Versuch 1
Installation und erste Schritte

Ubuntu ist eine verbreitete kostenlose Linux-Distribution der Firma *Canonicals*, welche sich zum Ziel gesetzt hatte, eine für „Otto Normalverbraucher“ einfach bedienbare Linux-Distribution anzubieten. Ubuntu basiert grösstenteils auf die Distribution *Debian*, einer ganz und gar nicht kommerziellen und von einer offenen Community entwickelten Linux-Distribution, welche v.a. auch im Server-Bereich weit verbreitet ist (*Debian* ist jedoch nicht ganz so einfach zu installieren wie *Ubuntu*).

Ubuntu gibt es in verschiedenen Versionen, einerseits im einem halbjährlich erscheinenden eher experimentellen „Entwickler-Release“, welcher jeweils nur 9 Monate lang mit Sicherheitsupdates versorgt wird. Andererseits und für die meisten Anwender empfehlenswerter, in einem alle 2 Jahre erscheinenden "Ubuntu *LTS"* Release, also mit „*Long Term Support*“, welcher lange 5 Jahre mit Sicherheitsupdates versorgt wird und weniger experimentell sondern immer stabil sein sollte. Wir installieren deshalb diesen.

- Um Linux wie in diesem Kurs bloss kennen zu lernen, installiert man Linux bevorzugt ohne grosse Risiken als „**Gast-System**“ unter dem bereits vorhandenen Windows- oder OS-X-Hostsystem. Hierzu wird eine **Virtualisierungs-Software** wie z.B. ***VirtualBox*** oder *VMWare Workstation Player* benötigt. Beide sind für nicht-kommerzielle Einsatzzwecke kostenlos (im Gegensatz zu *Parallels*).
- Oder **risikobehafteter** und **nicht wirklich praktisch**: **Linux** parallel zu Windows/OS-X in einer weiteren Diskpartition installieren, in welchem Fall beim Wechsel jeweils ein reboot nötig ist[1],
- Oder **kompromisslos**: die Windows- oder OS-X Partition komplett zu löschen und durch Linux ersetzen. (Und Windows z.B. noch als Gastsystem unter Virtualbox zu installieren → eher eine Ferienbeschäftigung!)

✗ Insbesondere bei den letzten beiden Varianten ist zuvor ein **Backup aller Userdaten empfehlenswert!!**

Linux Grundinstallation

Wenn Sie Linux also nur für Unterrichtszwecke benötigen, ist es am sichersten und einfachsten, wenn Sie Linux nur als „Gast-Betriebssystem“ unter einer „**Virtualisierungs-Software**“ wie *VirtualBox* installieren. Sie benötigen hierzu mindestens ca. **10..15 GB freien Diskspace** für die „virtuelle Disk“ und für flüssiges Arbeiten mindestens etwa **4 GB RAM**, denn es sind ja gleichzeitig zwei Betriebssysteme aktiv!

– Downloaden Sie hierzu vom AD oder der Ubuntu Website (https://www.ubuntu.com/download/desktop) das aktuelle ***Ubuntu LTS Desktop*** ISO-Image für die ***amd64*** Architektur (kompatibel zu Intel und AMD CPUs). Für leistungsschwächere Notebooks downloaden und installieren Sie hingegen besser das aktuelle ***Xubuntu LTS***, oder gar *Lubuntu,* welches eine schlankere/spartanischere Desktop-Oberfläche haben.

– Nebst obigem Linux-Image downloaden und installieren Sie eine geeignete Virtualisierungssoftware, bevorzugt das kostenlose ***VirtualBox*** ab **www.virtualbox.org,** passend für Ihr Hostsystem (Windows/OS-X) .
→ Sollte *VirtualBox* Probleme machen, verwenden Sie für Windows "*VMware Workstation Player*", für Mac OS-X "*Parallels*".

– Generieren Sie danach im ***VirtualBox Manager*** mittels Funktion „***Neu***“ eine neue „*Virtuelle Maschine*“ namens „**Ubuntu LTS**“ vom Betriebssystem-Typ **Linux / Ubuntu 64bit** und einer Hauptspeichergrösse von etwa **3GB** (aber nicht mehr als die Hälfte des Hauptspeichers!), wonach Sie eine neue „v*irtuelle Festplatte*“ von **mindestens 10GB** oder falls Sie genügend Diskspace haben besser **15GB** erzeugen. Auf dem Hostsystem ist diese "Virtuelle Disk" bloss eine „normale“ Datei, deren Grösse sich nachträglich jedoch nicht mehr einfach verändern lässt!

– Erst **nachdem** Sie das Ubuntu ISO-Image **ganz auf das Hostsystem herunter kopiert haben**, starten Sie im *Virtualbox Manager* die zuvor erstellte „*Virtuelle Maschine*“ und geben auf Aufforderung das heruntergeladene Ubuntu LTS ISO Image als virtuelles CD-Image an, wonach dieses booten sollte. **Sollte das CD-Image nicht booten, müssen Sie im BIOS Setup die HW-Virtualisierung (*VT-x* / *AMD-v*) einschalten.**

– Nachdem Ubuntu im Gastsystem gebootet hat, wählen Sie dann „**Install Ubuntu**“ und im späteren Verlauf die gewünschte Ländereinstellung (Sprache: **German/Deutsch**, Tastatur: ***Schweizerdeutsch***, Zeitzone: ***Zurich***)

– Das Partitionieren der „virtuellen Festplatte“ überlassen Sie dem Installer und wählen deshalb „**gesamte Disk verwenden**“, denn die "Disk" unter dem Gastsystem ist ja nur virtuell (es sei denn, Sie installieren nicht virtualisiert sondern „Dual-Boot“ ohne Virtualisierung, in welchem Fall Sie natürlich nicht die gesamte Disk

1 Eine Dual-Boot-Installation nur beim Booten via *EFI* oder *UEFI* anwenden, denn bei einer Installation via *BIOS* würde der Windows-Bootloader ersetzt, wodurch bei einem eventuellen späteren Entfernen von Linux Windows nicht mehr booten würde.

verwenden dürften!!! [2]).

