



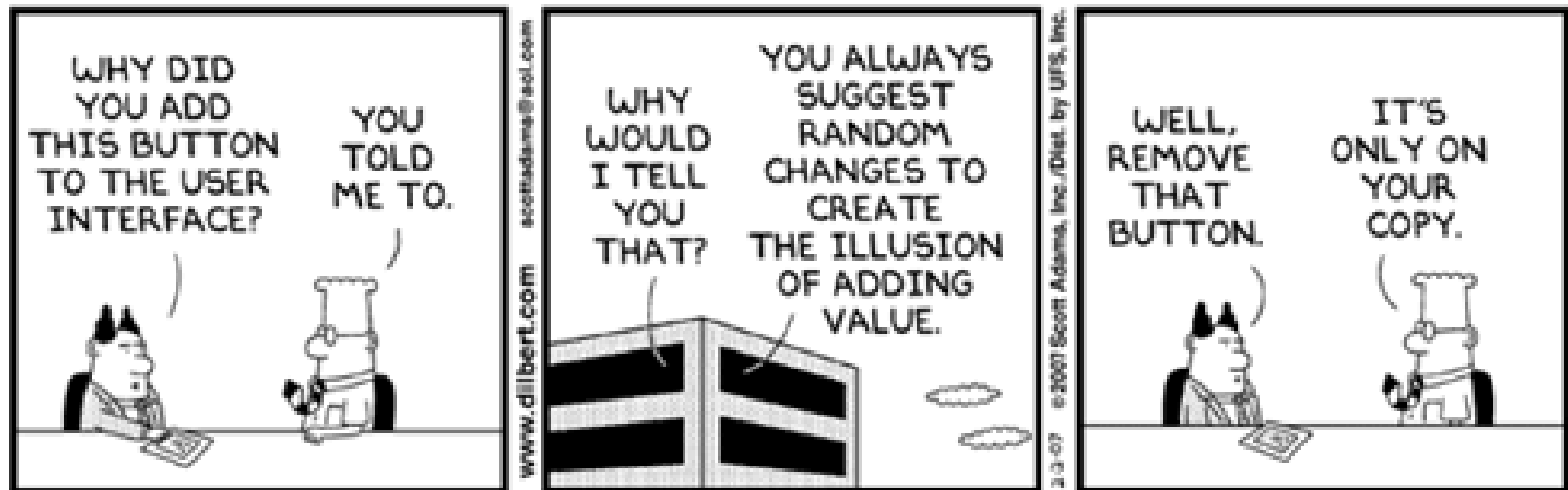
Feedback aus der Hausaufgabe

- Was ist Ihnen aufgefallen?
- Gab es grundlegende neue Erkenntnisse?
- Was hat gefehlt?
- Wieviel Zeit haben Sie aufgewendet?

Lektion 3: Benutzeroberfläche (Shell), Administration und Pflege

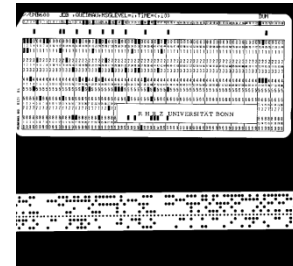
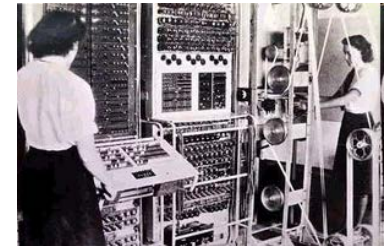


- Die "bash" Shell
- Scripting
- Systempflege
- Patches / Updates



Computer - Benutzerschnittstellen

- Mechanische Walzen
- Schalter / Glühbirnen
- Lochkarten / Lochstreifen
- Magnetband / Zeilendrucker
- Tastatur / CRT
- Grafische & akkustische Darstellung / Maus
- Virtual Reality / Haptische Geräte



- Die Shell (Befehlsübersetzer, Kommando-Interpreter) erlaubt dem Online-Benutzer die Interaktion mit dem Dateisystem, dem Prozess-Steuer-System sowie einfache Script-Programmierung.
- Die Login-Shell eines Benutzers wird in der Datei `/etc/passwd` definiert.
- Die Shell ist aus Sicht von Unix / Linux lediglich ein Benutzerprogramm ohne besondere Privilegien – es kann daher leicht ersetzt werden.

- Bourne Shell: sh (ursprüngliche Shell)
- C-Shell: csh (Scripting in C-Syntax)
- Korn-Shell: ksh (Mix aus sh und csh)

- Bourne Again Shell: bash (ist vollständig kompatibel zu der originalen Bourne-Shell, mit diversen Erweiterungen. Sie ist bei fast allen Linux-Systemen die Standard-Shell.

- Sobald die Shell bereit ist, Kommandoeingaben anzunehmen, meldet sie sich mit einem Bereitschaftszeichen (Prompt). Im einfachsten Fall ist dies ein "\$"-Zeichen als normales Prompt, dem "#" als Prompt für Superuser und dem ">"-Prompt, wenn noch weitere Eingaben erwartet werden. Wie vieles in der Shell, kann der Prompt beliebig modifiziert werden.
- Kommandozeile vom Terminal annehmen, das Kommando entschlüsseln und in seine Komponenten (Kommandoname, Optionen, Parameter) aufteilen
- Gewünschtes Programm suchen und starten. Der Suchweg wird durch eine Shellvariable PATH vorgegeben, diese und andere Variablen finden sich in entsprechenden Konfigurationsdateien, die beim Start der Shell eingelesen werden.
- Steuerung der Ein- und Ausgabe des Terminals. UNIX ist als Dialogsystem konzipiert und war ursprünglich für den Betrieb mit Fernschreibern (Teletype) und Datensichtgerät konzipiert. Diese sogenannten "Terminals" findet man heute noch in den *logischen* Terminals, z. B.:
 - den Konsolen, die dem Bildschirm und der Tastatur des PCs zugeordnet sind,
 - den Pseudo-Terminals, die einer Telnet-Verbindung zugeordnet werden oder
 - den X-Terminal-Fenstern auf der grafischen Benutzeroberfläche.

- Es gibt drei virtuelle Standarddateien, die dem aktuellen Terminal zugeordnet sind:
 - Standardeingabe (Tastatur, *stdin*)
 - Standardausgabe (Bildschirm, *stdout*)
 - Standard-Fehlerausgabe (Bildschirm, *stderr*)
- Die Shell kann angewiesen werden, die Ein- und Ausgaben umzuleiten.
- Ersetzung von Sonderzeichen und Befehlssubstitution. Kommando-eingaben können durch Sonderzeichen erweitert und der Kommandostring kann selbst durch die Shell expandiert werden.
- Ablaufsteuerung. Die Shell beherrscht viele Strukturen, wie man sie von Programmiersprachen her kennt (Test, bedingte Anweisung, Schleifen, Variablen, Rechenanweisungen).
- Erzeugen von Kind- und Hintergrundprozessen.
- Es gibt weiterhin die Möglichkeit, Programme "im Hintergrund" zu starten und am Bildschirm weiterzuarbeiten, während das Programm läuft.

Aufgaben der Shell III: Kommando-Interpretation

1. Ein Kommando wird erst ausgeführt, wenn der Benutzer am Ende der Kommandozeile die RETURN-Taste drückt.
2. Die Shell liest bis zum ersten Kommandotrenner („&“ „&&“ „||“ „;“ „>“ „<“ „>“) und stellt fest, ob Variablenzuweisungen erfolgen sollen oder die Ein-Ausgabe umgelenkt werden muss.
3. Die Shell zerlegt die Kommandozeile in einzelne Argumente. Sie trennt die einzelnen Argumente durch eines der Zeichen, die in der Shell-Variablen IFS (Internal Field Separator) stehen, normalerweise Leerzeichen, Tabs und Newline-Zeichen.
4. Variablenreferenzen, die in der Kommandozeile stehen, werden durch ihre Werte ersetzt.
5. Kommandos, die in `...` oder bei der bash in \$(...) stehen, werden ausgeführt und durch ihre Ausgabe ersetzt.
6. *stdin*, *stdout* und *stderr* werden auf ihre "Zielfile" umgelenkt.
7. Falls in der Kommandozeile noch Zuweisungen an Variablen stehen, werden diese ausgeführt.
8. Die Shell sucht nach Jokerzeichen und ersetzt diese durch passende Dateinamen.
9. Die Shell führt das Kommando aus.

