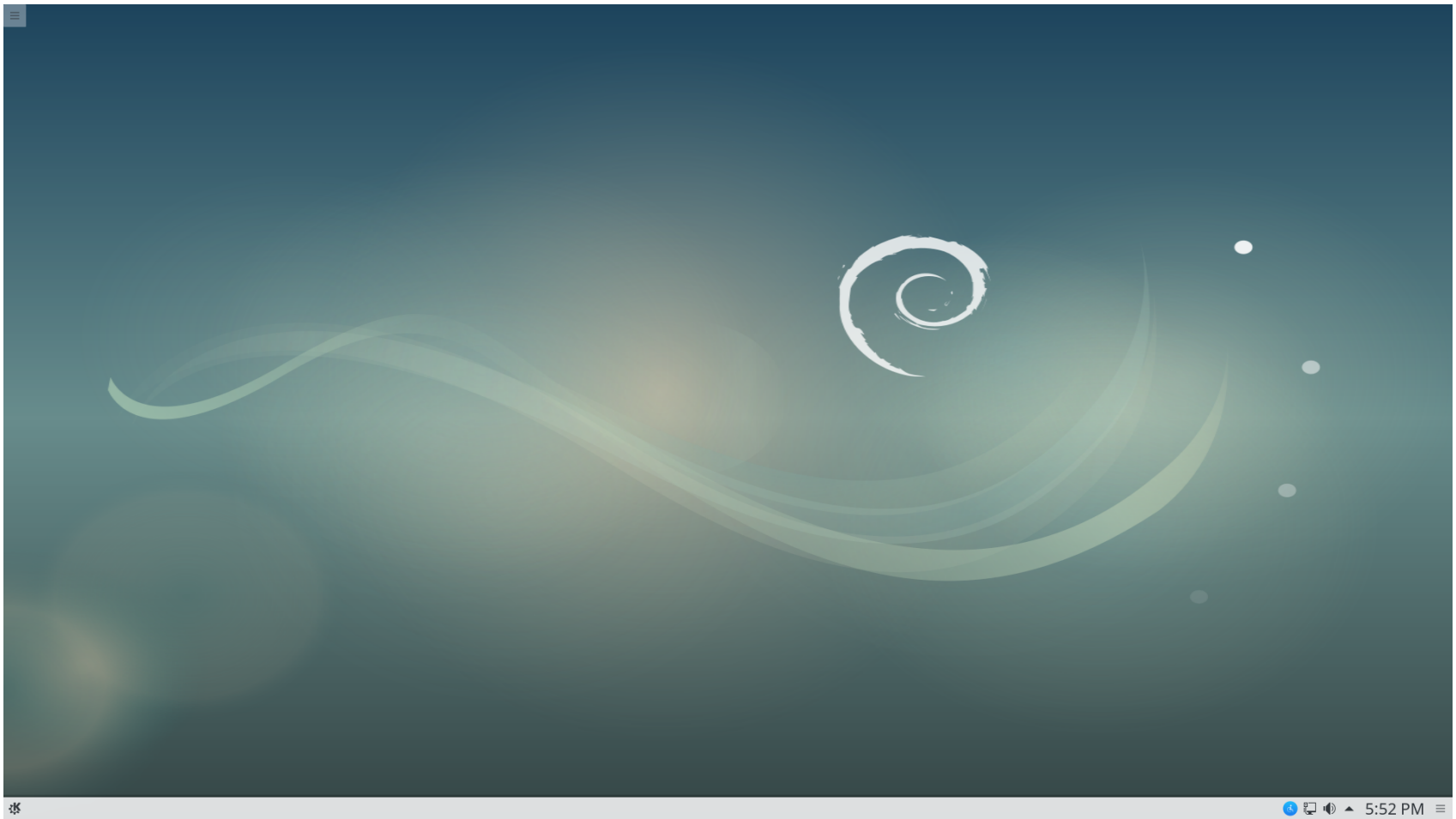
https://creativecommons.org/licenses/by-nd/4.0/

GNOME (Ubuntu) II



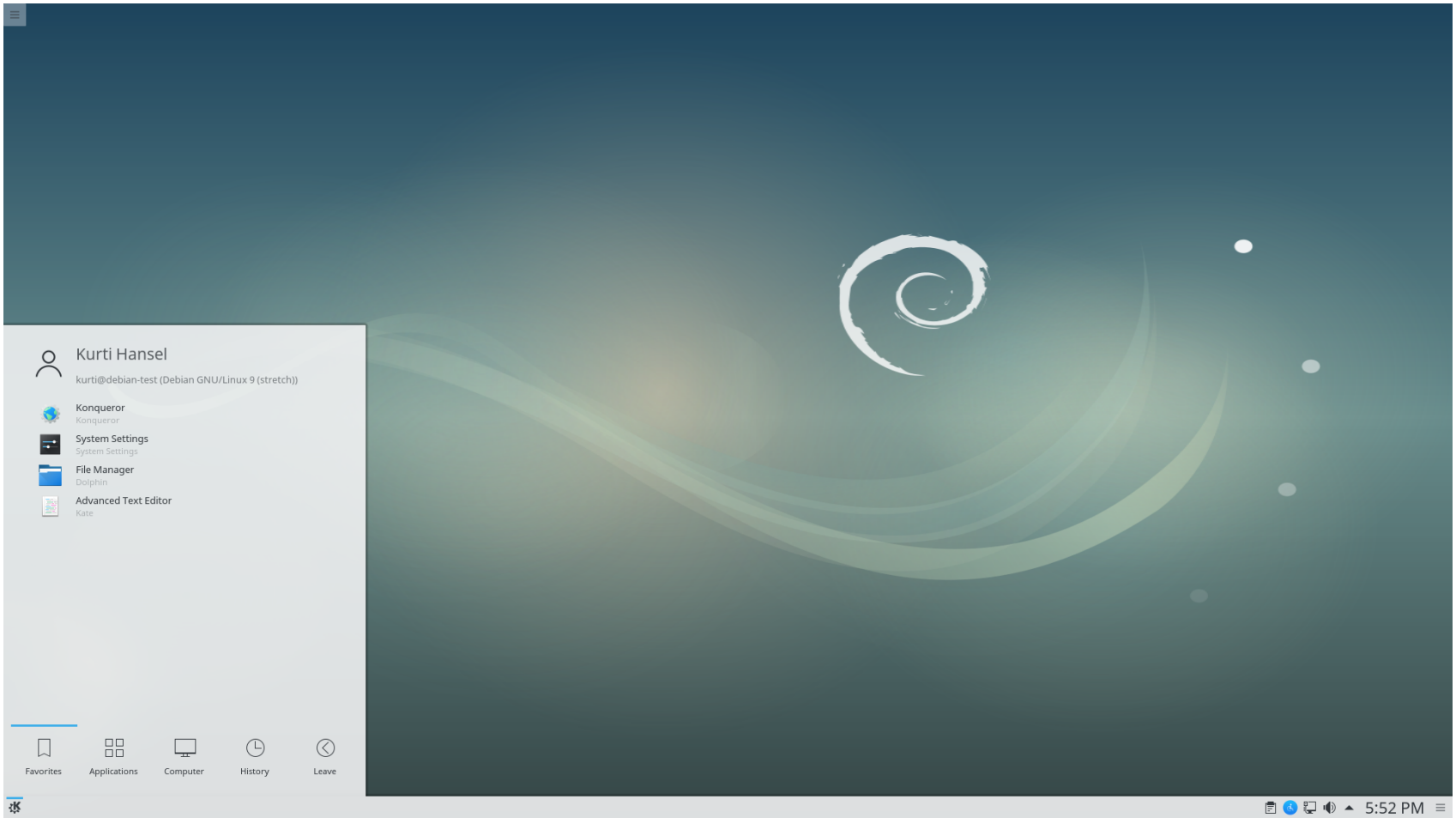
https://creativecommons.org/licenses/by-nd/4.0/

KDE Plasma I



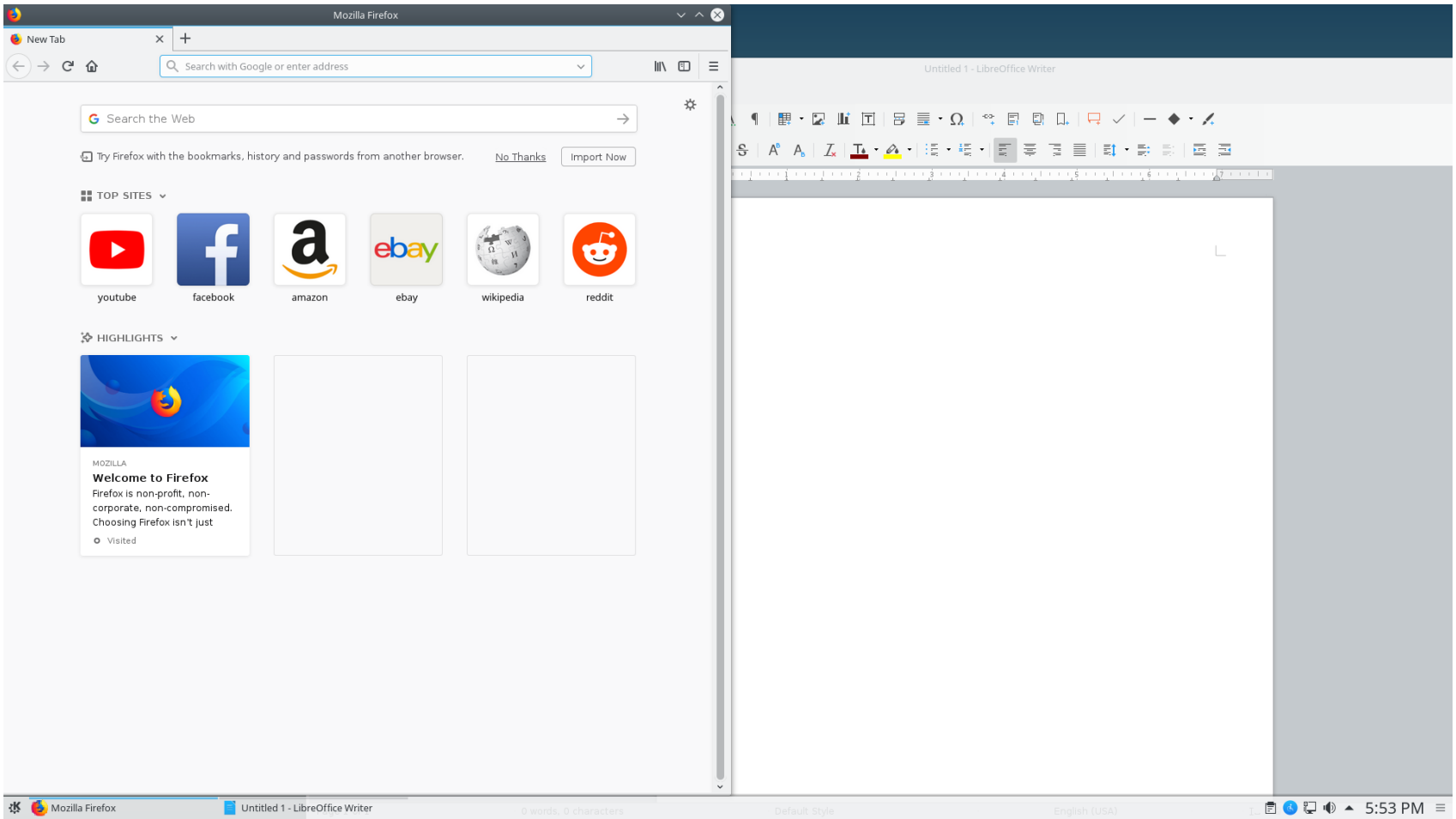
https://creativecommons.org/licenses/by-nd/4.0/

KDE Plasma II



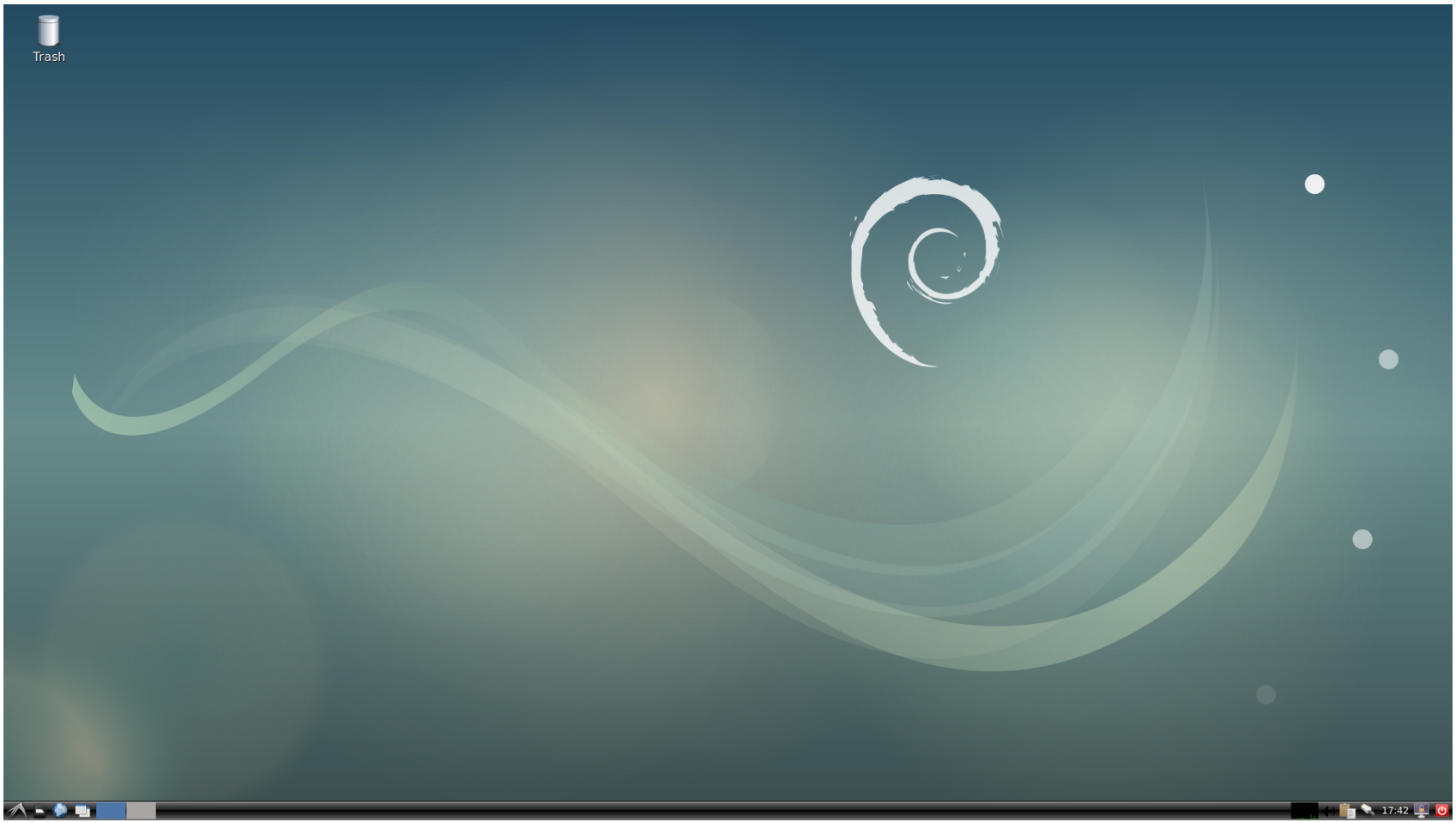
https://creativecommons.org/licenses/by-nd/4.0/

