Linux-Seminar

Linux Administration I (ADM1)

Dr.sc.nat. Michael J.M. Wagner, c//m//t*

Revision 1.52

*michael@wagnertech.de

Inhaltsverzeichnis

1 Einführung in Linux

1.1 Von UNIX zu Linux

Was ist ein Betriebssystem?

- Das grundlegende Computerprogramm
- steuert Hardware, koordiniert Ressourcenzugriffe
- Prozessmanagement (Laufzeit)
- Speichermanagement
- Steuerung und Abstraktion der Hardware (Gerätetreiber)
- Dateiverwaltung
  - Festplatten
  - CD
  - DVD
  - Halbleiterspeicher (nicht flüchtig)
  - ...
- Eingabemöglichkeiten für den Benutzer
  - Kommandozeile
  - GUI

Geschichte der Betriebssysteme

- Erst mit interaktiver Benutzung der Rechner nötig
- Lochkartenzeitalter (50/60'ger Jahre)
  - Entwickler geben Stapel von Lochkarten ab, erhalten Ergebnis am nächsten Tag
  - „Stapelverarbeitung" – „batch"
  - Häufig benötigte Programmteile (z.B. Interpreter, Compiler) stehen auf Magnetbändern zur Verfügung.
  - Operator muss diese Ressourcen vor der Stapelverarbeitung laden
  - Programme müssen diese Ressourcen anfordern -> „Vorlaufkarte"
  - Interpreter dieser Vorlaufkarten sind erste Formen eines Betriebssystems (operating system übernimmt Aufgaben des Operators)
- Terminals (Ende 60'ger Jahre)
  - Das interaktive Terminal verändert die Computerwelt
  - Ein-/Ausgabegerät am Arbeitsplatz

  - *time sharing operating system* (TSO)
    - *scheduler*
    - Kontrolle von Zugriffsrechten
- Im IBM-Betriebssystem MVS gibt es die Begriffe heute noch
- Massachusetts Institute of Technology (MIT) arbeitet am "Incompatible Timesharing System" ITS, einem UNIX-Vorläufer
- Kommerzielle Variante: MULTICS (*Multiplxed Information and Computing System*, AT&T, Bell Labs, hatte nie eine Verbreitung)

UNIX (70'ger Jahre bis heute)

- Aus der MULTICS-Entwicklung wird eine Einzelplatzversion abgespalten: UNICS
  - Mehrbenutzer-Fähigkeit wird bald nachgerüstet
  - Modularer Aufbau: Jeder Befehl kann einzeln ausgetauscht werden
  - Erste Implementierung in der Maschinensprache der PDP-7
  - Erster C-Compiler (Kernighan, Ritchie) führt zur raschen Verbreitung
- Bell Labs dürfen UNIX, wie es bald hieß, aus kartellrechtlichen Gründen nicht kommerziell vertreiben
  - Preisgünstige Abgabe an Hochschulen
- Berkley University entwickelt UNIX weiter und gibt die Berkley System Distribution (BSD) heraus
  - In den 80'gern erhalten AT&T / Bell Labs die Genehmigung ein Betriebssystem zu vertreiben und entwickeln das AT&T System V, was sich vom BSD deutlich unterschied.
  - Heutige UNIXe stellen meist eine BSD/System V-Mischung dar
  - UNIX-artige Betriebssysteme heute:
    - Sun Solaris
    - IBM AIX
    - HP UX
    - Linux
    - FreeBSD
    - Mac OS X
    - Android

  - Gemeinsamer Standard: POSIX (portable operating system interface) stellt wechselseitige Compilierbarkeit sicher
  - Warum gibt es heute noch UNIX?
    * Internet hat UNIX-Konzepte übernommen
    * Linux – open source-Szene

Open Source / freie Software

- Free Software Foundation (FSF, 1984): „frei“ wie in „Freiheit“, nicht wie „Freibier“
- GNU-Projekt (Richard Stallman, 1983)
  - freie UNIX-Tools (gcc, ...)
  - GNU General Public License (GPL)
    * freie Verwendung
    * freie Verbreitung
    * frei für Anpassungen
    * Anpassungen müssen/dürfen nur unter GPL weitergegeben werden
    * *Copyleft*: Diese einmal gewährten Rechte dürfen nicht wieder zurückgenommen werden
  - Projekte unter GPL
    * Linux
    * mySQL
    * gcc
    * ...
- BSD-Lizenz
  - weniger weitreichend
  - verzichtet auf *Copyleft*
  - ermöglicht das Einbinden von BSD-Software in kommerzielle Projekte
- Apache-Lizenz: „irgendwo dazwischen“
- Bemerkungen
  - Um sich von dem elitären Freiheitsbegriff der FSF abzugrenzen, etablierte sich der allgemeinere Begriff *open source software*
  - Linux steht zwar wegen der Einbindung vieler GNU-Tools unter GPL. Die Linux-Gemeinde ist im Allgemeinen weniger ideologisch und hat keinerlei Berührungsängste mit anderen Lizenzformen. Der Name „GNU/Linux“ konnte sich daher nicht durchsetzen.
  - Diese Verschiedenheit in den Grundsätzen zieht sich heute durch die Linux-Distributionen.

- Beispiele für open source software
  - LibreOffice (Basis: StarOffice/Openoffice von Star-Division/Sun/Oracle)
  - Mozilla-Suite (Netscape)
  - Eclipse (Basis: Visual Age von IBM)
  - MaxDB (Basis: SAP DB)

Die Entwicklung von Linux

- Situation 1991
  - x386 verfügbar (1985 - 2007)
    - 32 Bit Daten-, Adressbreite
    - *task switching*
    - „Space-Shuttle-Prozessor"
  - DOS 5.x
  - Windows 3.x
  - Minix, eine zu Lehrzwecken abgespeckte UNIX-Variante
  - GNU tools ohne passenden Kernel
- Ankündigung in der Minix-Newsgroup (25.8.1991):

  Hello everybody out there using minix – I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file system (due to practical reasons) among other things). I've currently ported bash (1.08) and gcc (1.40), and things seem to work. This implies that I'll get something practical within a few month, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

  Linus (torvalds@kruuna.helsinki.fi)

  PS. Yes – it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(

- Die Idee: Neuer Kernel + GNU-Tools = Neues OS
- Name des neuen OS: freax (free / freak UNIX)
- Durchgesetzt hat sich Linus' UNIX: Linux
- Tux, das Maskottchen (1996)
  - Tux: Torvalds' UNIX
  - Tux: Abk. f. *tuxedo* (Smoking)

Abbildung 1: TUX, das Linux-Maskottchen {wiki}

Linux heute

- wird auf der ganzen Welt weiterentwickelt
  - von Unternehmen (z.B. Google)
  - von Non-Profit-Organisationen
  - von Einzelpersonen
- Einsatzbereich
  - Server (vom Router bis zum Supercomputer)
    - Modularität ermöglicht schlanke, dezidierte Server
    - Marktanteil zwischen 1/3 und 1/2 (je nach Schätzung)[1]
    - Linux – Apache – mySQL – PHP
    - SAP R/3 - Installationen
  - Desktop
  - Embedded (Mobiltelefone, Switch)
- .[Kofler 2021, Kap. 1.5]

Distributionen

- Anfangs „Linux für Informatik-Studenten“
  - „Was für Bastler“
  - erste Studenten-Distributionen auf CD mit Installationsprogramm
- Heute gibt es stabile Distributionen mit Langzeitsupport (LTS)
- Wichtige Distributionen
  - Slackware (1993)
    - älteste, heute noch gepflegte Distribution
    - baute auf der ältesten Linux-Distribution überhaupt auf: „Softlanding Linux System“, SLS)

[1] https://de.wikipedia.org/wiki/Linux-Einsatzbereiche#Marktanteile_2 (13.3.2019)

        * „spartanisch“
    - RedHat Linux (in USA verbreitet)
        * zwei Linien: RedHat (business), Fedora (privat)
        * Schöpfer des RedHat Package Managers (RPM)
    - Debian GNU/Linux
        * nur SW unter GPL, auch das Installationsprogramm
        * Stabilität vor Aktualität
        * Mutter von Ubuntu, Knoppix, Mint, ...
    - Mandriva Linux
        * französisch/brasilianische Distribution
        * umfangreich
    - SuSE Linux (1994)
        * Software und Systementwicklung GmbH, Nürnberg 1992
        * großer Lieferumfang
        * „Windows aus Nürnberg“
        * 2001 SUSE Linux Enterprise Server (SLES)
        * 2003 Übernahme durch Novell
        * 2005 Umbenennung der „freien Linie“ in openSUSE
        * 2014 Übernahme durch MicroFocus
        * seit 2019 wieder selbständig
- Übersicht über verfügbare Distributionen: {distro}
- Kompatibilität
    - :-(
    - Linux Foundation soll Situation verbessern
    - Linux Standard Base, LSB
    - wird von den Distributionen unterschiedlich stark beherzigt
- .[Kofler 2021, S. 29-31]

Softwarepakete werden je nach Distribution in verschiedenen Formaten zur Verfügung gestellt:

- RPM: redhat package manager
- DEB: debian package manager
- SLP: slackware package manager
- TGZ: einfache Dateiarchieve

- FLATPAC: Anwendungen in abgeschotteter Umgebung[2]

Aufgabe:
Ausflug nach http://distrowatch.com

1.2 Elementare Bedienung

Ein UNIX-System wird traditionell über die Kommandozeile bedient. Auf Desktopsystemen ist dazu parallel eine graphische Bedienung möglich.

Um auf einem UNIX-System ohne graphische Oberfläche Befehle ausführen zu können, müssen diese „irgendwo“ eingegeben werden. Dieses „irgendwo“ ist die *shell*, die Kommandozeile.

- Warum shell?
  - Bei UNIX/Linux-Systemen ist in unteren run leveln keine grafische Benutzeroberfläche verfügbar.
  - Wiederkehrende Wartungsaufgaben (wie Backup) lassen sich über *shell scripts* automatisieren (s. Kap. *Scripting*)
- UNIX-Shells:
  - sh (Bourne Shell): „Kleinster gemeinsamer Nenner“
  - csh (C-Shell): Enthält spezielle Funktionen für C-Entwickler
  - bash (Bourne Again Shell): GNU-Weiterentwichlung von sh
  - ksh (Korn Shell): Kommerzielle Weiterentwicklung von sh
- Eingabevervollständigung (TAB)
  - Bei mehreren Möglichkeiten erfolgt bei 2xTAB eine Auflistung der Möglichkeiten
- Vorangegangene Befehle können über die Pfeiltasten zurückgeholt werden.
  - Mit Ctrl+r lässt sich in den vorangegangenen Befehlen suchen
- Shell-Copy-Paste
  - Mit der Maus markieren und mit mittlerer Maustaste/linke+rechte Maustaste einfügen
  - In der Windows-Powershell markieren Sie den gewünschten Text und drücken 2x die rechte Maustaste.

[2] Linux Magazin 02/18, S. 8.

Arbeiten als root

Die administrative Tätigkeit erfordert es oftmals mit root-Rechten zu arbeiten. Früher war es dazu üblich, sich mit dem Befehl `su -`[3] einen Rootprompt zu erzeugen. Da offenstehende Rootshells ein Sicherheitsrisiko darstellen, zudem bei mehreren Administratoren das Rootpasswort geteilt werden musste, ist es heute üblich bei Befehlen, die Rootrechte benötigen `sudo` voran zu stellen. In der Datei `/etc/sudoers` ist gestgelegt, wer das darf. Standardmäßig darf das jedes Mitglied der Gruppe `sudo`, dabei wird das Benutzerpasswort abgefragt.

Soll (beispielsweise auf Testsystemen) die Abfrage des Benutzerpassworts ausbleiben, kann der entsprechende Eintrag in der Datei `/etc/sudoers` so abgeändert werden:
`%sudo ALL=(ALL:ALL)NOPASSWD:ALL`
Editiert werden kann die Datei mit dem Befehl `sudo visudo`.