– Als Benutzernamen geben Sie z.B. Ihr Kürzel oder Vorname an, als Passwort reicht für unsere experimentellen Zwecke ein kurzes Trivialpasswort sowie "automatisches Einloggen" beim Starten.

Nutzen Sie die Zeit während der Installation um die nachfolgenden zwei grauen Blöcke zu lesen!

Journaling Filesysteme

Wie Sie vielleicht wissen, unterstützt Windows nebst den antiquierten Dateisystem **FAT,** VFAT und FAT32, welche oft noch auf Memory Sticks verwendet werden, auf Harddisks und SSDs das Dateisystem **NTFS.** NTFS ist ein so genanntes „Journalling Filesystem" und dadurch recht brauchbar und sicher und erlaubt unter Windows zudem auch Rechte-Vergabe (per *ACL = **A**ccess **C**ontrol **L**ist*). Alle Windows-Dateisysteme sind jedoch nicht „POSIX-Konform", weshalb sie sich nicht als vollwertige Linux Dateisysteme eignen: Insbesondere die Dateirechte können unter Linux nicht abgebildet werden!

Linux bevorzugt „eigene" POSIX-konforme Dateisysteme, wovon ***Ext4*** am weitesten verbreitet ist. Lesen und Schreiben kann Linux aber auch viele „fremde" Dateisysteme, nebst jenen von Microsoft (NTFS, FAT/VFAT, exFAT) oder jene von Apples OS-X (HFS, HFS+, APFS), jedoch jeweils ohne die "Dateirechte" abzubilden zu können.

Das Hauptproblem von antiquierten Dateisysteme wie FAT oder auch dem alten Linux *Ext2* Dateisystem ist, dass diese nach einem Systemabsturz resp. Stromausfall in einem inkonsistenten Zustand sein können, da beim Schreiben auf das Dateisystem ja nebst den eigentlichen Datei-Daten auch die sogenannten Metadaten nachgeführt werden müssen, also Dateinamen, Dateirechte, Dateilänge und aus welchen Disk-Blöcken die jeweilige Datei besteht. Beides (Daten und Metadaten) kann aber nicht gleichzeitig aktualisiert werden, weshalb ein Systemabsturz mitten in einer Schreib-Transaktion bei diesen einfachen Dateisystemen nebst dem unvermeidbaren Datenverlust auch zu einer Inkonsistenz im Dateisystem führt. (Beim Booten einer nicht ordnungsgemässem heruntergefahren Disk kann eine Inkonsistenz meist durch einen ev. recht lange dauernder Dateisystemcheck korrigiert werden).

Um derartige Dateisystem-Inkonsistenzen zu verhindern, verwenden modernere Dateisysteme ähnlich guten Datenbanken einen ***Journal-Mechanismus***, in welchem in einer *ersten Phase* die durchzuführenden Disk-Transaktionen - oder zumindest die Metadaten - vorerst nur in eine fortlaufende *Journaldatei* geschrieben werden. Periodisch werden dann die abgeschlossenen (*comitted*) *Transaktionen* auf das effektive Dateisystem der Disk übertragen, wonach das Journal jeweils zurückgesetzt wird. Stürzt das System nun mittendrin ab, werden einfach die Journal-Transaktionen beim nächsten Booten nochmals aus der Journaldatei eingespielt resp. die unvollständige Transaktionen gelöscht. Dadurch bleibt die Disk immer konsistent. Natürlich kann auch ein derartiges *Journalling Filesystem* keinen Datenverlust bei einem Systemabsturz vermeiden: was zum Zeitpunkt des Absturzes bloss im RAM und noch nicht oder nur unvollständig in die Journaldatei geschrieben wurde, ist natürlich verloren.

Microsofts ***NTFS*** oder Apples ***HFS+*** sowie das unter Linux verbreitete ***Ext4*** sind allesamt *Journaling Filesysteme.* Daneben gewinnen neuartigere Dateisysteme mit einem „*Copy-On-Write*" Mechanismus an Bedeutung, da diese auch ohne doppeltes Schreiben konsistent bleiben, was SSD-Harddisks schont und zusätzliche Features wie *Snapshots und Datei-Checksummen* ermöglicht, was für ein effizientes Backup dienlich ist. Das Linux Dateisystem ***Btrfs*** sowie Apples ***APFS*** sind von diesem Typ.

Man braucht daraus aber keine Religion zu machen, denn insgesamt sind für „normale" Desktop-Systeme alle genannten *Journalling* und „*Copy-on-Write*" Dateisysteme gut und die Performanceunterschiede wie auch die Limiten bezüglich der Datei-Maximalgrösse (im Pentabyte-Bereich) sind unbedeutend.

Übrigens wird ein „Undelete" von keinem Linux, Windows oder OS-X Dateisystemen wirklich unterstützt: Ein echtes „Filerecovery" ist also bestenfalls mit „Lowleveltools" möglich – sofern der gelöschte Platz nicht schon wieder mit neuen Daten überschrieben wurde! Zumindest die graphischen Oberflächen von Windows, Linux und OS-X stellen jedoch einen „Mülleimer" bereit, in welchen „gelöschte" Dateien effektiv bloss *hinein verschoben* werden...
→ https://de.wikipedia.org/wiki/Ext4 → https://btrfs.wiki.kernel.org , → https://en.wikipedia.org/wiki/List_of_file_systems

2 Für eine „Dual Boot" Installation parallel zu Windows (mit UEFI-Boot!) oder OS-X (EFI-Boot) ...
- schreiben Sie zuerst das Installer ISO-Image mit einem geeigneten Tool auf einen USB-Stick und booten diesen,
- verkleinern dann beim Partitionierungsschritt per „Resize Partition" zuerst die bestehende Betriebssystem-Partition,
- und legen dann im freigewordenen Raum eine Linux-Partition von mind. 15GB sowie eine kleinen 2GB Swap-Partition an. (Wenn Sie Linux langfristig produktiv nutzen möchten, sind ev. weitere Partitionen für */home* und */opt* sinnvoll…)
- sowie eine möglichst grosse /home Partition für die Nutzerdaten (zwecks einfacherer späterer System-Reinstallation)

Geschmackssache „Desktop“

Im Gegensatz zu *Windows* und *OS-X* gibt es für *Linux* nicht nur eine sondern viele verschiedene **Desktop-Umgebungen** mit Bezeichnungen wie *GNOME, KDE, Xfce, LXDE, LXQt,* und noch etliche weitere! *(vgl.* http://en.wikipedia.org/wiki/Desktop_environment).