KDE Plasma III



https://creativecommons.org/licenses/by-nd/4.0/

LXDE



https://creativecommons.org/licenses/by-nd/4.0/

Dateisystem in Linux

- Linux organisiert Dateien und Ordner in *einem*, monolithischen Baum mit der Wurzel /
  (zum Vergleich: Windows/DOS organisiert Dateien in mehreren Bäumen, mit dem Speichergerät als Wurzel)
- Datenträger (bzw. Partitionen) werden in diesen Baum als Ordner "eingehängt" (`mount`)
- Hierarchieseperator: /
  (zum Vergleich in Windows: \)
- "Versteckte" Ordner/Dateien starten mit "."
  Z.B. `/home/test/.versteckt`

https://creativecommons.org/licenses/by-nd/4.0/

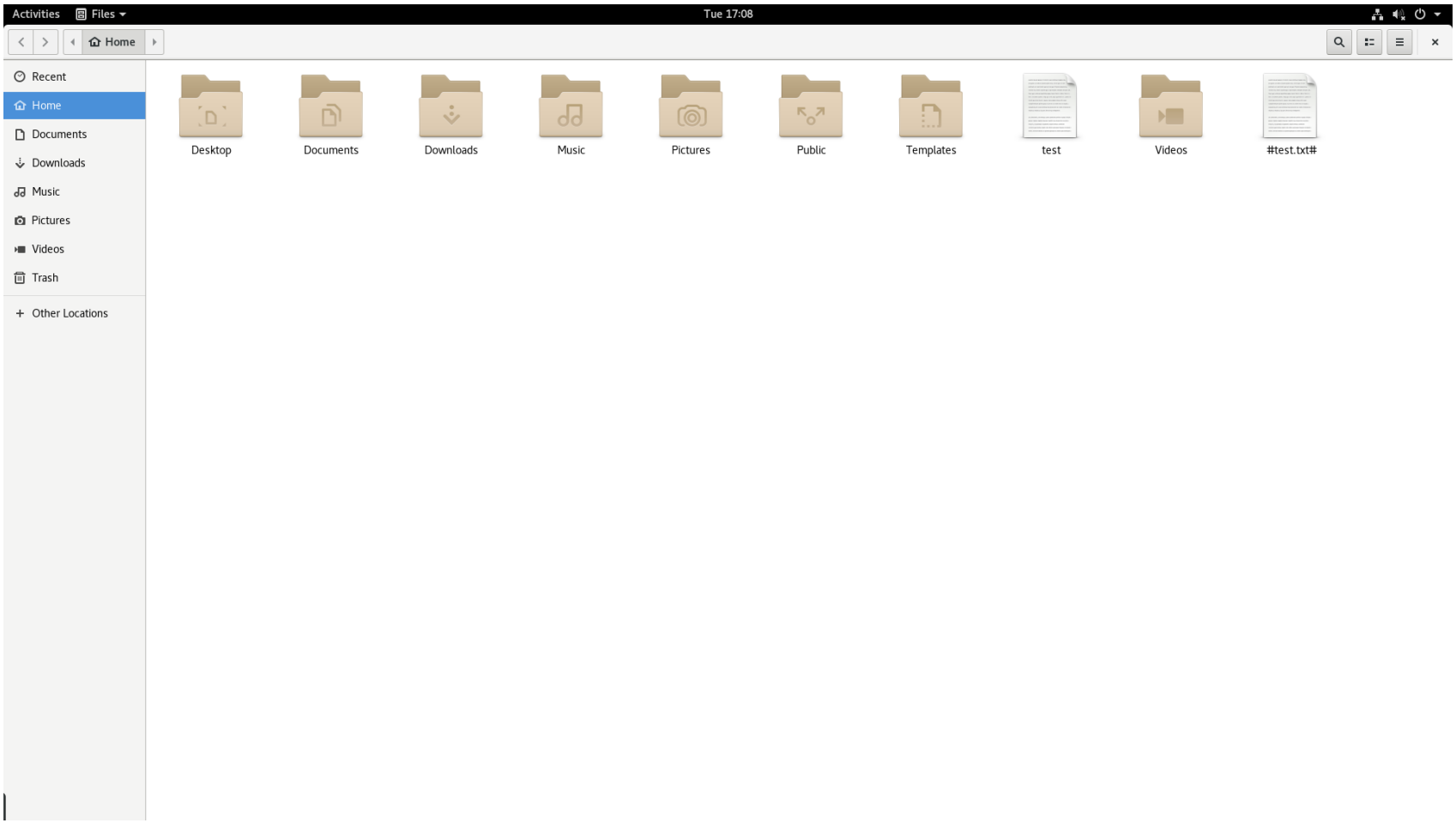
Konventionen – ein kurzer Überblick

Eine Auswahl der wichtigen, durch Konventionen festgelegte Verzeichnisfunktionen

- `/home/<username>` – Nutzerordner ("homes")
- `/home/<username>/.config` – Konfigurationsdateien von Programmen, lokal pro Nutzer (z.B. für Browser oder Mailprogramme)
- `/etc` – Konfigurationsdateien für globale Dienste
- `/boot` – Ablageort für den Betriebssystemkern (Kernel), Konfiguration für den Bootloader (z.B. grub)
- `/mnt` – Temporärer Mountpoint
- `/media` – Temporärer Mountpoint – je nach Distribution
- `/usr/bin, /usr/local/bin, /bin, /sbin, /opt` u.A. Ablageort für Anwendungen – je nach Distribution

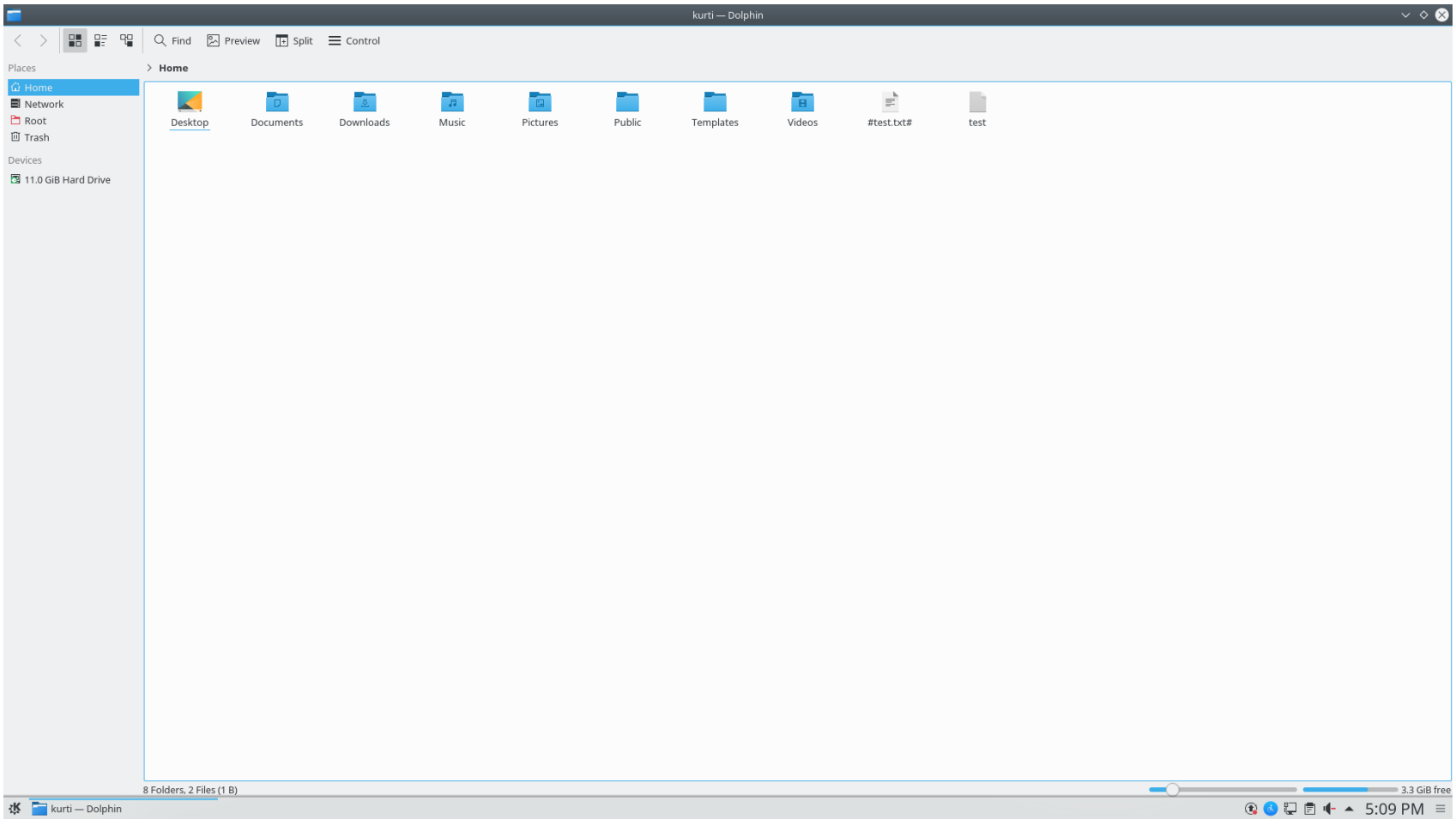
https://creativecommons.org/licenses/by-nd/4.0/

Dateiexplorer: GNOME



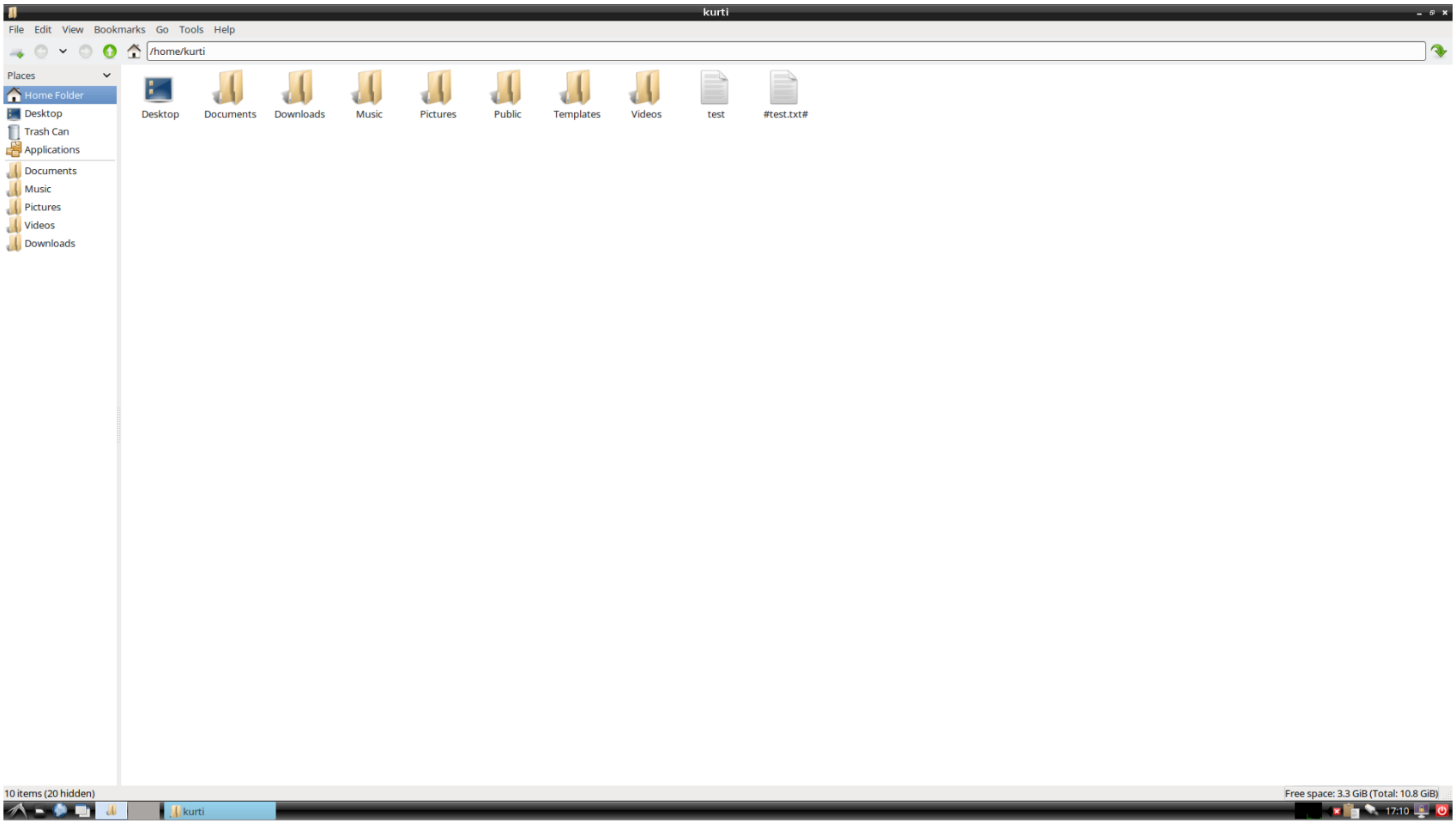
https://creativecommons.org/licenses/by-nd/4.0/

Dateiexplorer: KDE



https://creativecommons.org/licenses/by-nd/4.0/

Dateiexplorer: LXDE



https://creativecommons.org/licenses/by-nd/4.0/

Linux: Das Handwerkszeug für die tägliche Arbeit Teil II

Janis Streib

CC-BY-ND-4.0

09.-13.07.2018

https://creativecommons.org/licenses/by-nd/4.0/

Inhalt

- Shell
- Benutzer und Gruppen
  - ACLs
- Texteditoren
  - Graphische Texteditoren
  - Texteditoren in der Kommandozeile
- Paketverwaltung
- Noch mehr in der Kommandozeile
  - Nützliche Bash-Features
  - Nützliche Tools
  - Verifikation von Images
  - Umgebungsvariablen

https://creativecommons.org/licenses/by-nd/4.0/

Keine Angst vor der Konsole!

