IT-Infrastruktur

WS 2014/15

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

Foliensatz A: v1.0, 2014/09/15

- Einführung
- Seminararbeiten, Wissenschaftliches Arbeiten
- Datenformate und Wandlung

Zur Veranstaltung (1)

Service / Web-Seite: http://fom.hgesser.de

- Folien und Praktikumsaufgaben
- Vorlesungs-Videos („test, test “)
- Probeklausur gegen Semesterende
- Folien auch im Online-Campus

Prüfung und Benotung:

1. Lernfortschrittskontrolle (LFK)
2. Seminarausarbeitung
3. Klausur über 120 Minuten

Fragen: direkt in der Vorlesung oder danach
oder per E-Mail: h.g.esser@gmx.de

Zur Veranstaltung (2)

10 ECTS-Punkte – großes Modul

- Präsenzstudium: 48 Stunden *) (= 64 UE)
- Eigenstudium: 172 Stunden
- Lernfortschr.-Kontr.: 30 Stunden
- Workload: 250 Stunden

Klausur-relevant:

- Vorlesungen
- Seminarvorträge
- Überblick über TK-Skript

*) Stunde
= Zeitstunde,
= 60 min.

Über den Dozenten

Hans-Georg Eßer

- Dipl.-Math. (RWTH Aachen, 1997)
  Dipl.-Inform. (RWTH Aachen, 2005)
  Fachjournalist (FJS Berlin, 2006)
- Chefredakteur Linux-Zeitschrift (seit 2000) und Autor diverser Computerbücher
- LPI-zertifiziert (LPIC-1 und LPIC-2)
- seit 2006 Dozent (u. a. FOM, HS München, TH Nürnberg, Univ. Erlangen-Nürnberg): Betriebssysteme, Rechner-architektur, IT-Infrastruktur, Informatik-Grundlagen, System-programmierung, Betriebssystem-Entwicklung, IT-Sicherheit
- Seit 2010 Doktorand (Univ. Erlangen-Nürnberg), Doktorarbeit wird im September fertig

Inhaltsübersicht

Inhalte

- **A. Datenformate und Wandlung (von DTaus bis XML)**
  - Grafik, Video
  - Text
  - Applikationsformate (z.B. GIS, CAD usw.)
  - Kommunikationsformate
  - Konverter
  - Kodierungen (ASCII, ISO-Latin-*, Unicode)
- **B. PC als Arbeitsplatz**

Inhalte

- **C. Ergonomie und Arbeitsschutz**
- **D. Rechnerstrukturen**
  - Prozessor-Architekturen, Befehlssätze
- **E. Telekommunikation**
  - Geräte, Protokolle, Dienste
    → Eigenstudium, Seminararbeiten

  } Skript im Campus-System
- **F. Infrastruktur-Technologie**
- **G. Zentrale und verteilte IT-Infrastrukturen**

  } weitere Literatur folgt ggf.

Dieser Foliensatz

Folien A →

- Vorlesungsübersicht
- Seminar
- Wiss. Arbeiten
- Datenformate und Wandlung
- PC als Arbeitsplatz
- Ergonomie und Arbeitsschutz
- Rechnerstrukturen
- (Telekommunikation)
- Infrastruktur-Technologie
- Zentrale / verteilte IT-Infrastrukturen

Seminararbeiten

Seminararbeiten (1)

- 16 Teilnehmer laut Campus-System
- 18 Themenangebote aus dem Bereich „E. Telekommunikation“
- Literatur im Campus-System verfügbar

  [1] Firoz Kaderali: Skript „Kommunikationsnetze und Protokolle“

  [2] Peter Richert: Skript „Kommunikationssysteme“

  [3] H. R. van As: Skript „Datenkommunikation“
- Themen (ab 1. Vorlesung) im Campus-System
  → Anmeldung, bis Jahresende freigeschaltet

Seminararbeiten (2) – Themen

- alle Themenvorschläge auf Basis des Skripts „Kommunikationsnetze und Protokolle“ von Prof. Firoz Kaderali [1],
  http://www.kaderali.de/fileadmin/vorlesungsskripte/Buch KP (A4).pdf

1. Einführung: Offene Systeme (Skript, Kap. 1)
2. LANs und MANs (1) (Kap. 3.1, 3.2)
3. LANs und MANs (2) (Kap. 3.3)
4. LANs und MANs (3) (Kap. 3.4)
5. ISDN-Referenzmodell und Bit-Übertragungsschicht (Kap. 4.1, 4.2)

Seminararbeiten (3) – Themen

6. ISDN: Sicherungsschicht, Vermittlungsschicht (Kap. 4.3, 4.4)
7. ISDN: Signalisierung (Kap. 4.5)
8. ISDN: Dienste und Leistungsmerkmale (Kap. 4.6)
9. ATM als Basis für B-ISDN (Kap. 5.1, 5.2)
10. ATM: Referenzmodelle, phys. Schicht (Kap. 5.3, 5.4)
11. ATM-Schicht und Adaptionsschicht (Kap. 5.5, 5.6)
12. GSM-Grundlagen (Kap. 6.1 – 6.3)
13. GSM-Übertragungstechnik (Kap. 6.4)

Seminararbeiten (3) – Themen

14. GSM-Protokolle (Kap. 6.5)
15. WLAN (Kap. 7)
16. Internet-Protokolle, TCP/IP (Kap. 8.1 – 8.4)
17. TCP/IP: Vermittlungsschicht (Kap. 8.5)
18. TCP/IP: Transportschicht (Kap. 8.6)

Wissenschaftliches Arbeiten

Ausarbeitungen und Vorträge

Inhalte

- Wie lese ich Fachbücher/Skripte/Papers?
- Wie schreibe ich eine Seminararbeit?
- Wie halte ich einen Seminarvortrag?