[Kofler 2021, Kap. 12.2, 12.3]

Diverse Kommandos

- `cp <quelle> <ziel>` Kopieren von Dateien
- `mv <quelle> <ziel>` Verschieben von Dateien
- `rm <Datei(en)>` Löschen von Dateien
- `rm -r <Datei(en)/Verzeichnisse>` Rekursives Löschen von Dateien und Verzeichnissen
- `ls` Inhalt des aktuellen Verzeichnisses anschauen
  - `ls -l`: Langform
  - `ls -a`: Auch versteckte Dateien/Verzeichnisse (beginnen mit Punkt)
  - `ls -d`: Verzeichnisnamen statt Inhalt eines Verzeichnisses

  Die Optionen können auch kombiniert werden: `ls -la`
- `cd` Verzeichnis wechseln
- `mkdir` Verzeichnis erstellen
- `rmdir` (Leeres) Verzeichnis löschen
- `chmod` Rechte einer Datei ändern
- `echo <text>`: Gibt <text> aus
- `pwd`: *print working directory*
- `touch <datei>` aktualisiert den Zeitstempel der letzten Änderung einer bestehenden Datei. Ist die genannte Datei nicht vorhanden, wird eine leere Datei dieses Namens angelegt.
- `exit` oder `^D` zum Verlassen der *shell*

[3] Der Strich bewirkt, dass nicht nur UID/GID verändert werden, sondern auch die Loginskripte ausgeführt werden, was zur Erstellung der Pfadvariable wichtig ist.

2 Systeminstallation

2.1 Basisinstallation

Die verschiedenen Linuxdistributionen lassen sich heute über komfortable Installationsprogramme installieren. Grundsätzlich lassen sich dabei folgende Varianten unterscheiden:

- Reine Live-Systeme (z.B. Knoppix)
- Live-Systeme mit Installationsmöglichkeit (z.B. Linux-Mint)
- Systeme ohne Live-System (z.B. Debian-Linux)

Daneben gibt es noch die Unterscheidung zwischen Server- und Desktop-Systemen. Wichtige Serversysteme sind Server von Ubuntu und Debian.

Durch die Einführung von UEFI und *Secure Boot* ist die Installation etwas komplizierter geworden. Gundsätzlich gilt:

- UEFI verwenden, zusammen mit einer GPT-Partitionierung
- *Secure Boot* ausschalten: Dazu müsste ein Schlüssel des Bootloaders auf dem EPROM eingetragen sein. Nachdem es aber bei Linux nicht *den* Hersteller gibt, der diesen Schlüssel verwalten könnte (wie bei MS), ist es schwierig mit *Secure Boot*.

.[Kofler 2021, Kap. 2.2]

Aufgabe:

Erstellen Sie eine virtuelle Maschine mit einem der folgenden Systemen:

- Linux-Mint (Desktop)
- Debian-Server
- Ubuntu-Server

Dazu sind im VirtualBox-Manager folgende Schritte auszuführen:

- Neue Maschine anlegen
- Betriebsystem angeben, Speicher wählen, Festplatte (VDI) erzeugen.
- Maschine starten. Der Wizard fragt dann nach dem zu installierenden System (iso-Datei).

Loggt man sich auf Serversystemen ein, ist es oft von Interesse, vor welchem System man gerade sitzt. Die Information dazu findet sich in den Dateien `/etc/issue` und `/etc/os-release`.

Aufgabe:

Betrachten Sie diese Dateien.

Für den Bootprozess lassen sich Parameter mitgeben.[4]

- Ad hoc-Parameter im Bootmenu
  - Im Bootmenu `e` auf der gewünschten Konfiguration wählen
  - Zeile `linux ...` editieren
- Permanent Parameter ändern
  - In `/etc/default/grub` `GRUB_CMDLINE_LINUX_DEFAULT` anpassen

Typische Ubuntu-Bootparameter finden Sie hier.[5]

2.2 apt

Mit `apt` lässt sich das Debian-Paketsystem bedienen. Nach dem Kommando `apt` kommt der Befehl, den `apt` ausführen soll:

- `install`: Genanntes Paket aus dem Repository samt seiner Abhängigkeiten installieren
- `install -f`: Installiert „alles, was hängen geblieben ist"
- `remove`: Paket deinstallieren
- `purge`: Paket komplett entfernen

Weitere Befehle:

- `apt-cache showpkg PAKET` liefert Paketinformationen
- `apt-cache policy PAKET` zeigt Installationskandidaten (mit Version)
- `apt-cache search MUSTER` sucht nach Paketen, die dem `MUSTER` entsprechen
- `apt-mark hold PAKET`: Schließt Paket von der Aktualisierung aus

`apt` holt die Paketinformationen und Pakete aus den Debian-Reporitories, die in `/etc/apt/sources.list`, sowie den Dateien unter `/etc/apt/sources.list.d/` definiert sind. In diesen Dateien gilt für eine Paketquelle folgendes Format:

```
deb <host-url> <release> <component> [component1, ...]
```

Über diese Angabe kann der Paketmanager Informationen über vorhandene Pakete und deren Abhängigkeiten vom Repository laden. Dies sollte von Zeit zu Zeit vorgenommen werden. Dies erfolgt über `apt-get update` oder den entsprechenden Menupunkt im `aptitude`. Dazu ist es meist nötig den Schlüssel zur verschlüsselten Übertragung zu importieren:

- Schlüssel herunterladen: `wget SCHLUSSEL_URL`
- Schlüssel importieren: `apt-key add SCHLUSSEL_DATEI`

[4] https://wiki.ubuntu.com/Kernel/KernelBootParameters
[5] https://wiki.ubuntuusers.de/Bootoptionen/

.[Kofler 2021, Kap. 20.6]

Aufgabe:

- Fügen das Repository `http://wagnertech.de/debian stable main` mit dem Schlüssel `http://wagnertech.de/debian/conf/wagnertech.key` hinzu.
- Aktualisieren Sie die Paketinformationen des Systems.
- Installieren Sie das Paket `mbuild`

2.3 dpkg

Mit `dpkg` lassen sich weitere Informationen zu Paketen gewinnen oder Pakete, die als Datei vorliegen, installieren:

- `dpkg -i <deb>`: Installiert das als Datei vorliegene Paket
- `dpkg-deb -I <deb>`: Liefert die Paketinformationen zu einem als Datei vorliegenen Paket
- `dpkg-deb -c <deb>`: Zeigt die in einem als Datei vorliegenen Paket enthaltenen Dateien
- `dpkg -S <datei>`: Zeigt das zu einer auf dem Server befindlichen Datei mit welchem Paket sie installiert wurde
- `dpkg -l`: Zeigt alle installierten Pakete
- `dpkg -L <pkg>`: Zeigt alle zu einem installierten Paket gehörigen Dateien
- `dpkg-reconfigure <pkg>`: Führt die Installationsroutinen (`postinst`, ...) nochmal aus

.[Kofler 2021, Kap. 20.5]

Aufgaben:

Installation/Deinstallation mit `apt`

- Welche Dateien gehören zum Paket `apache2`? Schreiben Sie das Ergebnis in eine Datei.
- Deinstallieren Sie `apache2` mit `apt remove`.
- Überprüfen Sie erneut, welche Dateien zu diesem Paket auf dem System vorhanden sind.
- Löschen Sie nun mit `apt-get purge` auch die Konfigurationsdateien und überprüfen Sie das Ergebnis.

2.4 Software-Update

Zu einem Software-Update gehören folgende Schritte:

- Mit `apt update` werden die Paketinformationen aus den Paketquellen neu geholt
- `apt upgrade` aktualisiert das System. Dabei werden keinesfalls bestehende Pakte entfernt.
- `apt dist-upgrade` aktualisiert das System. Gegebenenfalls werden auch bestehende Pakte entfernt.

Dieses Vorgehen aktualisert alle aktualisierbaren Pakete. Will man, gerade bei einem automatisierten Aktualisierung, den Umfang einschränken, bietet sich die Funktionalität *unattended upgrade* an. Hier werden nur ausgewählte Paketquellen für den Upgrade berücksichtigt. Die Auswahl erfolgt in der Datei `/etc/apt/apt.conf.d/50unattended-upgrades`, eingeschaltet wir die Aktualisierung in `20auto-upgrades`. Die Frequenz bestimmt der Timer des `systemd` (s. Kap. **??**).

Manuell kann die Aktualisierung über `sudo unattended-upgrade -d`

Aufgabe:

- Führen Sie einen Software-Update Ihres Systems durch
- Überprüfen Sie die Systemdateien für den *unattended upgrade*.
- Führen Sie einen *unattended upgrade* druch.

Zur Bedienung weiterer Paketmanager: [Kofler 2021, Kap. 20.1-4,11]

2.5 Paketerstellung

Grundlagen

Debian-Pakete bestehen aus zwei Teilen, einem Verwaltungsteil und dem Paketinhalt. Zur Erstellung müssen Dateien in folgender Weise in einem Verzeichnis `$VERZ` angeordnet werden:

```
DEBIAN/
  control
  postinst
usr/
  bin/
    mein-prog
```

Der Befehl `fakeroot dpkg-deb --build $VERZ` baut dann ein Debian-Paket

Aufgabe:

- Installieren Sie `mbuild` aus der Paketquelle `http://wagnertech.de/debian main`.

- Schreiben Sie zwei Skriptdateien: `mein-prog` und `postinst` mit jeweils einer Konsolenausgabe.
- Erstellen Sie ein Paket
- Installieren und testen Sie das Paket.

Paketerstellung mit mbuild

Das Paket `mbuild`[6] bietet für die Paket erstellung folgende Unterstützung:

- Zusammenspiel mit den Versionierungswerkzeugen `svn` und `git`:

  Überprüfung, dass alle Dateien mit dem gewünschten Tag versioniert sind
- Überprüfung des Vorhandenseins eines *changelog*
- C++ - Build
- Cross Compiling
- Dokumentation in den *man pages*
- Unterstützung alter Paketformate (bis Debian 7)

Aufgabe:

- Erstellen Sie ein `git`-Repository mit `git init DIR`.
- Legen Sie Verzeichnisse und Dateien gemäß der `mbuild`-Dukomentation an.
- Taggen Sie den Stand mit einem Tag des Formats `PAKET_VERSION` (`git tag PAKET_VERSION`).
- Erstellen Sie das Paket mit dem Befehl `mconfigure PAKET`.
- Installieren und testen Sie das Paket.

Erstellung eines eigenen Repositories

Mit selbsterstellten Paketen lässt sich ein eigenes Debian-Repository aufbauen. Dazu wird auf der einen Seite ein Webserver, auf der anderen Seite das Repository benötigt.

Für den `apache2` kann ein Link vom Basisverzeichnis des Repositories zur `DocumentRoot` erstellt werden.

Das Repository wird dann mit folgenden Befehlen erstellt:

- GnuPG-Schlüssel zum Signieren anlegen
  - In `~/.gnupg/gpg.conf` die Zeile `digest-algo sha256` hinzufügen

[6] Paketquelle: http://wagnertech.de/debian main

  - Schlüssel anlegen: `gpg --gen-key`
  - Name des öffentlichen Schlüssels ermitteln: `gpg -k`
    Wenn die Ausgabe etwa so aussieht,

```
/home/michael/.gnupg/pubring.gpg
--------------------------------
pub   2048R/F9D5412B 2017-07-03
uid                  WagnerTech UG <mail@wagnertech.de>
sub   2048R/C1C27368 2017-07-03
```

    ist `C1C27368` der Schlüsselname.
  - Schlüssel exportieren: `gpg --armor --export SCHLUSSEL_NAME > SCHLUSSEL_DATEI`
  - `SCHLUSSEL_DATEI` zum Download bereitstellen (z.B. im `conf`-Verzeichnis, s.u.)

- Verzeichnisstruktur anlegen:

```
$DEBIAN/
  conf/
    distributions
```

- Inhalt von `distributions`:

```
Origin: VENDOR
Label: PROJEKT
Codename: RELEASE
Architectures: i386 amd64
Components: main
Description: DESCRIPTION
SignWith: SCHLUSSEL_NAME
```

  Dieser Block tritt für mehrere Releases (z.B. `testing`, `stable`) mehrfach auf.

