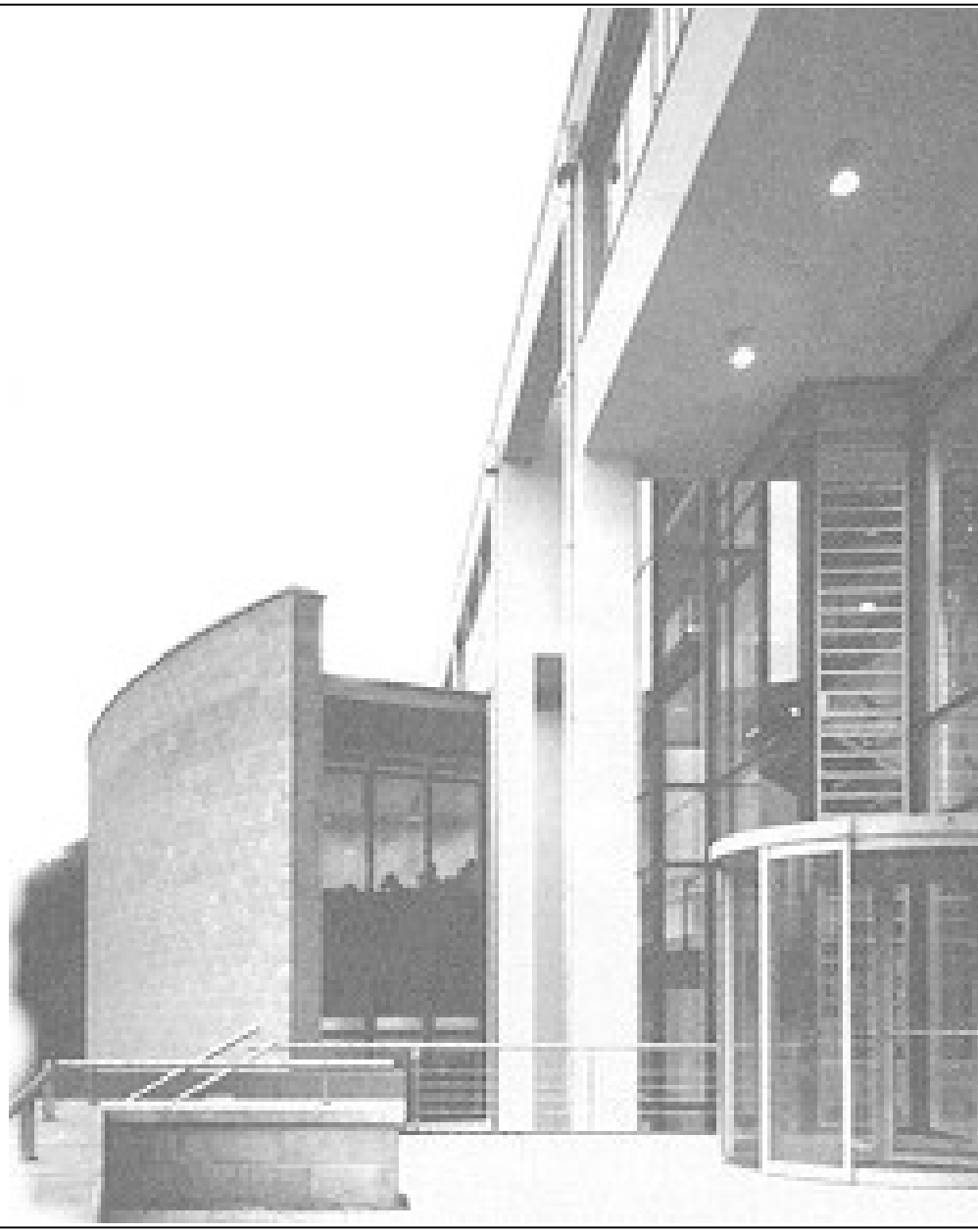
Studium neben dem Beruf

FOM Hochschule für Oekonomie & Management
University of Applied Sciences

IT-Infrastruktur

WS 2010/11

Hans-Georg Eßer
Dipl.-Math., Dipl.-Inform.