Quellen lesen

- Was ist das Themengebiet der Quelle?
- Was ist die zentrale Fragestellung?
  - Welches Thema wird im Detail behandelt?
  - Welche Aspekte sind besonders wichtig oder vernachlässigbar?
  - Welche Aspekte bleiben in der Quelle unklar?
- Literaturangaben auswerten
  - Grundlagen
  - Weiterführende Literatur

Seminararbeit schreiben

- Gliederung erstellen
- Typisch:
  - Kap. 1: Einleitung
  - Kap. 2: } (Inhalt)
  - Kap. $n-1$:
  - Kap. n: Zusammenfassung

Einleitung

- Einleitung besteht aus:
  - Motivation / Themenfeld
    - Welches Themengebiet behandelt die Arbeit
    - Warum ist es relevant?
  - Thema der (Seminar-) Arbeit
    - Fragestellung in dieser Ausarbeitung
    - Bezug auf das Paper / die Papers
    - Welche Lösung wird im Paper vorgeschlagen?
  - Struktur der Arbeit
    - Wie ist diese Arbeit aufgebaut? Roter Faden?

Einleitung

- Leser wollen i. d. R. nicht jedes Dokument vollständig lesen
- Darum: In der Einleitung vollständigen Überblick über die wichtigsten Inhalte geben (→ „Executive Summary“)

Inhaltliche Kapitel

- In den inhaltlichen Kapiteln kann sich die Grobstruktur der Ausarbeitung wiederholen, also
  - Kap. i.1: Einleitung (zu Thema i)
    - Worum geht es?
    - Roter Faden
    - Anknüpfen an vorangehendes Kapitel
  - Kap. i.2: …
  - Kap. $i.m$: Zusammenfassung
    - Worum ging es? Wieder: Roter Faden
    - ggf. Ausblick auf Folgekapitel

Redundanz ist OK, hilft bei Orientierung

Zusammenfassung

- Kurze Zusammenfassung der Ausarbeitung (Kenntnis der Arbeit kann hier vorausgesetzt werden)
- Falls sinnvoll: Hinweis auf offene Punkte, unbeantwortete Fragen

Inhaltsauswahl

- Häufiges Problem:
  Quelle viel zu lang / Quelle viel zu kurz
- Auswahl
- Ergänzung (weitere Literatur)
- Seminarausarbeitung und Vortrag müssen nicht deckungsgleich sein
- Ausarbeitung „in sich geschlossen“ (ohne weitere Literatur verständlich)
- Frage beantworten: Warum sollte man meine Ausarbeitung (statt der Originaltexte) lesen?

Quellen / Zitate (1)

- Beim Strukturieren bereits die relevanten Textstellen der Originalliteratur vermerken
  - Welcher Teil der Arbeit bezieht sich auf welchen Teil der Originalliteratur?
- Im Text eine Referenz einbauen falls ...
  - wesentliche Ideen, Zahlen, Bilder, Tabellen usw. übernommen werden
  - ganze Teile wortwörtlich übernommen werden (Zitate)
- Zitate als solche kenntlich machen („...“ [Referenz])
- Referenzen möglichst genau (am besten mit Seitenangabe, vor allem bei Büchern)

Kritischer Umgang

- Quellen kritisch bearbeiten
  - Nur weil etwas in einer Quelle steht, muss es nicht richtig sein → kein blindes Vertrauen
  - Technischen Beschreibungen und Erklärungen kann man auch widersprechen
  - Wenn unklar: Weitere Literatur zum selben Thema suchen, vergleichen
  - Kein direkter Zusammenhang zwischen „Zitierfähigkeit“ und Glaubwürdigkeit einer Quelle

Quellen / Zitate (2)

- Im Text stehen Referenzen (Verweise) auf das Literaturverzeichnis
  - Referenzen sind meist Kürzel in eckigen Klammern
  - Unterstützung durch LaTeX
- Arten des Zitierens:
  - Aktiv: In [1] wurde gezeigt … [1, S. 35]
  - Passiv: Maier und Müller [1] haben gezeigt …
  - Möglichst für eine Art entscheiden und konsistent verwenden

Quellen / Zitate (3)

- Informationen, die (unter Informatikern) allgemein bekannt sind, brauchen keinen Beleg durch Quellenangabe
  - „C ist eine häufig für systemnahe Programmierung verwendete Sprache.“ (ohne Beleg)
  - „Das Betriebssystem ZonkOS wurde nicht in C, sondern in Fortran implementiert [12].“ (mit Beleg)
- Zielgruppe: andere (Wirtsch.-) Informatik-Studenten, konkret: die übrigen Kursteilnehmer