- Die Konsole in Linux ist ein mächtiges Tool!
- Benutzbar über einen Terminalemulator oder direkt über einer der TTYs
- Die Shell bietet die Arbeitsumgebung in der Konsole.
  Häufige Vertreter:
  - `bash`
  - `sh`
  - `zsh`

https://creativecommons.org/licenses/by-nd/4.0/

Bash – Eine Übersicht

- Benutzer
- Hostname
- Aktueller Pfad (Current Working Directory, CWD)
  Ausgehend von diesem Pfad werden die Befehle ausgeführt

https://creativecommons.org/licenses/by-nd/4.0/

Die wichtigsten Basisbefehle (Bash)

Befehl	Beschreibung
`cd <Pfad>`	Wechsel in Verzeichnis `<Pfad>`
`ls [<Pfad>]`	Zeige Inhalt des CWD bzw. des Ordners `<Pfad>`
`cp <Datei1> <Datei2>`	Kopiere `<Datei1>` nach `<Datei2>`
`mv <Datei1> <Datei2>`	Verschiebe `<Datei1>` zu `<Datei2>` (äquivalent zu Umbenennen)
`cat <Datei>`	Inhalt von `<Datei>` anzeigen
`echo <Text>`	Zeige `<Text>` an

https://creativecommons.org/licenses/by-nd/4.0/

Relative Verzeichnispfade

Abkürzung	Bedeutung
.	Aktuelles Verzeichnis
..	Übergeordnetes Verzeichnis
~	Home-Verzeichnis

https://creativecommons.org/licenses/by-nd/4.0/

Navigation in der Shell – Beispiele

- `testuer@test:/opt$ cd ~/Downloads`
- `/home/testuser/Downloads`
- `testuer@test:/opt$ cd /home/janis/../testuser`
- `/home/testuser`
- `testuer@test:/opt$ cd ~/Downloads/.././../`
- `/home`

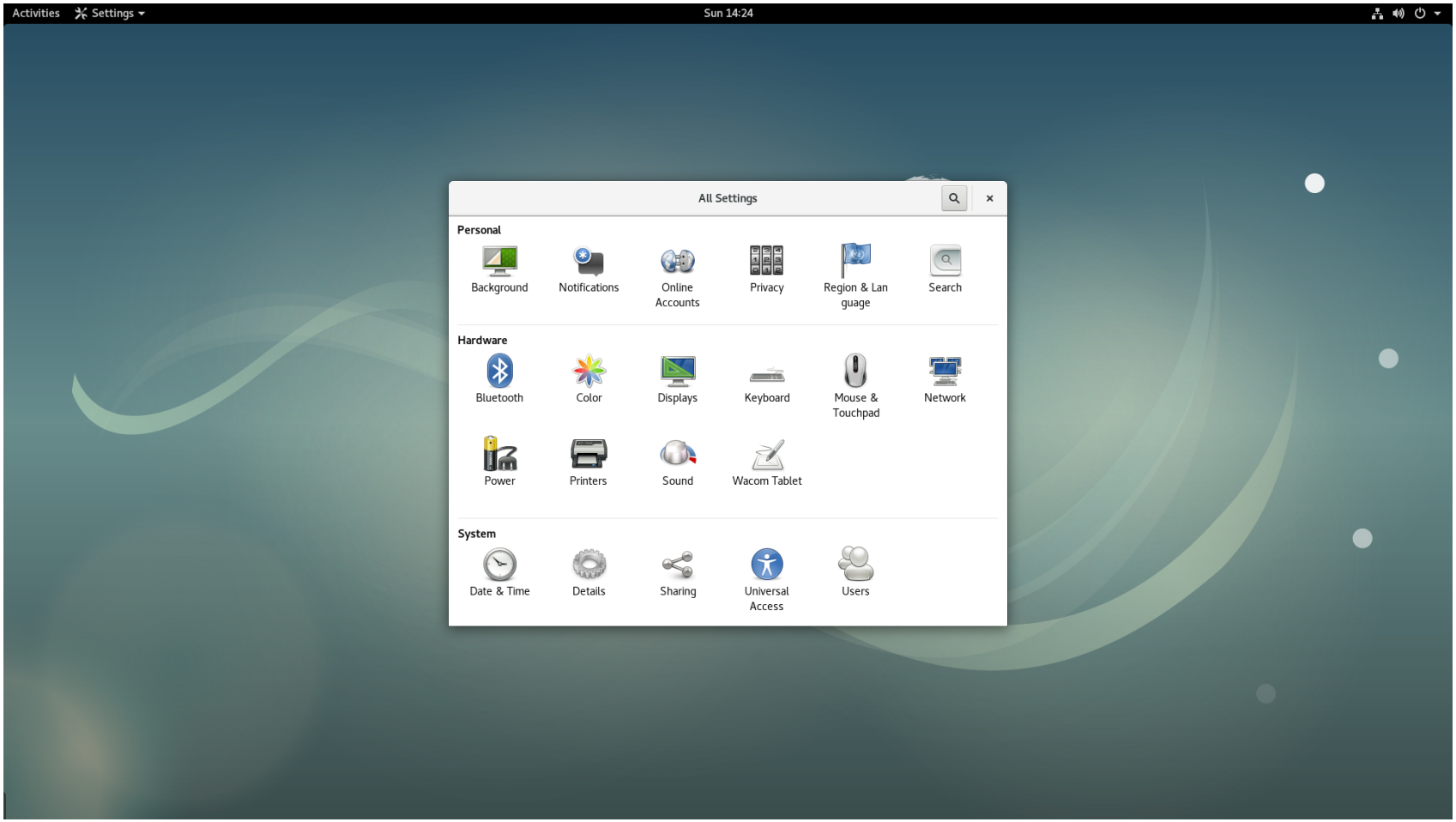
https://creativecommons.org/licenses/by-nd/4.0/

Hilfe!

- Basisbefehle der Shell können mithilfe des Kommandos `help` aufgelistet werden
- Viele Kommandos zeigen mittels des Parameters `-h` oder `--help` eine Liste der möglichen Parameter und Erklärungen zur grundsätzlichen Funktionsweise an
- `man <Command>` zeigt eine ausführliche Dokumentation zu fast allen Kommandos an
  - Volltextsuche (von Cursor aus nach unten) mittels "/"
  - "q" zum Beenden
- `https://unix.stackexchange.com/`, `https://serverfault.com/` und Google ;)

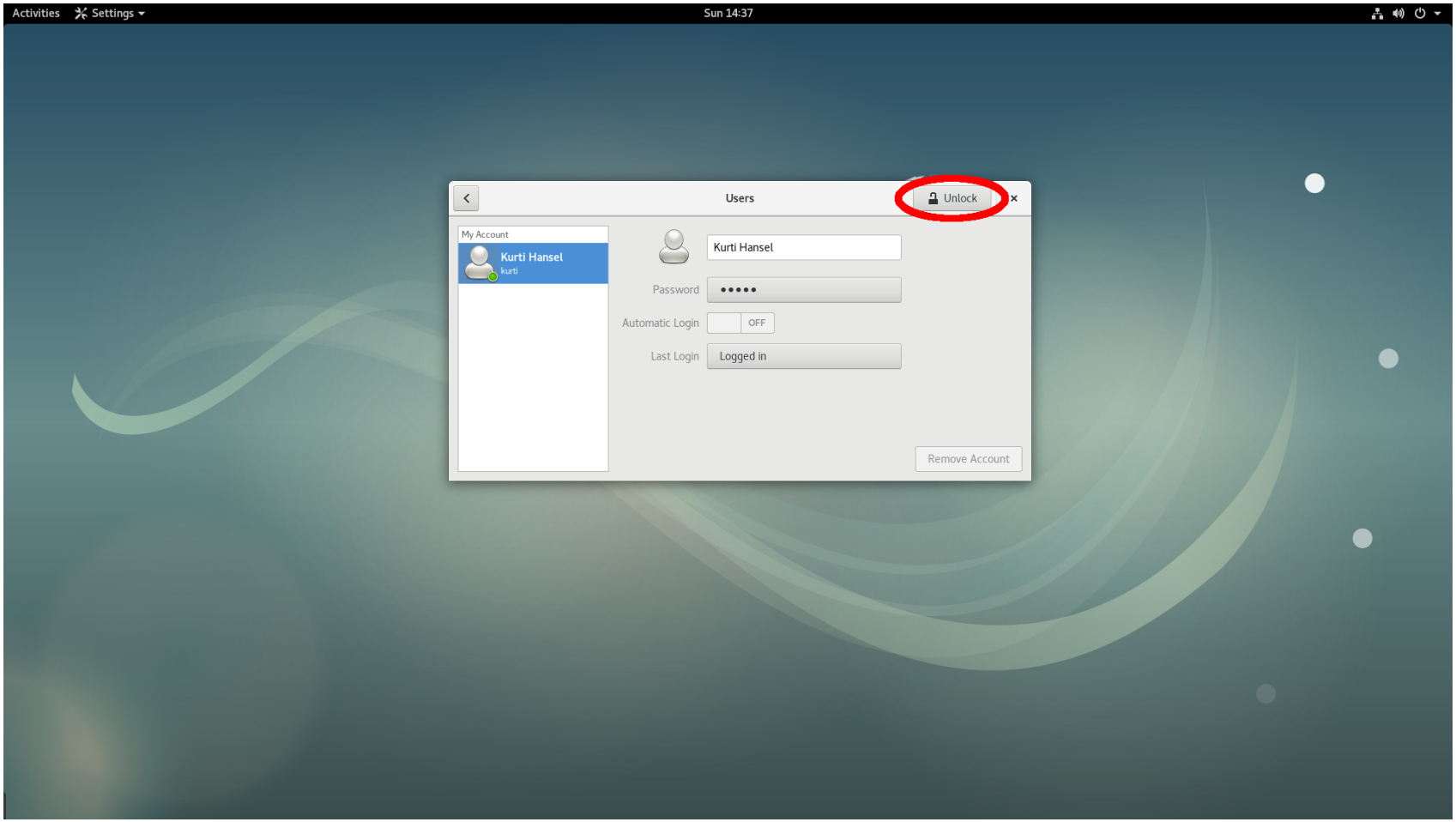
https://creativecommons.org/licenses/by-nd/4.0/

Graphische Nutzerverwaltung in GNOME I



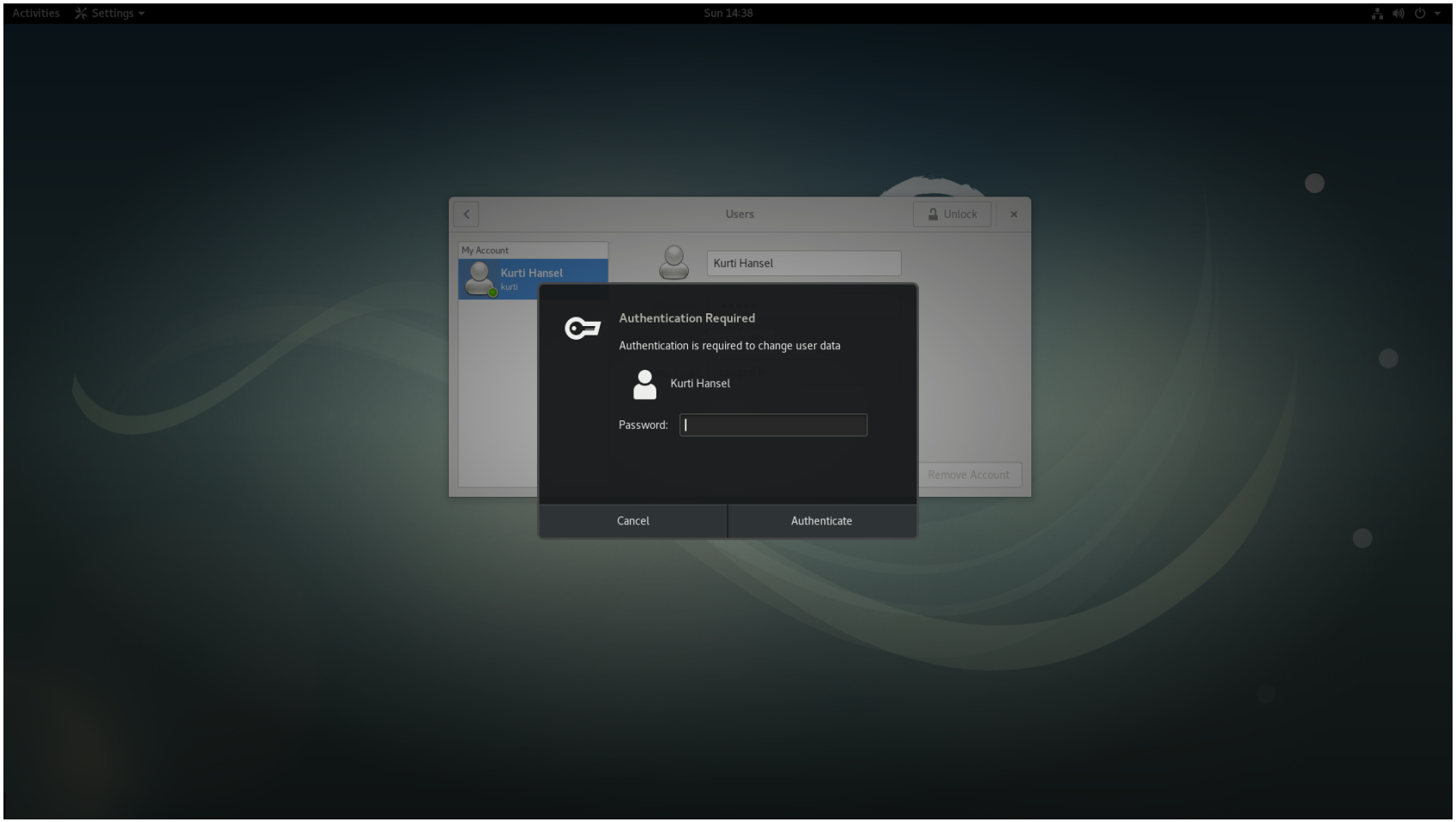
https://creativecommons.org/licenses/by-nd/4.0/

Graphische Nutzerverwaltung in GNOME II



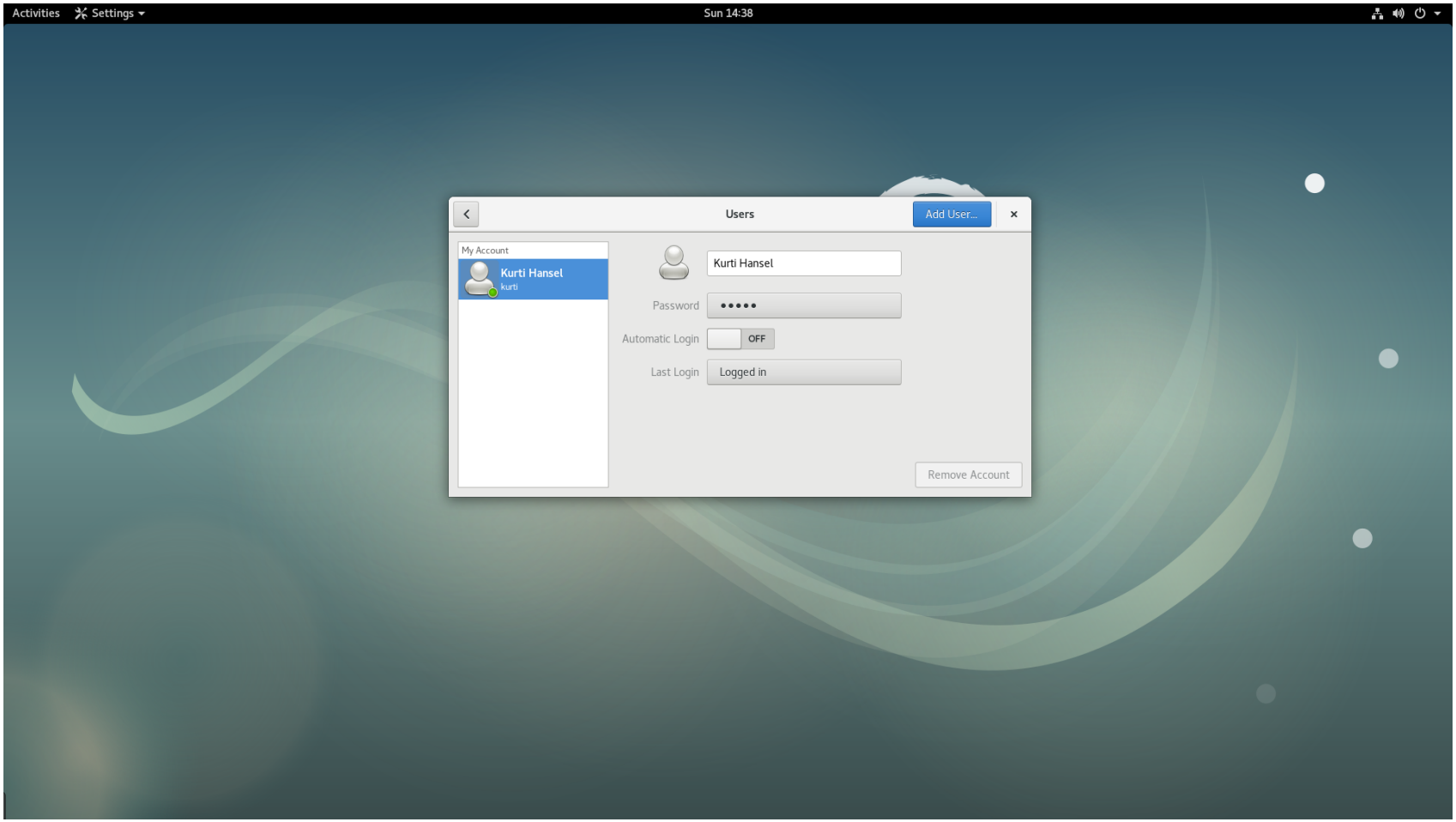
https://creativecommons.org/licenses/by-nd/4.0/