Foliensatz H (14.01.2011)
Archive und Pakete

Datenformate und Wandlung

Teil 4
Archive und Pakete

Heutiges Programm

- Kompressions- und Archivformate
  - Kompatibilität / Plattformen
- (Software-) Paketformate

  Installierbare Pakete für …
  - Windows: MSI
  - Linux: RPM, DEB
  - Mac OS: DMG

Kompression einzelner Dateien

- In der Linux-/Unix-Welt: diverse Komprimierer mit eigenen Dateiformaten
  - datei.txt → datei.txt.gz (gzip)
  - datei.txt → datei.txt.bz2 (bzip2)
  - datei.txt → datei.txt.Z (compress)

  etc.
- In der Windows-Welt: meist Archivformate
  - datei.txt → archiv.zip
    (auch wenn nur eine Datei komprimiert wird)

Kompression einzelner Dateien

```
esser@netbook:~/tmp$ ls -l
-rw-r--r-- 1 esser esser 14226 2010-07-12 21:07 index.html
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:03 kopie1.html
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:03 kopie2.html
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:03 kopie3.html
esser@netbook:~/tmp$ gzip kopie1.html ; bzip2 kopie2.html
esser@netbook:~/tmp$ compress kopie3.html ; ls -l
-rw-r--r-- 1 esser esser 14226 2010-07-12 21:07 index.html
-rw-r--r-- 1 esser esser  3806 2011-01-14 10:03 kopie1.html.gz
-rw-r--r-- 1 esser esser  3893 2011-01-14 10:03 kopie2.html.bz2
-rw-r--r-- 1 esser esser  6850 2011-01-14 10:03 kopie3.html.Z
esser@netbook:~/tmp$ file kopie*
kopie1.html.gz:  gzip compressed data, was "kopie1.html", from
                 Unix, last modified: Fri Jan 14 10:03:25 2011
kopie2.html.bz2: bzip2 compressed data, block size = 900k
kopie3.html.Z:   compress'd data 16 bits
```

Kompression einzelner Dateien

- komprimierte Datei enthält (meist) keine Metadaten
  - Dateiname des Originals:
    entsteht durch Weglassen der neuen Dateiendung (.gz, .bz2, .Z)
  - Besitzer, Gruppe, Zugriffsrechte des Originals:
    sind identisch in komprimierte Datei übernommen

```
esser@netbook:~/tmp$ ls -l programm
-rwxr-x--- 1 esser users 211 2011-01-14 10:32 programm
esser@netbook:~/tmp$ gzip programm
esser@netbook:~/tmp$ ls -l programm.gz
-rwxr-x--- 1 esser users  40 2011-01-14 10:32 programm.gz
```

Kompression einzelner Dateien

- komprimierte Datei enthält **meist** keine Metadaten, bei gzip aber Dateiname/Datum:

By default, gzip keeps the original file name and time-stamp in the compressed file. These are used when decompressing the file with the *-N* option. This is useful when the compressed file name was truncated or when the time stamp was not preserved after a file transfer.

-N --name
When compressing, always save the original file name and time stamp; this is the default. When decompressing, restore the original file name and time stamp if present. This option is useful on systems which have a limit on file name length or when the time stamp has been lost after a file transfer.

Kompression einzelner Dateien

```
esser@netbook:~$ ls -l
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:50 test1.html
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:50 test2.html
esser@netbook:~$ gzip test1.html
esser@netbook:~$ gzip -n test2.html   # (ohne Namen)
esser@netbook:~$ strings test1.html.gz | grep test
test1.html
esser@netbook:~$ strings test2.html.gz | grep test
esser@netbook:~$ mv test1.html.gz FALSCH1.gz
esser@netbook:~$ mv test2.html.gz FALSCH2.gz
esser@netbook:~$ gunzip -N FALSCH1.gz
esser@netbook:~$ gunzip -N FALSCH2.gz
esser@netbook:~$ ls -l
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:50 FALSCH2
-rw-r--r-- 1 esser esser 14226 2011-01-14 10:50 test1.html
```

Kompression einzelner Dateien

```
esser@netbook:~/tmp$ zip archiv.zip index.html
  adding: index.html (deflated 73%)

esser@netbook:~/tmp$ ls -l index.html archiv.zip
-rw-r--r-- 1 esser esser  3946 2011-01-14 10:08 archiv.zip
-rw-r--r-- 1 esser esser 14226 2010-07-12 21:07 index.html

esser@netbook:~/tmp$ unzip -l archiv.zip
Archive:  archiv.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
    14226  2010-07-12 21:07   index.html
---------                     -------
    14226                     1 file
```