Quellen / Zitate (4)

- Mehrere Referenzen, Seitenangaben
  - [1, 2, 4]
  - [1–3, 5]
  - [4, S. 15, 6, S. 34] → besser zerlegen
  - LaTeX macht das automatisch
- Viele Autoren
  - Müller et al.: ab vier Autoren; in engl. Texten ab drei Autoren, vgl. Wikipedia [3]
  - im Literaturverzeichnis aber alle Autoren nennen

Quellen / Zitate (5)

- Zitierweise
  - Bitte keine Fußnoten für Literatur-Verweise,
  - sondern einfach z. B. „[6]“ im Text als Verweis auf Eintrag [6] im Literaturverzeichnis
  - bei längeren Quellen (wie dem TK-Skript) immer mit Seitenangabe, in der Form „[6, S. 119]“

Literaturverzeichnis

- alle verwendeten (und nur die!) Quellen aufführen
  - sortiert nach Nachname des Erstautors, dann nach Datum
- Eintrag enthält
  - Namen der Autoren, Titel der Publikation/des Beitrags
  - ggf. Name der Zeitschrift/Konferenz, Band, Nummer, Seitenzahl / Ort
  - Erscheinungsjahr
  - möglichst genau
- Quellen im Web: Autoren, Titel, URL und Zugriffsdatum (wenn es eine Originalquelle gibt, diese zitieren!)

Literaturverzeichnis

- Wikipedia: u. U. akzeptable Quelle. Ganz gut für technische Dinge
- Whitepapers von Unternehmen: da fehlen oft wichtige Angaben (z. B. Autoren),
  → Firma als Hrsg.
- „keinen Schrott zitieren“

Literaturverzeichnis

- Negativ-Beispiel

Bei einem Vergleich forensisch nutzbarer Informationen in Windows-Dateisystemen sieht man u. a., dass FAT [1] nur einen einzigen Zeitstempel pro Datei speichert, während NTFS [2] für jede Datei vier Zeitstempel verwaltet [3]. Ergänzend gibt ...

[1] http://de.wikipedia.org/wiki/File_Allocation_Table
[2] http://de.wikipedia.org/wiki/NTFS
[3] http://articles.forensicfocus.com/2013/04/06/interpretation-of-ntfs-timestamps/
[4] http://books.google.de/books?id=fummOICB9IgC&pg=PA148&lpg=PA148&dq=ntfs+%22delete+time%22+timestamp&source=bl&ots=FOFB7xD3MY&sig=awkTk3V-f1dR3E6ivx46-Gc7xs4&hl=en&sa=X&ei=loFXU6mOC4LNtQbLuwE&ved=0CDwQ6AEwAg#v=onepage&q=ntfs%20%22delete%20time%22%20timestamp&f=false

Literaturverzeichnis

Positiv-Beispiel

Bei einem Vergleich forensisch nutzbarer Informationen in Windows-Dateisystemen sieht man u. a., dass FAT [1] nur einen einzigen Zeitstempel pro Datei speichert, während NTFS [2] für jede Datei vier Zeitstempel (create, modify, access, change) verwaltet [3]. Der vierte Zeitstempel (change) wird für gelöschte Dateien auch als delete-Timestamp betrachtet [4, S. 148].

[1] Microsoft: Microsoft EFI FAT32 File System Specification, 2000, Whitepaper, http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/fatgen103.doc, Onlineabruf am 23.04.2014
[2] Microsoft: NTFS Technical Reference, 2003, http://technet.microsoft.com/en-us/library/cc758691%28WS.10%29.aspx, Onlineabruf am 23.04.2014
[3] wpathulin: „Interpretation of NTFS Timestamps“, Forensic Focus, 2013, http://articles.forensicfocus.com/2013/04/06/interpretation-of-ntfs-timestamps/, Onlineabruf am 23.04.2014
[4] Greg Gogolin: *Digital Forensics Explained*, Auerbach Publications, 2012, ISBN: 1-4398-7495-6

Text produzieren

- In eigenen Worten formulieren
- Keine Textblöcke aus der Quelle übernehmen, auch keine Wort-für-Wort-Übersetzung
- kein Copy & Paste
  - Plagiat, ist in Prüfungs- und Seminararbeiten ein Grund für Nichtbestehen; an einigen Hochschulen sogar für offizielle Rügen oder Exmatrikulation (Abschlussarbeiten)
  - fällt auf (Stilbruch etc.) und wird geprüft
  - Reines Umformulieren von Satz für Satz ist aber genauso Plagiat!