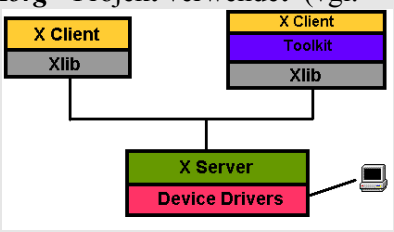
Diese Desktop-Umgebungen sind nicht Teil des Linux-Projektes sondern jeweils eigene Open Source Projekte und einige laufen auch auf anderen (nicht-Linux) POSIX Systemen wie BSD oder GNU-Hurd.

Daneben kann Linux auch ganz ohne graphische Desktop-Umgebung installiert werden, was insbesondere bei Display-losen Servern oder Embedded Systemen bevorzugt wird (z.B. Linux-basierte Router oder NAS) .

GNOME und ***KDE*** sind die „mächtigen“ und verbreitetsten Desktop-Umgebungen wogegen z.B. ***XFCE, LXDE*** und ***LXQt*** auf Schlankheit optimierte Desktop-Umgebungen sind, welche sich speziell für leistungsschwache Systeme oder virtualisierte Systeme mit wenig Gastsystem-RAM eignen. Auch Ubuntu gibt es mit verschiedenen Desktop-Umgebungen (vgl. http://www.tutonaut.de/tipp-10-desktops-fuer-ubuntu/)

Derartige Desktop-Umgebungen (engl. ***Desktop Environments)*** beinhalten nebst Programmen zur Darstellung, Dekoration und Manipulation der Fenster, Menus, Schnellstart- und Statusleiste etc. auch etliche Zubehör-Programme wie ***Dateimanager***, **ASCII-Editor**, ein **Terminal-Window** Programm, etc. Das Programm welches die Fenster mit Rahmen etc. dekoriert und die *Menus* darstellt wird dabei als „***Window Manager***“ bezeichnet.

Anwenderprogramme, welche unter einer graphischen Oberfläche laufen resp. den Fensterinhalt liefern, greifen dabei nicht direkt auf die Grafikkarte zu sondern über das "***Graphische Subsystem***". Derzeit wird hierfür meist noch "***X11***" verwendet, also ein "***X-Server***" welcher das "***X-Window-System Release 11***" Protokoll implementiert. Auch hierfür gab es früher verschiedene Implementationen, mittlerweile wird jedoch nur noch die Implementation vom "***Xorg***" Projekt verwendet (vgl. → www.x.org)



Damit der X-Server die vorhandene Grafik-Hardware nutzen kann, benötigt dieser ein Grafikkarten-spezifischer „***X-Server Display Driver***“. Manche Firmen (insbes. *Nvidia)* bieten auch eigene optimierte „Closed Source Treiber“ an. Je nach Linux-Distribution müssen diese aber manuell nachinstalliert werden.

Eine Anwendung welche den *X-Server* Dienst nutzt (wie z.B. der *Dateimanager,* das „*Terminal-Window*“ Programm oder ein *Firefox* Browser) wird hingegen als "***X-Client***" bezeichnet.

Da die Programmierung von Anwendungen direkt auf dem X11-Programmierinterface (also unmittelbar auf dem **Xlib** API) sehr umständlich wäre, verwendet man zur Entwicklung von *X-Clients* üblicherweise eine komfortableres objektorientiertes **GUI-Framework** – was allgemein als „***Toolkit***“ oder "***X-Toolkit***" bezeichnet wird. Oft werden diese zusammen mit einer Desktop-Umgebung (weiter-)entwickelt, was eine einheitliche Darstellung und ein einheitliches Handling (z.B. von Drucker-Dialogen etc.) gewährleistet (s. https://de.wikipedia.org/wiki/Liste_von_GUI-Bibliotheken).
Für viele Desktop-Umgebungen (*GNOME, XFCE, LXDE, ...)* wird hierfür das *GNOME-Toolkit* ***GTK+*** eingesetzt, welches auch Teil des GNOME Projektes ist (vgl. http://de.wikipedia.org/wiki/GIMP-Toolkit), wogegen nur wenige *(*KDE und LXQt) auf ein anderes Toolkit, nämlich auf die ***Qt****-Libraries* aufbauen. Derartige Grafik-***Toolkits*** können auch beliebig parallel installiert und verwendet werden, weshalb z.B. KDE-Anwendungen nach Installation der *Qt*-Libraries auch unter einem *GTK+* basierenden Desktop laufen, denn beide verwenden darunter X11.

Um spezielle Desktop-Effekte welche der X-Server nicht direkt unterstützt (z.B. Transparenz, wackeln, skalieren oder verzerren der Fenster) zu realisieren, kann der X-Server die Ausgabe zusätzlich über einen so genannten ***Composting Window Manager*** leiten (z.B. *Mutter, Compiz, KWin*) .

Da ein immer grösserer Teil der „Rendering“ Aufgaben (wie Grafikelemente zeichnen, Schrift darstellen, Schattieren) mittlerweile in der Grafik-Karten Hardware realisiert werden kann, ist das X-Server-Konzept überladen und veraltet, weshalb gegemwärtig ein Übergang zum moderneren, schlankeren und damit schnelleren ***Wayland*** Konzept stattfindet (→ http://wayland.freedesktop.org/). Sobald dieser neue Grafik-Unterbau stabil läuft und die wichtigen Grafik *Toolkits* (*GTK+, Qt*) an das *Wayland* API angepasst sind, laufen auch alle

darauf basierenden Anwendungen und Desktop-Umgebungen unter *Wayland* Display. Um dann auch X-Clients zu unterstützen, welche direkt auf das Xlib API basieren, beinhaltet *Wayland* zudem auch eine direkte *X-Server* Emulation (*XWayland)*.