Graphische Nutzerverwaltung in GNOME III



https://creativecommons.org/licenses/by-nd/4.0/

Graphische Nutzerverwaltung in GNOME IV



https://creativecommons.org/licenses/by-nd/4.0/

Nutzerverwaltung in der Konsole I

- Nutzer hinzufügen: `adduser <Nutzername>`
- Nutzer einer Gruppe hinzufügen: `adduser <Nutzername> <Gruppenname>`
- Nutzer entfernen: `deluser <Nutzername>`
- Nutzer aus Gruppe entfernen: `deluser <Nutzername> <Gruppenname>`
- Passowrt des aktuellen, bzw. eines Nutzers ändern: `passwd` bzw. `passwd <nutzer>`

https://creativecommons.org/licenses/by-nd/4.0/

Nutzerverwaltung in der Konsole II

- Nutzerdefinitionen: `/etc/passwd`
- Nutzerpasswörter: `/etc/shadow`
- Gruppendefinitionen und Mitgliedschaften: `/etc/group`

https://creativecommons.org/licenses/by-nd/4.0/

su

`su <Nutzername>` erlaubt uns in der Konsole ein anderer Nutzer zu werden (Passwort des Nutzers `<Nutzername>` benötigt).

https://creativecommons.org/licenses/by-nd/4.0/

Administrative Rechte

- Manche Befehle (z.B. `adduser`) müssen vom Nutzer "`root`" ausgeführt werden

⇒ Root ist der "ultimative" Super-User mit allen Rechten

- 1. Möglichkeit: Wir kennen das Passwort des Nutzers `root`
  ⇒ `su`
- 2. Möglichkeit: Wir fügen alle administrative Nutzer der Gruppe "`sudo`" hinzu
  - Alle Nutzer in der Gruppe "`sudo`" dürfen mithilfe des Kommandos "`sudo <Kommando>`" – nach Eingabe des Nutzerpassworts – Kommandos als Nutzer "`root`" durchführen
  - Der Nutzer "`root`" darf ohne Passworteingabe "`su`" ausführen

  ⇒ Durch "`sudo su`" kann jeder administrative Nutzer "`root`" werden

https://creativecommons.org/licenses/by-nd/4.0/

Benutzer- und Gruppenrechte I

- Dateien und Ordner gehören einem Nutzer und einer Gruppe

```
kurti@debian-test:~$ ls -al
total 100
drwxr-xr-x 19 kurti kurti 4096 Mar 23 20:33 .
drwxr-xr-x  3 root  root  4096 Mar 20 17:20 ..
-rw-------  1 kurti kurti   82 Mar 20 17:51 .bash_history
-rw-r--r--  1 kurti kurti  220 Mar 20 17:20 .bash_logout
-rw-r--r--  1 kurti kurti 3526 Mar 20 17:20 .bashrc
drwx------ 17 kurti kurti 4096 Mar 23 20:36 .cache
drwxr-xr-x  3 kurti kurti 4096 Mar 20 17:49 .cinnamon
drwx------ 28 kurti kurti 4096 Mar 23 20:33 .config
drwxr-xr-x  2 kurti kurti 4096 Mar 20 17:35 Desktop
drwxr-xr-x  2 kurti kurti 4096 Mar 20 17:35 Documents
```

Figure: `ls -al ~`

https://creativecommons.org/licenses/by-nd/4.0/

Benutzer- und Gruppenrechte II

- Für jede Datei / jeden Ordner kann festgelegt werden, ob gelesen, geschrieben und ausgeführt werden kann
- Bei Ordnern bedeutet "Ausführen" "Inhalt anzeigen" (`stat`)
- Eigentümer dürfen Zugriffsflags ändern

https://creativecommons.org/licenses/by-nd/4.0/

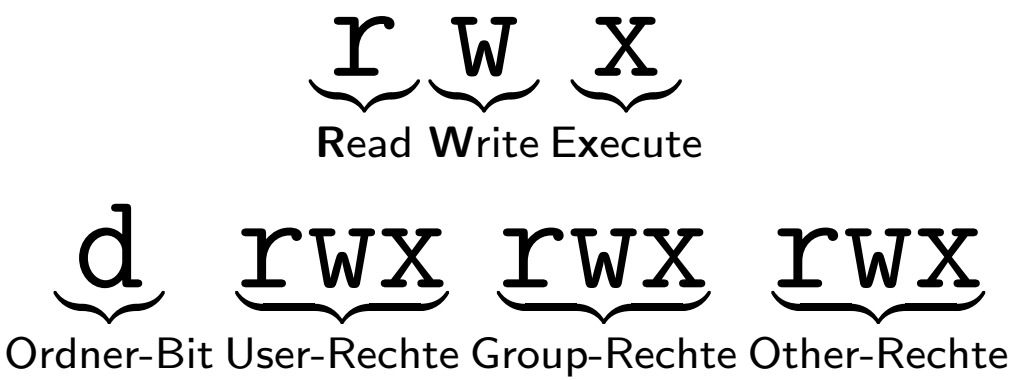
Benutzer- und Gruppenrechte III

```
kurti@debian-test:~$ ls -al
total 100
drwxr-xr-x 19 kurti kurti 4096 Mar 23 20:33 .
drwxr-xr-x  3 root  root  4096 Mar 20 17:20 ..
-rw-------  1 kurti kurti   82 Mar 20 17:51 .bash_history
-rw-r--r--  1 kurti kurti  220 Mar 20 17:20 .bash_logout
-rw-r--r--  1 kurti kurti 3526 Mar 20 17:20 .bashrc
drwx------ 17 kurti kurti 4096 Mar 23 20:36 .cache
drwxr-xr-x  3 kurti kurti 4096 Mar 20 17:49 .cinnamon
drwx------ 28 kurti kurti 4096 Mar 23 20:33 .config
drwxr-xr-x  2 kurti kurti 4096 Mar 20 17:35 Desktop
drwxr-xr-x  2 kurti kurti 4096 Mar 20 17:35 Documents
```

Figure: `ls -al ~`

https://creativecommons.org/licenses/by-nd/4.0/

Zugriffsflags



https://creativecommons.org/licenses/by-nd/4.0/

Spezial-Flags

`t`	Sticky-Bit. Ist dieses Bit gesetzt, so dürfen nur der Eigentümer und `root` die Datei/den Ordner verschieben oder löschen, ungeachtet des "`w`"-Flags
`s`	setuid-Bit. Ist dieses Bit gesetzt, so werden Dateien zusätzlich mit den Rechten des Eigentümers der Datei ausgeführt.

https://creativecommons.org/licenses/by-nd/4.0/

Eigentümer setzten

Befehl	Beschreibung
`chown <Nutzer> <Pfad>`	Setze den Eigentümer `<Nutzer>` für die Datei/ den Ordner `<Pfad>`
`chgrp <Gruppe> <Pfad>`	Setze die Eigentümergruppe `<Gruppe>` für die Datei/ den Ordner `<Pfad>`
`chown <Nutzer>:<Gruppe> <Pfad>`	Setze den Eigentümer `<Nutzer>` und die Eigentümergruppe `<Gruppe>` für die Datei/ den Ordner `<Pfad>`

https://creativecommons.org/licenses/by-nd/4.0/

Zugriffsflags setzten

`chmod <Flags als Bitmaske> <Pfad>`

Setze flags als dezimale Bitmaske für die Datei/ den Ordner `<Pfad>`.

`rwxrwxrwx` ⇒ 777
`rwxr-x---` ⇒ 750

`chmod <Liste der zu setzenden/ zu löschenden Bits> <Pfad>`

Setze flags als Bitliste für die Datei/ den Ordner `<Pfad>`.

`rwxrwxrwx` ⇒ `u+rwx,g+rwx,o+rwx`
`rwxr-x---` ⇒ `u+rwx,g+rx,g-w,o-rwx`

https://creativecommons.org/licenses/by-nd/4.0/

Access Control Lists (ACLs)

Möchte man weiteren Nutzern/ Gruppen spezielle Rechte auf eine Datei/ einen Ordner erteilen reichen die Standard Linux-Rechte nicht aus.
⇒ ACLs.
Konsolen-Tools:

- `getfacl <Datei> <Datei2> ...` – Anzeige der ACLs von Dateien/Ordnern
- `setfacl` – Setzten von ACLs

Hinweis

ACLs müssen vom Dateisystem unterstützt und aktiviert sein.
In Debian ist dies im Standardsetup gegeben. Das kann sich von Distribution zu Distribution unterscheiden.

https://creativecommons.org/licenses/by-nd/4.0/

ACL Masken

Einträge in die ACL werden mittels folgender Maskenstruktur dargestellt:

$$\underbrace{\texttt{u}}_{\text{User}} : \underbrace{\texttt{test}}_{\text{Nutzername}} : \underbrace{\texttt{-}}_{\text{Rechte (keine)}}$$

Analog dazu für Gruppen:

$$\underbrace{\texttt{g}}_{\text{Group}} : \underbrace{\texttt{testgroup}}_{\text{Gruppenname}} : \underbrace{\texttt{-}}_{\text{Rechte (keine)}}$$

Terme können durch Kommata zu einem zusammengefasst werden:

```
u:test:-,g:testgroup:-
```

Diese Regel entzieht dem Nutzer "test" und der Gruppe "testgroup" alle Rechte.

https://creativecommons.org/licenses/by-nd/4.0/

ACLs setzten

```
setfacl -m <maske> <Datei>
```


https://creativecommons.org/licenses/by-nd/4.0/

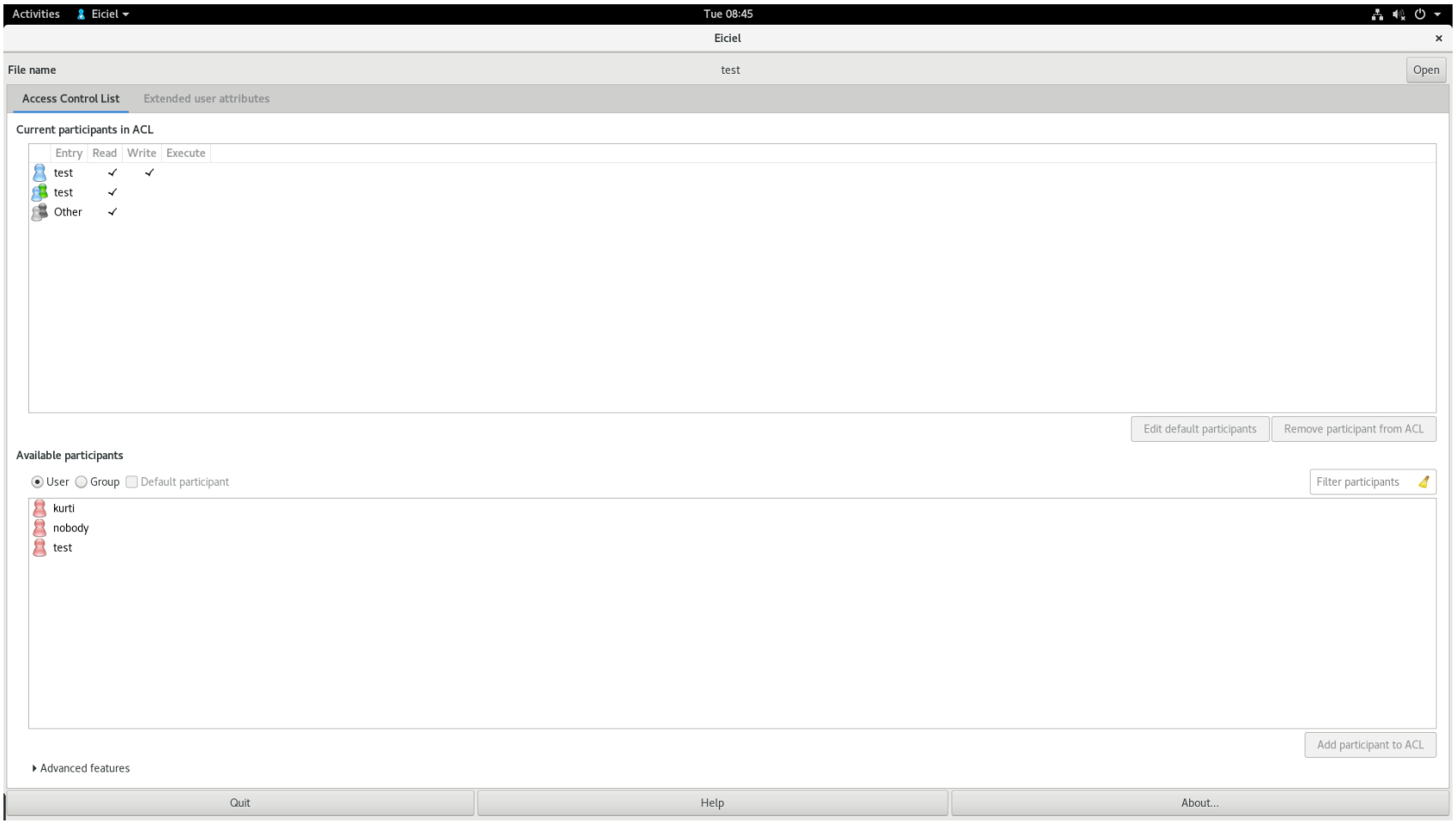
Auswertung der ACLs

- Die AC-Liste wird von oben nach Unten ausgewertet. Die erste Regel, die einen Zugriff erlaubt, wird genommen.
- Die `mask` legt fest, welche Rechte Nutzer und Gruppen maximal haben dürfen, die *nicht* Eigentümer sind

```
$ getfacl test
# file: test
# owner: janis
# group: janis
user::rw-
group::r--
group:audio:rwx
mask::rwx
other::r--
```