Text produzieren

- Eigene Beiträge kennzeichnen (wenn Sie z. B. eigene Praxistests etc. machen)
- Stil: Passivkonstruktionen, Substantivierungen meiden

Plagiat-Formen

- Copy & Paste (wortwörtliche Wiedergabe)
- Paraphrasierung (Wiedergabe mit eigenen Worten),
- Übersetzungsplagiat
- Strukturplagiat
- „Shake and Paste“ [1]

[1] http://plagiat.htw-berlin.de/ff/schule/3_2/wie

Aktuelles Plagiat (2013)

1 INTRODUCTION

CONSIDER a set of parties who wish to correctly compute some common function F of their local inputs, while keeping their local data as private as possible, but who do not trust each other, nor the channels by which they communicate. This is the problem of *Secure Multiparty Computation* (SMC) [1]. SMC is a very general security problem, i.e., it can be used to solve various real-life problems such as distributed voting, private bidding and online auctions, sharing of signature or decryption functions and so on. Unfortunately, solving SMC is—without extra assumptions—very expensive in terms of communication (number of messages), resilience (amount of redundancy), and time (number of synchronous rounds).

TrustedPals [2] is a smart card-based security framework for solving SMC which allows much more efficient solutions to the problem. Conceptually, TrustedPals considers a distributed system in which processes are locally equipped with tamper-proof security modules. In practice, processes are implemented as a Java desktop application and security

links: R. Cortiñas, F. C. Freiling, M. Ghajar-Azadanlou, A. Lafuente, M. Larrea, L. Draque Penso, I. Soraluze: "Secure Failure Detection and Consensus in „Trusted-Pals“, IEEE TDSC (July-Aug. 2012, vol. 9 no. 4, pp. 608-623)

rechts: Khaderbasha, Shakeel, Babu: „Secure Failure Detection and Aggrements In Trusted Pals“, Int. Journal of Innovative Technologies (Vol. 01, Issue 05, Oct 2013)

1. INTRODUCTION

Consider a set of parties who wish to correctly compute some common function F of their local inputs, while keeping their local data as private as possible, but who do not trust each other, nor the channels by which they communicate. This is the problem of Secure Multiparty Computation (SMC) [1]. SMC is a very general security problem, i.e., it can be used to solve various real-life problems such as distributed voting, private bidding and online auctions, sharing of signature or decryption functions and so on. Unfortunately, solving SMC is without extra assumptions very expensive in terms of communication (number of messages), resilience (amount of redundancy), and time (number of synchronous rounds).

TrustedPals [2] is a smart card-based security framework for solving SMC which allows much more efficient solutions to the problem. Conceptually, TrustedPals considers a distributed system in which processes are

Bilder und Tabellen

- Jedes Bild hat eine Bildunterschrift
  - beschreibt das Bild – ohne den restlichen Text vorauszusetzen
  - ist keine Kopie eines Satzes aus dem Text
- Für Tabellen gilt dasselbe analog

Umfang, Layout

- Ziel: 13–15 Seiten,
  - zzgl. Anhang, Verzeichnisse
- Format / Layout:
  - DIN-A4, 12pt Times New Roman, 1,5-zeilig, Blocksatz, autom. Silbentrennung
  - Abstände: oben 3 cm, unten 2 cm, links 4 cm, rechts 2 cm
  - Seitenzahlen: oben (zwischen Textblock und Papierrand)
  - kein Inhaltsverzeichnis

Abschließende Tipps

- Konsistent sein
  - beim Zitieren
  - bei der Verwendung von Fachbegriffen
- Sich an die (deutsche) Sprache halten
  - englische Wörter übersetzen und (wenn nötig) den Originalbegriff beim ersten Mal nennen
  - steht das Wort im Duden? Dann ist es deutsch
  - Fachbegriffe wie IP-Stack, Bridge, Gateway sind ok (1x erklären)
  - teilweise Geschmacksache („I/O“ oder „E/A“?)
- Sparsame Typographie

Häufige Fehler (1)

- Beschwerden, dass der Stoff zu umfangreich ist. Aus jedem Thema lässt sich problemlos ein 10-, 20- oder 60-minütiger Vortrag gestalten. Das ist eben die Vortragskunst.
- Eine Motivation wird vernachlässigt.
- Sich nicht mit dem roten Faden oder der „Message“ auseinanderzusetzen.
- Es werden Dinge vorausgesetzt, die bei Teilen des Publikums nicht bekannt sind.
- Fachbegriffe werden nicht erklärt, Abkürzungen nicht ausgeführt.

Quelle: Weicker & Weicker, Lehrstuhl 4, Univ. Erl.-Nbg. [5]

Häufige Fehler (2)

- Gliederung wird anhand von Stichworten erstellt, die dann in jeweils ein bis zwei Sätzen ausgeführt werden. Meist geht hierbei der rote Faden verloren: die Arbeit wirkt abgehackt und unverständlich
- Gliederung der Folien für den Vortrag oder Gliederung des Vortrags fü□r die Ausarbeitung verwenden
- Zu viel Zeit auf Folien verwenden, aber das Gesprochene improvisieren
- Den Folieninhalt nur vorlesen oder sich an den Folien entlang zu hangeln ohne den Vortrag lebendig werden zu lassen
- Mit Powerpoint-Effekten über inhaltliche Defizite hinwegtäuschen

Quelle: Weicker & Weicker, Lehrstuhl 4, Univ. Erl.-Nbg. [5]

Literatur

[1] Marcus Deininger, Horst Lichter, Jochen Ludewig, Kurt Schneider: Studienarbeiten. Ein Leitfaden zur Vorbereitung, Durchführung und Betreuung von Studien-, Diplom-, Abschluss- und Doktorarbeiten am Beispiel Informatik. vdf, 5. Auflage, 2005.