Archivformate

- speichern (meist komprimiert)
  - mehrere Dateien
  - ganze Verzeichnishierarchien
- besonders populär:
  - ZIP (Windows, Mac OS)
  - tar.gz, tar.bz2 (Linux, Unix)
  - DMG (Mac OS)
- eingeschränkt plattformübergreifend

Archivformate

- Probleme beim Transfer zwischen Betriebssystemen (Windows, Linux, Mac OS)
- Was soll man im Archiv (neben den Dateien selbst) speichern?
  - Timestamps – Erstellung, letzte Änderung, letzter Zugriff (nicht jedes Dateisystem hat alle drei Typen)
  - Zugriffsrechte – bei jedem OS anders geregelt: Besitzer, Gruppe, ACLs, klassische Unix-Dateirechte
  - speziell bei Mac OS: Resource Forks
  - speziell bei Windows: Streams („datei.txt:stream“)

Archivformate

- Unterschiedliche Dateiattribute (der versch. Betriebs- bzw. Dateisysteme) kann ein Archivformat durchaus handhaben
- Aber was tun beim Entpacken einer Datei, die auf einem fremden System gepackt wurde?
  - Im Idealfall: So viele Eigenschaften „mitnehmen“ wie möglich, z. B. Datum der letzten Änderung gibt es auf jedem System
  - Oft: Wegwerfen aller Attribute
- Weiteres Problem: Dateinamenskonventionen

Archivformat: ZIP

- ZIP-Format
  - populärer Klassiker, schon seit DOS-Zeiten benutzt
  - speichert ganze Unterordner, Archiv kann ergänzt werden
  - mittelmäßige Kompressionsrate, dafür schnell beim Packen und Entpacken
  - für alle Betriebssysteme verfügbar
  - kann mit speziellen Attributen der diversen BS umgehen (darunter auch Linux-Attribute und Mac OS Resource Forks)
  - Packprogramme mit GUI und für die Shell

Archivformat: ZIP

- Auch Dateien ohne Endung „.zip“ können ZIP-Archive sein, z. B. aktuelle Office-Dateien:
  - .docx, .xlsx, pptx – alle XML-basierten Office-Dateien von Microsoft Office
  - .odt, .ods, .odp – alle XML-basierten Office-Dateien von OpenOffice / Libre Office
- Solche Dateien kann man untersuchen, indem man sie in *.zip umbenennt
  (siehe alte Folien zu Office-Formaten)

Archivformat: tar

- tar (tape archive)
  - Historie: Unix-Tool, das ganze Verzeichnisse unkomprimiert auf ein Streamer-Band (ein tape) sichert. Nur unter Linux/Unix verbreitet
  - heute: Anlegen von tar-Dateien
  - tar-Kommando bietet Optionen, mit denen sich beliebige Kompressionsprogramme nutzen lassen (-z: gzip, -j: bzip2, -I tool: tool)
  - Nutzung ohne Kompression manchmal nützlich, etwa, beim Packen eines Ordners, der nur bereits komprimierte Dateien enthält

Archive ohne Kompression

- nur in der Linux-/Unix-Welt üblich
- Archivformate
  - tar (tape archive)
  - ar (archive)
  - cpio (copy in, copy out)
  - pax (portable archive exchange)
- Vorteil:
  - Keine Standardkompression integriert; nach Archivbildung Nutzung eines beliebigen Komprimierers

Beispiel: tar-Archiv (1)

```
esser@netbook:~/tmp$ echo "Hallo Welt" > datei1.txt

esser@netbook:~/tmp$ echo "Kleiner Test" > datei2.txt

esser@netbook:~/tmp$ tar cf archiv.tar datei*.txt

esser@netbook:~/tmp$ cat archiv.tar
datei1.txt0000644000175000017500000000000013115140124270120010ustar  esseresserHallo Welt
datei2.txt0000644000175000017500000000000015115140124360120040ustar  esseresserKleiner Test
```

(cat zeigt nur darstellbare Zeichen an)