- Paket hinzufügen: `reprepro includedeb RELEASE DEB_FILE`
- Weitere `reprepro`-Befehle:
  - Alle Pakete eines `RELEASE` anzeigen: `reprepro list RELEASE`
  - Vorhandene Versionen eines Pakets anzeigen: `reprepro ls PAKET`
  - Paket entfernen: `reprepro remove RELEASE PAKET`
  - Paket von einem Release zu einem anderen kopieren: `reprepro copy ZIEL QUELLE PAKET`

Aufgabe:

- Erstellen Sie ein Repository.
- Fügen Sie Ihr Paket hinzu.
- Fügen Sie das Repository Ihres Nachbarn* zu Ihren Paketquellen hinzu.
- Installieren Sie das Paket Ihres Nachbarn*.

3 Festplatten und Dateisysteme

3.1 Dateisystem

- Alles ist eine „Datei“
- Wurzel des Verzeichnisbaums ist `/` (absolute Pfade).
- Darunter liegen:
  - `bin` (binaries): Dieses Verzeichnis enthält die Systemprogramme, also Standardkonsolenkommandos wie `ls` (Verzeichnisinhalt anzeigen) oder `cat` (Textdateien ausgeben).
  - `sbin` (start binaries): Dieses Verzeichnis enthält weitere Dienstprogramme, vor allem Initialisierungsprogramme, die beim Systemstart aufgerufen werden.
  - `dev` (devices): Dort werden die Gerätedateien abgelegt, d.h. Dateien, die auf die einzelnen Hardwarekomponenten verweisen. Damit lässt sich der Zufriff auf Geräte genau wie bei einzelnen Dateien über Benutzerrechte regeln.
  - `usr` (user, unix system resources): Hier finden Sie diverse Unterverzeichnisse für die Komponenten der wichtigsten Anwendungsprogramme. Beispiele: Unter `/usr/bin` befinden sich die ausführbaren Dateien, `/usr/lib` ist das Verzeichnis für gemeinsam genutzte Bibliotheken, `/usr/include` enthält die C-Header-Dateien des Systems und der Bibliotheken – wichtig für die Kompilierung gängiger Open Source-Software, die im Quellcode geliefert wird.

    > `/usr` usually contains by far the largest share of data on a system. Hence, this is one of the most important directories in the system as it contains all the user binaries, their documentation, libraries, header files, etc.... X and its supporting libraries can be found here. User programs like `telnet`, `ftp`, etc.... are also placed here. In the original Unix implementations, `/usr` was where the home directories of the users were placed (that is to say, `/usr/someone` was then the directory now known as `/home/someone`). In current Unices, `/usr` is where user-land programs and data (as opposed to 'system land' programs and data) are. The name hasn't changed, but it's meaning has narrowed and lengthened from „everything user related" to „user usable programs and data". As such, some people may now refer to this directory as meaning 'User System Resources' and not 'user' as was originally intended. {tldp}

  - `opt` (optional): Enthält zusätzliche Anwendungen, die weniger häufig benötigt oder aber nachträglich installiert werden.
  - `proc`: Spezielles Verzeichnis, das Informationen über laufende Prozesse, sowie weitere Systeminformation enthält. Das Verzeichnis ist hierarchisch angelegt und ermöglicht es auf einheitliche Weise Prozessinformationen abzufragen, ohne direkt an den Kernel zu müssen.

  - `etc`: Das Verzeichnis etc enthält die meisten systemweiten Konfigurationsdaten, sowohl für das Betriebsystem selbst als auch für viele Serverdienste und Anwendungen. Eine besondere Bedeutung haben/hatten z.B. die Init-Skripte unter `/etc/init.d`, die für den automatischen Start von Programmen beim Booten zuständig sind (System V-Init). Zusätzlich zu den globalen Einstellungen gibt es bei vielen Programmen auch benutzerspezifische Konfigurationsdaten. Diese werden in den Home-Verzeichnissen der jeweiligen Benutzer gespeichert; ihre Namen beginnen meist mit einem Punkt.
  - `var`: Dieses Verzeichnis enthält variable Daten, vor allen Dingen Logdateien, in die Fehlermeldungen und Hinweise vom Betriebsystem und aus anderen Quellen eingetragen werden.
  - `home`: enthält für jeden (menschlichen) Benutzer, der dem System bekannt ist, ein Home-Verzeichnis. Hier werden alle Anwendungsdaten dieses Benutzers abgelegt. Zusätzlich werden hier auch seine persönlichen Einstellungen für die verschiedenen Anwendungs- und Systemprogramme gespeichert. Gewöhnliche Benutzer haben außerhalb ihres Home-Verzeichnisses in aller Regel keine Schreibrechte, so dass sie weder vorsätzlich noch aus Unkenntnis das System oder die Dateien anderer User beschädigen können.
  - `root`: Dieses Verzeichnis ist das spezielle Home-Verzeichnis des gleichnamigen Superusers. Es liegt nicht im Verzeichnis `/home` wie die anderen Benutzerverzeichnisse. `/home` wird nämlich oft so eingerichtet, dass es auf einem anderen physikalischen Datenträger oder zumindest in einer anderen Partition liegt als der Rest des Betriebsystems. Möglicherweise steht es also nicht zur Verfügung, wenn ein Fehler auftritt, den root beheben muss. Da Sie normale Aufgaben am Rechner aus Sicherheitsgründen nicht als root erledigen sollten, dient dieses Verzeichnis in der Regel nicht der Speicherung von Arbeitsdateien, sondern beherbergt vor allem die persönlichen Konfigurationseinstellungen von root.
  - `tmp`: Dieses Verzeichnis enthält temporäre Dateien. Dieses Verzeichnis wird bei jedem Systemstart gelöscht. Da bei Linux-Servern Systemstarts sehr selten sein können, ist darauf zu achten, dass dieses Verzeichnis nicht zu voll wird.
  - `boot`: Enthält den Kernel und das Verzeichnis für das Bootmenu (`grub`)

  .[Kofler 2021, Kap. 11.8]

- Relative Pfade werden relativ zum aktuellen Verzeichnis interpretiert.
- Besondere Kürzel:
  - ~: Home-Verzeichnis des aktuellen Benutzers
  - .: Aktuelles Verzeichnis
  - ..: Verzeichnis oberhalb des aktuellen Verzeichnis
- `ls -l` zeigt auch die Zugriffsrechte an (z.B. -rwxr-xr-x), sowie die Anzahl der Hardlinks
- `ls -la` zeigt alle Dateieinträge
- `ls -i` zeigt INode-Nummern

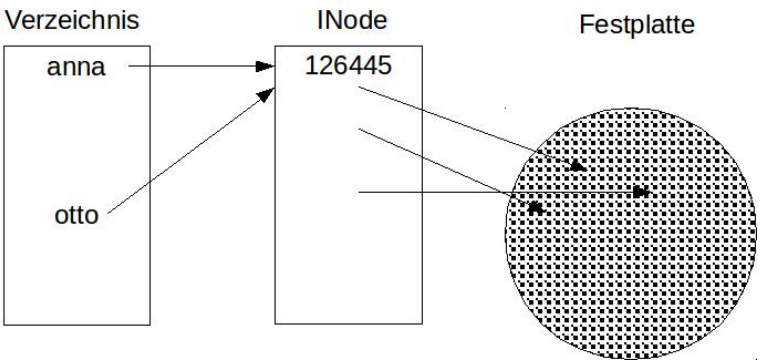

Abbildung 2: Inodes

- Auf Dateien wird nicht über den Namen, sondern über eine beim Namen hinterlegte Nummer (inode) zugegriffen: Abbildung 2.
- `ln` erstellt einen Inode-Eintrag
- Mit `find -inum NUM` kann nach Verzeichniseinträgen zu einer bestimmten Inode-Nummer gesucht werden.

Aufgabe:

```
> echo Test >otto   # schreibt Test in otto
> ln otto anna      # legt weiteren Verzeichniseintrag an
> ls -li            # gibt das Verzeichnis aus
> nano anna         # anna wird verändert
> less otto         # gibt otto aus
> rm otto           # löscht otto
> less anna         # gibt anna aus
> rm anna           # löscht anna
```

- Symbolische Links (`ln -s`) verweisen auf Dateien. Sie können auch ins „Nirvana“ zeigen. `ln -sf` überschreibt ggf. einen bereits bestehenden Link.

Aufgabe:

```
> echo Test >otto   # schreibt Test in otto
> ln -s otto anna   # legt symbolischen Link an
> ls -li            # gibt das Verzeichnis aus
> nano anna         # anna wird verändert
> less otto         # gibt otto aus
> rm otto           # löscht otto
> less anna         # gibt anna aus
> rm anna           # löscht anna
```

- Dateien, Verzeichnisse, die mit einem Punkt beginnen sind „versteckte" Dateien/Verzeichnisse.
  - Sie werden in den üblichen Dateibrowsern nicht angezeigt.
  - Sie werden bei `ls -l` nicht angezeigt.
  - Sie werden mit `ls -la` angezeigt.
  - Beispiele: `~/.profile ~/.ssh`

3.2 Festplatten

Partitionen und *mount*

- Partitionierung ist Unterteilung der Festplatte
  - Trotz „Volllaufen" einzelner Partitionen kann Betriebssicherheit gewährleistet werden (wenn man es richtig macht :-)
  - `df` (disk free) gibt Auskunft über die Belegung der Dateisysteme
  - `lsblk` listet alle *block devices*

> Aufgabe:
>
> Betrachten Sie die Plattenpartionierung (`df` oder in der Systemüberwachung).

- Partitionen werden entweder als *swap* oder als Dateisystem verwendet, wobei viele Systeme aufgrund der oft großen zur Verfügung stehenden RAM-Speicher gar keinen *swap* mehr vorsehen oder diesen als Auslagerungsdatei vorsehen. Um eine solche anzulegen sind folgende Befehle nötig:

```
sudo dd if=/dev/zero of=/swapfile bs=1024 count=4194304
sudo mkswap /swapfile
sudo chmod 600 /swapfile
sudo swapon /swapfile
/swapfile      -       swap    - 0 0 in /etc/fstab
```

  [Kofler 2021, Kap. 22.16]

- Wichtige Dateisysteme:
  - Ext3/4 (Linux Extended File System 3/4)
    - Mit Journalfunktion für die Wiederherstellung nach Stromausfall o.ä.
  - FAT16: für Disketten
  - FAT32: Altes Windows-Dateisystem, z.B. für USB-Sticks
  - NTFS: Aktuelles Windows-Dateisystem
    - Kann auch unter Linux zugegriffen werden

        * Kennt, im Gegensatz zu den FAT-Systemen, auch Zugriffsrechte und Benutzer
    - ISO 9660: CD-ROM
    - Swap: Die „Auslagerungsdatei", kann mit `swapon` (ggf. `/sbin/swapon`) abgefragt werden.

  Weitere Infos unter [Kofler 2021, Kap. 22.7]

- `mount` dient zum Einhängen von Dateisystemen in den Verzeichnisbaum.
- Mit `mount` können auch entfernte Dateisysteme (Samba, NFS) eingehängt werden.
- .[Kofler 2021, S. 58-61]

Verschiedene *mount*-Befehle:

- Befehl zum Einhängen eines Windows-Laufwerks/Freigabe:

```
> mount -t cifs //WinServer/Freigabe /mnt -o username=name -o password=pass \
  -o uid=UID,gid=GID
```

  `UID`, `GID` sind *user id* und *group id* des Benutzers, der nach dem Einhängen die Daten benutzen können soll.

  Ob in Windows Verzeichnisse freigibt, können Sie lokal mit der Eingabe von `\\localhost` im Windowsexplorer herausfinden.

- Eine weitere Möglichkeit entfernte Dateisysteme einzuhängen ist der *ssh mount*. Dazu muss auf dem Server ein ssh-Server laufen:

```
> sshfs [user@]host:[dir] mountpoint
```

- Loop-Devices

  Einzelne Dateien können als Dateisysteme gemounted werden (ISO-Dateien als `iso9660`).

```
sudo losetup /dev/loop0 IMAGE_NAME.iso # create loop device
sudo mount -t iso9660 /dev/loop0 /mnt # mount loop device
```

- Befehl zum Aushängen eines Dateisystems

```
> umount /mnt
```

Aufgabe:

- Legen Sie in /mnt eine Datei an (`sudo touch /mnt/otto`)
- Überprüfen Sie das Ergebnis (`ls -l /mnt`)
- Mounten Sie ein `sshfs`, *loop device* oder eine Windows-Freigabe auf `/mnt`.
- Überprüfen Sie das Ergebnis (`ls -l /mnt`)
- Lösen Sie den *mount*
- Überprüfen Sie das Ergebnis (`ls -l /mnt`)

Dieses Beispiel zeigt, wie ein *mount* Inhalte des Dateisystems überdecken kann.