[2] Albers et al.: Gute wissenschaftliche Praxis für das Verfassen wissenschaftlicher Qualifikationsarbeiten, http://www.hochschulverband.de/cms1/uploads/media/Gute_wiss._Praxis_Fakultaetentage.pdf, Juli 2012.

[3] Wikipedia zu „et al.“: http://de.wikipedia.org/wiki/Et_al.

[4] Peter Rechenberg: Technisches Schreiben. Hanser, 2. Auflage, 2003.

[5] Karsten Weicker, Nicole Weicker: Seminarrichtlinien am Lehrstuhl 4, https://www4.cs.fau.de/Lehre/SS11/PS_KVBK/info/Weicker.pdf

[6] Markus Rath: Leitfaden zur Anfertigung von Seminar-, Studien- und Diplomarbeiten, http://www.wi-inf.uni-duisburg-essen.de/FGFrank/documents/Lehre/Leitfaden_WissArbeit.pdf

Weitere Hinweise zum Seminar

- FOM-Dokument:
  „Leitfaden zur formalen Gestaltung von Seminar- und Abschlussarbeiten“
- über Campus-System verfügbar (von Kursseite aus verlinkt)

Datenformate und Wandlung

Gliederung

Gliederung (1)

- Daten und Informationen
- Zahlensysteme
- ASCII und andere Zeichenkodierungen (ISO-Latin-*, Unicode / UTF)
- Digitalisierung (Bilder, Audio, Video)
- Kompression

Gliederung (2)

- Datenformate
  - Textformate
  - Grafikformate
  - Konverter
- Kommunikationsformate
- Applikationsformate
- Archive und Programmpakete

A. Datenformate und Wandlung

Daten und Informationen (1/7)

Was bedeutet „Information“?

- eine Form von Wissen
- Information hat immer eine Bedeutung
  - Text in einer unbekannten Fremdsprache: bedeutungslos
  - Bedeutung also abhängig vom Betrachter
  - Zusammenhang wichtig (etwa: Text aus einem fremden Fachgebiet)
- Information informiert:
  - Herr Müller informiert Frau Meier
  - Ich informiere mich (selbst) über ...

Daten und Informationen (2/7)

Informationen...

- vielseitig, z. B.
  - Fakten(wissen)
  - Konzepte (die Antwort auf: „Was sind Informationen?“ ist auch eine Information)
  - Anleitungen (z. B. Algorithmen)
- Information an sich hat keine „standardisierte“ Darstellung, etwa in Schriftform

Daten und Informationen (3/7)

Weitergabe von Informationen

- Viele Wege denkbar
  - im direkten Gespräch erzählen
  - eine Vorlesungsstunde darüber halten
  - aufschreiben (Buch? Notizzettel? Tafel?)
  - etwas (ohne Worte) vorführen

Daten und Informationen (4/7)

und „Daten“?

- Repräsentation von Information(en)
- mit oder ohne Struktur

Name	Telefon	Ort
Anton Müller	089/1234567	München
Berta Meier	0241/7343	Aachen
Christian Bauer	0211/64834	Düsseldorf
Dagmar Grün	02131/46734	Neuss

Wegbeschreibung: aus dem Münchener HBf nach Ankunft links raus, nach dem Ausgang links halten bis zur nächsten großen Kreuzung, diagonal die Kreuzung überqueren, Eingang der FOM liegt an Hauptstraße, im Gebäude in den ersten Stock zur Bibliothek

Daten und Informationen (5/7)

Daten laut Wikipedia:

„In der Informatik und Datenverarbeitung versteht man Daten als (Maschinen-) lesbare und bearbeitbare in der Regel **digitale Repräsentation von Information**. Die Information wird dazu meist zunächst in Zeichen (bzw. Zeichenketten) **kodiert**, deren Aufbau strengen Regeln folgt, der so genannten **Syntax**. Um aus Daten Informationen zu gewinnen, müssen sie in einem **Bedeutungskontext** interpretiert werden.“

Quelle: http://de.wikipedia.org/wiki/Daten

Daten und Informationen (6/7)

- **Repräsentation**
  - eine Form der Darstellung, die gewisse Regeln einhält
  - z. B.: 5 vs. „fünf“ vs. IIIII vs. ~~IIII~~ vs.
  - für den Betrachter: offensichtlich alle gleichwertig
- Digitale Repräsentation: „im Computer“
- Frage: **Was** kann man **wie** im Computer speichern?