Beispiel: tar-Archiv (2)

```
esser@netbook:~/tmp$ hexdump -C archiv.tar
00000000  64 61 74 65 69 31 2e 74  78 74 00 00 00 00 00 00  |datei1.txt......|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 30 30 30 30  36 34 34 00 30 30 30 31  |....0000644.0001|
00000070  37 35 30 00 30 30 30 31  37 35 30 00 30 30 30 30  |750.0001750.0000|
00000080  30 30 30 30 30 31 33 00  31 31 35 31 34 30 31 32  |0000013.11514012|
00000090  34 32 37 00 30 31 32 30  30 31 00 20 30 00 00 00  |427.012001. 0...|
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000100  00 75 73 74 61 72 20 20  00 65 73 73 65 72 00 00  |.ustar  .esser..|
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000120  00 00 00 00 00 00 00 00  00 65 73 73 65 72 00 00  |.........esser..|
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000200  48 61 6c 6c 6f 20 57 65  6c 74 0a 00 00 00 00 00  |Hallo Welt......|
00000210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  64 61 74 65 69 32 2e 74  78 74 00 00 00 00 00 00  |datei2.txt......|
[...]
```

Beispiel: tar-Archiv (3)

```
[...]
00000400  64 61 74 65 69 32 2e 74  78 74 00 00 00 00 00 00  |datei2.txt......|
00000410  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000460  00 00 00 00 30 30 30 30  36 34 34 00 30 30 30 31  |....0000644.0001|
00000470  37 35 30 00 30 30 30 31  37 35 30 00 30 30 30 30  |750.0001750.0000|
00000480  30 30 30 30 30 31 35 00  31 31 35 31 34 30 31 32  |0000015.11514012|
00000490  34 33 36 00 30 31 32 30  30 34 00 20 30 00 00 00  |436.012004. 0...|
000004a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000500  00 75 73 74 61 72 20 20  00 65 73 73 65 72 00 00  |.ustar  .esser..|
00000510  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000520  00 00 00 00 00 00 00 00  00 65 73 73 65 72 00 00  |.........esser..|
00000530  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000600  4b 6c 65 69 6e 65 72 20  54 65 73 74 0a 00 00 00  |Kleiner Test....|
00000610  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00002800
```

Syntax des tar-Kommandos

- Drei Haupt-„Betriebsarten“:
  - **c**reate: `tar -cf Archivname.tar Dateien`
  - lis**t**: `tar -tf Archivname.tar`
  - e**x**tract: `tar -xf Archivname.tar [Dateien]`
  - Option -v (verbose): gibt immer Namen der bearbeiteten Dateien aus

Archivformat: tar.gz, tar.bz2

- tar.gz (auch: tgz) / tar.bz2:
  - eigentlich kein eigenständiges Format:

    Dateien entstehen (eigentlich) in zwei Schritten
    - tar-Archiv erstellen (archiv.tar)
    - dieses Archiv komprimieren; gzip: archiv.tar.gz
    - dank tar-Optionen aus Anwendersicht nur ein Schritt
  - Standardformat für Archive unter Linux/Unix
    - z. B. Quellcode von Software fast immer als tar.gz- oder tar.bz2-Paket erhältlich
    - bzip2: bessere Kompression als gzip, dafür etwas langsamer beim Packen

Archivformat: DMG

- DMG (Apple Disk Image, Mac OS)
  - Standardformat unter Mac OS für Software-Downloads
  - Kompression und Passwortschutz möglich
  - DMG-Images werden von Mac OS automatisch „gemountet“ und erlauben dann die Installation der enthaltenen Programme (meist einfaches Kopieren in den Anwendungen-Ordner)
  - technisch: in DMG-Datei steckt ein Disk Image, vergleichbar mit der Raw-Kopie einer Plattenpartition

Linux-Paketformate

- Die meisten Linux-Distributionen setzen eine Paketverwaltung ein:
  - Installation von Paketen aus sog. **Repositories** (Repos, Paketquellen)
  - Bequemes Entfernen, Aktualisieren von Paketen
  - Paketverwaltung löst automatisch **Abhängigkeiten** und **Konflikte** auf
- Es gibt viele Paketsysteme. Zwei sind populär:
  - RPM (Red Hat Package Manager)
  - DEB (Debian Package Manager) & APT (Advanced Package Tool)