https://creativecommons.org/licenses/by-nd/4.0/

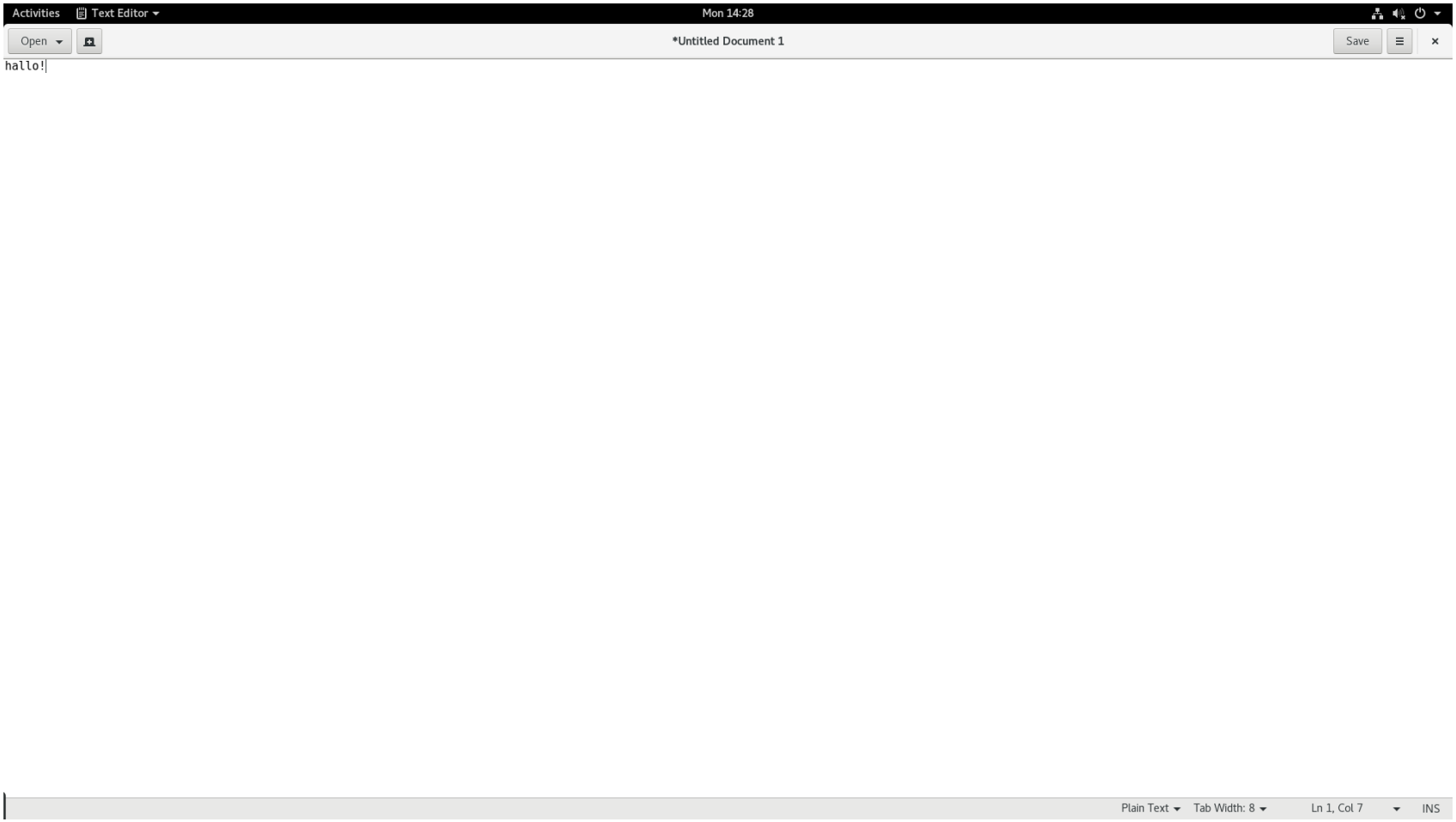
Graphische ACL-Verwaltung mit `eiciel`



https://creativecommons.org/licenses/by-nd/4.0/

Graphische Texteditoren

gedit



https://creativecommons.org/licenses/by-nd/4.0/

nano

https://creativecommons.org/licenses/by-nd/4.0/

emacs I

https://creativecommons.org/licenses/by-nd/4.0/

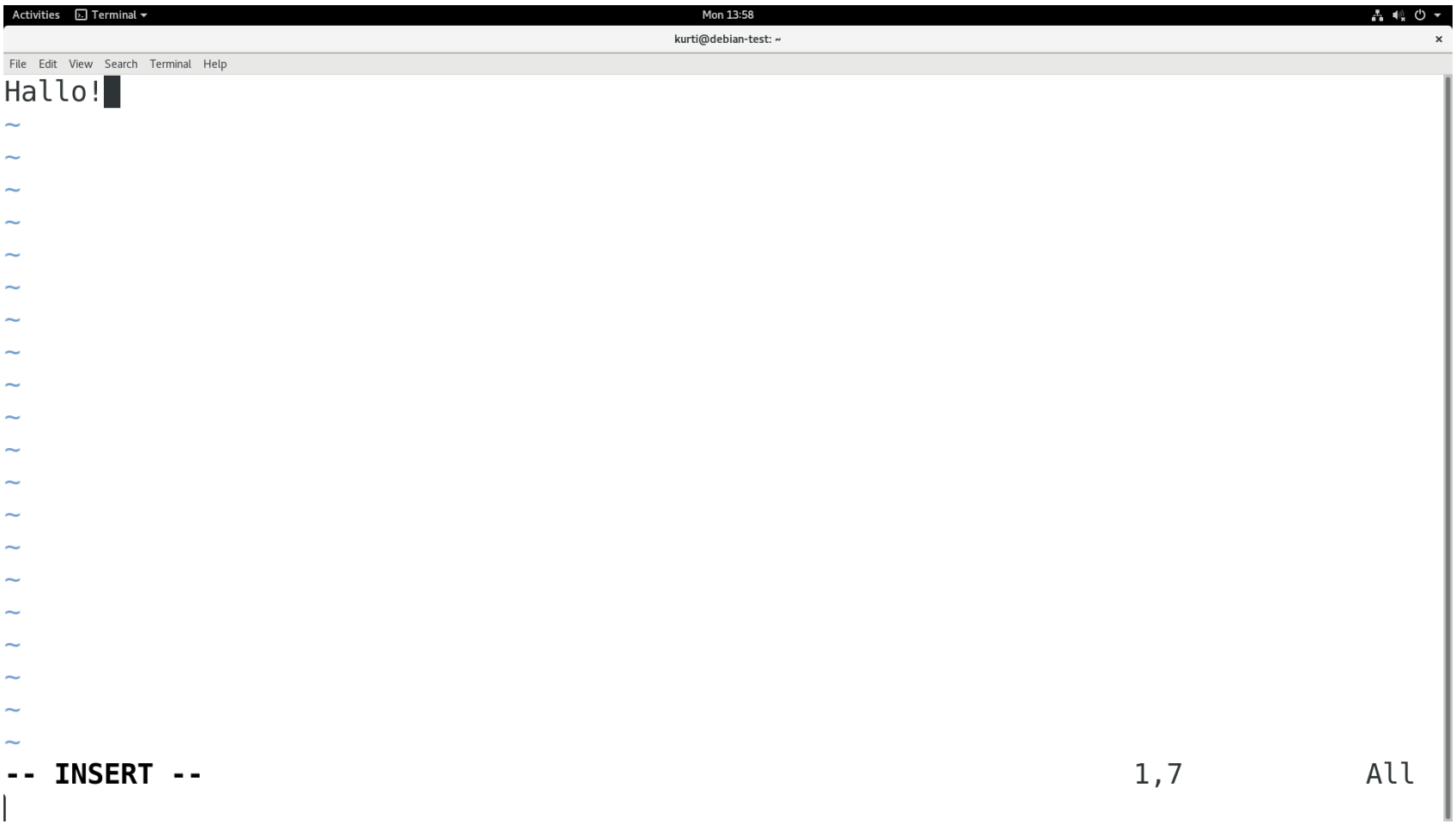
emacs II

- `Strg + x Strg + s`: Speichern
- `Strg + x Strg + c`: Schließen
- `Alt + x doctor`: Psychologe
- Cheat sheet: `https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf`

https://creativecommons.org/licenses/by-nd/4.0/

Texteditoren in der Kommandozeile

vi/vim I



https://creativecommons.org/licenses/by-nd/4.0/

vi/vim II

- `:q` – Beenden
- `:w` – Speichern
- `:wq` – Speichern und beenden
- `i` – Insert-Modus
- `Esc` – Modus verlassen

Bildquelle:
`https://www.commitstrip.com/en/2017/05/29/trapped/`

`https://creativecommons.org/licenses/by-nd/4.0/`

Debian, Ubuntu: dpkg mit APT

- Paketformat: `.deb`, Ubuntu Pakete sind *häufig* kompatibel mit Debian
- Paket-tool: `dpkg`
- Paketverwaltung: `apt` oder `apt-get`
- Paketquellen aktualisieren:
  `apt update` oder `apt-get update`
- Alle Pakete aktualisieren:
  `apt dist-upgrade` oder `apt-get dist-upgrade`
- Paket installieren:
  `apt install <paketname>` oder
  `apt-get install <paketname>`
- Paket entfernen:
  `apt remove <paketname>` oder
  `apt-get rmeove <paketname>`
- Paketsuche:
  `apt search <text>` oder `apt-cache search <text>`

https://creativecommons.org/licenses/by-nd/4.0/

Fedora, CentOS, RHEL: rpm mit yum

- Paketformat: `.rpm`, Pakete i.d.R. zwischen den Distributionen kompatibel
- Paket-tool: `rpm`
- Paketverwaltung: `yum`
- Paketquellen-cache aktualisieren:
  `yum makecache`
- Paket installieren:
  `yum install <paketname>`
- Alle Pakete aktualisieren:
  `yum update`
- Paket entfernen:
  `yum remove <paketname>`
- Paketsuche:
  `yum search <text>`
- Praktisches Cheat sheet:
  `https://access.redhat.com/articles/yum-cheat-sheet`

https://creativecommons.org/licenses/by-nd/4.0/

OpenSuSE, SLES: rpm mit zypper

- Paketformat: `.rpm`, Pakete i.d.R. *nicht* mit RHEL/CentOS/Fedora kompatibel
- Paket-tool: `rpm`
- Paketverwaltung: `zypper`
- Paketquellen-cache aktualisieren:
  `zypper refresh`
- Paket installieren:
  `zypper install <paketname>`
- Alle Pakete aktualisieren:
  `zypper update`
- Paket entfernen:
  `zypper remove <paketname>`
- Paketsuche:
  `zypper search <text>`
- Praktische Cheat sheets:
  `https://en.opensuse.org/images/1/17/Zypper-cheat-sheet-1.pdf`,
  `https://en.opensuse.org/images/3/30/Zypper-cheat-sheet-2.pdf`

https://creativecommons.org/licenses/by-nd/4.0/

Nicht-paketierte Software installieren

- Manchmal gibt es für bestimmte Software keine Pakete für die eigene Distribution, sondern ein "generisches" `.tar.gz`-Archiv (mehr zu `tar` später)
- Konvention: In Ordner in `/opt` entpacken und das Hauptprogramm in `/usr/local/bin` linken

⇒ `ln -s <Pfad zur Anwendung> /usr/local/bin/<anwendung>`

https://creativecommons.org/licenses/by-nd/4.0/

Autovervollständigung

- Kommandos und Pfade können durch drücken von `Tab` oder `Esc` autovervollständigt werden
- Für viele Kommandos werden bei der Installation eigene Autovervollständigungen für Bash mitgeliefert, sodass in vielen Fällen auch Parameter semantisch korrekt vervollständigt werden können

https://creativecommons.org/licenses/by-nd/4.0/

Einfache Dateimatcher

- Möchte man auf mehrere Dateien mit gleichen Dateinamenteile matchen, so kann man den Wildcard-Operator "`*`" verwenden.

  Beispiele:

  - `*.png` – Matched alle Dateien und Ordner, die mit "`.png`" enden.
  - `bilder_*.png` – Matched alle Dateien und Ordner, die mit "`bilder_`" anfangen und mit "`.png`" enden.
- Möchte man bestimmte (z.B. durchnummerierte) Dateien matchen, kann man die Bash Brace Expansion (`{..}`) nutzen.

  Beispiele:

  - `doc_{1..4}.pdf` oder `doc_{1,2,3,4}.pdf` – Matched doc_1.pdf, doc_2.pdf, doc_3.pdf, doc_4.pdf
  - Auch Alphabetisch möglich: `doc_{a..d}.pdf` oder `doc_{d,c,a,d}.pdf`
  - ...oder mit Listen: `doc_{teil1,teil8}.pdf`

https://creativecommons.org/licenses/by-nd/4.0/

History und Bash Reverse Search

- Die Bash führt eine History über abgesetzte Kommandos
- Mittels der ↑- und ↓-Tasten kann man in der History navigieren und alte Kommandos nochmals abschicken oder modifizieren
- Mit der Tastenkombination `Strg` + `R` kann man einen Text in der History suchen und sich entsprechende Kommandos aus der Vergangenheit zurückholen

https://creativecommons.org/licenses/by-nd/4.0/

Logische Operationen

- Führe `<Kommando2>` *nur* dann aus, wenn `<Kommando1>` erfolgreich war (logisches UND, nicht erfolgreich: `$?` $\neq 0$):
  `<Kommando1> && <Kommando2>`
- Führe `<Kommando2>` *nur* dann aus, wenn `<Kommando1>` *nicht* erfolgreich war (logisches ODER):
  `<Kommando1> || <Kommando2>`
- `true` wird *immer* erfolgreich ausgeführt
- `false` schlägt *immer* fehl

$\Rightarrow$ `false || true` *immer* erfolgreich

https://creativecommons.org/licenses/by-nd/4.0/

Pipes und Redirects

- Mit dem Pipe-Operator "|" kann man die Textausgabe von Programmen wieder einem anderen Programm als Eingabe weiterleiten
- Das Programm hinter der Pipe muss dazu die Eingabe von der Konsole lesen. Möchte das Programm hinter der Pipe aber eine "Datei" lesen, so kann man den "Ausgabekanal" als einzelnes "-" wieder in das Programm geben
- Ausgaben können auch in Dateien mittels "`<Kommando> > <Dateiname>`" geleitet werden (überschreibt dabei die Datei `<Dateiname>` sofern sie schon existiert!)
- Möchte man an die Datei anhängen, anstatt zu überschreiben, kann stattdessen "`<Kommando> >> <Dateiname>`" genutzt werden

https://creativecommons.org/licenses/by-nd/4.0/

grep

- Filtert die Eingabe
- Beispiel: `cat <Datei> | grep -F <zu suchender Text>`
- Suche kann mit `-v` invertiert werden
- Mit `-F` wird der Matching-Modus auf "text" gesetzt. Ohne `-E` wird der Suchtext als regulärer Ausdruck interpretiert
- Rekursive Suche in Verzeichnis oder mehreren Dateien mit `-r`

https://creativecommons.org/licenses/by-nd/4.0/

Exkurs: Reguläre Ausdrücke (RegEx) mit `grep` und `sed`

- Text-Ausschnitte können einfach gemachted werden.
- Zusätzlich einige Zeichen mit speziellen Funktionen (Auswahl):
  - `$`: Ende der Zeile
  - `^`: Anfang der Zeile
  - `.`: Ein beliebiges Zeichen
  - `*`: Das Muster zuvor kann 0 bis beliebig oft wiederholt werden
  - `[ ]`: Treffe Liste an Buchstaben, bzw. Buchstabnebereichen, z.B.:
    `[a-z,A-Z,0-9]`

https://creativecommons.org/licenses/by-nd/4.0/

sed

- **S**tream **Ed**itor
- Nimmt Eingabe und gibt sie wieder aus
- Wendet dabei einen RegEx an und ersetzt gemachtchte Ausschnitte mit einem anderen Text
- Syntax:
  `sed 's/<muster>/<ersetzung>/<option>'`
  Optionen (Auswahl):
  - *keine* – ersetzt nur erste Matches pro Zeile
  - *n* – Ersetzt nur das *n*-te Vorkommen
  - *g* – Global: Ersetzt alle Matches

https://creativecommons.org/licenses/by-nd/4.0/

sort

- Sortiert Zeilen
- Mit der Option `-h` können SI-Präfixe (`k`,`g`,`t` usw.) beachtet werden
- Mit der Option `-n` kann numerisch sortiert werden

https://creativecommons.org/licenses/by-nd/4.0/

`wc`

Zählt Zeilen (`-l`), Wörter (`-w`), Zeichen (`-m`) und Bytes (`-c`) einer Eingabe.