Daten und Informationen (7/7)

- **Kodierung durch Zeichen (-ketten)**
  - Was ist ein Zeichen?
  - Vorschlag: A-Z, a-z, 0-9 und Leerzeichen – gut?
  - Wie speichert der Computer Zeichen?
  - Prinzipiell kennt der Rechner nur zwei Werte: 0 und 1
  - Für alles andere: Umwandlung in Kombinationen aus Nullen und Einsen

Bits und Bytes

- **Bit:** 0 / 1 an / aus wahr / falsch (binary digit – Binärziffer)
- **Bit-Folge:** Zeichenkette, die aus Nullen und Einsen besteht, z. B. 00101, 10
- **Byte:** Bit-Folge aus acht Bits (auch: Oktett), z. B. 01101011, 11111111, 00000000
- Führende Nullen kann man weglassen: 00000010 = 10
- **Wie viele (unterschiedliche) Bytes gibt es?**

Dualzahlen (1/3)

- Bitfolgen auch als Zahlenwerte auffassen: $1_b = 1$, $10_b = 2$, $11_b = 3$, $100_b = 4$, $101_b = 5$ ...
- **Dualzahlen:** Zahlensystem, das zur Darstellung von Zahlen nur die Ziffern 0 und 1 verwendet. Auch **Binärzahlen** genannt
- Rechnen mit Dualzahlen funktioniert wie mit normalen Zahlen. Hier:
  - Addieren + Subtrahieren
  - Multiplizieren

Dualzahlen (2/3)

- Addieren:

```
  001001001
+ 010101110
      1
-----------
= 011110111
```

Übertrag

- Subtrahieren:

```
  011010111
– 010101110
    1  1
-----------
= 000101001
```

Übertrag

Dualzahlen (3/3)

- Multiplizieren

```
101001 x 1001
-------------
       101001
      000000
     000000
    101001
       1
-------------
    101110001
```

Übertrag

Probe:

$101001_b = 41$
$1001_b = 9$
$41 \times 9 = 369$
$101110001_b = 369$

Umrechnung dual → dezimal (1/3)

- Grundlage: Wie funktionieren unsere Zahlen?
- $4982 = 4 \times 1000 + 9 \times 100 + 8 \times 10 + 2 \times 1$
  $= 4 \times 10^3 + 9 \times 10^2 + 8 \times 10^1 + 2 \times 10^0$
  (von rechts nach links: Einser, Zehner, Hunderter, Tausender, ...)
- Bei Dualzahlen geht das genauso:
- $10011_b = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
  $= 1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$
  $= 19$

Umrechnung dual → dezimal (2/3)

umgekehrt: dezimal → dual etwas komplizierter

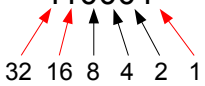
49 als Dualzahl?

49 : 2 = 24,	Rest	1	Einser
24 : 2 = 12,	Rest	0	Zweier
12 : 2 = 6,	Rest	0	Vierer
6 : 2 = 3,	Rest	0	Achter
3 : 2 = 1,	Rest	1	
1 : 2 = 0,	Rest	1	-> 110001_b (32+16+1)

Warum geht das? Rechnen Sie mal direkt
110001_b : 2... $110001_b = 2 \times 11000_b + 1$

Umrechnung dual → dezimal (1/3)

- oder durch „Kopfrechnen“:
  - kleinste 2er-Potenz (2, 4, 8, 16, ...), die in die Zahl 49 „passt“: 32 (nächst größere: 64)
  - also 49 = 32 + ? = 32 + 17
  - 17 = 16 + ? = 16 +1
  - Damit: 49 = 32 + 16 + 1
    = 100000 + 010000 + 000001
    = 110001



Bit, Byte, (Doppel-) Wort

Einheit	Erklärung	Werte	Anzahl Werte
Bit	Binary Digit	0, 1	2
Byte	8 Bit	0 – 255	256
Wort	2 Byte, 16 Bit	0 – 65535	65536
Doppelwort	2 Worte (4 Byte)	0 – 4294967295	4294967296

Kodierung (1/3)

- Was der Computer gut kann:
  Bits und Bytes speichern; auch (Doppel-) Worte und längere Bitfolgen (mehrere Bytes verwenden)
- Problem: allgemeine Daten speichern
- Lösung: „beliebige“ Daten auf Bitfolgen abbilden

Kodierung (2/3)

- Beispiel: „A-Z“ als 1-26 speichern, binär:
  A = 00001, B = 00010, ...

A	00001	H	01000	O	01111	V	10110	unbenutzt
B	00010	I	01001	P	10000	W	10111	00000
C	00011	J	01010	Q	10001	X	11000	11011
D	00100	K	01011	R	10010	Y	11001	11100
E	00101	L	01100	S	10011	Z	11010	11101
F	00110	M	01101	T	10100			11110
G	00111	N	01110	U	10101			11111

- Also 5 Bit pro Buchstabe
- HALLO → 01000 00001 01100 01100 01111

Kodierung (3/3)

- Praxis: „nur Großbuchstaben“ unbrauchbar
- Kleinbuchstaben, Zahlen, Sonderzeichen
- erster Standard: ASCII, enthält die Zeichen: 0-9, A-Z, a-z, das Leerzeichen „ “ und: `!"#$%&\'()*+,-./:;<=>?@[\]^_`{|}~`

ASCII-Tabelle (1/2)