- Die drei dem Terminal zugeordneten Dateikanäle *stdin*, *stdout* und *stderr* können jederzeit auf Dateien umgeleitet werden. Den drei Standarddateien sind die Filehandles 0 (*stdin*), 1 (*stdout*) und 2 (*stderr*) zugeordnet.
- **Eingabeumleitung:** („mail lubich < dateiname“) Das Programm liest nun nicht mehr von der Tastatur (*stdin*), sondern aus einer Datei, bis das Dateiende erreicht ist.
- **Ausgabeumleitung:** („ls > dateiname“) Die Ausgabe des Programms wird nicht auf dem Bildschirm (*stdout*) ausgegeben, sondern in einer Datei geschrieben.
- **Umlenkung der Fehlerausgabe (*stderr*):** („ls 2> logdatei“) Die Umleitung der Fehlerausgabe erfolgt genauso, wie die Ausgabeumleitung, jedoch wird hier die Zeichenfolge "2>" verwendet, da *stderr* die Terminalnummer 2 hat. Natürlich ist eine beliebige Kombination von Ein- und Ausgabeumleitung möglich, z. B. „Kommando < Eingabedatei > Ausgabedatei 2> Fehlerdatei“
- **Anhängen von Infos an eine Datei:** („ls >> dateiname“) Es ist auch möglich, die Ausgabe des Programms an eine bereits vorhandene Datei anzuhängen.

- Die Umleitung von Ein- und Ausgabe lässt sich auch unterdrücken. Für die Ausgabe schreibt man „Kommando > /dev/null“, oder für die Fehlerausgabe „Kommando 2> /dev/null“.
- Beides lässt sich kombinieren: „Kommando > Ergebnisdatei 2> /dev/null“.
- Will man ein Programm mit einem beliebigen Eingabedatenstrom versorgen, schreibt man „Kommando < /dev/zero“.
- Die Umleitung von *stdout* und *stderr* in dieselbe Datei würde prinzipiell eine zweimalige Angabe der Datei (eventuell mit einem langen Pfad) erfordern. Für die Standarddateien werden in solchen Fällen spezielle Platzhalter verwendet:
 - &0 Standardeingabe
 - &1 Standardausgabe
 - &2 Standard-Fehlerausgabe
 - Beispiel: „Kommando > ausgabe 2>&1“

- Eine „Pipe“ (Röhre) verbindet zwei Kommandos über einen temporären Puffer, d. h. die Ausgabe vom ersten Programm wird als Eingabe vom zweiten Programm verwendet. Alles, was das erste Programm in den Puffer schreibt, wird in der gleichen Reihenfolge vom zweiten Programm gelesen. Pufferung und Synchronisation werden vom Betriebssystem vorgenommen. Der Ablauf beider Prozesse kann nebenläufig erfolgen, die Flusskontrolle regelt das Betriebssystem. In einer Kommando-folge können mehrere Pipes vorkommen.
- Der Pipe-Mechanismus wird durch das Zeichen "|" (senkrechter Strich) aktiviert. Beispiel: „ls | more“.

- Es können auch mehrere Kommandos hintereinander durch Pipes verbunden werden: „Kommando 1 | Kommando 2 | Kommando 3 | Kommando 4 | ...“
- Pipelines haben Vorrang vor anderen Formen der Ein- und Ausgabeumleitung. Bevor die Shell mit der Ausführung der Einzelbefehle und der dazugehörenden Umleitungen beginnt, baut sie die gesamte Pipeline zusammen: Sie erzeugt für jeden Abschnitt einen neuen Prozess und verbindet die Standardausgabe jedes Prozesses mit der Standardeingabe des Nächsten.
- Will der Anwender die Fehlerausgabe eines Programms ebenfalls durch die Pipeline schicken, kann er in Bourne-Shell-Skripten „Programm1 2>&1 | Programm2“ schreiben. Da die Shell die Pipeline-Verbindung zuerst herstellt, landen die Fehlermeldungen vom Programm1 auf der Standard-Eingabe von Programm2.

- Kommandofolgen, die durch Pipes verbunden sind, werden als "Filter" bezeichnet. Einige nützliche Filter sind in jedem UNIX/Linux-System verfügbar, z.B.:
 - „head [datei(en)]“: Ausgabe der ersten n Zeilen aus den angegebenen Dateien. Voreinstellung ist 10 Zeilen. Wird keine Datei angegeben, liest head von der Standardeingabe.
 - „tail [-/+n] [datei]“: Ausgabe der letzten n Zeilen einer Datei. Voreinstellung für n ist 10. Wird keine Datei angegeben, liest tail von der Standardeingabe.
 - „more“: seitenweise Ausgabe des *stdin*.
 - „wc“ (word count): Zählen von Wörtern im *stdin*.
 - „sort“: Sortieren des *stdin* nach diversen Kriterien.
 - tee [datei]“: Pipe mit T-Stück: Kopiert von *stdin* nach *stdout* und schreibt die Daten gleichzeitig in die angegebene Datei

Sonderzeichen bei der Expansion von Dateinamen

- *: Der Stern steht für eine beliebige Zeichenfolge - oder für überhaupt kein Zeichen. Beispiel: "ab*" steht für alle Dateinamen, die mit "ab" anfangen, auch für "ab" selbst ("ab", "abc", "abcd", "abxyz", usw.).
- ?: Das Fragezeichen steht für genau ein beliebiges Zeichen. Beispiel: "?bc" steht für alle Dateinamen mit 3 Zeichen, die auf "bc" enden ("abc", "bbc", "1bc", "vbc", "xbc", usw.), nicht jedoch für "bc".
- []: Die eckige Klammer wird ersetzt durch **eines** der in der Klammer stehenden Zeichen. Auch ein Bereich ist möglich, z. B. [a-k] = [abcdefghijk]. Beispiel: "a[bcd]" wird ersetzt durch "ab", "ac" und "ad". Soll das Minuszeichen selbst in die Zeichenmenge aufgenommen werden, muss es an erster Stelle stehen (gleich nach der öffnenden Klammer).
- [!]: Die eckige Klammer mit Ausrufezeichen wird ersetzt durch eines der **nicht** in der Klammer stehenden Zeichen. Beispiel: "[!abc]" wird ersetzt durch ein beliebiges Zeichen außer a, b oder c. Soll das Ausrufezeichen selbst in die Zeichenmenge aufgenommen werden, muß es an letzter Stelle stehen.
- \: Der Backslash hebt den Ersetzungsmechanismus für das folgende Zeichen auf. Beispiel: "ab\?cd" wird zu "ab?cd" - das Fragezeichen wird übernommen. **Wichtig:** Bei der Umleitung von Ein- und Ausgabe werden Metazeichen in den Dateinamen hinter dem Umleitungszeichen nicht ersetzt.
- Achtung: Der * ist ein gefährliches Zeichen, Tippfehler können zum Fiasko führen, wenn aus Versehen ein Leerzeichen zuviel getippt wird („rm a*" löscht beispielsweise alle Dateien, die mit "a" anfangen, „rm a *" löscht dagegen erst die Datei "a" und dann alle Dateien im Verzeichnis – ohne Rückfrage).

Sonderzeichen bei der Expansion von Zeichenketten

- Um bestimmte Sonderzeichen (z. B. *, ?, [], Leerzeichen, Punkt) zu übergeben, ohne daß sie von der Shell durch Dateinamen ersetzt werden, werden Anführungszeichen verwendet, die auch ineinander geschachtelt werden können.
- Dabei haben die drei verschiedenen Anführungszeichen Doublequote ("), Quote (') und Backquote (`) unterschiedliche Bedeutung:
- `"`, `'`: Keine Ersetzung der Metazeichen * ? [], jedoch Ersetzung von Shellvariablen (siehe unten) und Ersetzung durch die Ergebnisse von Kommandos (Backquote). Auch `\` funktioniert weiterhin. Beispiel:
 - `echo Der *` wird hier durch alle Dateinamen ersetzt
 - `echo "Der * wird hier nicht ersetzt"`
- `'` 'Das einfache Anführungszeichen unterdrückt jede Substitution. Beispiel:
 - `echo 'Weder * noch `pwd` werden ersetzt'`
- ``` `Zwischen Backquote (Accent Grave) gesetzte Kommandos werden ausgeführt und das Ergebnis wird dann als Parameter übergeben (d. h. die Ausgabe des Kommandos landet als Parameter in der Kommandozeile). Dabei werden Zeilenwechsel zu Leerzeichen. Braucht dieses Kommando Parameter, tritt die normale Parameterersetzung in Kraft. Beispiel:
 - `echo "Aktuelles Verzeichnis: `pwd`"`
- Weil die verschiedenen Quotes manchmal schwer zu unterscheiden sind, wurde bei der *bash* eine weitere Möglichkeit eingeführt. Statt in Backquotes wird die Kommandofolge in `$(...)` eingeschlossen., z. B.:
 - `echo "Aktuelles Verzeichnis: $(pwd)"`

Navigation im Dateisystem

- Das Unix/Linux-Dateisystem entspricht einer Baumstruktur mit der Wurzel ganz oben und Directories als Strukturelement.
- Pfadnamen sind absolut („/bin/ls“) oder relativ, wobei „.“ das aktuelle und „..“ das Elterndirectory angibt („./bin/ls“ oder „../../directory1/datei1“).
- Anzeigen des aktuellen Directory: `pwd`
- Wechseln des Directories: „`cd [Pfad]`“
- Anzeige des Inhalts eines Directories: „`ls [Pfad]`“

Basiskommandos unter Linux

- ls: Anzeige eines Directories
- more: seitenweise Ausgabe
- cat: Ausgabe ohne Zwischenstopps
- cp: Kopien von Dateien
- rm: Löschen von Dateien
- mkdir / rmdir: Anlage / Löschen von Directories
- mv: Verschieben von Dateien
- chown, chmod: Ändern von Zugriffsrechten (Lesen; Schreiben, Ausführen für Besitzer, Gruppe und Rest) von Dateien / Directories
- ps: Anzeigen der Prozessliste
- kill: Terminieren von laufenden Prozessen
- date: Anzeige von Datum und Uhrzeit
- passwd: Ändern des Passwortes
- man: Abfrage von Online-Hilfetexten
- exit: Ausloggen / Terminieren der Shell



Übung (ca. 30 min.)

- Aufgabe(n) gemäss separatem Aufgabenblatt
- Lösungsansatz: Einzelarbeit oder Gruppen von max. 3 Personen
- Hilfsmittel: beliebig
- Besprechung möglicher Lösungen in der Klasse (es gibt meist nicht die eine «Musterlösung»)

Übungsbesprechung (ca. 15 min.)

- Stellen Sie Ihre jeweilige Lösung der Klasse vor.
- Zeigen Sie auf, warum ihre Lösung korrekt, vollständig und effizient ist.
- Diskutieren Sie ggf. Design-Entscheide, Alternativen oder abweichende Lösungsansätze.
- Gibt es Unklarheiten? Stellen Sie Fragen.



- Viele Unix-Programme (insbes. Shells) erlauben das Schreiben fast beliebig komplexer Scripte (d.h. interpretiert ablaufender Programmschritte mittels einfacher Sprach-/Strukturelemente und I/O-Redirection / Pipelining).
- Zusätzlich existieren dedizierte Script-Sprachen und Umgebungen (z.B. perl, awk, sed).
- Viele Unix/Linux-Kommandos sind als Filter zur Verwendung in Pipelines / Scripts programmiert.
- Scripting ist für kleine, schnelle Problemlösungen gedacht und geeignet, es ist aber kein Ersatz für systematisches Programmieren.

Ein erstes sh-Script