- Datei `/etc/fstab` enthält die standardmäßig eingehängten Dateisysteme.

Aufgabe:

- Betrachten Sie mit `mount`, welche Dateisysteme in den Verzeichnisbaum eingehängt sind.
- Betrachten sie die Datei `/etc/fstab`.
- Überprüfen Sie mit `df` die Belegung der Dateisysteme

Partitionierung von Platten

Es gibt aktuell zwei Verfahren zur Verwaltung der Partitionierungsinformationen auf der Festplatte:

- MBR: Die Partitionierungskonzepte auf Basis der MBR-Partitionstabellen reichen bis in die DOS-Zeit zurück, und entsprechend angestaubt wirken manche Regeln und Einschränkungen. Mit MBR kommen Sie heute fast nur noch in virtuellen Maschinen oder bei externen Datenträgern in Kontakt (USB-Sticks, SD-Karten).
- GPT: Um die vielen MBR-Einschränkungen zu umgehen, gibt es seit vielen Jahren die viel moderneren GUID Partition Tables. Apple ist schon 2005 auf GPT umge- stiegen. Der PC-Markt hat diesen Schritt im Herbst 2012 mit der Markteinführung von Windows 8 vollzogen.[7]

Aktuelle Linux-Distrubutionen unterstützen GPT mit UEFI-Boot.

`sudo cfdisk DEVICE` startet ein interakative Konsolenanwendung.

Eine neu erstellte Partition muss dann mit einem Dateisystem initialisiert werden: `mkfs.SYSTEM PART`

`SYSTEM`: Filesystemtyp, z.B. `mkfs.ext4`
`PART`: Partition, z.B. `/dev/sdb2`

Aufgabe:

Mounten eines Dateisystems:

- Legen Sie mit Hilfe von VirtualBox eine weitere Festplatte an.
- Fügen Sie die Platte dem System hinzu (im ausgeschalteten Zustand)
- Überprüfen Sie mit `lsblk` das Ergebnis.
- Legen Sie mit `cfdisk` zwei Partitionen an.
- Legen Sie auf den Partitionen ein Dateisystem an:
  `mkfs.ext4 PART`

[7] Kofler 2021, S. 48f.

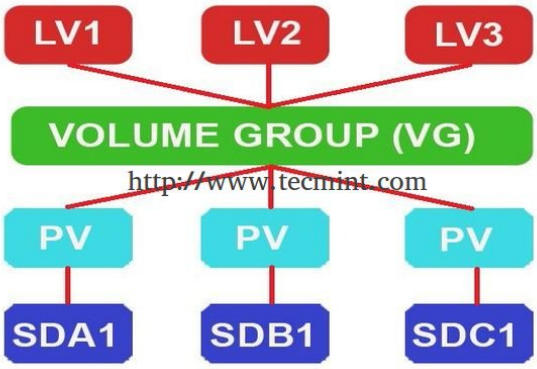

Abbildung 3: Logical Volume Manager

- Legen Sie zwei Mountpoints an (`/mnt/p1`, `/mnt/p2`)
- Mounten Sie die Partitionen auf die zwei Mountpoints und fügen Sie Dateien hinzu.
- Hängen Sie `/mnt/p2` wieder aus und mounten Sie die Partition (irrtümlich) auf `/mnt`. Was ist mit `/mnt/p1` passiert?

Logical Volume Manager (LVM)

Um eine größere Flexibilität bei der Verwendung von Festplatten zu erreichen, kann ein *Physical Volume Manager* verwendet werden. Dazu werden die Partitionen, die über mehrere Festplatten verteilt sein können (*Physical Volumes*, PV) zu einer *Volume Group* (VG) zusammengefasst. Eine VG stellt eine große vituelle Festplatte dar. Diese kann nun ihrerseits wieder partitioniert werden. Die Partitionen der VG heißen *Logical Volumes* (LV) (s. Abb. 3[8]).

Vergrößern einer LVM-Partition[9]:

- Vergrößern der Platte im VirtualBox-Manager für virtuelle Medien oder neue virtuelle Platte anlegen
- Kontrolle mit `sudo cfdisk`: Hier muss *free space* erscheinen.
- *free space* auswählen, neue (physische) Partition anlegen, Partitionen festschreiben
- Mit `sudo partprobe` Partitionstabelle neu einliesen und mit `ls -l /dev/sd*` prüfen.
- *Physical Volume* erstellen `sudo pvcreate /dev/sdaX` und mit `sudo pvs` prüfen.
- Mit `sudo vgs` den VG-Namen ermitteln
- Mit `sudo vgextend VG_NAME PYS_PART` neue Partition der VG hinzufügen.
- Mit `df` den Namen der logischen Partition ermitteln
- Mit `sudo lvextend -L 20G LOG_PART` logische Partition vergrößern. Kontrolle mit `lvdisplay`

[8] https://www.tecmint.com/wp-content/uploads/2014/07/Create-Logical-Volume-Storage.jpg (7.9.2020)
[9] https://www.thomas-krenn.com/de/wiki/LVM_vergrößern (2.8.2020)

- Mit `sudo resize2fs -p LOG_PART` Dateisystem vergrößern und mit `df` überprüfen.

> Aufgabe:
>
> Mounten Sie die oben erstellten Partitionen nicht direkt, sondern vergrößern Sie damit die Rootpartition.
>
> Hat Ihr System noch kein LVM, so kann in dieser Übung auch ein solches Schritt für Schritt angelegt werden. In diesem Fall müssen statt den `extend`-Befehlen die jeweiligen `create`-Befehle verwendet werden.

3.3 Bootvorgang

Beim Bootvorgang muss man heute Altsysteme mit BIOS/GRUB und aktuelle System mit EFI/-Secureboot unterscheiden. Da GRUB eine bewährte Methode ist, Linux-Systeme zu starten, wird diese nach wie vor auch in EFI-Systemen verwendet. [Kofler 2021, Abb. 23.1] zeigt die Zusammengänge.

Secureboot ist im Zusammenhang mit Linux problematisch: [Kofler 2021, S. 866f.]

GRUB bietet grundsätzlich die verschiedenen Linux-Kernel zur Auswahl an. Zu jedem Kernel gehört eine passende initrd-Datei, eine initiale RAM-Disk, die die für den Start benötigten Kernelmodule enthält.

Viele Distributionen versuchen den GRUB zu verstecken, insbesondere, wenn Linux das einzige System am Rechner ist. Will man dennoch in den GRUB, muss rechtzeitig eine Taste gedrückt werden: [Kofler 2021, Kap. 23.2]

Ist beispielsweise nach einer Windows-Reinstallation auf einem Dual-Boot-System GRUB zerschossen, muss GRUB neu installiert werden: [Kofler 2021, Kap. 23.4]

Grundsätzlich gibt es mit EFI auch die Möglichkeit Linux ohne GRUB zu starten: [Kofler 2021, Kap. 23.5]

> Aufgabe:
>
> Sehen Sie sich auf Ihrem System in folgenden Verzeichnissen um:
>
> - `/boot/efi`: Die EFI-Partition
> - `/etc/grub.d`: Die GRUB-Konfiguration

3.4 Systemstart (systemd)

Für den Start von UNIX-Systemen gibt es mittlerweile drei verbreitete Systeme. Das älteste ist das *System V Init*. Dieses startet zuerst den `init`-Prozess, der dann über viele, viele Skripte alle Dienste startet. System-V-Init ist robust und bewährt. Das Starten der vielen Skripte hat aber zur Folge, dass der Shell-Interpreter sehr oft gestartet werden muss, was entsprechend

Zeit kostet. Canonical, die Firma hinter Ubuntu, hat daher mit *Upstart* ein eigenes Startsystem entwickelt. In den Ubuntu-Distributionen wird dies seit Version 6.10 verwendet {upstart} und wurde Schritt für Schritt weiter verbessert. Eine wichtige Neuerung gegenüber dem System-V-Init ist die Parallelisierung des Systemstarts, was bei Mehrkernsystemen eine deutliche Beschleunigung bringt.[10]

Mit *systemd* hat die Linux-Gemeinde auf ein neues Startsystem entwickelt, das mittlerweile in die meisten Distributionen Einzug gehalten hat {wiki}:

- Debian 8
- Ubuntu 15
- Red Hat RHEL 7
- openSUSE 12

Oftmals entstanden dann aber Abspaltungen, die auf den `systemd` verzichten, z.B. Devuan bei Debian.

systemd managed nicht nur Dienste, sondern auch Geräte und Mountpoints. All diese Dinge werden als *units* bezeichnet. Units werden durch *unit files* gesteuert. Sie sind der Nachfolger der Initskripte. Die Unitfiles stehen im Verzeichnis `/etc/systemd/system`. Unitfiles sind Konfigurationsfiles. Ein Beispiel findet sich in Abb. 4. Es zeigt den Start eines Mysql-Servers.

Listing 1: MySQL-Unitfile

```
01 [Unit]
02 Description=MySQL 5.6 database server
03 After=syslog.target
04 After=network.target
05
06 [Service]
07 Type=simple
08 User=mysql
09 Group=mysql
10
11 # Execute pre and post scripts as root
12 PermissionsStartOnly=true
13
14 ExecStartPre=/usr/libexec/mysql-check-socket
15 ExecStartPre=/usr/libexec/
   mysql-prepare-db-dir %n
16 ExecStart=/usr/bin/mysqld_safe --basedir=/usr
17 ExecStartPost=/usr/libexec/mysql-wait-ready
   $MAINPID
18 ExecStartPost=/usr/libexec/
   mysql-check-upgrade
19
20 # Give a reasonable amount of time for the
   server to start up/shut down
21 TimeoutSec=300
22
23 # Place temp files in a secure directory, not
   /tmp
24 PrivateTmp=true
25
26 [Install]
27 WantedBy=multi-user.target
```

Abbildung 4: Unitfile für MySQL-Server [LM 02/2016]

Die Datei enthält folgende Informationen:

- `[Unit]`
  - `Description`: Klartextbeschreibung
  - `After`: Vorgabe, welche andere Dienste vorher gestartet sein müssen
- `[Service]`
  - Die Rechte (Benutzer/Gruppe), unter denen der Dienst laufen soll
  - `Type=simple` besagt, dass der zu startende Dienst unter `ExecStart` zu finden ist.
  - `ExecStartPre`: Skripte, die vor dem Dienst gestartet werden
  - `ExecStartPost`: Skripte, die nach dem Start des Dienstes ausgeführt werden

[10] [LM 02/2016] S. 22 ff.

  - `TimeoutSec`: Maximale Startzeit des Dienstes
- Unter `[Install]` wird der MySQL-Server dem `multi-user.target` zugeordnet.

Über weitere Parameter in der `[Service]`-Sektion lassen sich Sicherheitsmerkmale steuern:

- `PrivateTmp`: Verwendung eines eigenen `tmp`-Verzeichnisses
- `PrivateNetwork`: Verhinderung eines Netzwerkzugriffs
- `ReadonlyDirectories`: Verzeichnisse, auf die nur lesend zugegriffen werden dürfen
- `InaccessibleDirectories`: Verzeichnisse, auf die nicht zugegriffen werden dürfen

Darüber hinaus lassen sich weitere Resourcen, wie Dateigrößen oder die Anzahl der Kindprozesse beschränken.

Eine Übersicht im laufenden System schafft der Befehl `systemctl`:

- `systemctl --state=failed`: Auflistung aller Dienste, die nicht gestartet werden konnten
- `systemctl status <service>`: Status des Dienstes
- `systemctl start|stop|kill|restart|reload <service>`: Startet/beendet einen Dienst
- `systemctl enable|disable <service>`: Nimmt einen Dienst in den/aus dem Systemstart

Aufgabe:

- Betrachten Sie das Unitfile des `sshd`.
- Prüfen Sie, ob beim Systemstart Dienste nicht gestartet werden konnten.
- Betrachten Sie die Statusinformationen der Dienste `sshd` und `nfs-kernel-server`.
- Starten Sie den `nfs-kernel-server` neu.
- Definieren Sie Ihren eigenen Dienst:
  - Legen Sie die Datei `dater` an und machen Sie diese ausführbar.

```
#!/bin/bash
while true
do
  date >> /tmp/dater.out
  sleep 10
done
```

  - Legen Sie Sie Datei `/etc/systemd/system/dater.service` an.
  - Starten Sie den Dienst und überprüfen Sie den Erfolg.
  - Nehmen Sie den Dienst in den Systemstart
  - Starten Sie den Server neu und prüfen Sie, ob der Dienst mit dem Systemstart gestartet wurde.