- **ASCII = A**merican **S**tandard **C**ode for **I**nformation **I**nterchange:

32		48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	

ASCII-Tabelle (2/2)

- ASCII – und was ist mit Umlauten?
  - „einfach drauf verzichten“ – ae, oe, ue, Ae, Oe, Ue und ss stoeren hoechstens die Uebergenauen
  - Zeichen aus der Zeichentabelle werfen und durch Umlaute ersetzen, etwa
    [→ ä,] → ö, \ → ü,
    { → Ä, } → Ö, | → Ü usw.
  - ASCII verwendet nur 7 Bit (0-127); über achtes Bit neuen Zeichensatz mit Sonderzeichen definieren, z. B. ISO-8859-15 (Westeuropa mit €-Zeichen). Dort: 0-127 wie ASCII, 128-255 Zusatzzeichen

ISO-8859-15

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
NBSP	¡	¢	£	€	¥	Š	§	š	©	ª	«	¬	SHY	®	¯
°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Quelle: http://de.wikipedia.org/wiki/ISO_8859-15

Oktal- und Hexadezimalzahlen (1/9)

- Noch zwei weitere Zahlensysteme
- Bisher:
  - Dualzahlen – Basis 2
  - Dezimalzahlen – Basis 10
- Jetzt:
  - Oktalzahlen – Basis 8
  - Hexadezimalzahlen – Basis 16

Oktal- und Hexadezimalzahlen (2/9)

Oktalzahlen

Basis 8, also 8 Ziffern:
0, 1, 2, 3, 4, 5, 6, 7

$7_o + 1_o = 10_o$

oktal	dez.	binär	oktal	dez.	binär
0	0	0	14	12	1100
1	1	1	15	13	1101
2	2	10	16	14	1110
3	3	11	17	15	1111
4	4	100	20	16	10000
5	5	101	...		
6	6	110	77	63	111111
7	7	111	100	64	1000000
10	8	1000	101	65	1000001
11	9	1001	102	66	1000010
12	10	1010	103	67	1000011
13	11	1011	104	68	1000100

Oktal- und Hexadezimalzahlen (3/9)

Hexadezimalzahlen

- Basis 16, also 16 Ziffern:
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  10 11 12 13 14 15
- $9_h + 1_h = A_h$,
  $F_h + 1_h = 10_h$ (mit Wert 16)

Oktal- und Hexadezimalzahlen (4/9)

hex.	dez.	binär	hex.	dez.	binär	hex.	dez.	binär
0	0	0	10	16	10000	20	32	100000
1	1	1	11	17	10001	21	33	100001
2	2	10	12	18	10010	22	34	100010
3	3	11	13	19	10011	23	35	100011
4	4	100	14	20	10100	24	36	100100
5	5	101	15	21	10101	25	37	100101
6	6	110	16	22	10110	26	38	100110
7	7	111	17	23	10111	27	39	100111
8	8	1000	18	24	11000	...		
9	9	1001	19	25	11001	FF	255	11111111
A	10	1010	1A	26	11010	100	256	100000000
B	11	1011	1B	27	11011	101	257	100000001
C	12	1100	1C	28	11100	102	258	100000010
D	13	1101	1D	29	11101	...		
E	14	1110	1E	30	11110	FFF	4095	111111111111
F	15	1111	1F	31	11111	1000	4096	1000000000000

Oktal- und Hexadezimalzahlen (5/9)

- Umrechnen Dual ↔ Oktal
  und Dual ↔ Hexadezimal
  ist also leicht:
- Jede Ziffer einer Oktalzahl wird zu drei Bits:
  $307_o = 011\ 000\ 111_b$
- Jede Ziffer einer Hex.-Zahl wird zu vier Bits:
  $3AF_h = 0011\ 1010\ 1111_b$

Oktal- und Hexadezimalzahlen (6/9)

Umrechnen in Dezimalzahlen: Wie bei Dualzahlen, aber mit anderer Basis

- $1101_b = 1 \times \underline{2}^3 + 1 \times \underline{2}^2 + 0 \times \underline{2}^1 + 1 \times \underline{2}^0 = 13$
- $1734_o = 1 \times \underline{8}^3 + 7 \times \underline{8}^2 + 3 \times \underline{8}^1 + 4 \times \underline{8}^0 = 988$
- $1A3F_h = 1 \times \underline{16}^3 + 10 \times \underline{16}^2 + 3 \times \underline{16}^1 + 15 \times \underline{16}^0 = 6719$
- zur Erinnerung im Dezimalsystem:
  $3921 = 3 \times 10^3 + 9 \times \underline{10}^2 + 2 \times \underline{10}^1 + 1 \times \underline{10}^0 = 6719$

Oktal- und Hexadezimalzahlen (7/9)

Umrechnen von Dezimal- in Oktal- oder Hexadezimalzahlen:

- entweder in zwei Schritten – erst in Dualzahlen umrechnen, dann 3er- bzw. 4er-Gruppen zusammenfassen:
  $80 = 001\ 010\ 000_b = 120_o = 0101\ 0000_b = 50_h$
- oder durch schriftliches Dividieren wie bei Umrechnen in Dualzahl, aber diesmal mit Divisor 8 oder 16

Oktal- und Hexadezimalzahlen (8/9)

942 als Hexadezimalzahl?