```
lubich@ubuntu: cat ./test1.sh  
echo "Hello, World"
```

```
lubich@ubuntu: /bin/sh ./test1.sh  
Hello, World  
lubich@ubuntu:
```

Ein zweites sh-Script

```
lubich@ubuntu: cat ./test2.sh
#!/bin/sh
echo -n "Anzahl Dateien in /etc ist: "
ls -lasg /etc | wc -l
echo "Done"
lubich@ubuntu: chmod 700 test2.sh
lubich@ubuntu: ./test2.sh
Anzahl Dateien in /etc ist: 229
Done
lubich@ubuntu:
```

Sprachelemente: Variablen

- Deklaration: `VAR = Wert`
- Benutzung `$VAR` oder `${VAR}`, wenn anderer Text folgt.

```
COLOR=yellow
```

```
echo This looks $COLORish
```

```
echo This seems ${COLOR}ish
```

- ergibt:

```
This looks
```

```
This seems yellowish
```

Sprachelemente: Spezielle Variablen

- Argumente der Kommandozeile: \$1, \$2, ... bis zu \$9
- \$0 ist der Name des Scripts
- \$* = gesamte Kommandozeile
- \$# = Anzahl der Argumente
- \$? = exit-Status des letzten Kommandos
- \$- = alle aktiven Shell-Optionen
- \$\$ = Prozess-ID
- \$! = Prozess-ID des letzten Hintergrund-Kommandos
- ...

- „*“ matcht null oder mehr Zeichen
- „?“ matcht genau ein Zeichen
- [Wertebereich] matcht jedes Zeichen im Wertebereich
- [ak3] matcht a, k, oder 3.
- [a-z] matcht alle Kleinbuchstaben
- [a-mz] matcht alle Kleinbuchstaben von a bis m, sowie z
- Taucht ein Muster innerhalb eines Kommandos auf, ersetzt die Shell vor der Ausführung die entsprechenden Muster (globbing):
 - `ls *.c; ls ?.c; ls [a-d]*;`

Sprachelemente: Flusskontrolle I

```
if Bedingung; then
    Kommandos
elif Bedingung; then
    Kommandos
else
    Kommandos
fi
```

```
#!/bin/sh
myname=`whoami`
if [ $myname = root ]; then
    echo "Welcome"
else
    echo "Must be root to continue"
    exit 1
fi
```

Sprachelemente: Flusskontrolle II

```
while Bedingung; do  
    Kommandos  
done
```

```
for Variable in Liste; do  
    Kommandos  
done
```

Beide Loops können durch `break` und `continue` gesteuert werden.

Sprachelemente: Flusskontrolle III

```
case Ausdruck in
    Muster1) Kommandos;;
    Muster2) Kommandos;;
esac
```

Pipelining, Input-/Output Redirection: bekannt

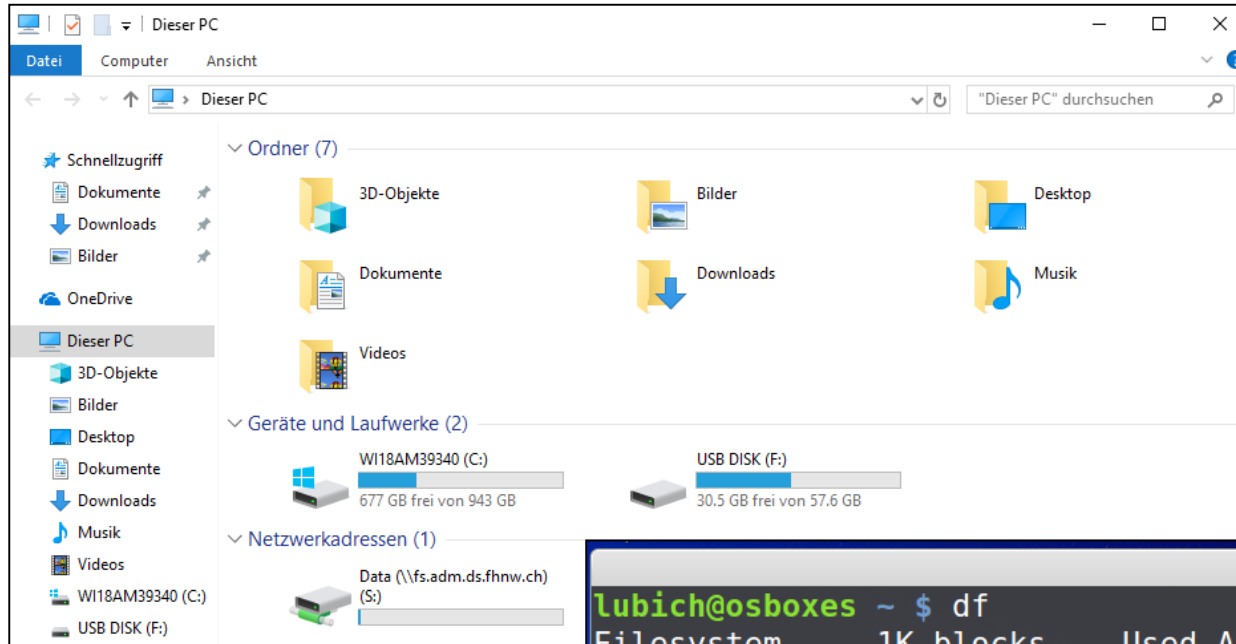
Funktionen:

```
Name ( ) {
    Kommandos
}
```

Aufruf:

```
Name Argument1, Argument2, ...
```

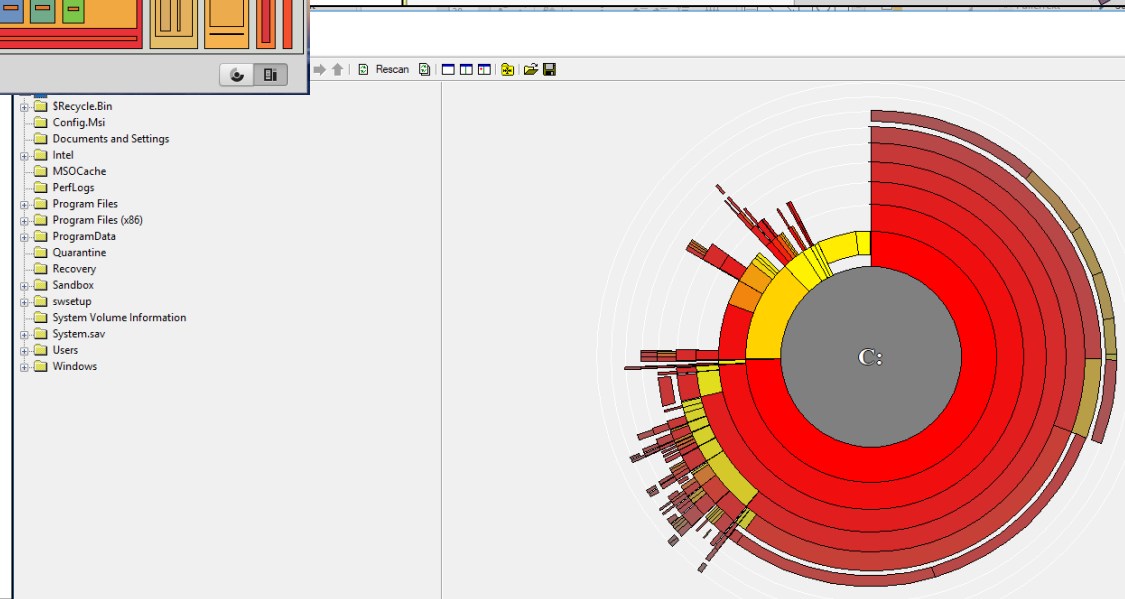
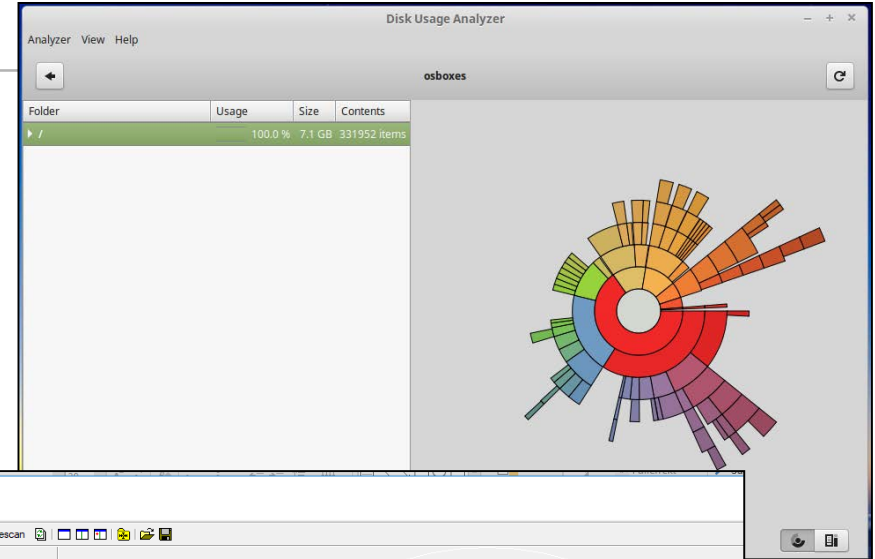
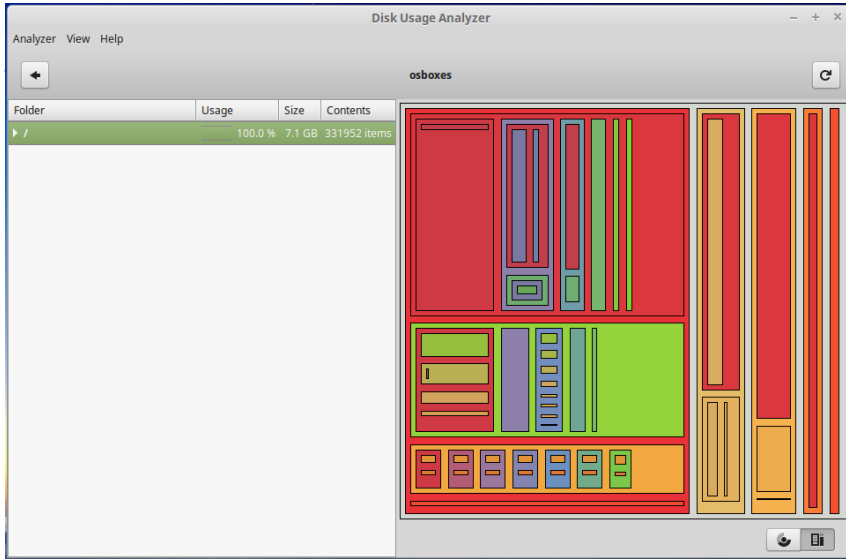




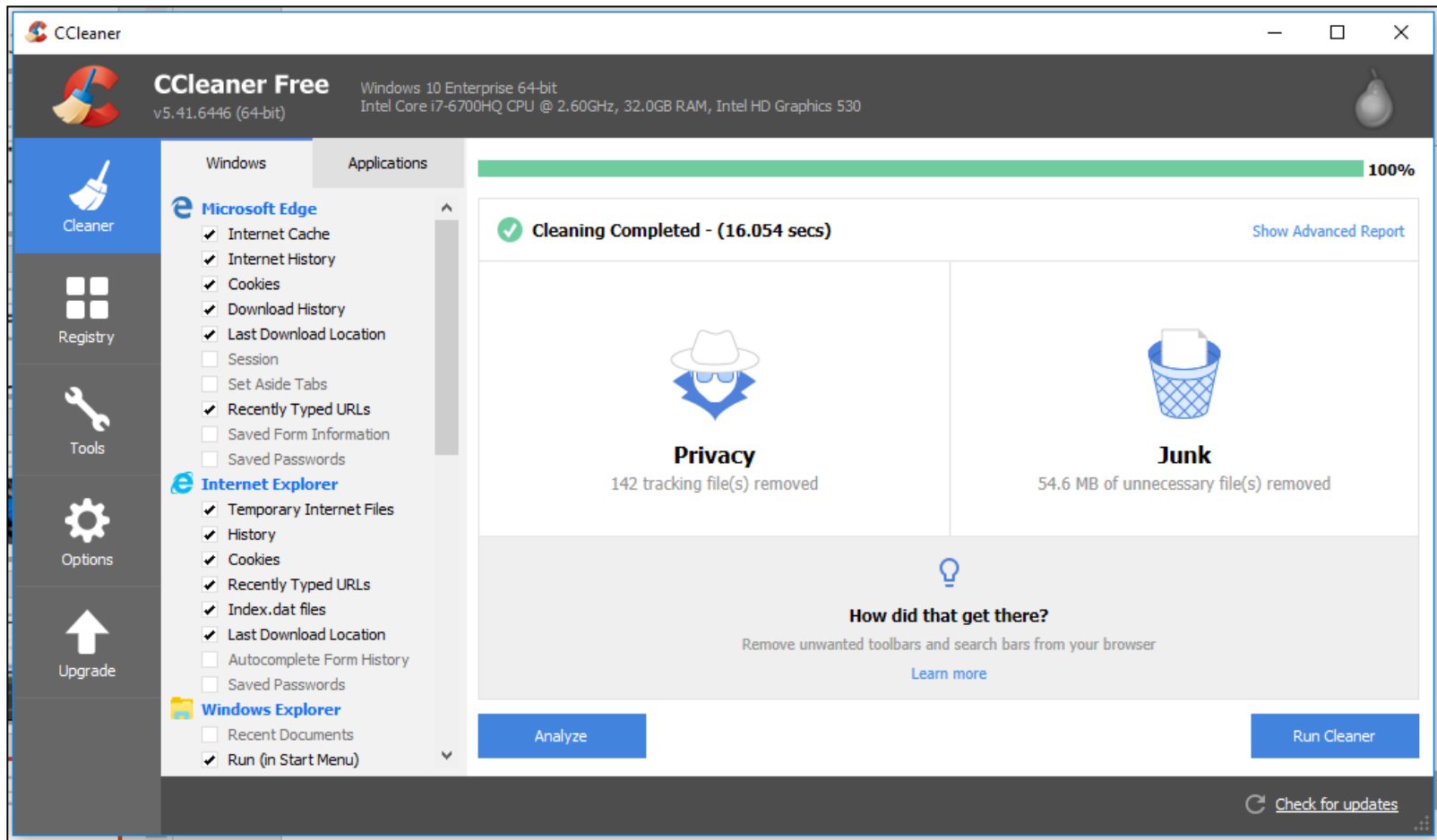
```