Neben kontinuierlich laufenden Diensten kann `systemd` auch periodisch laufende Dieste verwalten. Dazu muss `Type=oneshot` gesetzt und in einer `SERVICE.timer`-Datei bestimmt werden, wie oft der Dienst ausgeführt werden soll.

Aufgabe:

- Betrachten Sie die Dateien `/lib/systemd/system/apt*`.
- Wandeln Sie das `dater`-Beispiel so um, dass das Programm minütlich ausgeführt wird.

4 Systemadministration

4.1 Shell-Kommandos (bash, ksh)

Um z.B. zu sehen, wer auf einem System eingeloggt ist, kann der Befehl who verwendet werden. Ein einzelner Befehl kann auch mehrere Programme zur selben Zeit ausführen oder Programme so miteinander verknüpfen, dass sie interagieren. Hier ein Befehl, der die Ausgabe von `who` als Eingabe von `wc -l` verwendet, das die Textzeilen in einer Datei zählt; das Ergebnis ist die Anzahl der Zeilen in der Ausgabe von who, also die Anzahl der eingeloggten Benutzer: `who | wc -l` Der senkrechte Strich, *pipe* genennt, stellt die Verbindung zwischen `who` und `wc` her.

- Platzhalter

  Mit Platzhaltern kann man eine Menge von Dateien mit ähnlichen Namen ansprechen. Zum Beispiel steht `a*` für alle Dateien, deren Name mit dem Buchstaben „a“ beginnt. Platzhalter werden von der Shell zu der Menge von Dateinamen „expandiert“, die dieser Spezifikation entsprechen. Wenn Sie also Folgendes eingeben:

  ```
  ls a*
  ```

  ermittelt die Shell zunächst alle Dateinamen im aktuellen Arbeitsverzeichnis, die mit „a“ beginnen. Gerade so, als ob Sie Folgendes eingegeben hätten:

  ```
  ls anika anita anna
  ```

  `ls` weiß nichts davon, ob Sie Platzhalter verwendet haben: Es sieht nur die komplette Liste von Dateinamen nach der Expansion durch die Shell.

  Soll einem Programm wirklich die Zeichenfolge „a*“ als Parameter übergeben werden, so muss die Expansion durch die Shell verhindert werden. Dies erfolgt durch das Setzen in Hochkomma.

  Weitere Platzhalter s. Tabelle 1.

- Tilde

  Die Shell behandelt Tilden (~) als Sonderzeichen, falls sie alleine oder am Anfang eines Wortes stehen. Dabei bedeutet:

  ~: Ihr Home-Verzeichnis

Tabelle 1: Shell-Platzhalter [Lin:32]

Platzhalter	Bedeutung
`*`	jede Art von Zeichen außer einem führenden Punkt
`?`	ein beliebiges einzelnes Zeichen
`[menge]`	ein einzelnes Zeichen aus der angegebenen `menge`, üblicherweise eine Sequenz von Zeichen wie `[aeiuoAEIUO]` für alle Vokale, oder ein Bereich mit einem Bindestrich, wie `[A-Z]` für alle Großbuchstaben
`[^menge]`	ein einzelnes Zeichen, das *nicht* in der angegebenen `menge` enthalten ist
`[!menge]`	(wie oben)

`~anna`: das Home-Verzeichnis des Benutzers anna

- Shell-Variablen

  Variablen werden durch Zuweisung definiert: `VAR=3`

  Um auf einen Variablenwert zuzugreifen wird ein Dollarzeichen vorangesetzt: echo `$VAR`

  Einige Variablen werden von der Shell bereits beim Anmelden definiert. Alle globalen Variablen einer Shell werden durch `env` ausgegeben.

  Der Gültigkeitsbereich einer Variable (d.h. welche Programme diese sehen) ist normalerweise die Shell, in der sie definiert wurde. Um eine Variable auch Programmen zugänglich zu machen, die von der Shell aufgerufen werden (d.h. Subshells), wird der Befehl `export` verwendet: `export VAR`

  Das „Exportieren“ kann auch gleich mit der Zuweisung erfolgen: `export VAR=2`

  Eine exportierte Variable wird auch Umgebungsvariable genannt, da sie jetzt auch für andere Programme in der „Umgebung” verfügbar ist.

  [Kofler 2021, Kap. 9.7]

- Pfad-Variable

  Eine sehr wichtige Variable ist `PATH`, da sie der Shell mitteilt, wo sie Programme findet. Wenn Sie einen Befehl wie z.B. `who` eingeben, muss die Shell das entsprechende Programm finden. Dazu wertet die die Variable `PATH` aus, die aus einer Sequenz von Verzeichnissen besteht, die durch Doppelpunkt getrennt werden. Die Shell sucht nach einem Befehl in jedem dieser Verzeichnisse.

  Der Pfad eines ausfühbaren Programms kann mit
  `which <kommando>`
  ermittelt werden.

- Aliase

  Der eingebaute Befehl `alias` definiert bequeme Abkürzungen für lange Befehle und erspart so Tipparbeit. Beispiel: `alias ll='ls -l'` Um Änderungen an Umgebungsvariablen oder Aliase stets zur Verfügung zu haben, werden diese in die Datei `.bashrc` eingetragen.

[Kofler 2021, S. 322f.]

Aufgabe:

- Betrachten Sie mit `alias`, welche Aliasse bereits definiert sind.
- Erstellen Sie einen neuen Alias, der eine Datei ausführbar macht (`chmod a+x`).

- Umleitung von Ein- und Ausgaben

  Die Shell kann die Standardeingabe, die Standardausgabe und Standard-Error in bzw. aus Dateien umleiten:

  `meinbefehl < eingabedatei`

  `meinbefehl > ausgabedatei` (erzeuge/überschreibe ausgabedatei)

  `meinbefehl >> ausgabedatei` (an ausgabedatei anhängen)

  Auch die Ausgabe eines Befehls, der nach Standard-Error schreibt, kann in eine Datei umgeleitet werden:
  `meinbefehl 2> fehlerdatei`

  Folgendermaßen können sowohl Standardausgabe als auch Standard-Error in eine Datei umgeleitet werden:
  `meinbefehl > ausgabedatei 2> fehlerdatei` (separate Dateien)
  `meinbefehl > datei 2>&1` (eine Datei)
  `meinbefehl &> datei` (eine Datei)

  Mit dem Pipe-Operator | können Sie die Standardausgabe eines Befehls zu Standardeingabe eines anderen machen, mit `|&` können sowohl stdout als auch stderr weitergeleitet werden.

  [Kofler 2021, Kap. 9.4]

- Befehle kombinieren

  Um mehrere Befehle nacheinander auf einer einzigen Kommandozeile auszuführen, werden die mit Semikolons getrennt:

  `befehl1 ; befehl2 ; befehl3`

  Um eine Sequenz von Befehlen auszuführen, die Bearbeitung aber beim Scheitern eines Befehls abzubrechen, werden sie mit `&&` („und“) getrennt.

  Falls gerade im Fehlerfall weiterzumachen ist, werden die Befele mit `||` („oder“) getrennt.

  Befehle können dabei mit geschweiften Klammer geklammert werden:
  `probier_was || ( echo "Da ist was schief gelaufen" ; exit 1 )`

  [Kofler 2021, Kap. 9.5]

- Quoting

  Normalerweise behandelt die Shell Leerstellen einfach als Trennsymbole der Wörter der Kommandozeile. Soll das Wort Leerstellen enthalten (z.B. ein Dateiname mit Leerstelle), so ist es in einfache oder doppelte Anführungszeichen zu setzen. Einfache Anführungszeichen behandeln den Inhalt wörtlich, während bei doppelten Shell-Variablen ausgewertet werden.

  `echo '$HOME'` (ergibt: `$HOME`)

  `echo "$HOME"` (ergibt: `/home/anna`)

Aufgabe:

- Leiten Sie die Ausgabe von `who` in eine Datei um. Verwenden Sie die eben geschriebene Datei als Eingabe für den Befehl `wc -l`
- Führen Sie folgende Befehle für Dateien durch, die es gibt und für Dateien, die es nicht gibt:

```
ls <datei> && cat <datei>
ls <datei> || echo "Fehler: Datei nicht gefunden"
```

- Um die Wirkung der verschiedenen Hochkommata zu demonstrieren, führen Sie folgende Befehle aus. Welche Dateien werden angelegt?

```
datei="anna lena"
touch $datei
touch "$datei"
touch '$datei'
```

  Anmerkung: `touch` legt eine neue leere Datei an.

- Ausgabe eines Befehls als Textkette verwenden: `$(<befehl>)`

  ```
  echo "Zur Zeit ist der Benutzer $(whoami)angemeldet"
  ```

  Früher wurde dafür der *backtick* verwendet:

  ```
  echo "Zur Zeit ist der Benutzer `whoami` angemeldet"
  ```

  [Kofler 2021, S. 329, Tab. 9.5]

- Suchbefehl: find

```
### Suche nach Name
find . -name "otto*"
find . -iname "otto*" # not case sensitive
```

  Der erste Parameter von `find` ist das Basisverzeichnis der Suche. Statt dem aktuellen Verzeichnis (`.`) kann auch ein anderes Verzeichnis angegeben werden.

  Weitere Optionen:

  - `-type d|f`: Schränkt die Suche nach Verzeichnissen oder Dateien ein
  - `-exec ... \;`: Lässt den Anwender mit der gefundenen Datei etwas tun.

```
find . -name "*.cpp" -type f -exec grep "TODO" {} /dev/null \;
grep "TODO" $(find . -name "*.cpp" -type f)
find / -name .bashrc 2>/dev/null
```

Weitere Möglichkeiten zur Dateisuche: [Kofler 2021, Kap. 11.4]

- Kommandos zur Anzeige von Textdateien
  - `cat <datei>`: Gibt den Inhalt der Datei aus
  - `less <datei>`: Gibt den Inhalt der Datei seitenweise aus
    - Mit `less` kann man auch blättern, suchen, ... Die Befehle sind ähnlich wie beim vi:
      `/`: Suchen, `n` springt zum nächsten Treffer, `N` zum letzten
      `?`: Rückwärts suchen
      `g`: An den Anfang springen
      `G`: An das Ende springen
      `q`: Beenden
    - `less -S` schneidet lange Zeilen rechts ab.
  - `tail <datei>`: Gibt die letzten (10) Zeilen einer Datei aus.
    - `tail -f <datei>` ist dynamisch. Praktisch bei log-Dateien, wird mit `CTRL-C` verlassen.
  - `diff <datei1> <datei2>`: Vergleicht Datei1 mit Datei2.
    `diff -yW 80`: Gegenüberstellung mit einer Breite von 80 Zeichen
- Zusammenfassung wichtiger bash-Sonderzeichen: [Kofler 2021, Kap. 9.12]

Aufgabe:

- Erstellen Sie die Textdatei `otto`.
- Kopieren Sie die Textdatei in die Datei `anna`.
- Verändern Sie anna.
- Betrachten Sie den Unterschied mit `diff`
- Hängen Sie `otto` an `anna` an: `cat otto >> anna`
- Hängen Sie eine weitere Zeile an `anna` an: `echo <weitere Zeile> >> anna`
- Öffnen Sie ein weiteres Fenster, in dem Sie mit `tail -f` Änderungen an `anna` verfolgen können.
- Hängen Sie nun im ersten Fenster eine weitere Zeile an `anna` an.
- Betrachten Sie `anna` mit `less` und suchen Sie nach einem Ausdruck.