Beim installierten Ubuntu kann auf dem Anmeldebilschirm zwischen *Wayland* und *X11-Server* gewechselt werden (welcher Display Manager aktiv ist, werden Sie dann später im Versuch herausfinden).

Übrigens können sofern nötig und zugelassen *X-Clients* und *X-Server* auch auf verschiedenen System laufen - über eine TCP/IP-Netzwerkverbindung. Gegebenenfalls läuft dann der *X-Server* auf dem Client-PC (also dem Rechner mit dem Display) und die *X-Clients* (also das auszuführende Anwender-Programm) z.B. auf einem zentralen Applikations-Server ohne Display. Auch Windows kennt diese Möglichkeit und nennt dies „*Remote Desktop*“, wobei dann das *Remote-Desktop-Protokoll (RDP)* verwendet wird (nicht X11).

Nachbearbeitung der Installation

Loggen Sie sich mit *Ihrem* User-Account ein (also mit Ihrem Kürzel oder Vornamen als Benutzername).

Virtualbox „Gasterweiterungen“ installieren

Durch Installation der „*Virtualbox Gasterweiterungen*“ wird eine bessere Integration zwischen Host- und Gastsystem erreicht. Einerseits lässt sich damit die **Grösse des „Gastsystem Displays“ beliebig** wählen, andererseits ist mit diesen ein Zugriff auf „**Gemeinsame Order**“ möglich, d.h. vom Gastsystem aus kann damit auf ausgewählte Ordner des Hostsystems zugegriffen werden.

Da diese *Virtualbox Gasterweiterungen* jeweils auf dem Gastsystem compiliert werden müssen, benötigen Sie einige C-Compiler-Tools. Diese installieren Sie mittels eines "Metapackages" namens ***build-essential*** :

- Clicken Sie auf das Ubuntu Symbol (links oben) und tippen Sie dann „**Terminal“** ein…
  Nach starten des „Terminal“ Programms geben Sie im zugehörigen "Terminal-Window" folgendes ein:

  $ **sudo apt update**

  $ **sudo apt install build-essential**

- Installieren Sie nun folgendermassen die **Virtualbox-Gasterweiterungen** :
  - Clicken Sie im Virtualbox-Gastsystem Menubalken auf ***Geräte > Gasterweiterungen installieren***
    - Wenn das nicht klappt, führen Sie ***Geräte > Optische Laufwerke > Medium Entfernen*** aus und versuchen es nochmals!
  - Falls danach die Kompilation der Treiber Gasterweiterungen nicht automatisch startet (was dann Minutenlang dauert!)...
    → öffnen Sie im Gastsystem den Dateimanager (per Click auf den „*Aktenschrank*“)
    → clicken darauf in der linken Liste auf die virtuelle CD (namens „**VBOXADDITIONS“**)
    → Damit sollte die Installation der "Guest Addtions" starten. Wenn dies nicht automatisch erfolgt, müssen Sie „**autorun.sh“** aus der virtuellen CD ausführen – aber leider klappt dies meist nicht per einfachem Doppelclick, weshalb Sie dieses Programm wie folgt über ein „*Terminal Window*“ (resp. „Shell-Console“) starten müssen:
    → Clicken Sie hierzu auf das Ubuntu Symbol (links oben) und tippen Sie dann „**Terminal“** ein. Nach starten des „*Terminal*“ Programms, geben Sie im erscheinenden Fenster folgendes ein:
    $ **sh /media/***<Tab><Tab>/autorun.sh (wobei <Tab> die Tabulator-Taste links auf der Tastatur ist)*
    → Mit jedem <Tab> sollte automatisch eine Ordnerebene tiefer gewechselt werden un beim zweiten der VBOXADDITIONS-Ordner angezeigt werden. Geben Sie dann noch „autorun.sh“ ein worauf nach einem <Enter> eine Dialogbox zur Eingabe des Passwortes angezeigt werden sollte.
    → Geben Sie Ihr das Linux-Passwort ein und bestätigen Sie selbiges ...
  - Klappt es nicht oder dauert die Ausführung danach zu schnell (es dauert **minutenlang!**)
    hat die Installation wohl nicht geklappt! (Eventuell haben Sie das falsche Passwort angegeben?)
  - Nach erfolgreicher Installation der Gasterweiterungen **rebooten** Sie das Gastsystem …
    wonach sich einerseits die Grösse des Gastsystem-Fensters nun beliebig ändern lassen sollte und wie nachfolgend erklärt auch der Zugriff auf Ordner des Hostsystem möglich sein sollte….

Achtung: sollte später mal die Fenstergrösse des Gastsystem wieder unveränderbar und klein sein, so wurde höchstwahrscheinlich vorgängig eine Systemaktualisierung des Gastsystems durchgeführt, wobei der **Linux-Kernel** aktualisiert wurde (Package *linux-image-...*).
In diesem Fall müssen Sie die „Virtualbox-Gasterweiterungen" wieder wie oben neu installieren, denn damit werden die Linux-Kernel-Treiber der „Gastsystemerweituerungen" neu generiert, welcher für jede neue Kernel-Revision neu übersetzt werden müssen.

Virtualbox Shared Folders (Zugriff auf „Gemeinsame Ordner")

Ein Zugriff aus dem Linux-Gastsystem auf beliebige Ordner des Hostsystems (Windows, OS-X, …) setzt ebenfalls obige Installation der „Virtualbox Gasterweiterungen" voraus (vgl. vorheriges Kapitel).