Terminal
lubich@osboxes ~ $ df
Filesystem      1K-blocks    Used Available Use% Mounted on
udev             2051808         4   2051804   1% /dev
tmpfs            413920       1012    412908   1% /run
/dev/sda1       49409840 7330928 39546000  16% /
none              4              0         4   0% /sys/fs/cgroup
none             5120           0        5120   0% /run/lock
none            2069584        960   2068624   1% /run/shm
none            102400         12    102388   1% /run/user
lubich@osboxes ~ $
  
```

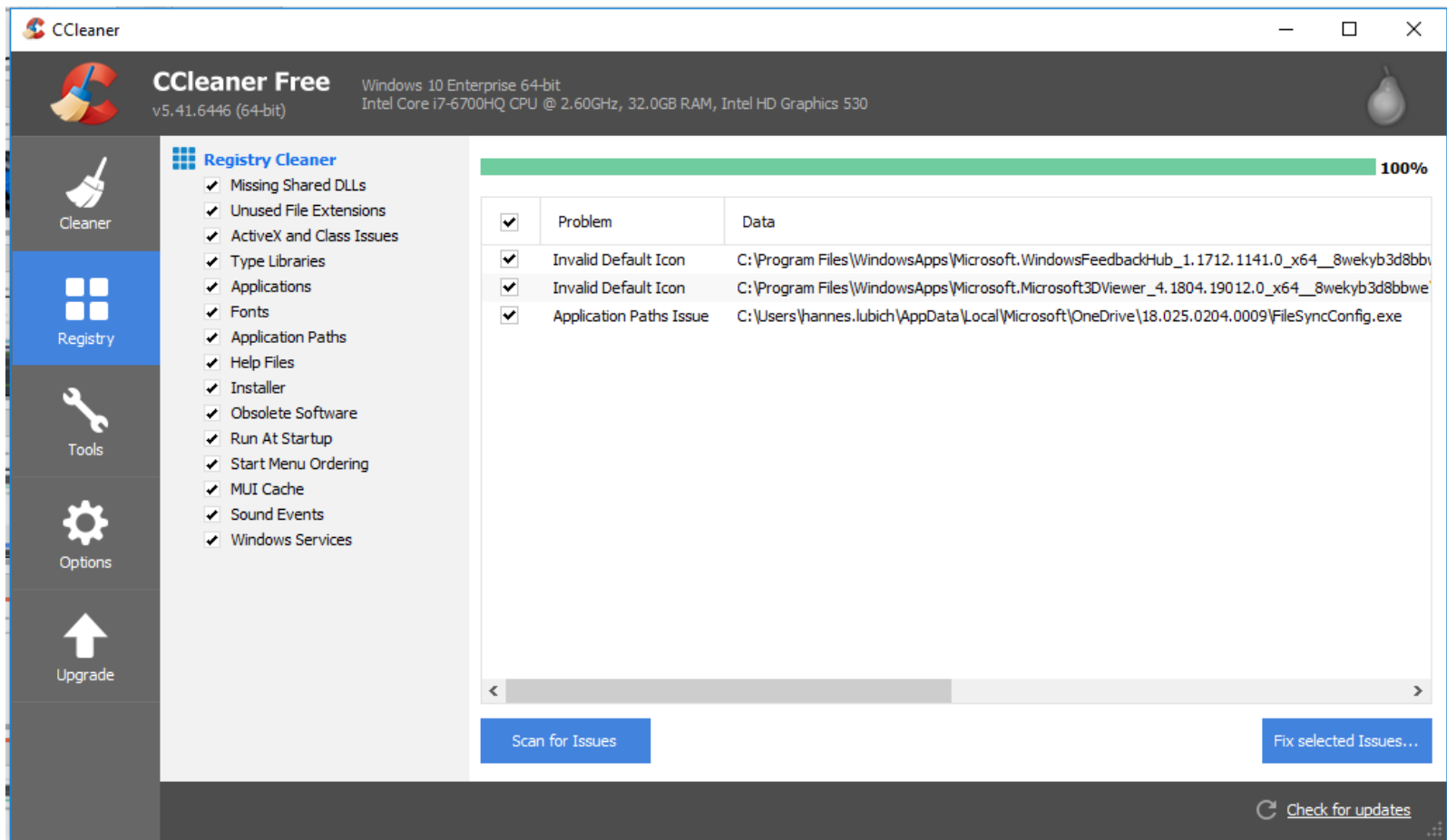
Habe ich noch Platz?



Aufräumen Windows: CCleaner

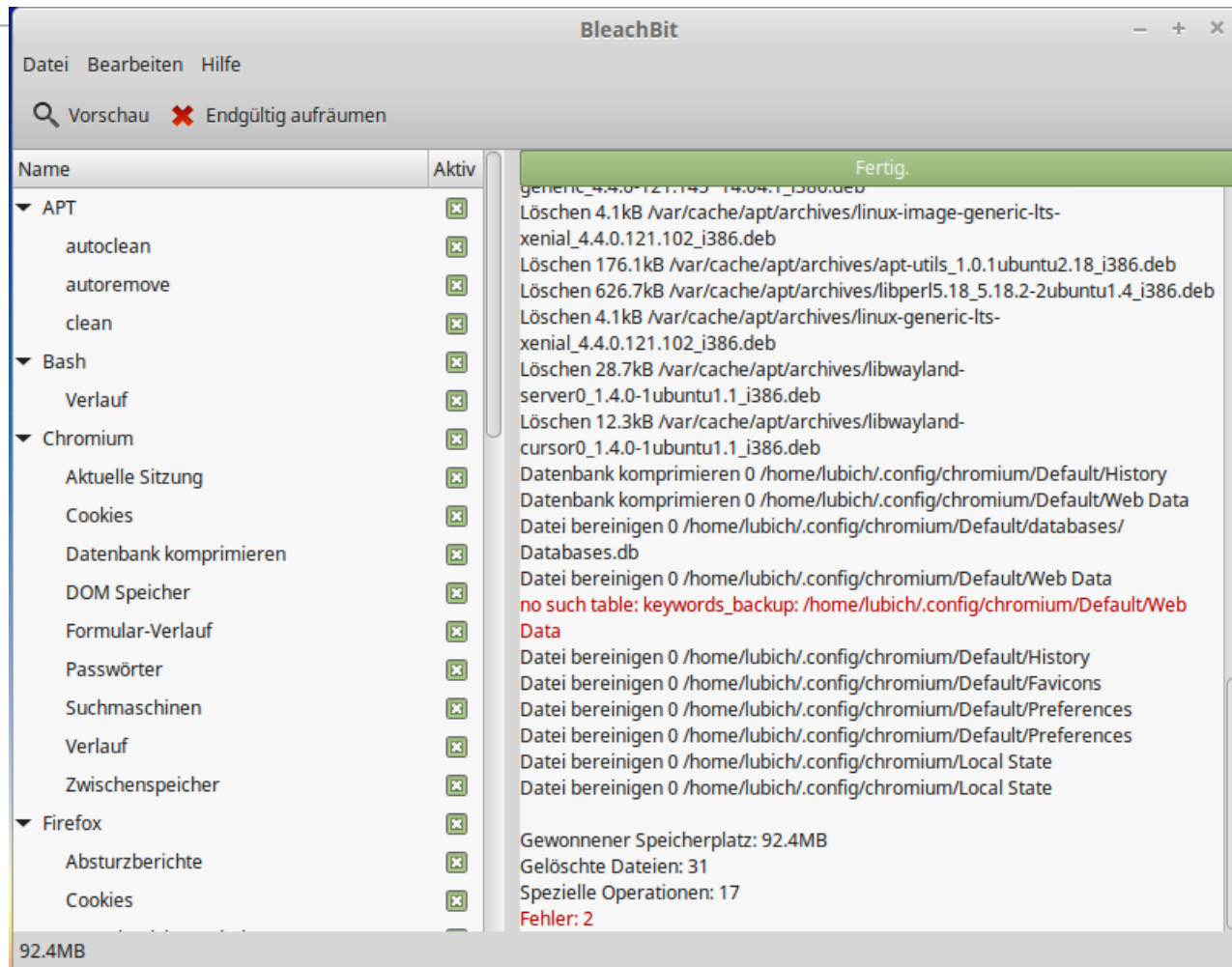


Aufräumen Windows: CCleaner



Aufräumen

Linux: Bleachbit (als root)



Aufräumen

Linux: /lib/modules