4.2 Systeminformation und Prozesse

- `uname` liefert diverse Systeminformationen. Die Option -a liefert alle mit uname lieferbaren Informationen: Kernel, Host, Kernel-Release, Kernel-Version (Build-Datum), Maschine, Prozessor, Hardwareplattform, Betriebssystem.
- `uptime`: Wie lange läuft die Maschine?
- `who`: Wer ist angemeldet?
- `cat /proc/cpuinfo`: CPU des Rechners
- `cat /proc/meminfo`: Speicher des Rechners
- `free -h`: Freier Speicher
- `cat /etc/issue, cat /etc/os-release`: Instaliertes Betriebssystem
- `inxi -F`: Gibt eine ausfühliche Systeminformation, muss aber meist erst installiert werden (Paket: `inxi`).

Ein Prozess ist eine Arbeitseinheit auf einem UNIX-System. Jedes ausgeführte Programm besteht aus einem oder mehreren Prozessen. Jeder Prozess (außer der Init-Prozess) hat einen Vaterprozess, der ihn gestartet hat.

- `ps -ef` oder `ps aux`: Welche Prozesse laufen?
- `ps -efH`: Prozesshierarchie: Wer hat wen gestartet?
- `top`: Welche Prozesse sind aktiv?

  Dieses Programm lässt sich mit folgenden Tastenbefehlen steuern:

  - *P*: Sortierung nach Prozessorlast (`%CPU`), default
  - *M*: Sortierung nach Speicherbedarf (`%MEM`)
  - *k*: Sende ein Signal an den obersten Prozess der Liste.
  - *q*: Beende das Programm.
- Mit `kill [-<signal>] <pid>` kann man einem Prozess Signale schicken. Die wichtigsten sind:
  - `HUP` (ggf. Neulesen der Konfiguration)
  - `TERM` (default, Beenden, kann aber vom Programm abgefangen werden)
  - `KILL` (Beenden)
- `&` am Ende einer Befehlszeile startet einen Prozess im Hintergrund.

  Mehr zum Verwalten von Prozessen: [Kofler 2021, Kap. 12.1]

Aufgabe:

- Schreiben Sie ein Endlosprogramm `endless`

```
#!/bin/bash
while true
do
  echo hallo
done
```

- Machen Sie das Programm ausführbar (`chmod 755 <datei>`) und rufen Sie es auf (`./endless`)
- Suchen Sie das Programm mit `top` in einem anderen Fenster
- Im top können Sie mit „k“ einen Prozess „killen“
- Starten Sie `endless` erneut
- Suchen Sie den Prozess mit `ps -ef`
- Suchen Sie den Prozess in der Prozesshierarchie (`ps -efH`)
- Beenden Sie den Prozess mit `kill <pid>`
- Holen Sie sich mit `inxi -F` eine ausführliche Systeminformation. Ggf. muss das Programm erst installiert werden.

4.3 Benutzer und Gruppen

- Benutzer sind in `/etc/passwd` abgelegt.

  Hier ist auch hinterlegt mit welcher Shell in welchem Verzeichnis sich der Benutzer im System nach dem Login befindet. Bei einem Login werden als erstes in der Shell Konfigurationsdateien ausgeführt, darunter die persönliche Steuerdatei `~/.profile`.

- Passwörter in `/etc/shadow`
- Benutzergruppen in `/etc/group`

  Die Gruppen, zu denen ein Benutzer gehört, sind auf folgende Weise festgelegt:

  – Jeder Benutzer gehört zu einer *Primärgruppe*. Deren Gruppennnummer ist in `/etc/passwd` an 4. Position festgelegt, ihr Name in `/etc/group`.

  – Ein Benutzer kann weiteren Gruppen zugeordnet werden, indem in `/etc/group` hinter der Gruppendefinition weitere Benutzernamen kommasepariert angefügt werden.

  Mit dem Befehl `groups` kann die aktuelle Gruppenzuordnung eines Benutzers überprüft werden.

Aufgabe:

- Betrachten die die oben genannten Systemdateien.
- Legen Sie einen neuen Nutzer mit der grafischen Oberfläche an (falls vorhanden).
- Legen Sie einen neuen Nutzer auf der Kommandozeile mit dem Befehl

  `sudo adduser <user>` an.
- Betrachten Sie erneut die Systemdateien

Weitere Adminstrationsbefehle:

- `deluser`: Löscht einen Benutzer
- `passwd`: Ändert das Passwort des aktuellen Benutzers
- `sudo passwd USER` setzt das Kennwort des Benutzers `USER`.
- Um nicht in den Dateien selbst Benutzerinfomrationen ändern zu müssen existiert der Befehl `usermod`. Seine Verwendung kann auf der *man page* angeschaut werden.

4.4 Rechteverwaltung

Der Befehl `ls -l` zeigt auch die Zugriffsrechte von Dateien und Verzeichnissen an (z.B. `-rwxr-xr-x`).

Die erste Stelle gibt den Dateityp an: '-' für eine gewöhnliche Datei, 'd' für ein Dateiverzeichnis oder 'l' für einen Symlink. Die neun nachfolgenden Stellen zeigen in Dreiergruppen die Zugriffsrechte an. Ein Buchstabe steht dafür, dass ein Zugriffsrecht gewährt wird, ein Strich bedeutet dagegen, dass es nicht gewährt wird. Im vorliegenden Fall darf der Eigentümer (die ersten drei der neun Zeichen) die Datei lesen, schreiben (dazu gehören in der Regel auch Löschen und Umbenennen) und ausführen. Die Gruppe (die nächsten drei Zeichen) und der Rest der Welt (die letzten drei) dürfen nur lesen und ausführen, aber keinerlei Änderungen durchführen. Das Recht der Ausführung ist nur für Programme und für Verzeichnisse sinnvoll. Letztere lassen sich zwar natürlich nicht wie ein Programm ausführen, aber wenn das x nicht gesetzt ist, können sie nicht als Arbeitsverzeichnis ausgewählt werden.

Intern werden die Zufriffsrechte übrigens als dreistellige Oktalzahl gespeichert. Die erste Stelle enthält die Benutzerrechte des Eigentümers, die zweite die der Gruppe und die dritte die der anderen Benutzer. Der Wert jeder Stelle ist die Summe aus den gewährten Benutzerrechten: 4 steht für Lesen, 2 für Schreiben und 1 für Ausführen. Das Zugriffsrecht `rwxr-xr-x` lässt sich also als 755 darstellen. [S10]

Bei Debian/Ubuntu/Mint hat jeder Nutzer seine eigene Gruppe. Standardmäßig kann also kein Nutzer die Dateien eines anderen Nutzers ändern, auch wenn dieser die Rechte für die Gruppe freigegeben hat. Damit ist eine sehr feingranulare Rechteverwaltung möglich.

Bei der Neuanlage von Dateien/Verzeichnissen bestimmt die `umask`, welche Rechte das neue Objekt *nicht* hat. Der Standardwert von `0022` legt Objekte ohne Schreibrecht für Gruppe und Welt an.

Eigentumsverhältnisse und Zugrifsrechte können mit den Befehlen `chown`, `chgrp` und `chmod` geändert werden. [Kofler 2021, Kap. 11.5]

Welche Rechte für welche Dateioperationen benötigt werden, ist hier dargestellt: [Kofler 2021, S. 399, Tab. 11.8]

Aufgabe:

- Erstellen Sie eine Datei in Ihrem Verzeichnis
- Wechseln Sie zu einem der neu angelegten Nutzer: `su <user>`
- Versuchen Sie die gerade angelegte Datei zu editieren
- Geben Sie als Besitzer der Datei der Gruppe Schreibrechte:
  `chmod g+w <datei> oder chmod 664 <datei>`
- Versuchen Sie erneut als anderer Nutzer auf Ihre Datei zuzugreifen
- Nehmen Sie den neuen Nutzer in Ihre Gruppe
  `sudo vi /etc/group`
  `<owner>:x:1000:<new user>`
  Alternativ können Sie den Befehl `usermod` verwenden.
- Prüfen Sie mit dem Befehl `groups` zu welchen Gruppen der neue Nutzer nun gehört. Hinweis: Erst bei einem erneuten Login wird die Gruppenzugehörigkeit eines Benutzers neu vergeben.
- Versuchen Sie erneut als anderer Nutzer auf Ihre Datei zuzugreifen
- Nun versuchen Sie die Sache umgekehrt: Der neue Nutzer legt eine Datei in seinem Verzeichnis an und gibt der Gruppe Schreibrechte. Sie versuchen diese Datei zu ändern.

Ein weiteres Instrument der Rechteverwaltung ist das S-UID-Bit. Ist bei einer ausführbaren Binärdatei dieses Bit gesetzt, so nimmt der Nutzer während der Ausführung die Identität des Eigentümers der ausführbaren Datei an.

Aufgabe:

- Versuchen Sie als gewöhnlicher Benutzer mit dem Editor `nano` die Datei `/etc/passwd` zu editieren (`nano /etc/passwd`).
- Setzen Sie das s-Bit: `sudo chmod u+s /usr/bin/nano`.
  Falls sich `nano` an einem anderen Ort befindet, kann dies mit `which nano` herausgefunden werden.
- Versuchen Sie erneut die Datei `/etc/passwd` zu editieren.
- Machen Sie die Änderungen wieder rückgängig.

Weitere Spezial-Bits s. [Kofler 2021, Kap. 11.6].

4.5 Systemprotokollierung

Der Linuxkernel selbst und die diversen Deamons, die in einem Linux-System laufen, protokollieren ihren Betriebszustand in Logdateien. Die Programme schreiben dabei nicht selbst in Dateien, sondern bedienen sich dabei eines Dienstes, des Syslog.

Die trsditionelle Implementierung ist `rsyslogd`, die nach und nach vom Logger des `systemd` (`systemd-journald`) abgelöst wird.

syslog

Syslog empfängt die Meldungen und schreibt sie in die verschiedenen Logdateien. Eine Logmeldung besteht aus drei Teilen:

- der Quelle (*facility*). Folgende Quellen sind definiert:

```
auth authpriv cron daemon kern lpr mail mark news security syslog
user uucp ftp local0 local1 local2 local3 local4 local5 local6 local7
```

- der Dringlichkeit (*priority*). Folgende Stufen sind definiert:

```
emerg alert crit error warn notice info debug
```

- dem Meldungstext.

Die Regeln, die die Nachrichten auf die Dateien verteilen, sind in der Konfigurationsdatei `/etc/rsyslog.conf` zu finden. Damit der `rsyslog` zu älteren Implementierungen kompatibel ist, versteht er verschiedene Varianten in der Konfigurationsdatei. Zwei Formate sind dabei empfohlen:

- `sysklogd`-Format

```
<facility>.<priority> <file>
```

  Für *facility* bzw. *priority* kann dabei auch der „*" als *wildcard* eingesetzt werden. Ein „-" vor dem Dateinamen sagt, dass nicht jeder Eintrag geflasht werden muss.

- *RainerScript*: Dieses Format ermöglicht weitergehende Konfigurationsmöglichkeiten. Es können Bedingungen und Ausnahmen definiert werden. Hier ein Beispiel:

```
if $fromhost == 'host1' then {
   mail.* action(type="omfile" file="/var/log/host1/mail.log")
   *.err /var/log/host1/errlog # this is also still valid
   #
   # more "old-style rules" ...
   #
} else {
   mail.* action(type="omfile" file="/var/log/mail.log")
   *.err /var/log/errlog
   #
   # more "old-style rules" ...
   #
```

```
}
```

Die Log-Dateien werden regelmäßig abgeschlossen und umgespeichert. Diese Aufgabe übernimmt `logrotate`. `logrotate` wird über den `cron`-Dienst regelmäßig aufgerufen. Es prüft anhand der Einträge in `/etc/logrotate.conf`, ob eine der Dateien „rotiert" werden muss. Eine Rotation umfasst folgende Schritte:

- Löschen der ältesten Logdatei
- Erhöhung der Versionsnummern größer 2
- Kompression der Version 1 und Umbenennung auf 2
- Kopieren der aktuellen Datei auf Version 1

Ein nützliches Kommando, um sowohl kompromierte als auch nicht kompromierte Dateien zu durchsuchen, ist `zgrep`.