Linux-Paketformate

- Vorteile der Paketverwaltung gegenüber „install.exe“ aus Windows-Welt:
  - automatische „Masseninstallation“ von vielen Anwendungen möglich
  - automatische Aktualisierung aller Programme, für die es ein Update gibt
  - automatische Aktualisierung des ganzen Betriebssystems auf eine neuere Version

Linux-Paketformate: RPM

- Ein **RPM-Paket** besteht aus …
  - einem **Archiv** mit den zu installierenden Dateien
  - **Metadaten** (Paketname, Version, Datum der Erstellung, für welche Linux-Version, Kontakt zum Paketbauer, Abhängigkeiten, Konflikte)
  - Skripte, die vor/nach Installation/Deinstallation ausgeführt werden
- Ein **RPM-Repository** listet eine Sammlung von RPM-Paketen und deren Download-Adressen auf (etwas vereinfacht...)

Linux-Paketformate: RPM

- Manche Programme sind in mehrere RPM-Pakete unterteilt, von denen evtl. nur eine Auswahl benötigt wird. Beispiel:
  - prog-1.2.3.i586.rpm
  - prog-1.2.3.x86_64.rpm
  - prog-lib-1.2.3.i586.rpm
  - prog-lib-1.2.3.x86_64.rpm
  - prog-debug-1.2.3.i586.rpm
  - prog-debug-1.2.3.x86_64.rpm
  - prog-data-1.2.3.noarch.rpm

Linux-Paketformate: RPM

- RPM-Pakete mit Kommando rpm installieren:
  rpm -i paketname.rpm
- Update eines schon installierten Pakets:
  rpm -U paketname.rpm
- Entfernen eines Pakets:
  rpm -e paketname (ohne „.rpm“)
- Überprüfen eines installierten Pakets:
  rpm -V paketname (ohne „.rpm“)

Linux-Paketformate: RPM

- Problem:
  - Es gibt viele verschiedene Linux-Distributionen (OpenSuse, Fedora, Red Hat, Mandriva, ...), die alle RPM-Pakete verwenden.
  - Pakete verschiedener Distributionen sind meist inkompatibel; in der Regel auch Pakete verschiedener Versionen (z. B. OpenSuse 11.2 / 11.3)
- Lösung: Repositories
  - Jeder Distributor verwaltet eines oder mehrere Repositories für jede Version seiner Distribution; mehr dazu später

Linux-Paketformate: RPM

- Problem: Abhängigkeiten
  - Ein Programm X benötigt eine Bibliothek Y (shared library, wie DLL-Datei bei Windows)
  - Das Paket, das X enthält, enthält kein Y
  - Der Paketmanager (RPM) weiß nicht, in welchem Paket sich die Bibliothek Y befindet – wenn es überhaupt eines für diese Linux-Version gibt
  - Installation des X-Pakets schlägt fehl (immerhin), weil RPM erkennt, dass die Bibliothek fehlt
- Lösung: auch Repositories (→ später)

Linux-Paketformate: RPM

- (Kein) Problem: Konflikte
  - Im RPM-Paket X findet sich der Hinweis, dass es nicht parallel zu Paket Y installiert sein darf
  - Y ist installiert, Anwender will X installieren
  - RPM verweigert Installation
  - Problem oder Feature?
- Lösung:
  - Es gibt die Option --force, die das Ignorieren der Konflikte erzwingt (fast immer eine schlechte Idee)

RPM: Repositories

- Installation am besten nur über Repositories
- Je nach Distribution verschiedene Tools für Zugriff auf die Repos:
  - OpenSuse: Zypper (zypper install paketname)
  - Fedora/Red Hat: YUM (yum install paketname)
  - Mandriva: URPMI (urpmi paketname)

  usw.
- Zugriff auf verschiedene Medien möglich

RPM: Repositories

- Repositories auf verschiedenen Medien:
  - HTTP
  - FTP
  - lokales Verzeichnis
  - CD/DVD

  u. a.
- Repositories nach Inhalten aufgeteilt:
  - Pakete
  - Updates
  - Quellpakete

RPM: Repositories

- Neben den offiziellen Repos vom Distributor gibt es oft viele Zusatz-Repos von Drittanbietern, z. B.
  - für spezielle Multimedia-Player und -Codecs
  - für proprietäre Hardware-Treiber (etwa Grafikkarten von Nvidia oder ATI/AMD)
  - von Software-Entwicklern, die ihre Programme nicht in den „regulären“ Repos unterbringen konnten
- Wie schützt man sich vor „falschen“ Repos?