```
Terminal
lubich@osboxes ~ $ cd /lib/modules
lubich@osboxes /lib/modules $ ls
3.16.0-38-generic  4.4.0-111-generic  4.4.0-116-generic  4.4.0-121-generic
4.4.0-109-generic  4.4.0-112-generic  4.4.0-119-generic
lubich@osboxes /lib/modules $ du -hs * | sort -hr
155M    4.4.0-121-generic
155M    4.4.0-119-generic
145M    3.16.0-38-generic
37M     4.4.0-116-generic
36M     4.4.0-112-generic
36M     4.4.0-111-generic
36M     4.4.0-109-generic
lubich@osboxes /lib/modules $
```


Aufräumen Linux: Synaptic Paket Mngr

Synaptic-Paketverwaltung

Datei Bearbeiten Paket Einstellungen Hilfe

Schnellauswahl-Filter

Neu laden Anwenden Eigenschaften Suche

Alle	S	Paket	Installierte Version	Neueste Version	Beschreibung
Installiert		linux-generic-his-xenial	4.4.0-121.102	4.4.0-121.102	Complete Generic Linux kernel and headers
Installiert (aktualisierbar)		usbutils	1:007-2ubuntu1.1	1:007-2ubuntu1.1	Linux USB-Hilfsprogramme
Installiert (automatisch entfernbar)		libbluetooth3	4.101-0ubuntu13.3	4.101-0ubuntu13.3	Bibliothek zur Nutzung des Bluetooth-Protokollstacks BlueZ
Installiert (lokal oder veraltet)		manpages-dev	3.54-1ubuntu1	3.54-1ubuntu1	Handbuchseiten für die Software-Entwicklung unter GNU/Linux
Installiert (manuell)		libsctp1	1.0.15+dfsg-1	1.0.15+dfsg-1	Benutzerbereichszugriff auf Linux-Kernel-SCTP - Gemeinsame Bibliothek
Nicht installiert		linux-headers-3.16.0-38	3.16.0-38.52~14.04.1	3.16.0-38.52~14.04.1	Header files related to Linux kernel version 3.16.0
Nicht installiert (zurückgebliebene)		linux-headers-4.4.0-109	4.4.0-109.132~14.04.1	4.4.0-109.132~14.04.1	Header files related to Linux kernel version 4.4.0
		linux-headers-4.4.0-111	4.4.0-111.134~14.04.1	4.4.0-111.134~14.04.1	Header files related to Linux kernel version 4.4.0
		linux-headers-4.4.0-112	4.4.0-112.135~14.04.1	4.4.0-112.135~14.04.1	Header files related to Linux kernel version 4.4.0
		linux-headers-4.4.0-116	4.4.0-116.140~14.04.1	4.4.0-116.140~14.04.1	Header files related to Linux kernel version 4.4.0
		linux-headers-4.4.0-119	4.4.0-119.143~14.04.1	4.4.0-119.143~14.04.1	Header files related to Linux kernel version 4.4.0
		linux-headers-4.4.0-121	4.4.0-121.145~14.04.1	4.4.0-121.145~14.04.1	Header files related to Linux kernel version 4.4.0
		fonts-lklug-sinhala	0.6-3	0.6-3	Singhalesische Unicode-Schrift der Lanka Linux User Group
		kmod	15-0ubuntu6	15-0ubuntu6	Werkzeuge zum Verwalten von Linux-Kernel-Modulen
		linux-headers-3.16.0-38-generic	3.16.0-38.52~14.04.1	3.16.0-38.52~14.04.1	Linux kernel headers for version 3.16.0 on 32 bit x86 SMP