- Danach im Virtualbox-Gastsystem-Menubalken auf ***Geräte > Gemeinsame Ordner*** clicken,
- sodann in der erscheinenden Dialogbox rechts auf das Icon  clicken ….
- und in der neu erscheinenden Dialogbox den freizugebenden Ordner-Pfad des Hostsystems auswählen (z.B. den „Downloads" Order des Hostsystems) sowie nach ev. Korrigieren des Ordner-Freigabenamens noch: ***[x] permanent erzeugen*** sowie **[x] automatisch einbinden** anwählen.
- Nach Bestätigen muss das Gastsystem nochmals rebootet werden, wonach der freigegebene Ordner im Dateimanager des Gastsystems in der Liste links unter ***sf_….*** sichtbar sein sollte (*sf = Shared Folder)*
- Möglicherweise ist der Ordner zwar sichtbar, aber auf den Inhalt kann mangels Zugriffsrecht noch nicht zugegriffen werden. Dies kann (nur) in einem *Terminal* Window durch zufügen der Sicherheitsgruppe ***vboxsf*** zur eigenen User-Id korrigiert werden, per:
  $ ***sudo usermod -a -G vboxsf $USER***
  Erst nach erneutem rebooten des Gastsystems sollte der Zugriff nun endlich auch unter Ihrer UserId klappen (denn Sicherheitsgruppen werden nur beim neu Einloggen in den Linux-Kernel eingelesen).

Administrationsrecht per 'sudo'

Zur Systemadministration müssen gelegentlich Programme (resp. „Commands") mit Systemadministrationsrecht gestartet werden – also unter «***root***-Recht» – denn der User mit uneingeschränktem Recht heisst auf allen POSIX-kompatiblen Systemen '**root**'. (Also auch unter Linux).
Je nach Linux-Distribution wird bei Systeminstallation das direkte Einloggen unter dem *root*-User durch Vergabe eines root-Passwortes ermöglicht. Alternativ wird die Installer dem bei der Systeminstallation erzeugten Benutzer das Recht geben, über das Programm '*sudo*' beliebige andere Programme unter «root-Recht» zu starten.

Beispiel: Oben wurde auf der Kommandozeile zuerst das Programm '*sudo*' gestartet, welches zuerst prüft, ob Ihre Benutzer-Id „*sudo*-berechtigt" ist (also Mitgliedschaft in der Sicherheitsgruppe '*sudo*' hat) wonach gegebenenfalls das dahinter angegebene Programm ('*usermod*') unter der User-Id *root* gestartet wird – das sicherheitskritische Programm '**sudo**' ist offensichtlich in der Lage, temporär die Benutzerrechte zu ändern!

Öffnen Sie ein *Terminal*-Window und geben Sie darin folgendes ein:

$ **id**

Dies listet nebst Ihrem Benutzername und Ihrer numerische User-Id (kurz *uid*) auch alle Gruppenmitgliedschaften Ihrer Benutzer-Id auf: u.a. auch '***sudo***' sowie die soeben zugewiesene Gruppe '***vboxsf***'.

Geben Sie nun '*id*' mit vorangestelltem '*sudo*' ein, also:

$ **sudo id**

Wodurch erkennbar ist, dass nun das Programm '*id*' tatsächlich unter der Id des Users *root* gestartet wurde. Der root-User hat normalerweise keine weiteren Gruppenmitgleidschaften – er brauch dies nicht, da er sowieso uneingeschränkte Rechte hat – und zwar aufgrund seiner numerischen User-ID = ...

Der „Command Prompt" (also was links unten auf der Eingabezeile des Terminals angezeigt wird) hat übrigens folgende Struktur: *<ihr-Benutzername>@<rechnername>:<ordnerpfad>**$***

Geben Sie nun folgendes ein:

$ **sudo su**

Wonach der Commandprompt in zweierlei Hinsicht wechselt: einerseits wird zuvorderst nun *root* angezeigt, andererseits endet der Prompt mit '#' statt '$' . Offensichtlich haben Sie nun eine „root-Shell“ offen, was durch die Raute '#' verdeutlicht wird! D.h. all nun gestarteten Programme werden mit root-Recht gestartet!

Es ist jedoch eine **schlechte Angewohnheit**, **permanent in einer „root-Shell“ zu arbeiten!**
Die Gefahr ist nämlich zu gross, dass Sie irgendwann mal durch einen Fehlgriff das Linux-System „abschiessen“, denn auf der Kommandozeile wird nicht dauernd nachgefragt „*ob Sie dies oder das wirklich durchführen oder löschen möchten“.* Besser als eine permanent offene root-Shell ist, nur wo wirklich nötig durch punktuelle Verwendung des Programms „**sudo**“ einzelne Commands unter root-Recht zu starten.

Wenn Sie *sudo* ausführen, wird zur Sicherheit Ihr Benutzer-Passwort abgefragt. Danach ist '*sudo*' in der betreffenden Shell-Console für einige Minuten „authentifiziert“, d.h. bei kurz aufeinander folgenden *sudo*-Aufrufen muss das Passwort nicht erneut eingeben werden [3].

- Schliessen Sie deshalb die offene „root-Shell“ z.B. per Eingabe des Commands '***exit***' .
- Statt die Verwendung des '*sudo*' Programmes könnte man wie erwähnt auch direkt unter dem *root* User einloggen sofern diesem ein gültiges Passwort vergeben wurde (z.B. per Programm '*passwd*'). Weshalb ist dies insbesondere bei Systemen mit mehreren Administratoren nicht sinnvoll?

Aktualisieren des Systems

Jedes System sollte zwecks „bug-fixing“ und aus Sicherheitsgründen regelmässig aktualisiert werden - beim installierten Ubuntu werden Sie wie bei *Windows* oder *OS-X* gelegentlich automatisch hierzu aufgefordert.

Manchmal ist es aber erwünscht, diese Systemaktualisierung manuell auszulösen, was wie folgt auf Console-Ebene, d.h. in einem Terminal-Window durchgeführt werden kann:

- Zuerst **aktualisieren** Sie die **lokale Paketliste** , denn um effizient Pakete zu installieren muss unsere Installation wissen, welches Softwarepackage in welcher Version auf dem Server vorhanden ist:

  $ **sudo apt update**
  Achtung: damit werden *nicht* die installierten Programmpakete selbst aktualisiert, sondern bloss die lokale Liste aller auf dem Ubuntu Server-Pool vorhandenen Programmpakete! Ein '*apt update*' sollte sowohl vor Installation neuer Software als auch vor jedem Update durchgeführt werden!