RPM: Repositories

- Signierte Pakete / Repo-Keys:
  - Public-Key-Kryptographie (GPG, PGP etc.)
  - Jedes Repo bietet ein Paket mit öffentlichen Schlüsseln dieses Repos an.
    Wollen Sie das Repo nutzen? → Keys installieren
  - Jedes Paket ist mit dem privaten Key des Repos signiert (diesen Key kennt nur der Repo-Betreiber)
  - Bei Installation eines Pakets aus dem Repo wird die Signatur mit dem vorhandenen public key geprüft. Gibt es keinen Key oder einen Fehler, erscheint eine Warnung

Linux-Paketformate: DEB

- Gleicher Ansatz wie bei RPM:
  - Pakete über Tool dpkg installieren, updaten, löschen, Status prüfen etc.
  - Pakete enthalten Archive, Metadaten (inkl. Abhängigkeiten und Konflikten), Installations- und Deinstallationsskripte
  - Admin-Tool dpkg bietet ähnliche Funktionen wie RPM-Tool rpm
- Andere Liste von Distributionen: Debian, Ubuntu, Knoppix, ...

Linux-Paketformate: DEB

- DEB-Pakete nicht auf RPM-Systemen installierbar, und
- RPM-Pakete nicht auf DEB-Systemen installierbar („zwei Welten)
- aber: Programm alien konvertiert zwischen DEB- und RPM-Formaten
- Wie in der RPM-Welt: Ein DEB-Paket zu haben, heißt nicht, es installieren zu können
- Lösung aller Probleme auch hier: Repos

APT: Repositories

- Anders als in der RPM-Welt gibt es ein einheitliches Tool für die Verwaltung von Repos: APT (Advanced Package Manager)
- Zwar müssen auch APT-Anwender für jede Distribution (und jede Version davon) separate Repos nutzen, aber die APT-Kommandos sind immer gleich
  - apt-get install paketname
  - apt-cache search paketname
  - apt-get remove paketname

APT: Repositories

Verwaltung der Repos in /etc/apt/resources.list

```
deb cdrom:[Ubuntu-Netbook 10.04 _Lucid Lynx_ - Release i386
  (20100429.4)]/ lucid main restricted
deb http://de.archive.ubuntu.com/ubuntu/ lucid main restricted
deb-src http://de.archive.ubuntu.com/ubuntu/ lucid main restricted

## Bug Fixes, Updates
deb http://de.archive.ubuntu.com/ubuntu/ lucid-updates main restricted

## Universe Repos
deb http://de.archive.ubuntu.com/ubuntu/ lucid universe
deb http://de.archive.ubuntu.com/ubuntu/ lucid-updates universe
deb http://de.archive.ubuntu.com/ubuntu/ lucid multiverse
deb http://de.archive.ubuntu.com/ubuntu/ lucid-updates multiverse

## Fremdanbieter
deb http://download.virtualbox.org/virtualbox/debian lucid non-free
deb http://ppa.launchpad.net/d.filoni/dillo/ubuntu lucid main
```

Repos und Pakete aus Anbietersicht

- Aus Sicht eines Software-Anbieters:
  - Chaos – Debian, RPM, weitere Paketformate, diverse inkompatible Distributionen und deren Versionen
  - Wer eigene Repos anbieten will, muss diese für alle zu unterstützenden Distributionen separat pflegen
  - Beim RPM-Format heißt das auch: Nutzung verschiedener Tools, mit denen man Repos erstellt, denn Suse != Fedora != Mandriva...
  - Beim DEB-Format: immerhin einheitliche Tool-Welt (APT)

Beispiel: RPM-Paket „bauen“

- Um ein RPM-Paket zu bauen, ist ein sog. Specfile nötig (specification file)
- Beispiel für ein Specfile
  - Programm „Bomberclone“
  - Quelle: T. Schürmann, LinuxUser 07/2006, S. 52 ff., http://www.linux-user.de/ausgabe/2006/07/052-rpm/

RPM-Specfile (1/3)