https://creativecommons.org/licenses/by-nd/4.0/

find

- Findet Dateien
- Viele Filteroptionen für Namen, Pfadnamen, Dateimetadaten u.v.m.
- Beispiel:
  `find -iname "*.pdf" .`
  Sucht im aktuellen Pfad nach Dateien, die mit "`.pdf`" enden (ignoriert Groß- und Kleinschreiben)

https://creativecommons.org/licenses/by-nd/4.0/

`locate`

- Hält optimierte Suchdatenbank aller Dateien im Dateibaum
  - $\Rightarrow$ Datenbank muss mittels `updatedb` aktualisiert werden
- `locate <suchbegriff>`, bzw. `locate -i <suchbergriff>`, um nicht zwischen Groß- und Kleinschreibung zu unterscheiden

https://creativecommons.org/licenses/by-nd/4.0/

Pager

- Pager erlauben das Scrollen und durchsuchen von langen Ausgaben auch ohne Terminal-Emulator
- Häufigster Vertreter: `less`
- Suche mittels /

https://creativecommons.org/licenses/by-nd/4.0/

Überprüfung des Images (Ubuntu) I

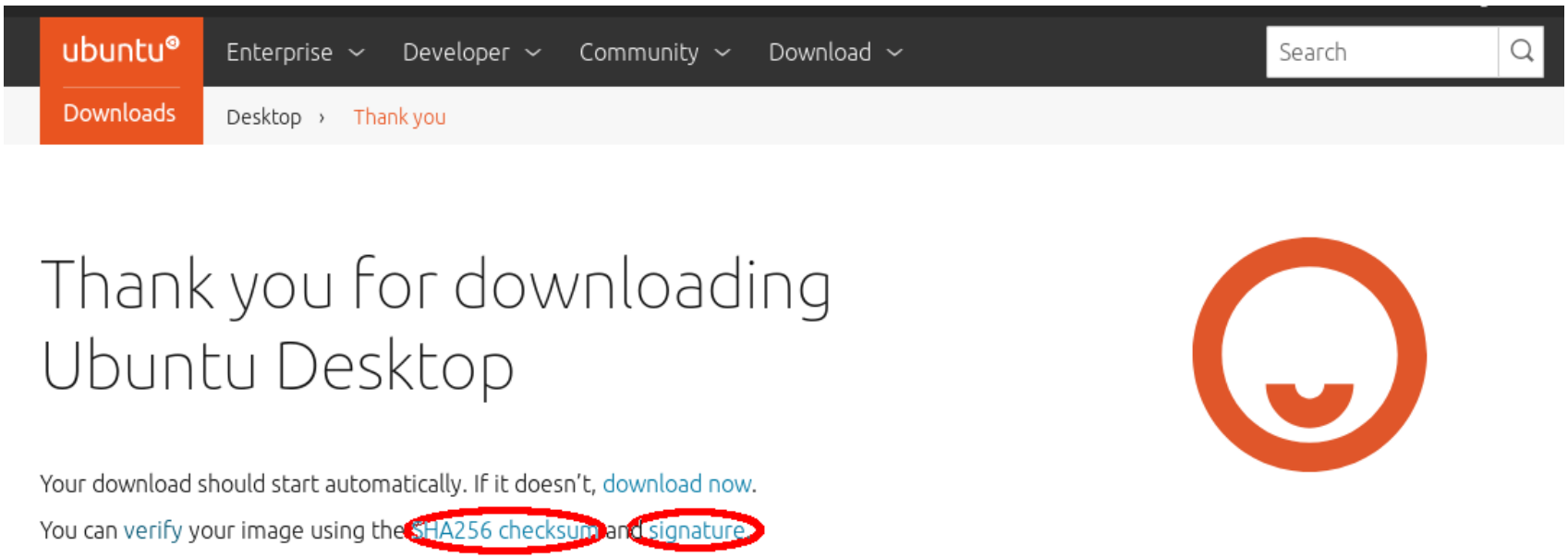


Figure: Kryptographische Signatur und Prüfsumme eines Ubuntu-Images

Man erhält SHA256-Summen und deren Signaturen

https://creativecommons.org/licenses/by-nd/4.0/

Überprüfung des Images (Ubuntu) II

Verifikation der Signaturen:

```
# 1. Importieren des oeffentlichen Schluessels von Ubuntu ueber einen Keyserver
$ gpg --keyid-format long --keyserver hkp://keyserver.ubuntu.com \
                          --recv-keys 0x46181433FBB75451 0xD94AA3F0EFE21092
# 2. Ueberpruefung der Signatur der Pruefsummen
$ gpg --keyid-format long --verify SHA256SUMS.gpg SHA256SUMS
```

Die Ausgabe sollte dann den Text "`Good signature`" enthalten. Ist eine Signatur "`Bad`", so kann die Authentizität ("Echtheit") des Images *nicht* gewährleistet werden (Sicherheitsrisiko!)

https://creativecommons.org/licenses/by-nd/4.0/

Überprüfung des Images (Ubuntu) III

Überprüfung der Prüfsummen:

```
$ sha256sum -c SHA256SUMS 2>&1 | grep OK
```

Ausgabe (beispielhaft) bei erfolgreicher Überprüfung des Images:

```
ubuntu-18.04-desktop-amd64.iso: OK
```


https://creativecommons.org/licenses/by-nd/4.0/

Umgebungsvariablen

- Globale (in der shell-Instanz gültige, aber an Kindprozesse übergebene) Umgebungsvariablen können mittels `export <VARIABLE>=<inhalt>` gesetzt werden
- Liste der aktuellen Umgebungsvariablen: `env`
- Referenzierung von Variablen immer mit `$<variable>`

https://creativecommons.org/licenses/by-nd/4.0/

Wichtige Umgebungsvariablen und ihre Funktionen

Variable	Funktion
`$PATH`	":"-getrennte Liste von Ordnern, in denen sich die Programme befinden
`$HOME`	Home-Directory des aktuellen Nutzers
`$PWD`	CWD
`$LOGNAME`, `$USER`, `$USERNAME`	Aktueller Nutzername
`$EDITOR`	Standart-Texteditor

https://creativecommons.org/licenses/by-nd/4.0/

Linux: Das Handwerkszeug für die tägliche Arbeit Teil III

Janis Streib
CC-BY-ND-4.0

09.-13.07.2018

https://creativecommons.org/licenses/by-nd/4.0/

Inhalt

Prozessverwaltung

Remote-Zugriff

Datensicherung

Einfache Automatisierung

Drucken
    CUPS

https://creativecommons.org/licenses/by-nd/4.0/

Prozesse in Linux

- Jeder Prozess erhält für seine Laufzeit eine ID vom Linux-Kern (PID)
- Jeder Prozess hat einen Elter (außer dem init-Prozess, PID 1) $\Rightarrow$ Baum
- Stirbt ein Prozess, gibt er einen Code zurück (return code), den der Elter-Prozess behandeln muss (return code $\neq 0 \Rightarrow$ Fehler)
- Stirbt ein Elter-Prozess, werden seine Kinder von PID 1 "adoptiert"
- Prozessen können Signale geschickt werden, auf die die Prozesse unterschiedlich reagieren können
  - Spezialfall: `SIGKILL (9)`: Hier wird der Prozess "mit Gewalt" (ohne Interaktion mit dem Prozess) von außen getötet
- Prozesse laufen immer unter einem Nutzer, mit den jeweiligen Rechten des Nutzers
  - Der Pseudo-Nutzer `nobody` hat keine Rechte

https://creativecommons.org/licenses/by-nd/4.0/

ps

Zeige Prozessliste

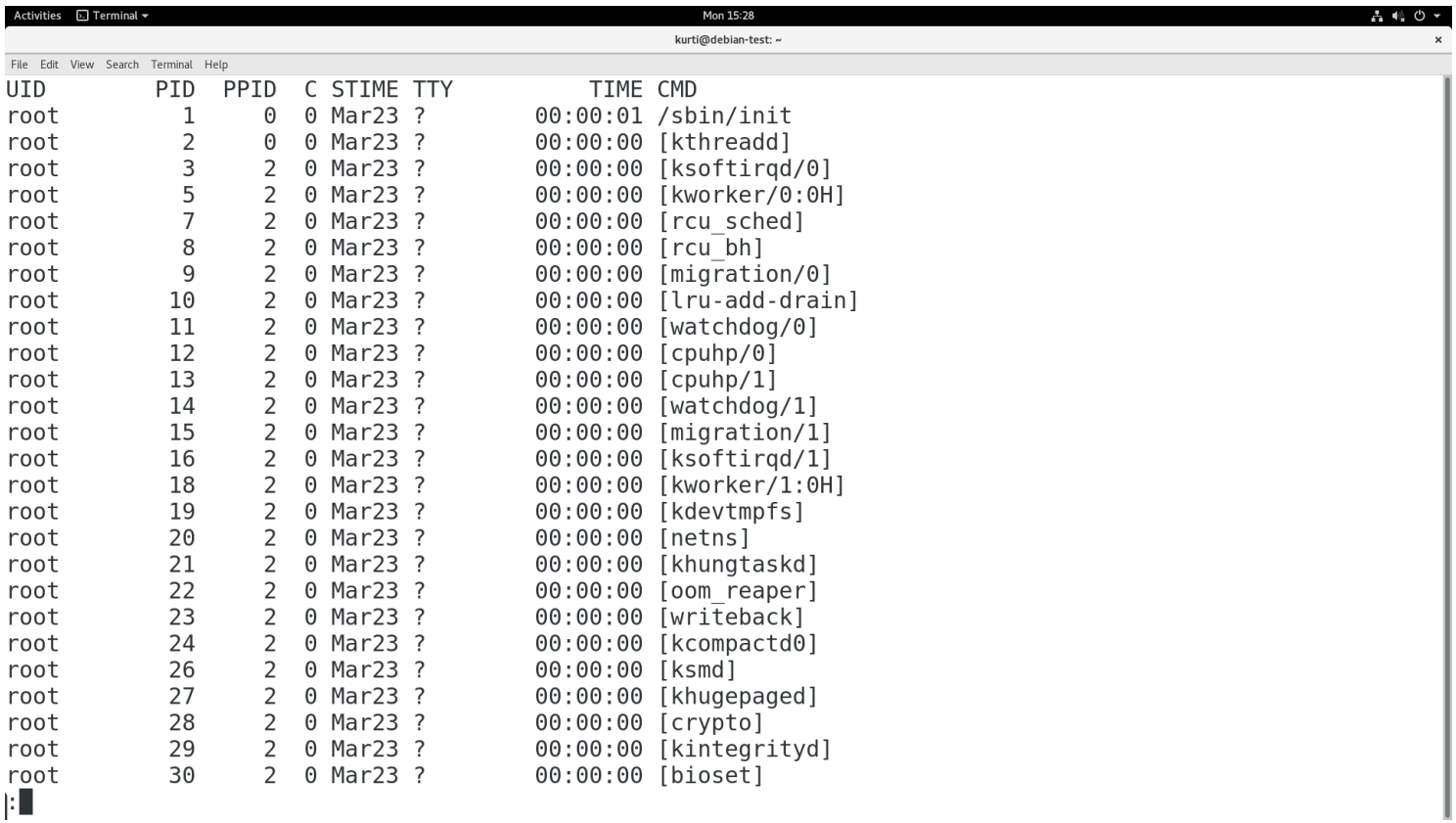


Figure: `ps -ef | less`

https://creativecommons.org/licenses/by-nd/4.0/

top

Zeige sich aktualisierende Prozessliste, mit RAM- und CPU-Auslastung

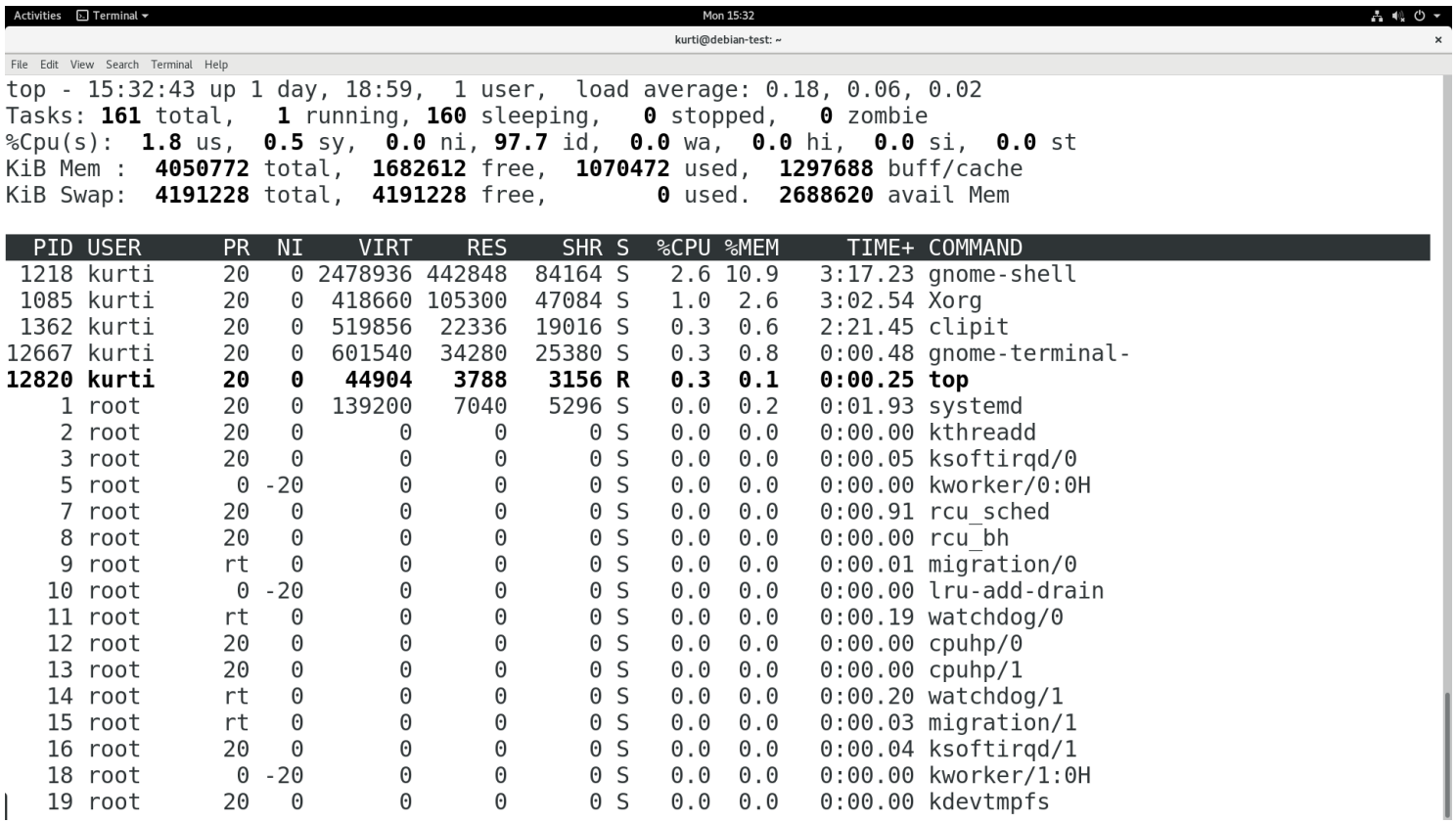


Figure: top

https://creativecommons.org/licenses/by-nd/4.0/

htop

Zeige sich aktualisierende, bunte Prozessliste, mit graphischer RAM- und CPU-Auslastung