- Nach erfolgreicher[4] Aktualisierung dieser Paketliste können alle veralteten installierten Programmpakete wie folgt aktualisiert resp. „***upgraded***“ werden:

  $ **sudo apt upgrade**

Statt dem Programm ***apt*** könnte auch das Programme ***apt-get*** mit den gleichen Argumenten (*update, upgrade ...*) verwende werden – *apt* unterstützt jedoch auch Befehle wie '*apt search ...*' und '*apt show …*'

Beide *apt*-Programme wie auch die Paketmanager-Programme auf der graphischen Oberfläche von Ubuntu (namens „*Ubuntu-Software*“ und "*Aktualisierungsverwaltung*" etc. verwenden die gleiche lokale Paket-Datenbank.
Es spielt also keine Rolle mit welchem “*Packagemanager*“ aus dem Pool der Distribution Programme installiert, upgedated oder entfernt werden, das System behält über eine interne Datenbank immer die Übersicht und weiss jederzeit was woher und wo installiert wurde – solange die Installation aus dem Pool der Distribution erfolgte.

Potentiell problematisch sind bloss Programme, welche direkt via Internet-Browser heruntergeladen und

3 Die Passwort-Abfrage bei 'sudo' lässt sich auch ganz umgehen, indem in der Datei */etc/sudoers* die Zeile *%sudo ... zuhinterst* geändert wird auf: *%sudo ALL=(ALL:ALL) NOPASSWD:ALL* (Aber Achtung: bei Fehleingabe funktioniert '*sudo*' nicht mehr!)

4 Falls das „Aktualisieren der Paketliste“ nicht klappt, sind mögliche Fehlerquellen: keine Internet-Verbindung zum Server-Pool, eine zu grosse Abweichung der Systemzeit, oder eine falsche eingestellte lokale Zeitzone (!)

nicht über einen Packagemanager der Distribution installiert werden – dies sollte man wenn immer möglich vermeiden! Grund hierfür sind die "Abhängigkeiten", denn die meisten Open Source Programme benötigen selbst wiederum zahlreiche Libraries (Programmbibliotheken) in einer spezifischen Version, und diese werden in jeweils separaten Programm-Packages auf dem Server gehalten.
D.h. Sie sollten wenn immer möglich Programme nur aus dem Pool der Distribution über einen der obigen Packet Manager der Distribution installieren (in unserem Fall per 'apt install' oder "Ubuntu-Software") und nicht separat vom Internet herunterladen und manuell installieren!

Natürlich können Programme per '*apt*' auch wieder entfernt werden, entweder mit '**apt remove** *paketname*' oder wenn gleichzeitig auch die Programmeinstellungen entfernt werden sollen per '**apt purge** *paketname*'.

„Shell“ Kommandointerpreter

Vielleicht kennen Sie den „*Command Interpreter*“ von Windows, das Programm ***cmd.exe***. oder die leistungsfähigere "Power-Shell". Wie der Name sagt, ***interpretieren*** derartige Kommando-Interpreter die Benutzer-Eingaben auf der *„Kommandozeile“*.
Die auf der Kommandozeile eingegebenen „***Commands“*** können dabei Programmnamen sein, an welche optional auch „**Kommandozeilen-*Argumente“*** übergeben werden können (vgl z.B. 'apt update' wobei *apt* der Programmname ist und *update* das 1. Argument, welches an das Programm *apt* übergeben wird).
Nebst Starten von beliebigen Programmen beinhaltet ein Kommando-Interpreter auch interne Commands sowie Commands zur Ablaufsteuerung wie *if..., for..., while ...*

Unter POSIX-kompatiblen Systemen (wie Linux oder OS-X) wird der Kommando-Interpreter als „***Shell“*** bezeichnet, in Analogie zu einer Muschel, welche eine Perle umschliesst - die Perle ist der Betriebsystem-Kern (Linux).

Wird eine Abfolge mehrerer Commands in eine Datei geschrieben, spricht man unter Windows von einem „Batchfile“ (zu deutsch „Stapeldatei“), unter POSIX-Systemen hingegen von einem „***Shell script****“*.

Im UNIX- resp. POSIX-Bereich gibt es verschiedene Shell-Programme (*sh, bash, ksh, csh, ash*...) mit teilweise leicht unterschiedlichem internen „Sprachumfang“ und Syntax. Die meisten Shells sind jedoch POSIX-Konform, d.h. zur ursprünglichen **UNIX Bourne-Shell** (***'sh'***) kompatibel, denn dieser Shell-Sprachumfang ist in einem der POSIX-Standards definiert.

Unter *Linux* und OS-X wird als interaktive Shell (in einem „Terminal“ Window) üblicherweise die '***bash'*** Shell verwendet (eine Open Source aus dem GNU-Projekt und ebenfalls POSIX-konform) wogegen für Shellscripts meist die einfache '***sh'*** Shell verwendet wird.
(vgl. http://en.wikipedia.org/wiki/POSIX, http://www.gnu.org/software/bash/)

Basic File Sytem Commands

- Starten Sie auf der graphischen Oberfläche ein Shell-Window („*Terminal“)* worin automatisch eine ***bash****-Shell* gestartet wird, welche die von Ihnen eingegebenen Kommandozeilen interpretiert...
- Generell unterscheidet man zwischen Command-Argumenten und Command-Optionen. Letztere sind erkennbar an einem oder zwei '-' vor der Option. Zuerst einige wenige Dateisystem-Commands.
  Die meisten Commands zeigen eine Kurzhilfe und zulässige Command-Optionen an, wenn das betreffende Programm mittels Option '***--help'*** aufgerufen wird (versuchen Sie's!). Die vollständige Hilfe erhalten Sie hingegen durch Aufruf von '***man*** *command* ***'*** , also z.B. '***man ls'*** . Dies klappt jedoch nicht bei den direkt in der Shell eingebauten, so genannten „intrinsic" Commands, deren ausführliche Hilfe nur mit '***man bash'*** angezeigt wird.

cd / .....

ls .....

ls *-a* .....

ls *-l* .....

cd .....

mkdir xxx .....

cd xxx .....

pwd .....

cd .. ..

rmdir xxx .....