- Specfile bomberclone.spec:

```
# Specfile fuer BomberClone
Summary: Bomberman-Klon
Name: bomberclone
Version: 0.11.6.2
Release: 1
Copyright: GPL
Group: Games/Action
Source: bomberclone-0.11.6.2.tar.gz
URL: http://www.bomberclone.de
Distribution: Suse Linux 10.0
Packager: Tim Schuermann <tschuermann@linux-user.de>
```

RPM-Specfile (2/3)

```
%description
Ein Bomberman-Klon, bei dem sich mehrere Spieler mit
kleinen Bomben heftig unter Druck setzen.
Mehrspielerpartien in einem Netzwerk sind ebenfalls
möglich.

%prep
%setup
./configure

%build
make

%install
make install
```

RPM-Specfile (3/3)

```
%files
/usr/local/bin/bomberclone
/usr/local/share/games/bomberclone
%doc /usr/local/doc/bomberclone/README
%doc AUTHORS TODO INSTALL NEWS COPYING ChangeLog

%post
ldconfig
```

- Paket bauen mit
  rpmbuild -bb bomberclone.spec

Windows-Installer

- MSI-Dateien
  - historisch: „Microsoft Installer“; für Windows-Installer
  - enthalten keine eigene Installationsroutine, sondern das Software-Paket und eine „Anleitung“ für den Windows-Installer
  - komplexe Struktur, MSI-Dateien enthalten Datenbanktabellen
  - MS bietet Tool, das aus XML-Dateien eine MSI-Datei erstellt (Vorgang deutlich komplexer als beim Bauen von RPM- / DEB-Paketen

Windows-Installer

- MSI-Dateien / Windows-Installer erlauben
  - Roll-Back bei Installationsfehlern: Alle schon vorgenommenen Änderungen rückgängig machen (z.B. auch Registry-Edits)
  - Einrichten der Deinstallationsroutine (Kontrollzentrum, Softwareverwaltung)
  - Aufteilung der Software in Komponenten, Anwender kann bei Installation auswählen, welche Teile installiert werden
  - ...

Windows-Installer

- MSI-Dateien / Windows-Installer erlauben
  - On-Demand-Installation und „Advertise“-Feature:
    - Programm enthält Menüpunkte für *nicht* installierte Komponenten
    - bei Aufruf dynamische Nachinstallation der benötigten Komponenten über den Windows-Installer – dafür keine neue Eingabe des Administrator-Passworts nötig

Mac-OS-Paketformate??

- kein spezielles Paketformat für Mac OS
- Software meist in DMG-Image
- Anwendung besteht häufig aus Verzeichnis
  - z. B.: Eintrag „Safari“ im Mac-OS-Programmordner ist in Wirklichkeit ein Verzeichnis „Safari.app“ (→ nächste Folie)
- Einige Programme bringen Installer mit (vgl. setup.exe/Windows)
- Neu: Mac OS AppStore (wie für iPhone & Co.)

Mac OS: Beispiel Safari

```
imac27:Applications esser$ ls -l Safari.app/
drwxr-xr-x  12 root  wheel  408 Nov 18 21:25 Contents

imac27:Applications esser$ ls -l Safari.app/Contents/
lrwxr-xr-x    1 root  wheel      28 Nov 18 21:23 CodeResources ->
_CodeSignature/CodeResources
-rw-r--r--    1 root  wheel   14809 Oct 15 02:40 Info.plist
drwxr-xr-x    3 root  wheel     102 Nov 18 21:25 MacOS
-rw-r--r--    1 root  wheel       8 Oct 15 02:40 PkgInfo
drwxr-xr-x  475 root  wheel   16150 Nov 18 21:25 Resources
-rwxr-xr-x    1 root  wheel  214752 Oct 15 02:40 Safari Webpage Preview
Fetcher
drwxr-xr-x    3 root  wheel     102 Nov 18 21:27 SafariSyncClient.app
drwxr-xr-x    3 root  wheel     102 Apr 20  2010 WebApplicationCore.bundle
drwxr-xr-x    3 root  wheel     102 Nov 18 21:25 _CodeSignature
-rw-r--r--    1 root  wheel     458 Oct 15 02:42 version.plist

imac27:Applications esser$ ls -l Safari.app/Contents/MacOS/
-rwxr-xr-x  1 root  wheel  13518992 Oct 15 02:42 Safari
```