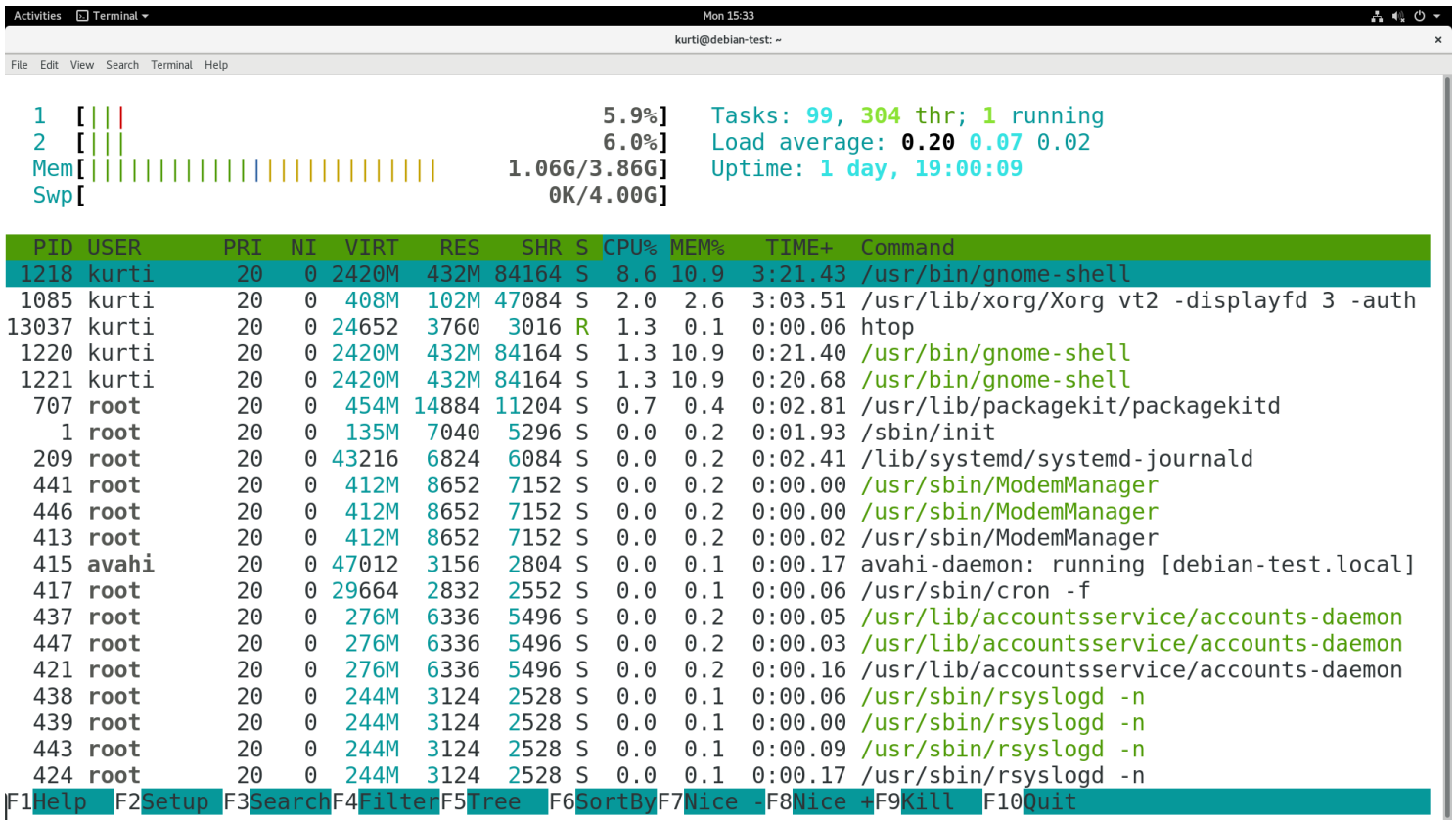


Figure: htop

https://creativecommons.org/licenses/by-nd/4.0/

kill

- `kill -<Signalname/-Nummer> <PID>` schickt dem Prozess mit der PID `<PID>` ein Signal.
  - Beispiel: `kill -9 1234`
- PID kann mittels `ps` herausgefunden werden.
  - `killall -<Signal> <Prozessname>` schickt *allen* Prozessen mit dem Namen `<Prozessname>` ein Signal

  $\Rightarrow$ Prozesse sind nicht eindeutig mit ihrem Namen identifizierbar!
- `kill` kann beispielsweise aus `htop` heraus mittels der Taste F9 auf den ausgewählten Prozess aufgerufen werden

https://creativecommons.org/licenses/by-nd/4.0/

SSH

- **S**ecure **Sh**ell – Protokoll und Programm zur sicheren Herstellung einer verschlüsselten Verbindung
- `ssh <user>@<host>` öffnet eine Shell auf Host `<host>` als Benutzer `<user>`.
  - $\Rightarrow$ SSH-Server muss auf entferntem Host laufen
- Authentizität wird durch Speichern des Fingerprints des öffentlichen Schlüssels des Hosts sichergestellt (Bestätigung und manuelle Überprüfung durch Nutzer bei erster Verbindung zum Server).
- SSH erlaubt auch Transport von Daten über den sicheren Kanal (mehr dazu später)

https://creativecommons.org/licenses/by-nd/4.0/

tar

- **T**ape **Ar**chiver
- Packprogramm und Dateiformat für Dateien (ohne Kompression!)
- Häufig zusammen mit Kompression verwendet (gepacktes Archiv wird komprimiert) wie z.B. `tar.gz` oder `tar.xz`
- Auspacken:
  `tar xf <datei>`
- Auspacken (mit gzip-Kompression):
  `tar xzf <datei>`
- Einpacken:
  `tar cf <archiv> <dateien>`
- Einpacken (mit gzip-Kompression):
  `tar czf <datei> <dateien>`
- Inhalt überprüfen (**t**):
  `tar tf <archiv>`
- Möchte man, welche Dateien ein-/ ausgepackt werden, kann man noch das Flag "`v`" vor das "`f`" platzieren

https://creativecommons.org/licenses/by-nd/4.0/

cpio (veraltet)

- **c**o**p**y **i**n, copy **o**ut
- Archivprogramm vergleichbar zu `tar`
- Aktuelles Verzeichnis einpacken:
  `ls | cpio -o > verzeichnis.cpio`
- Archiv in Verzeichnis auspacken:
  `cpio -i < archiv.cpio`

https://creativecommons.org/licenses/by-nd/4.0/

scp

- Analog zu `cp`, nur kann man über `ssh` Dateien auf einen Server kopieren
  - Beispiel: Lade die Datei "`test.txt`" auf den Server "`meinserver.tld`" in dem sich im Home-directory des Nutzers "`kurti`" befindlichen Ordner "`Unterlagen`"
    `scp test.txt kurti@meinserver.tld:Unterlagen/`

https://creativecommons.org/licenses/by-nd/4.0/

rsync

- Dateien zu Ziel kopieren
- Inkrementelles Senden abhängig vom am Ziel vorhandenen Dateien
- Unvollständig kopierte Dateien können fortgesetzt werden

https://creativecommons.org/licenses/by-nd/4.0/

cron

- Dienst, der die Ausführung von Kommandos zu festen Uhrzeiten oder in Intervallen erlaubt
- Als Nutzer oder als Root
- Editieren der Cron-Jobs: `crontab -e`
  - Öffnet einen Texteditor mit den Job-Definitionen
  - Zeile-Syntax: `<Minute> <Stunde> <Tag des Monats> <Monat> <Tag der Woche> <Kommando>`
  - Beispiel: Führe alle 5 Minuten aus
    `*/5 * * * * <Kommando>`
  - Beispiel: Führe am Anfang jeder Woche um 5 Uhr aus:
    `0 5 * * 1 <Kommando>`

https://creativecommons.org/licenses/by-nd/4.0/

at

- Führe ein Kommando einmalig zu einer bestimmten Zeit aus
- Beispiel:
  `at 8:00pm`
  Anschließend die Liste der auszuführenden Kommandos angeben und mit `Strg + D` Abschließen
- Zeitangaben relativ oder absolut möglich, z.B.: `now + 2hours`
- Weiter Infos zu den Zeitangaben: `man at`

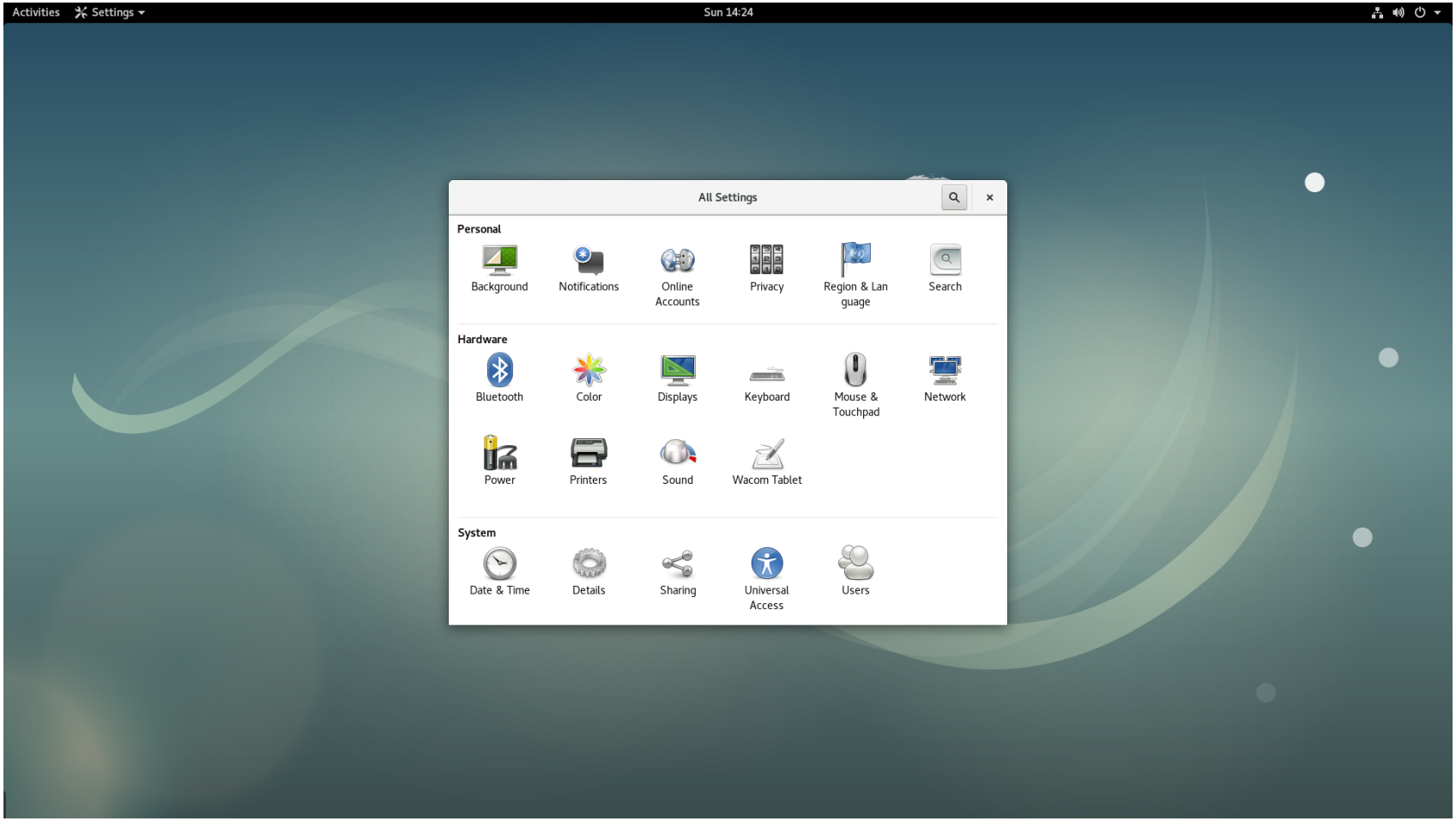
https://creativecommons.org/licenses/by-nd/4.0/

CUPS

- **C**ommon **U**nix **P**rinting **S**ystem
- Findet Verwendung auf vielen Unix-artigen System, z.B. auch auf MacOS
- In Linux:
  - Um drucken zu dürfen, müssen Nutzer in der Gruppe `lp` sein
  - Um CUPS verwalten zu dürfen, müssen Nutzer in der Gruppe `lpadmin` sein