- Achtung: Mit ***rmdir*** können nur leere Verzeichnisse gelöscht werden! Um ein Verzeichnis *rekursiv* samt Inhalt und allfälligen Unterverzeichnissen zu löschen, verwenden Sie ***'rm -r** ordnerpfad' 'rm -r'* ist deshalb ein recht gefährliches Kommando zumal es auf Shellebene weder ein „Papierkorb“ noch eine Nachfrage gibt, ob Sie die Aktion tatsächlich ausführt werden soll!

Linux kennt wie alle POSIX-kompatiblen Systeme im Gegensatz zu Windows **keine Laufwerks-buchstaben (**C: , D:, ...) sondern nur ***eine*** **Ordner- resp. Verzeichnis-Hierarchie** .

Verschiedene Laufwerke oder auch Freigaben können an beliebiger Stelle in diese eingebunden werden.

Beginnen Pfadangaben mit einem **Slash** '/', handelt es sich um einen **„absoluten“ Pfad** (z.B. */xxx/yy/zz)* Absolute Pfade beziehen sich auf die **oberste Verzeichnisebene**, dem so genannten „*Root-Verzeichnis“ (also der Wurzel des Verzeichnisbaumes*) und ein Slash '/' ohne weitere Angabe ist folglich das oberste Verzeichnis in der Hierarchie, das so genannte root-Verzeichnis.
Ohne einen führenden Slash sind Pfadangaben hingegen „***relativ“*** auf das **gerade *aktive* Verzeichnis** bezogen (z.B. *xxx/yy/zz)*.

Der Begriff „*root*“ ist unter Linux/Unix somit doppeldeutig: einerseits bezeichnet „*root*“ ja jenen Benutzer mit uneingeschränktem Recht, andererseits wird die oberste Verzeichnisebene als *root* Verzeichnis resp. *root directory* bezeichnet.

Ein weiterer wichtiger Ordner in der Verzeichnis-Hierarchie ist das so genannte *Homeverzeichnis* (engl. *home directory*). Alle Homeverzeichnisse sind bei POSIX-Systemen üblicherweise unter dem Verzeichnis */home* und haben den Namen der betreffenden Benutzer-Id.

Auf Shell-Ebene kann der Pfad zum eigenen Homeverzeichnis auch mit dem Tilde-Zeichen (~) angegeben werden, denn die Shell ersetzt dieses Zeichen durch den Pfad des Homeverzeichnisses.

– Mit welchen vier '***cd***' Varianten können Sie aus einem aktiven Verzeichnis **~/xxx** zurück ins Home-Verzeichnis wechseln?

.....

– Wo in der Verzeichnis-Hierarchie wurde der „Shared Folder“ des Hostsystems eingebunden? .....

Prozesse auflisten

Mittels Command '***ps***' lassen sich Prozesse auflisten. Standardmässig werden bloss die Prozesse aufgelistet, deren (Text-)Ausgabe im gleichen Terminal Window erfolgt...

– Versuchen Sie's und studieren Sie die Ausgabe!

Jedem neu gestartete Prozess (resp. Programm) wird zur Identifizierung eine ***Process Id*** (**PID**) zugeteilt.

Der Prozess, welcher einen neuen (Child-)Prozess gestartet hat wird „Parent Process“ genannt, dssen Id wird als **PPID** angezeigt (Parent Process ID).

– Geben Sie nun '***ps -ef***' ein, womit einerseits aufgrund von Option *-e alle* auf dem System laufenden Prozesse dargestellt werden, andererseits aufgrund von Option *-f* detailliertere Information aufgelistet wird – u.a. die UserId unter welche das Programm läuft und in der dritten Spalte die *PPID* etc.

– Von welchem Programm wurde das '*ps*' Programm gestartet? .....

– Und von welchem Programm selbiges? .....

Commands lassen sich auch kombinieren, z.B. den Output des einen Commands in den Input eines weiteren Commands leiten! Wie folgt kann z.B. der Output von '*ps -ef*' per '*grep*' Command nach einem grossen 'X' durchsucht werden, womit erkennbar wird, welches Grafische Subsystem läuft:

$ **ps -ef | grep X** → aktuell läuft als Grafisches Subsystem: .....

Bash Command Completition und Command History

- Wird ein Command nur teilweise eingegeben und die <Tab>-Taste gedrückt, wird der Command oder Ordner soweit eindeutig vervollständigt oder durch nochmaliges Drücken der <Tab> Taste alle Möglichkeiten angezeigt (die bash Shell versucht auch Commandline-Argumente sinnvoll zu ergänzen).
  Beispiel: ***ls<Tab><Tab>*** → zeigt alle Commands welche mit „ls“ beginnen.
- Zuvor in einer Shell ausgeführte Commands können einfach per Pfeil-Taste ↑ wieder geholt werden.
- Die gesamte *Command-History* lässt sich mittels Shell-Kommando '***history***' anzeigen.
- Möchten Sie hingegen ein früher eingegebenes Kommando gezielt in der Shell-History suchen, dann tippen Sie auf der leeren Kommandozeile zuerst ***Ctrl-R*** und geben darauf eine beliebige Such-Zeichenfolge ein welche in der gesuchten Commandozeile enthalten war. Die Suche erfolgt dabei inkrementell rückwärts in der Command-History.
  Mit ***Enter*** übernehmen Sie eine vorgeschlagene/angezeigte Kommandozeile und führen diese aus, wogegen ein wiederholtes ***Ctrl-R*** immer weiter rückwärts in der Command-History sucht.
  Betätigen Sie die Anzeige statt mit *Enter* mit einer der horizontalen Pfeiltasten ← →, kann der gerade angezeigte Eintrag vor Ausführung noch bearbeitet werden. Per ***Ctrl-C*** ist hingegen jederzeit ein Abbruch möglich.
- *Copy&Paste* funktioniert in einem Terminal-Window entweder nach markieren per Maus-links per Popup-Menu (Maus-Rechts „Kopieren“ / „Einfügen“) oder alternativ per ***Shift-Ctrl-C*** und ***Shift-Ctrl-V*** .
  (Ctrl-C / Ctrl-V funktioniert hingegen nicht, da die Shell Ctrl-C wie vorher erwähnt als „Terminate“ Signal interpretiert).