[Kofler 2021, Kap. 18.12]

Aufgabe:

- Betrachten Sie die Logging-Regeln in `/etc/rsyslog.conf`, resp. `/etc/rsyslog.d/*`. Anmerkung: Das Minuszeichen vor Dateipfaden kann aussagen, dass hier auf die unmittelbare Synchronisation der Datei verzichtet werden kann.
- Schreiben Sie einen Logeintrag unter Zuhilfenahme des Programms `logger`:
  `logger -t <process> -p <facility>.<priority> <message>`
- Suchen Sie den Logeintrag. In welche Dateien wurde er geschrieben?
- Ergänzen Sie die Konfigurationsdatei um einen Eintrag für `news.info`.
- Starten Sie `rsyslog` neu und testen Sie die neue Konfiguration.

systemd-journald

Das Init-System systemd, das in Abschnitt 24.1 beschrieben wird, enthält eigene Logging-Funktionen, das sogenannte Journal. Für die Protokollierung ist der Hintergrundprozess `systemd-journald` verantwortlich. Im Vergleich zu traditionellen Syslog-Diensten gibt es drei fundamentale große Unterschiede:

- Das Journal wird platzsparend in einem binären Format gespeichert. Dieses Format ist aber auch der größte Nachteil des Journals: Unzählige Tools setzen voraus, dass die Logging-Dateien im Textformat vorliegen und unkompliziert mit grep ausgewertet werden können.
- Das Journal ist gegen nachträgliche Änderungen geschützt. Damit ist es für einen Einbrecher unmöglich, seine Spuren zu beseitigen.
- Das gesamte Journal wird an einem Ort gespeichert. Die bei Syslog übliche Trennung in mehrere Logging-Dateien entfällt (und damit auch die entsprechende Konfiguration). Die Extraktion bestimmter Meldungen erfolgt mit `journalctl`.

Davon abgesehen ist das Journal aber Syslog-kompatibel. Alle Dienste, die bisher Syslog zum Protokollieren verwendet haben, können ohne Änderungen das Journal nutzen. Auch das im vorigen Abschnitt beschriebene Kommando logger kooperiert problemlos mit dem Journal.

Wenn das Verzeichnis `/var/log/journal` existiert, speichert das Journal seine Protokolle dort. Existiert dieses Verzeichnis hingegen nicht, kommt `/run/log/journal` als Speicherort zur Anwendung. Auf den ersten Blick scheint das kein großer Unterschied zu sein, allerdings ist `/run` üblicherweise ein temporäres Dateisystem. Dort gespeicherte Protokolle gehen daher mit jedem Neustart verloren![11]

Zur Verwendung von `journalctl` und der Konfiguration von `journald`: [Kofler 2021, S. 620f.]

Aufgabe:

- Schreiben Sie einen Logeintrag unter Zuhilfenahme des Programms `logger`:
  `logger -t <process> -p <facility>.<priority> <message>`
- Suchen Sie `journalctl` den Logeintrag.

4.6 Kernel[12]

Aufgaben des Kernels

Der Kernel eines Betriebssystems bildet die hardwareabstrahierende Schicht, das heißt, er stellt der auf dieser Basis aufsetzenden Software eine einheitliche Schnittstelle (API) zur Verfügung, die unabhängig von der Rechnerarchitektur ist. Die Software kann so immer auf die Schnittstelle zugreifen und braucht die Hardware selbst, die sie nutzt, nicht genauer zu kennen. Der Linuxkernel stellt folgende Funktionalität zur Verfügung:

- Speicherverwaltung
- Prozessverwaltung, Multitasking, Lastverteilung
- Sicherheitserzwingung
- Eingabe/Ausgabe-Operationen auf verschiedenen Geräten
- Virtualisierung
- Routing

[11] Kofler 2021, Kap. 18.13
[12] https://de.wikipedia.org/wiki/Linux_(Kernel) (3.1.2018)

Funktionsweise

Bei einem strikt monolithischen Kernel wird der gesamte Quellcode inklusive aller Treiber in das Kernel-Image (den ausführbaren Kernel) kompiliert. Im Gegensatz dazu kann Linux Module benutzen, die während des Betriebs geladen und wieder entfernt werden können. Damit wird die Flexibilität erreicht, um unterschiedlichste Hardware ansprechen zu können, ohne sämtliche (auch nicht benötigte) Treiber und andere Systemteile im Speicher halten zu müssen (modular monolithischer Kernel). Nahezu jeder Treiber kann allerdings auch als Modul zur Verfügung stehen und vom System dann dynamisch nachgeladen werden. Ausgenommen davon sind Treiber, die für das Starten des Systems verantwortlich sind, bevor auf das Dateisystem zugegriffen werden kann. Der Kernel besteht aus mehreren Tausend Quelldateien mit Millionen Zeilen C-Code.

Der Kernel ist ein Betriebssystemkern und darf nicht als das eigentliche Betriebssystem verstanden werden. Dieses setzt sich aus dem Kernel und weiteren grundlegenden Bibliotheken und Programmen (die den Computer erst bedienbar machen) zusammen.

Architektur

Da der Kernel für viele Systeme kompilert wurde, ist Linux heute das meist-portierte Betriebssystem:

- Intel 80386
- Intel 8086
- Intel IA-64 (Itanium)
- AMD64 (Athlon, Opteron)
- PowerPC
- Alpha
- Sun SPARC, UltraSparc
- Motorola 68020
- MIPS (RISC von Silicon Graphics)
- HP PA-RISK
- IBM zSeries
- Embedded Prozessoren, (Realtime bis etwa 20 ms)

Entwicklungsprozess

Die Entwicklung von Linux liegt durch die GNU General Public License und durch ein sehr offenes Entwicklungsmodell nicht in der Hand von Einzelpersonen, Konzernen oder Ländern, sondern in der Hand einer weltweiten Gemeinschaft vieler Programmierer, die sich hauptsächlich über das Internet austauschen. Bei der Entwicklung kommunizieren die Entwickler fast ausschließlich über E-Mail, da Linus Torvalds behauptet, dass so die Meinungen nicht direkt aufeinander prallen. In vielen Mailinglisten, aber auch in Foren und im Usenet besteht für jedermann die Möglichkeit, die Diskussionen über den Kernel zu verfolgen, sich daran zu beteiligen und auch aktive Beiträge zur Entwicklung zu leisten. Durch diese unkomplizierte Vorgehensweise ist eine schnelle und stetige Entwicklung gewährleistet, die auch die Möglichkeit mit sich bringt, dass jeder dem Kernel Fähigkeiten zukommen lassen kann, die er benötigt.

Eingegrenzt wird dies nur durch die Kontrolle von Linus Torvalds und einigen besonders verdienten Programmierern, die das letzte Wort über die Aufnahme von Verbesserungen und Patches in die offizielle Version haben. Manche Linux-Distributoren bauen auch eigene Funktionen in den Kernel ein, die im offiziellen Kernel (noch) nicht vorhanden sind.

Die Versionskontrolle des Kernels unterliegt dem Programm Git. Dies wurde speziell für den Kernel entwickelt und auf dessen Bedürfnisse hin optimiert.

Eine eigene Geschichte ist die Nummerierung der Kernelversionen. Im Lauf der nun mehrere Jahzehnte dauernden Kernelentwicklung wurden die Schemata der Nummerierung immer wieder verändert. Hier eine Aufstellung der Historie:

Linux kernel version history[13]

Logging

Die Meldungen des Kernel werden über zweierlei Wege festgehalten. Der Kernel schreibt Meldungen

- in einen Ringspeicher, der mit `dmesg` ausgelesen werden kann.
- über den Logging-Dienst des Systems

Da während des Bootvorgangs der Logging-Dienst erst gestartet wird, können frühe Meldungen nur über den Ringspeicher ausgelesen werden.

Aufgabe:

- Welche Kernel-Version läuft auf Ihrem System (`uname -a`)
- Welche Module hat der Kernel geladen (`lsmod`)
- Betrachten Sie die Kernel-Logs `dmesg` und `cat /var/log/syslog` / `journalctl -k`.

[13] https://en.wikipedia.org/wiki/Linux_kernel_version_history

Kernelmodule

Zum manuellen Nachladen von Kernelmodulen stehen folgende Befehle zur Verfügung:

- `insmod MODPATH`: Lädt das unter dem Pfad `MODPATH` liegende Modul in der Kernel.
- `modprobe MOD`: Lädt das Modul `MOD`. `modprobe` berücksichtigt Modulabhängigkeiten und sucht sich den Pfad selbst.

Die Modulabhängigkeiten sind in `/etc/modprobe.d/` abgelegt.

[Kofler 2021, Kap. 25.1]

Wie man bei einem WLAN-Adapter schrittweise von der Hardwareerkennung über das Laden der Treiber zum Netzwerk kommt, ist hier[14] beschrieben.

4.7 Drucken unter Linux

Unter UNIX/Linux gibt es für das Drucken ein eigenes Protokoll: CUPS (*common unix printing system*).

Kern dieses Systems ist der *sceduler*, der die verschiedenen Druckaufgaben entgegennimmt und in die Warteschlangen der verschiedenen Drucker stellt. CUPS enthält einen eigenen HTTP-Server (Port 631), über den es vom Netz aus ansprechbar und über einen Browser administrierbar ist.[15]

Der Betrieb eines reinen Druckservers unter Linux gestaltet sich hingegen oft als schwierig: Die Druckertreiber sind of proprietäre Software, die erst mal für Windows implementiert wird und dann auf Linux portiert wird. Dabei ist es (z.B. bei HP) so, dass der Treiber zur Installation eine graphische Oberfläche benötigt.

Umgekehrt bieten viele Netzdrucker selbst eine CUPS-Schnittstelle an, so dass das Drucken aus Linux-System heraus gut unterstützt wird. Diese Drucker werden von Linux-Desktop-Systemen automatisch erkannt und stehen sofort zur Verfügung.

Für einen automtischen Druck steht das Kommendo `lp` zur Verfügung. Da die meisten Drucker postscriptfähig sind, lassen sich Postscriptdateien direkt an den Drucker schicken. Der Befehl dafür lautet:

```
lp -d DRUCKERNAME PS_DATEI
```

Ein Druck über die durch `ghostscript` realisierte Umwandlung in Postscript erfolgt über:

```
lpr -P DRUCKERNAME PS_DATEI
```

[Kofler 2021: Kap. 18.11]

[14] `https://hilfe.wagnertech.de/index.php/wlan-analyse/` (7.12.2023)
[15] [S:260.]

5 Shell-Programmierung

5.1 Grundlagen

Ein erstes Beispiel

```
#!/bin/bash
set -e  # Abbruch bei einem Fehler (RC != 0)
set -x  # debug mode

summe=0
for n in $*
do
  case $n in
  *[!0-9]*)
    echo "Keine Zahl: $n"
    ;;
  *)
    let summe=$summe+$n
    echo "Zahl: $n"
    ;;
  esac
done
echo "Summe: $summe"
```

Aufgabe:

- Skript in die Datei `summe.sh` tippen
- `summe.sh` ausführbar machen (`chmod a+x summe.sh`)
- `summe.sh` ausführen (mit verschiedenen Parametern)

Variablen

Die Zuweisung von Variablenwerten erfolgt mit `variable=<wert>` (kein Leerzeichen!). Die Werte, die in Variablen gespeichert sind, sind immer Zeichenketten. Sie werden in einem entsprechenden Kontext in Zahlen umgewandelt.

Werden in einem Shell-Skript auf Variablen zugegriffen, sollten diese immer in doppelte Anführungszeichen gesetzt werden, um bestimmte Laufzeitfehler zu vermeiden. Eine undefinierte Variable oder eine Variable, deren Wert Leerstellen enthält, wird nicht richtig aufgelöst, wenn sie nicht in doppelten Anführungszeichen steht.

```
a="Erster Fall"
if [ $a = "Erster Fall" ]; then echo "Erster Fall"; fi (Fehler!)
```

Falls ein Variablenname neben anderen Zeichen steht, muss er mit geschweiften Klammern umschlossen werden:

```
a=Apfel
echo "${a}ernte"
```

Booleans und Rückgabewerte

Für die Shell steht der Wert 0 für wahr oder Erfolg und alle anderen Werte für falsch oder Fehler. Außerdem gibt jeder Befehl beim Beenden einen Integerwert an die Shell zurück, der mit `$?` abgefragt werden kann:

```
who
echo $?
```

- Die Rückgabewerte eines Befehls sind normalerweise in dessen Manpage dokumentiert.
- Der Befehl `test`, der mit `[` abgekürzt werden kann, evaluiert einfache boolsche Ausdrücke.
  `test 10 -lt 5` (Rückgabewert 1)
  `[ 10 -lt 5 ]` (dasselbe; Achtung Leerstellen!)