		linux-headers-4.4.0-109-generic	4.4.0-109.132~14.04.1	4.4.0-109.132~14.04.1	Linux kernel headers for version 4.4.0 on 32 bit x86 SMP
		linux-headers-4.4.0-111-generic	4.4.0-111.134~14.04.1	4.4.0-111.134~14.04.1	Linux kernel headers for version 4.4.0 on 32 bit x86 SMP
		linux-headers-4.4.0-112-generic	4.4.0-112.135~14.04.1	4.4.0-112.135~14.04.1	Linux kernel headers for version 4.4.0 on 32 bit x86 SMP
		linux-headers-4.4.0-116-generic	4.4.0-116.140~14.04.1	4.4.0-116.140~14.04.1	Linux kernel headers for version 4.4.0 on 32 bit x86 SMP
		linux-headers-4.4.0-119-generic	4.4.0-119.143~14.04.1	4.4.0-119.143~14.04.1	Linux kernel headers for version 4.4.0 on 32 bit x86 SMP

Sektionen

Status

Ursprung

Benutzerdefinierte Filter

Suchergebnisse

Architecture

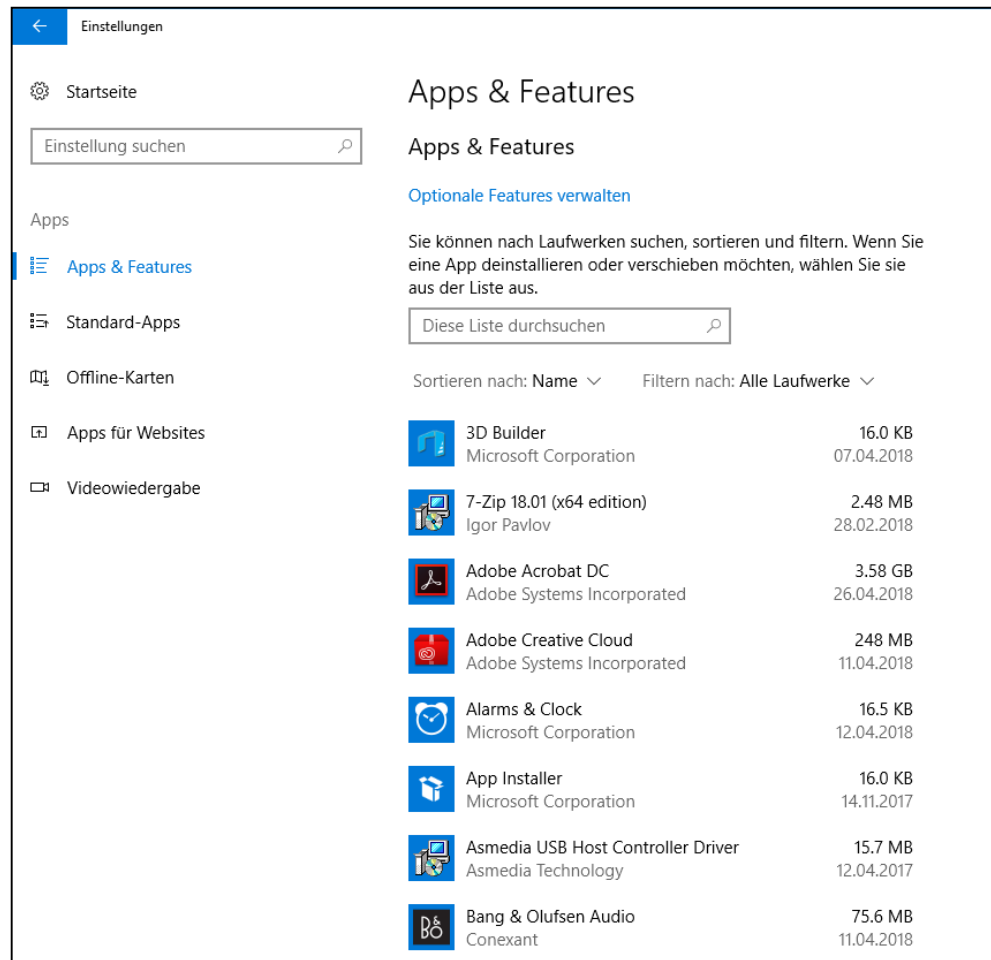
Header files related to Linux kernel version 4.4.0

Bildschirmfoto herunterladen Änderungsprotokoll abrufen

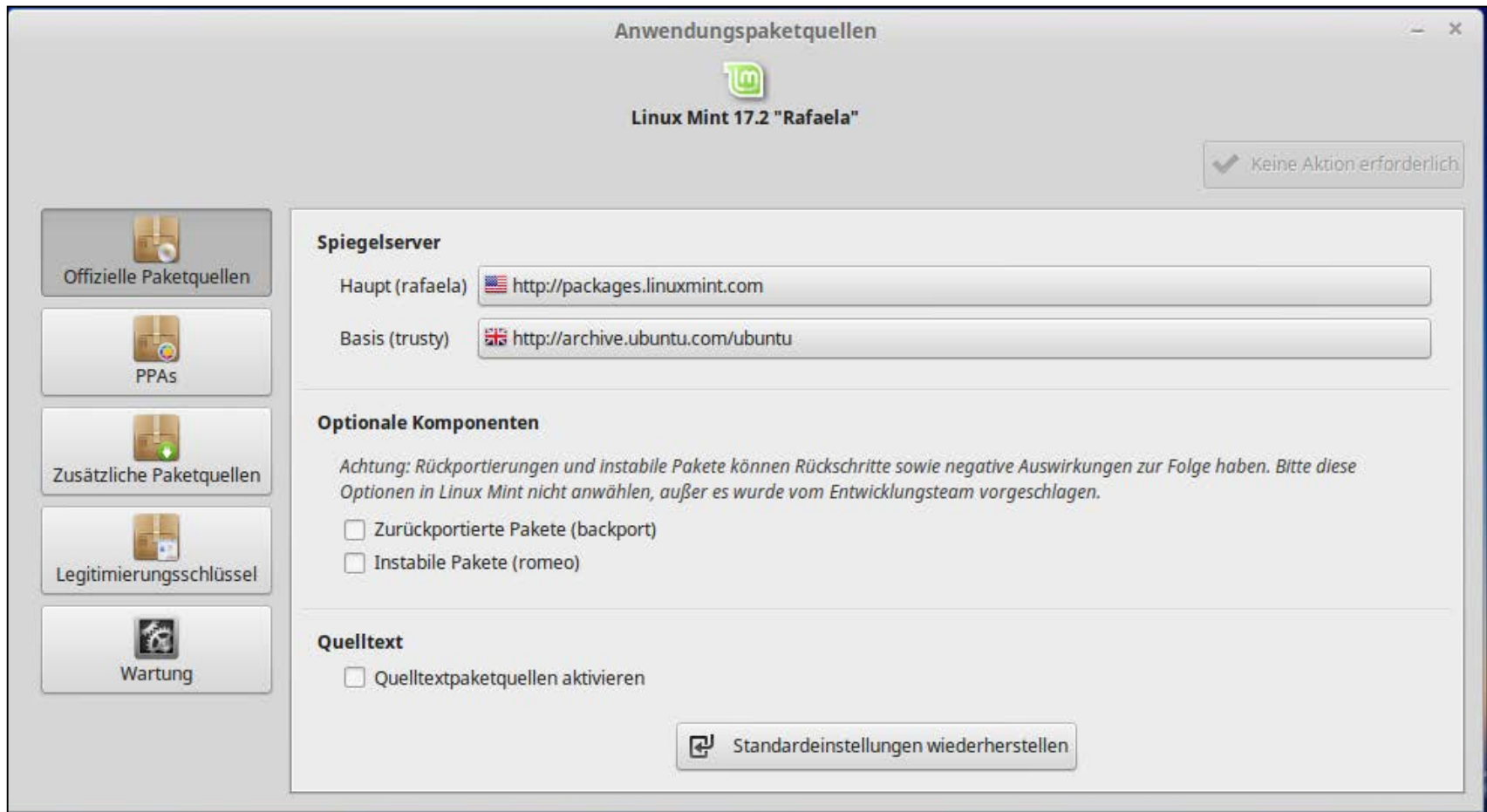
This package provides kernel header files for version 4.4.0, for sites

206 Pakete aufgelistet, 1985 installiert, 0 defekt, 0 werden installiert oder aktualisiert, 10 werden entfernt, 425 MB werden frei

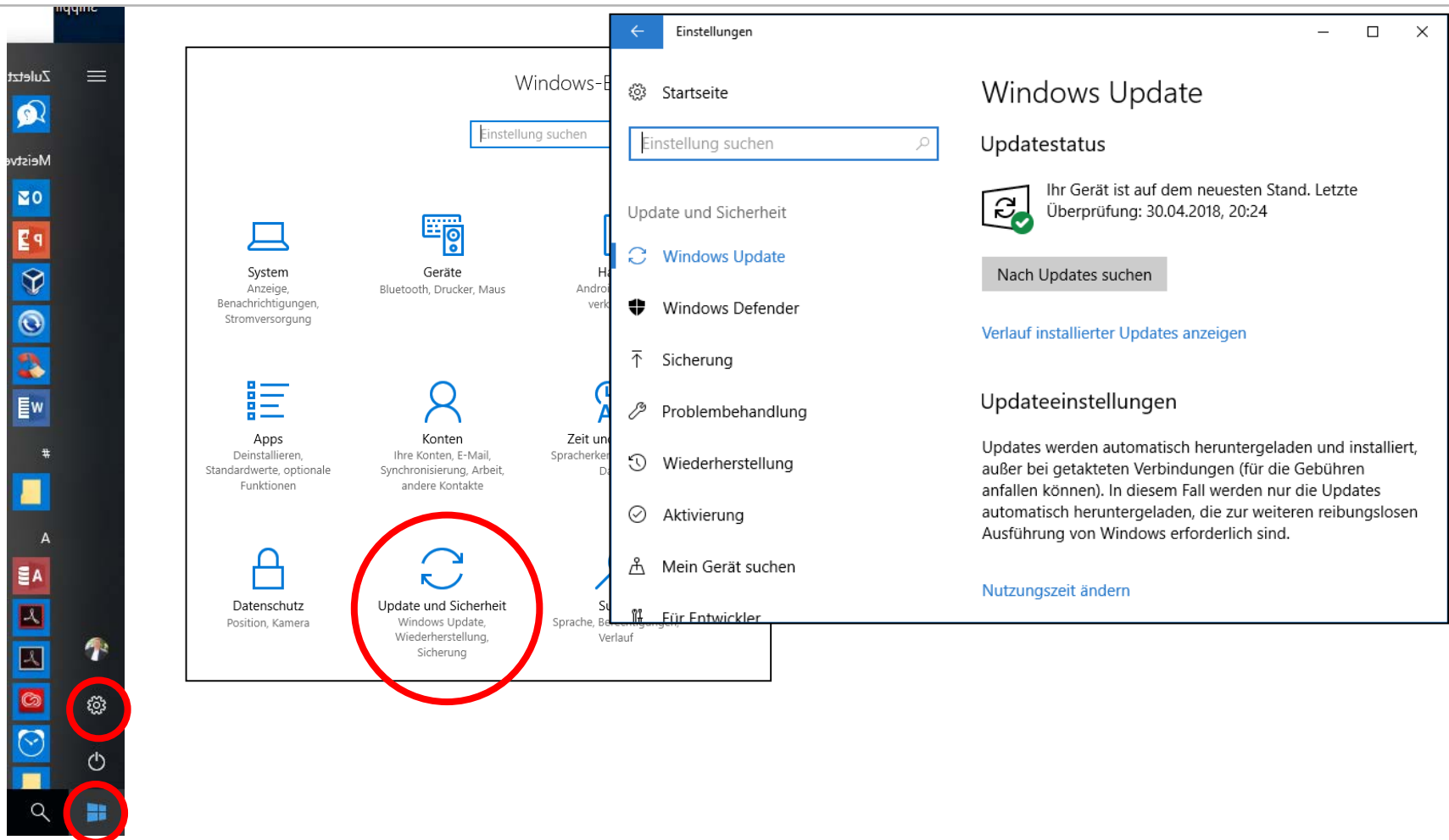
Software-Quellen: Windows-10



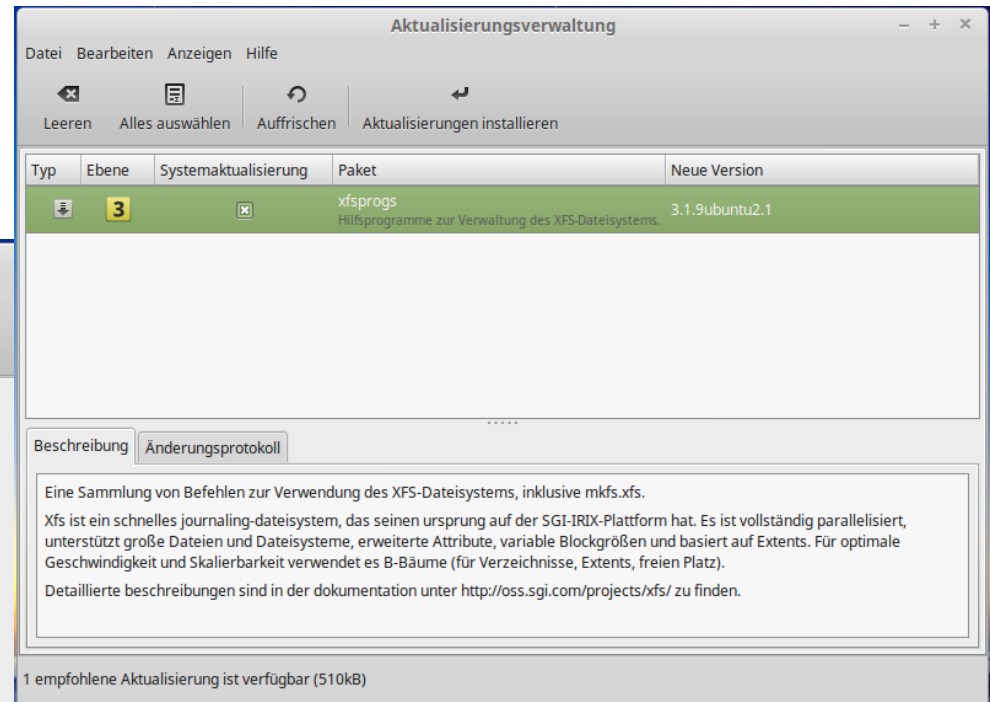
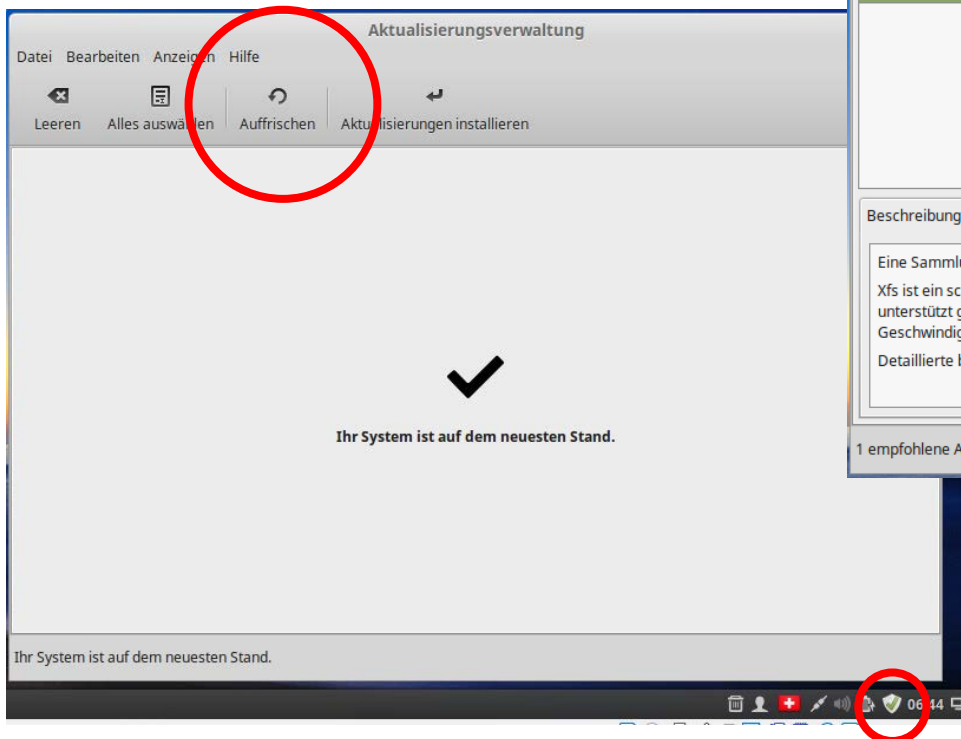
Software-Quellen: Linux



Software Updates: Windows-10 (monatlich)



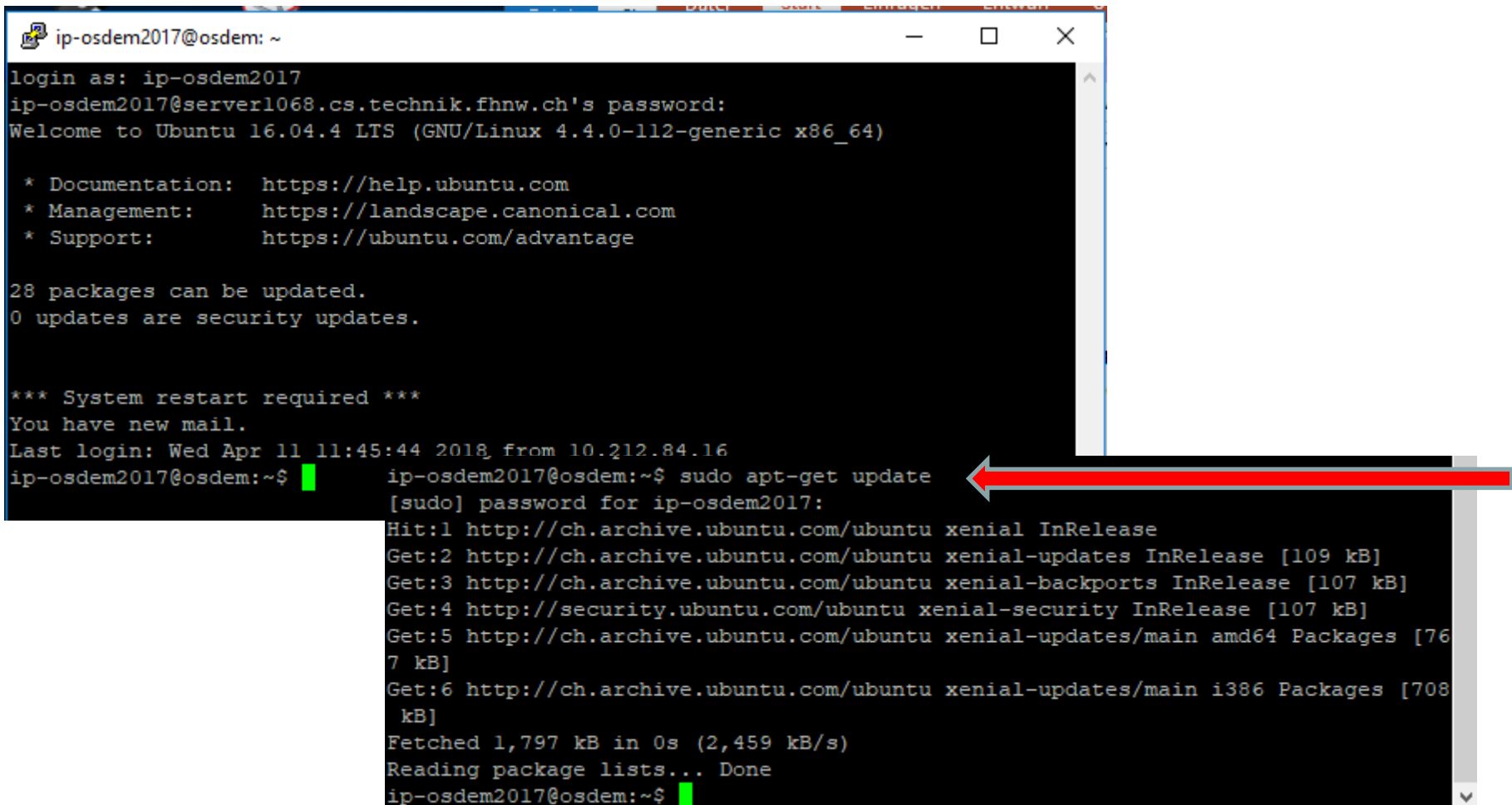
Software Updates: Linux (täglich/wöchentlich)



Software Updates: Linux (täglich/wöchentlich)

- `sudo apt-get update`
- `sudo apt-get upgrade`
- `sudo apt-get clean` (falls `/var/cache/apt/archives` zu viel Platz einnimmt)
- Nützlich bei «Remote Shell» (z.B. `rsh`, `telnet`, `ssh`) Verbindungen sowie bei GUI-Problemen

Software Updates: Linux (täglich/wöchentlich)

A terminal window titled 'ip-osdem2017@osdem: ~' showing a login session. The user 'ip-osdem2017' logs in from 'server1068.cs.technik.fhnw.ch'. The system is Ubuntu 16.04.4 LTS. It lists update links for documentation, management, and support. It reports that 28 packages can be updated, with 0 security updates. A message states '*** System restart required ***' and 'You have new mail.' The last login was on Wed Apr 11 11:45:44 2018 from 10.212.84.16. The user then runs 'sudo apt-get update'. A red arrow points to this command. The terminal shows the progress of updating package lists from various sources, including the ch.archive.ubuntu.com mirrors and security.ubuntu.com. It reports that 1,797 kB were fetched in 0s at a speed of 2,459 kB/s. The process is completed successfully.