Virtualbox und USB 2.0 / USB 3.0

- Damit auch USB 2.0 und USB 3.0 aus dem Gastsystem funktioniert, kann das „**Virtualbox Extension Pack“** nachinstalliert werden. Laden Sie dieses Package hierzu von der Virtualbox Downloadpage auf das Hostsystem (*Windows* oder *OS-X*) herunter, also von www.virtualbox.org/wiki/Downloads ...
- Und installieren Sie dieses in der ***Virtualbox Management Console*** per:
  Datei > Einstellungen > Zusatzpakete …

Zugriff auf Windows-Netzwerkfreigaben

Wenn Sie Linux als Gastsystem verwenden, können Sie Zugriffe auf Netzwerkfreigaben der FHNW wie oben beschrieben indirekt über das Hostsystem und dann über einen „Gemeinsamen Ordner“ der Gastsystemerweiterungen realisieren.

Zugriffe auf Windows Dateifreigaben erfolgen über das Netzwerkprotokoll ***SMB*** *(Server Message Block)*.
Linux unterstützt auch dieses Windows-Protokoll direkt. Die "Active Directory" resp. „Distributed File System“ Freigaben der FHNW entsprechend dem Netzwerklaufwerk 'S:' haben unter Windows den Freigabepfad:
\\fs.edu.ds.fhnw.ch\data\
Um diese auf dem Linux Desktop direkt anzuzeigen ...

- Starten Sie den Linux Dateimanager per Click auf das **Dateimanager-Icon**
- geben in diesem ***Ctrl-L*** ein, wodurch eine **Adresszeile** im Dateimanager angezeigt werden sollte.
- Auf der Adresszeile geben Sie den Freigabepfad in der Form ***smb://Server/Freigabe*** ein, also
  smb://fs.edu.ds.fhnw.ch/data/ ***Achtung:*** *unter Linux Vorwärts-slashs statt Back-slashes wie unter Windows!*
- Worauf eine Dialogbox mit Benutzername, Domäne und Passwort Abfrage angezeigt werden sollte.
  - ✗ Ist dies nicht der Fall, liegt ein Verbindungsproblem vor: auf dem Hostsystem musste der Zugriff auf das AD schon vor Start des Gastsystems möglich sein. Sollte dies nicht der Fall sein, aktivieren Sie auf dem Hostsystem den Zugriff aufs AD (→ VPN verbinden) und deaktivieren/ aktivieren darauf im Gastsystem das Netzwerk oder rebooten selbiges.
  - ✗ Klappe es dann immer noch nicht, fehlt ev. das Package '**gvfs-backend**' (sollte unter Ubuntu vorinstalliert sein).
- Als Benutzernamen müssen Sie Ihr FHNW-Benutzername angeben (*vorname.name*) sowie **als Domäne: EDU** und natürlich Ihr FHNW-Passwort .
- Sobald die Freigabe erfolgreich öffnet, können Sie zwecks späterem Schnellzugriff per Ctrl-D ein '**Lesezeichen**' für diese Freigabe oder für beliebige Unterordner anlegen.
- Alternativ zur obigen „Distributed Filesystem Freigabe“ ist auch ein direkter Zugriff auf FHNW-Server möglich, wodurch der Verbindungsaufbau etwas schneller erfolgt. Der Dateiserver für die Domäne EDU, auf welche Studierende unserer Hochschule Zugriff haben ist unter Windows auf:
  *\\fsemu18.edu.ds.fhnw.ch\e_18_data11$* resp. unter Linux auf: ***smb://fsemu18.edu.ds.fhnw.ch/e_18_data11$***
- Ihre eigenen Benutzerdaten sind entsprechend unter: *smb://fsemu18.edu.ds.fhnw.ch/e_18_home11$/vorname.name*

Anhang (nur für nativ installierte Linux-Systeme)

WLAN und VPN-Verbindungen unter Linux

- Das Einrichten eines Wireless-Netzwerks geschieht bequem mittels „*Network-Manager"* per Click oben rechts auf das Netzwerk-Icon.
  An der FHNW können Sie entweder ohne Authentifikation über das WLAN „*fhnw-public*" verbinden, wonach Sie sich über den Browser authentifizieren oder das VPN verbinden müssen (s. unten). Alternativ könnten Sie auch über das WLAN „*eduram*" über das 802.1x-Protokoll Authentifizieren, wozu in den Netzwerkeinstellungen u.a. „geschütztes EAP (PEAP)" und MSCHAPv2 gewählt werden muss…
- Zwecks VPN-Zugriff unterstützt die FHNW das Cisco VPN Protokoll „*AnyConnect*" (ein **SSL**-basiertes VPN-Protokoll), wozu es auch einen Cisco Any-Connect Client gibt. [5]

Drucken unter Linux

Als Drucksystem wird auf Linux-Clients üblicherweise das ursprünglich von Apple entwickelte „CUPS" verwendet. Per CUPS kann man auch auf Windows-Druckserver zugreifen.
Leider erlaubt die FHNW-Systemadministration aber nur noch den Zugriff von Windows und OS-X Sytemen über entsprechend zertifizierte Canon-Druckertreiber, weshalb der direkte Zugriff auf FHNW-Drucker für Linux-User nicht mehr möglich ist.
Alternative: via Mailprint, USB-Stick oder über ein Windows-Gastsystem drucken!

5 Leider ist aufgrund der 2-Faktor-Authentifizierung das Verbinden mit dem FHNW VPN mittels der Open source '*openconnect*' nicht mehr einfach möglich.