  Weitere Möglichkeiten siehe `man test`.
- Will ein Programm selbst mit einem Rückgeabewert ungleich null enden:
  `exit N`

Programmelemente

- echo: Standardausgabe
- read: Standardeingabe
- let var=<wert>: Zuweisung eines arithmetischen Wertes
- for – do – done: Schleife ({bash} Kap. 11.1)
- while – do – done: Schleife mit Abbruchbedingung ({bash} Kap. 11.1)
- case – esac: Fallunterscheidung ({bash} Kap. 11.4)
- if – else – fi: Fallunterscheidung ({bash} Kap. 7)
  - Achtung: Beim if statement sind Leerzeichen zu beachten!
- Aus Schleifen kann mit break herausgesprungen werden ([Lin] S. 189)

Syntaxbeispiel

- for statement

```
for var in 17 18 19
do
   echo $var
done
```

Hinweis: Normalerweise sind die Wörter duch Leerzeichen getrennt. Soll ein anderes Trennzeichen verwendet werden, kann dieses durch `IFS` festgelegt werden:

```
IFS=","
data="17,18,19"
for var in $data
do
   echo $var
done
```

- while statement

```
while read input
do
   echo "$input wurde eingegeben"
done
```

Soll zeilenweise aus einer Datei gelesen werden, wird der Dateiname als Eingabe hinter das `done` geschrieben:

`done < FILE`

Eine Eingabezeile kann auch gleich auf mehrere Variablen aufgeteilt werden:

`while read v1 v2 v3`

Die Aufteilung erfolgt beim `$IFS` (default: Leerzeichen)

- case statement

```
case $var in
A)
   echo "A gefunden"
   ;;
B)
   echo "B gefunden"
   ;;
*)
   echo "Was anderes gefunden"
   ;;
esac
```

- if statement

```
if [ -f datei.txt ]
then
   echo "Es existiert die Datei datei.txt"
elif [ "$var" = "hallo" ]
then
   echo "Variable hat den Wert 'hallo'"
```

```
elif rm datei1.txt
then
   echo "Datei datei1.txt konnte geloescht werden."
elif [ $i -lt 100 ]
then
   echo "i ist kleiner 100"
else
   echo "Von obigen Bedingungen hat keine gepasst."
fi
```

Spezielle Variablen

- `$0`: Name des gerufenen Skripts
- `$1 .. $n`: Aktualparameter 1 .. n
- `$*`: Alle Parameter der Kommandozeile
- `$#`: Anzahl der Parameter der Kommandozeile
- `$?`: Rückgebewert des letzen Aufrufs

Funktionen

Die Parameterübergabe erfolgt in gleicher Weise, wie bei Skriptaufruf von der Kommandozeile.

```
Funktionsname()
{
   Befehle
}
#
# oder
#
function Funktionsname
{
   Befehle
}
```

Aufgabe:[16]

Schreiben Sie das Skript `filetest.sh`, mit folgenden Eigenschaften:

- Das Sktipt muss mit 2 Parametern aufgerufen werden, der erste ist die Option, der zweite der Name einer Datei.
- Hat der erste Parameter den Wert

  `-a`: Prüfe, ob es sich beim 2. Parameter um eine Datei handelt. Hänge das aktulelle Datum (`date`) an die Datei.

  `-d`: Prüfe, ob es sich beim 2. Parameter um eine Datei handelt. Lösche die Datei.

  `-v`: Gib Programmname und Version aus

Aufgabe:[17]

Schreiben Sie das Skript `uhr.sh`, das in Abhängigkeit von der Tastatureingabe folgendes ausführt:

`d`: Ausgabe des aktuellen Datums
`v`: Ausgabe des aktuellen Verzeichnisses
`e`: Skript wird beendet
Andere Eingaben: Fehlermeldung

Aufgabe:

Schreiben Sie ein weiteres Skript `uhr1.sh`, das `uhr.sh` in drei Weisen mit Eingaben versorgt:

- Pipe: `cat input.txt | uhr.sh`
- Redirekt: `uhr.sh < input.txt`
- HERE-Dokument:

```
uhr.sh <<STOP
d
v
e
STOP
```

Zeichenketten

Ein weiteres nützliches Instrument bei der Shell-Programmierung ist das Abschneiden von Suffixes (%) und Präfixes (#). Werden die Zeichen einfach verwendet, wird die minimale Treffermenge abgeschnitten, werden sie doppelt verwendet (%% und ##), so wird die maximale Menge abgeschnitten.

```
name=MichaelWagner.txt
echo ${name%.txt}
echo ${name%.*}
# liefert MichaelWagner
echo ${name#Michael}
echo ${name#*l}
# liefert Wagner.txt
echo ${name#*e}
# liefert lWagner.txt
echo ${name##*e}
# liefert r.txt
```

Weitere Möglichkeiten der Zeichenketten-Verarbeitung unter `man bash` im Abschnitt „Parameter Expansion".

[16] [S10:795]
[17] [S10:801]

Aufgabe:

Jedes Skript bekommt unter dem Parameter `$0` seinen Aufrufpfad übergeben.

- Schreiben Sie ein Skript `aufruf.sh`, das
  - diesen Aufrufpfad ausgibt.
  - den Dateipfad vorne abschneidet und nur den Skriptnamen ausgibt.
- Verlinken Sie das Skript auf einen anderen Namen (`ln -s`) und rufen Sie das Skript erneut auf.

Kommandozeilenargumente

Die Verarbeitung von Kommandozeilenargumenten kann auch über die `getopts`-Funktionalität realisiert werden. Die Aufrufsyntax lautet:
`getopts <optstring> <var>`

`<optstring>` beschreibt die möglichen Optionen (ohne '-'): Falls die Optionen `-d -e -f` möglich sind, lautet der String: "def". Optionen, die von einem Wert gefolgt werden (z.B. `-d wert`) erhalten im *optstring* ein ':', also "d:ef".

`<var>` ist der Name der Variablen, in die die Option geschrieben wird. Hat die Option einen Wert, wird dieser in die Variable OPTARG geschrieben.

Hier ein Beispiel:

```
while getopts d:ef o
do  case "$o" in
  d)  echo "Wert=$OPTARG";;
  e)  echo "-e gesetzt";;
  f)  echo "-f gesetzt";;
  [?]) echo "Usage: $0 [-d wert] [-e] [-f]" >&2
    exit 1;;
  esac
done
```

Aufgabe:

- Testen Sie das kleine Beispiel für `getopts`
- Setzen Sie das Beispiel `filetest.sh` auf `getopts` um.

Alternativ steht auch `getopt` zur Verfügung: Dies unterstützt auch alternative Langnamen. Eine Beschreibung findet sich in `/usr/share/doc/util-linux/examples`.

5.2 Ein größeres Beispiel

In diesem Beispiel soll mit Skripten eine Büchereiverwaltung simuliert werden. Als Datenbank dienen drei Dateien:

- `buch.csv`

  ```
  <signatur>;<Autor>;<Titel>
  ```

- `benutzer.csv`

  `<benutzer-id>;<Nachname>;<Vorname>;<e-mail>`

- `ausleihe.csv`

  ```
  <ausleihe-id>;<signatur>;<benutzer-id>;<ausleihe-dat>; \\
     <rueckgabe-soll>;<rueckgabe-ist>
  ```

  Ist ein Buch ausgeliehen, so fehlt der Eintrag für `<rueckgabe-ist>`. Als Datumsformat wird `YYYY-MM-DD` verwendet.

Elementare Funktionen

Aufgabe:

Schreiben Sie eine Datei `funktion.sh` mit folgenden Funktionen:

- `get_buch <signatur>`: Anhand der Signatur wird die entsprechende Zeile auf `stdout` ausgegeben. Falls die Signatur nicht gefunden wird, wird ein entsprechender Rückgabewert gesetzt.
- `insert_buch <signatur> <Autor> <Titel>` hängt nach einer Prüfung, ob die Signatur nicht schon in Verwendung ist, den Datensatz an `buch.csv` an.
- `delete_buch <signatur>` entfernt das Buch mit der gegebenen Signatur aus `buch.csv`.
- `get_ausleihen` liefert alle aktuellen Ausleihen auf `stdout`.
- `get_nutzer <nutzer-id>` liefert den angeforderten Nutzer auf `stdout`. Falls der Nutzer nicht gefunden wird, wird ein entsprechender Rückgabewert gesetzt.
- `get_value <pos> <datensatz>` liefert den Wert an Position <pos> aus dem kommaseparierten Datensatz auf `stdout`.

Schreiben Sie eine Datei `buecherei.sh`, das die oben beschriebenen Funktionen testet. Die Datei mit den Funktionen wird über `. funktion.sh` geladen. Vergessen Sie nicht, die Rückgabewerte abzufragen.

Mahnmail

Aufgabe:

Schreiben Sie ein Skript, das eine Datei `mahn.csv` in folgenndem Format erstellt:

```
<e-mail>;<Nachname>;<Vorname>;<Signatur>;<Autor>;<Titel>;<rueckgabe-soll>
```

Gehen Sie dazu in folgenden Schritten vor:

- Lesen Sie alle aktuellen Ausleihen.
- Überprüfen Sie, ob `<rueckgabe-soll>` bereits in der Vergangenheit liegt.
  Hinweis: Der Befehl `date +%s -d DSTRING` wandelt die Datums-Zeichenkette `DSTRING` in Epochensekunden um.
- Falls ja, lesen Sie Buch und Nutzer und schreiben Sie eine Zeile in die Ausgabedatei.

Abverkauf von Büchern

Aufgabe:

Erstellen Sie sich eine Eingabedatei `verkauf.csv` mit folgendem Format:

```
<Signatur>;<preis>
```

Erstellen Sie ein Skript, das folgende Schritte ausführt:

- Titel und Autor holen, sowie das Exemplar in der Datenbank löschen
- Das Skript erstellt pro Buch eine txt-Datei, die Titel, Autor und Preis enthält.

6 Quellen

6.1 Literatur

Hildebrand (2008) Hildebrandt, Ralf; Koetter, Patrick Ben (2008): Postfix. Einrichtung, Betrieb und Wartung.

Kofler (2021) Kofler, Michael (2021): Linux. Das umfassende Handbuch, Rheinwerk Verlag GmbH, Bonn 2021

[Lin] D.J. Barret, „Linux, kurz&gut“, O'Reilly 2004.

[S10] S. Kersken,“SUSE Linux 10.x“, Galileo Computing 2006.

[TCP/IP] ICN TI Enabling. Kurs ICP/IP Grundlagen. Siemens 1999.

6.2 Quellen im Internet

{apache} Apache-Foundation, http://www.apache.org

{awk} awk-Manual, http://www.grymoire.com/Unix/Awk.html

{bash} Bash-Manual, http://tldp.org/LDP/abs/html

{bashdb} http://www.rodericksmith.plus.com/outlines/manuals/bashdbOutline.html

{bounce} https://en.wikipedia.org/wiki/Bounce_message

{dhcp} https://wiki.debian.org/DHCP_Server

{dnsmasq} https://wiki.debian.org/HowTo/dnsmasq

{distro} Übersicht über Linux-Distributionen, http://distrowatch.com

{ibm} http://www.ibm.com/developerworks/linux/library/l-job-scheduling/index.html

{ldap} https://www.debian.org/doc/manuals/debian-handbook/sect.ldap-directory.en.html

{lightsquid} http://lightsquid.sourceforge.net/

{linuxconfig} https://linuxconfig.org/linux-dns-server-bind-configuration

{masq} http://tldp.org/HOWTO/IP-Masquerade-HOWTO/

{mk} http://mathias-kettner.de/

{nfs} http://thesystemadministrator.net/linux-administration/how-to-install-and-configure-nfs-server-and-client-on-linux-remote-disk-access-with-nfs

{regex} Reguläre Ausdrücke, http://gnosis.cx/publish/programming/regular_expressions.html

{tldp} http://tldp.org/

{rsyslog} http://www.rsyslog.com/doc/v8-stable/configuration/basic_structure.html

{upstart} https://wiki.ubuntuusers.de/Upstart/

{wiki} Internet-Lexikon www.wikipedia.de. Stichworte: Linux, Apache Software Foundation