```
ip-osdem2017@osdem: ~
login as: ip-osdem2017
ip-osdem2017@server1068.cs.technik.fhnw.ch's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-112-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

28 packages can be updated.
0 updates are security updates.

*** System restart required ***
You have new mail.
Last login: Wed Apr 11 11:45:44 2018 from 10.212.84.16
ip-osdem2017@osdem:~$ ip-osdem2017@osdem:~$ sudo apt-get update
[sudo] password for ip-osdem2017:
Hit:1 http://ch.archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://ch.archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:3 http://ch.archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Get:4 http://security.ubuntu.com/ubuntu xenial-security InRelease [107 kB]
Get:5 http://ch.archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [767 kB]
Get:6 http://ch.archive.ubuntu.com/ubuntu xenial-updates/main i386 Packages [708 kB]
Fetched 1,797 kB in 0s (2,459 kB/s)
Reading package lists... Done
ip-osdem2017@osdem:~$
```

Software Updates: Linux (täglich/wöchentlich)

```
ip-osdem2017@osdem:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  linux-headers-4.4.0-101 linux-headers-4.4.0-101-generic
  linux-headers-4.4.0-103 linux-headers-4.4.0-103-generic
  linux-headers-4.4.0-104 linux-headers-4.4.0-104-generic
```

```
  linux-image-extra-4.4.0-96-generic linux-image-extra-4.4.0-97-generic
  linux-image-extra-4.4.0-98-generic snap-confine
Use 'sudo apt autoremove' to remove them.
The following packages have been kept back:
  libdrm2 linux-generic linux-headers-generic linux-image-generic
The following packages will be upgraded:
  apport grub-legacy-ec2 ifupdown libavahi-client3 libavahi-common-data
  libavahi-common3 libpam-modules libpam-modules-bin libpam-runtime libpam0g
  libplymouth4 linux-base plymouth plymouth-theme-ubuntu-text python3-apport
  python3-problem-report snap-confine snapd ubuntu-core-launcher
19 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
Need to get 14.7 MB of archives.
After this operation, 2,633 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

```
Found kernel: /boot/vmlinuz-4.4.0-59-generic
Updating /boot/grub/menu.lst ... done

Processing triggers for libc-bin (2.23-0ubuntu10) ...
Processing triggers for initramfs-tools (0.122ubuntu8.11) ...
update-initramfs: Generating /boot/initrd.img-4.4.0-121-generic
ip-osdem2017@osdem:~$
```


Zusammenfassung der Lektion 3 und Hausaufgabe

- Die Benutzung der Shell
- Scripting
- Administrations- und Pflegeaufgaben im System
- Updates / Patches
- Hausaufgabe:
 - Repetieren und Üben Sie den Stoff dieser Lektion anhand der Web-Seite „<http://www.netzmafia.de/skripten/unix/unix2.html>“ und Üben Sie zudem die Benutzung der Shell sowie der grafischen Benutzeroberfläche.
 - Nutzen Sie die vorgestellte Administrationssoftware regelmässig.