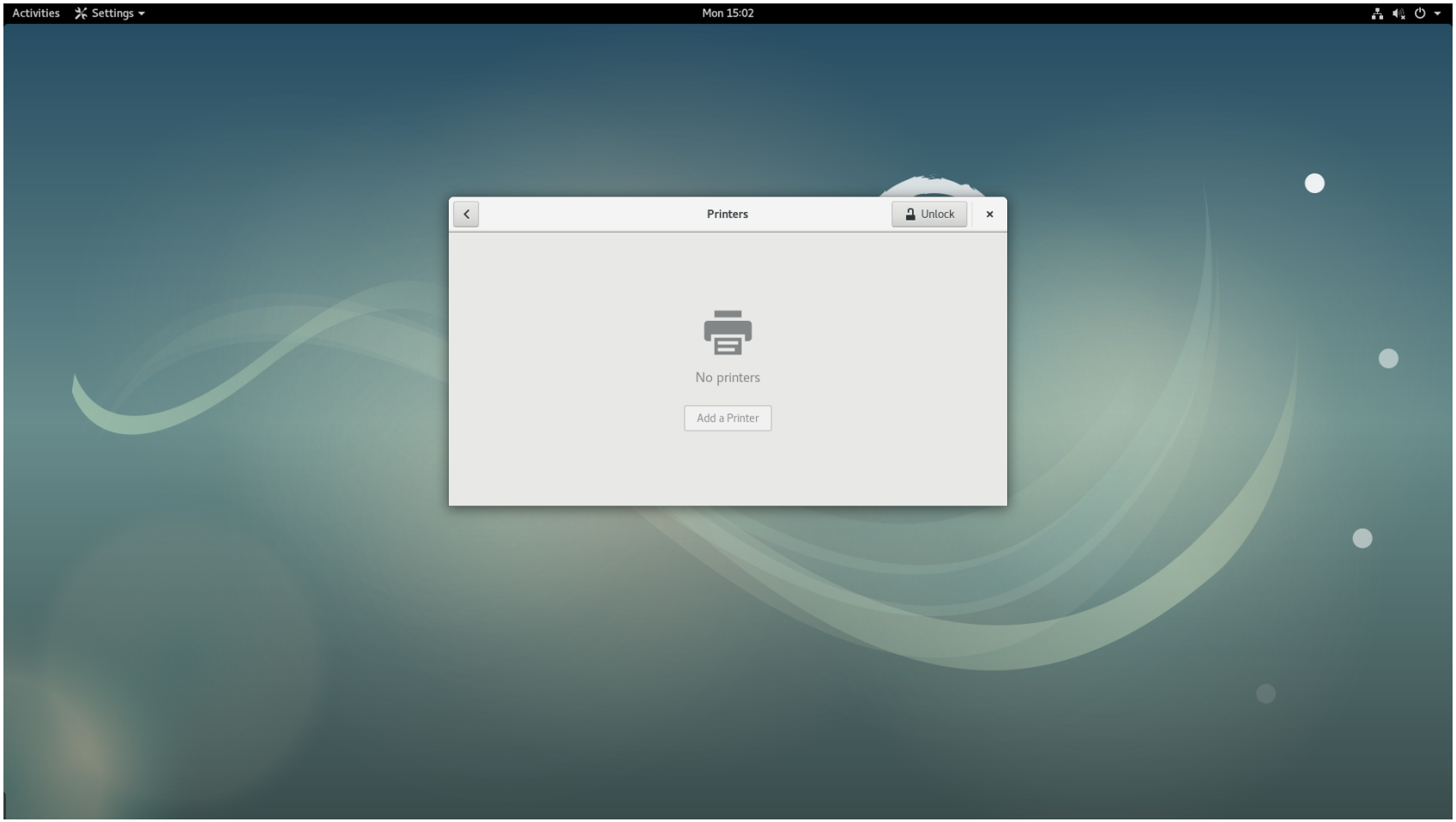
https://creativecommons.org/licenses/by-nd/4.0/

CUPS-Verwaltung in GNOME I



https://creativecommons.org/licenses/by-nd/4.0/

CUPS-Verwaltung in GNOME II



https://creativecommons.org/licenses/by-nd/4.0/

CUPS-Verwaltung im Web-Interface

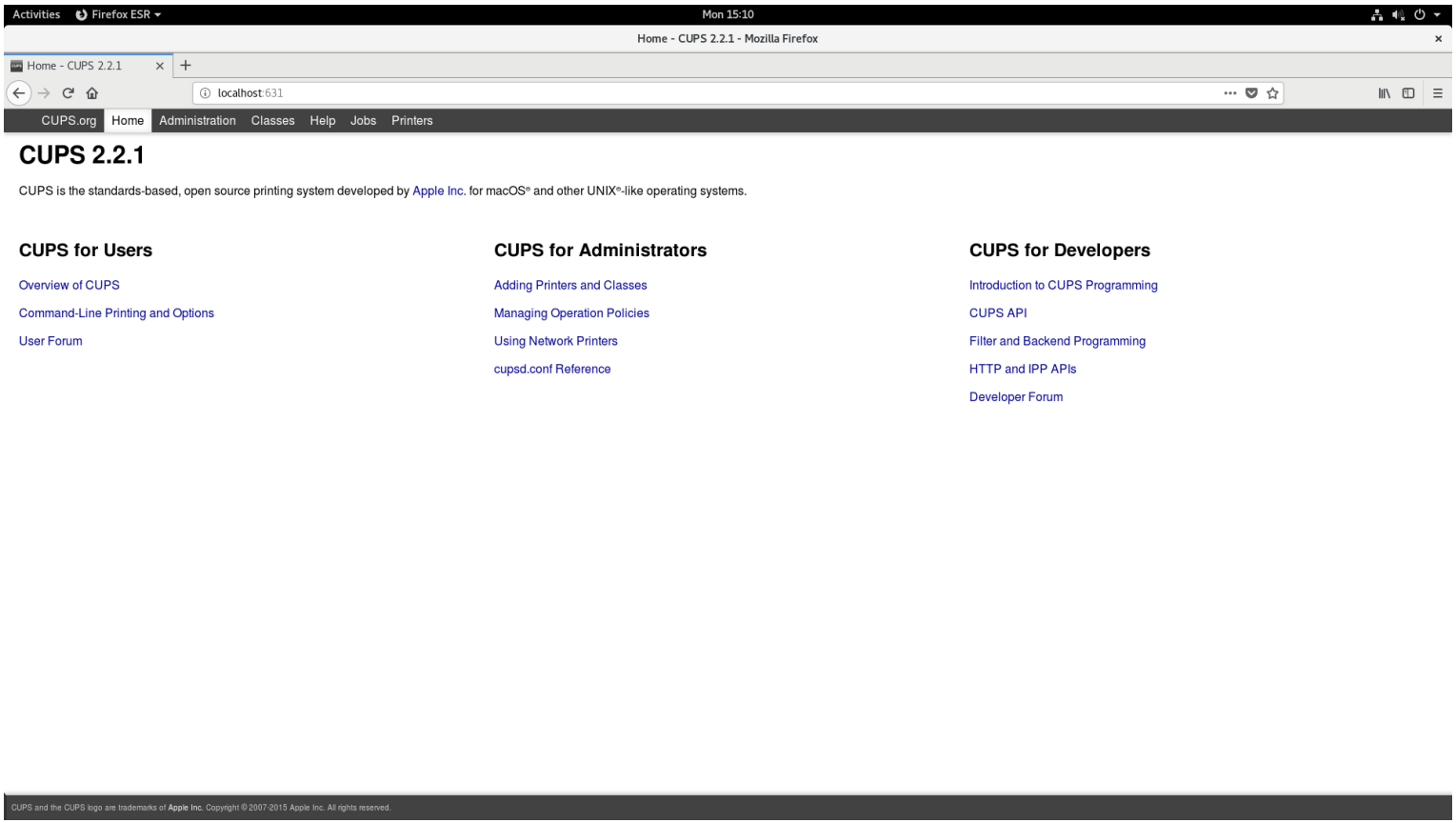


Figure: Verwaltung der lokalen CUPS-Instanz im Browser über die URL `http://localhost:631`

https://creativecommons.org/licenses/by-nd/4.0/

`lp` – Drucken aus der Kommandozeile

- Drucken auf der lokalen CUPS-Instanz mit dem Standarddrucker mit standard Druckereinstellungen:
  `lp <datei>`
- Drucken auf der lokalen CUPS-Instanz mit einem spezifischen Drucker mit standard Druckereinstellungen:
  `lp -d <druckername im CUPS> <datei>`

https://creativecommons.org/licenses/by-nd/4.0/

Linux: Das Handwerkszeug für die tägliche Arbeit
Bonus: SSH für Fortgeschrittene

Janis Streib
CC-BY-ND-4.0

09.-13.07.2018

https://creativecommons.org/licenses/by-nd/4.0/

Inhalt

Authentifizierung
- Keys
- SSH-Agent

Tunnel

https://creativecommons.org/licenses/by-nd/4.0/

SSH Authentifizierung

- SSH erlaubt verschiedene Mechanismen zur Authentifizierung, u.A.:
  - Passwort
  - Authentifikation mit Key

https://creativecommons.org/licenses/by-nd/4.0/

Öffentlicher und Privater Schlüssel

- SSH verwendet asymmetrische Kryptographie
  - Es gibt einen öffentlichen und einen privaten Schlüssel
  - Nachrichten können mit dem privaten Schlüssel verschlüsselt werden und ein Gesprächspartner kann mittels des öffentlichen Schlüssels feststellen, dass die Nachricht mit dem privaten Schlüssel verschlüsselt wurde (Signatur, die Nachricht kann nicht entschlüsselt werden)
  - Nachrichten können mit dem öffentlichen Schlüssel verschlüsselt werden, aber nur mit dem privaten entschlüsselt
- Durch Hinterlegung des eigenen öffentlichen Schlüssels auf dem Server ist eine Authentifikation des Nutzers möglich

Wichtig!

Der private Schlüssel darf **nie** den eigenen Rechner verlassen!

https://creativecommons.org/licenses/by-nd/4.0/

Schlüsselpaare erzeugen mit `ssh-keygen`

- Durch Ausführung von `ssh-keygen` kann (ohne Angabe von Parametern) ein SSH-Schlüsselpaar erzeugt werden
- Standard-Schlüsselname: `id_rsa`
- Standard-Ablageort: `~/.ssh`
- Der private Schlüssel kann noch zusätzlich mit einem Passwort verschlüsselt werden (Diebstahlschutz!)

https://creativecommons.org/licenses/by-nd/4.0/

Hinterlegen des öffentlichen Schlüssels auf einem Server

- Um sich auf einem entfernten Rechner mit dem Schlüssle zu authentifizieren, muss dieser dort hinterlegt werden
- Alle zugelassenen Schlüssel werden auf dem Server in der Datei `/.ssh/authorized_keys` hinterlegt (einer pro Zeile)
- Automatischer Schlüsselupload und Hinterlegung in der Datei mittels `ssh-copy-id <nutzer>@<host>`

https://creativecommons.org/licenses/by-nd/4.0/

Der SSH-Agent

- Dienst, der lokal den Authentifizierungsprozess mit dem eigenen Schlüsselpaar übernimmt
- Hinzufügen der Standardschlüssels `id_rsa`: `ssh-add`
- Allgemein: `ssh-add <pfad zum schlüssel>`

https://creativecommons.org/licenses/by-nd/4.0/

SSH-Agent-Forwarding

- Der Agent kann zur Authentifizierung z.B. auf einen Jumphost getunnelt werden

⇒ Erlaubt die Nutzung der Authentifizierung mit seinem persönlichen Schlüssel an weiteren Servern vom Jumphost aus, ohne, dass der private Schlüssel den eigenen Rechner verlässt!

- `ssh -A <nutzer>@<host>`

https://creativecommons.org/licenses/by-nd/4.0/

Tunnel: Beispiel & Motivation

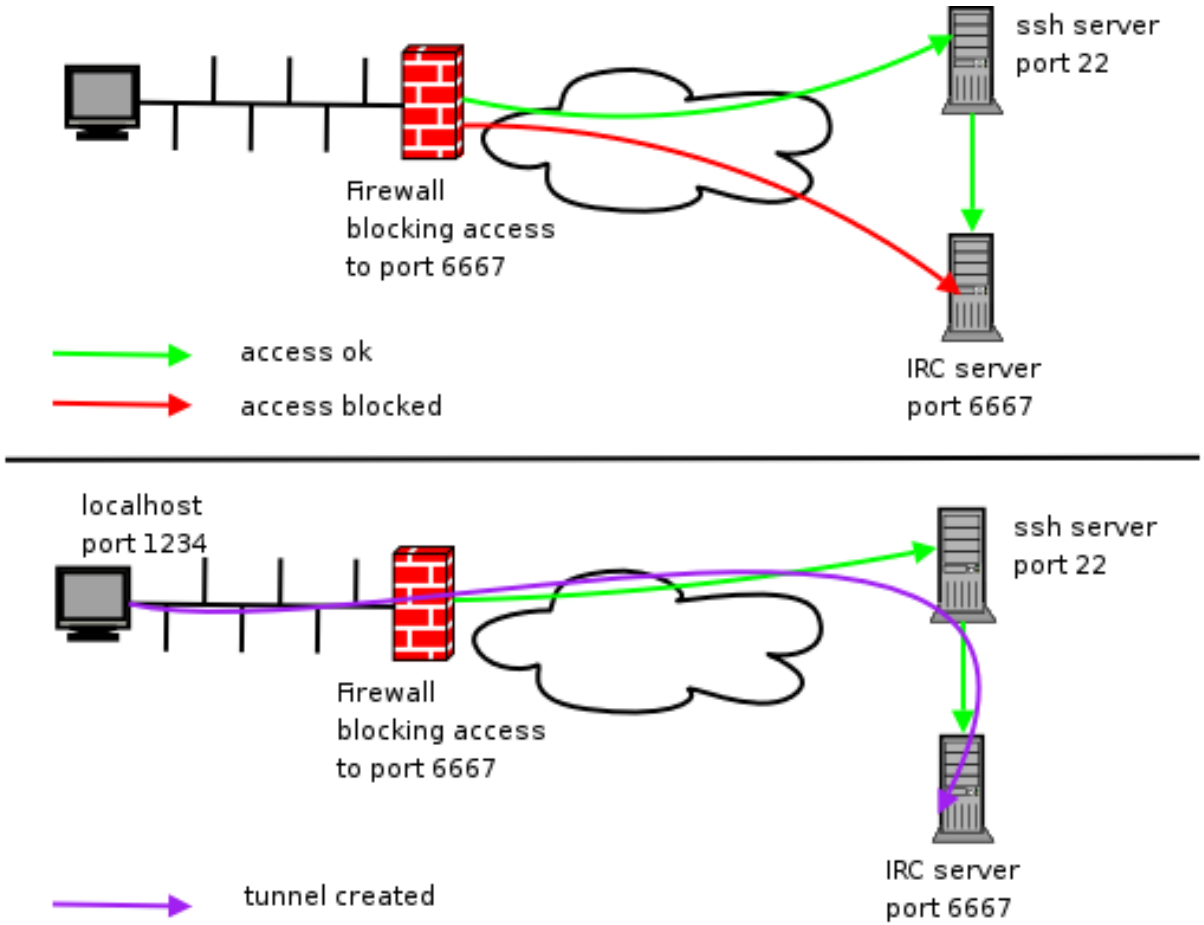


Figure: Quelle: `https://www.gaia-gis.it/fossil/virtualpg/wiki?name=port-forwarding`

`https://creativecommons.org/licenses/by-nd/4.0/`

SSH-Port-Forwarding

- Ziel: Erreichen eines entfernten TCP-Ports, den man nicht von sich aus (direkt) erreichen kann
- Lösung: (local) SSH-Port-Forwarding
- `ssh -L <lokaler Port>:<host>:<zu erreichender Port> <nutzer>@<jumphost>`
- So kann nun auf den lokalen Rechner über "`localhost:<lokaler Port>`" den entfernten Port `<zu erreichender Port>` auf `<host>` zugreifen, indem dieser durch `<jumphost>` und die SSH-Verbindung geleitet wird

https://creativecommons.org/licenses/by-nd/4.0/

SSH Proxy-Jump

- Möchte man einen SSH-Host erreichen, der nur durch einen anderen SSH-Host erreichbar ist, so kann die Option "ProxyJump" (`-J`) verwendet werden
- Beispiel:
  `ssh -J <nutzer>@<jumphost> <nutzer>@<zielhost>`

https://creativecommons.org/licenses/by-nd/4.0/