942 : 16	= 58,	Rest	14 (E)	Einser	(16°)
58 : 16	= 3,	Rest	10 (A)	16er	(16^1)
3 : 16	= 0,	Rest	3	256er	(16^2)

→ $3AE_h$ (3 x 256 + 10 x 16 + 14 x 1)

Warum geht das? Rechnen Sie mal direkt
$3AE_h$: 16... $3AE_h = 16 \times 3A_h + E_h$

Oktal- und Hexadezimalzahlen (9/9)

942 als Oktalzahl?

942 : 8	=	117,	Rest	6	Einser	(8^0)
117 : 8	=	14,	Rest	5	8er	(8^1)
14 : 8	=	1,	Rest	6	64er	(8^2)
1 : 8	=	0,	Rest	1	512er	(8^3)

→ 1656_o (1 x 512 + 6 x 64 + 5 x 8 + 6 x 1)

Wie bei Hexadez.-Zahlen: Rechnen Sie mal direkt
1656_o : 8... $1656_o = 8 \times 165_o + 6_o$

Warum hexadezimal? (1/3)

- deckt praktische (häufig benutzte) Bereiche ab
- z. B.: Home-Computer mit 64 KByte RAM
  64 KByte = 64 x 1024 Byte = 65 536 Byte
  → hexadezimal: 10000_h Byte
  also Adressen: 0000_h – $FFFF_h$
- z. B.: PC mit 1 GByte RAM
  1 GByte = 1024 x 1024 x 1024 Byte
  = 1 073 741 824 Byte
  → hexadezimal: 40000000_h Byte
  also Adressen: 00000000_h – $3FFFFFFF_h$

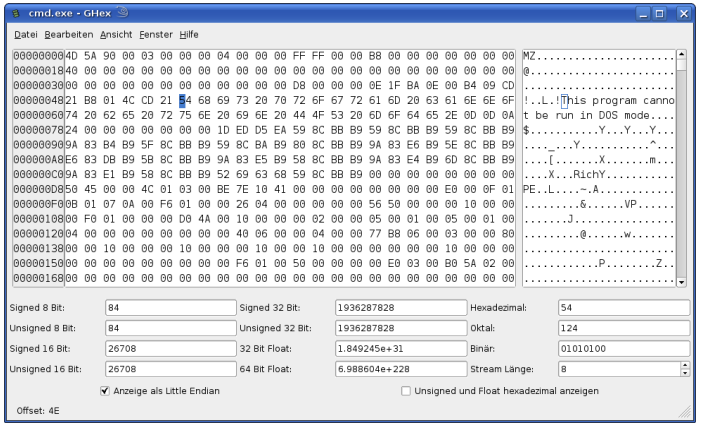
Warum hexadezimal? (2/3)

- Kodierung von Bytes (z. B.: ASCII-Zeichen), also 8 Bit, durch zwei Hex-Zahlen:

hex	dez	hex	dez	hex	dez	hex	dez	hex	dez	hex	dez	hex	dez	hex	dez		hex	dez
00	0	10	16	20	32	30	48	40	64	50	80	60	96	70	112		F0	240
01	1	11	17	21	33	31	49	41	65	51	81	61	97	71	113		F1	241
02	2	12	18	22	34	32	50	42	66	52	82	62	98	72	114		F2	242
03	3	13	19	23	35	33	51	43	67	53	83	63	99	73	115		F3	243
04	4	14	20	24	36	34	52	44	68	54	84	64	100	74	116		F4	244
05	5	15	21	25	37	35	53	45	69	55	85	65	101	75	117		F5	245
06	6	16	22	26	38	36	54	46	70	56	86	66	102	76	118	...	F6	246
07	7	17	23	27	39	37	55	47	71	57	87	67	103	77	119		F7	247
08	8	18	24	28	40	38	56	48	72	58	88	68	104	78	120		F8	248
09	9	19	25	29	41	39	57	49	73	59	89	69	105	79	121		F9	249
0A	10	1A	26	2A	42	3A	58	4A	74	5A	90	6A	106	7A	122		FA	250
0B	11	1B	27	2B	43	3B	59	4B	75	5B	91	6B	107	7B	123		FB	251
0C	12	1C	28	2C	44	3C	60	4C	76	5C	92	6C	108	7C	124		FC	252
0D	13	1D	29	2D	45	3D	61	4D	77	5D	93	6D	109	7D	125		FD	253
0E	14	1E	30	2E	46	3E	62	4E	78	5E	94	6E	110	7E	126		FE	254
0F	15	1F	31	2F	47	3F	63	4F	79	5F	95	6F	111	7F	127		FF	255

Warum hexadezimal? (3/3)

- Für viele Aufgaben ist Hex-Darstellung der Standard, z. B. Analyse von Binärdateien:



Zahlensysteme

- Sie kennen nun bereits vier Zahlensysteme:

System	Basis	Ziffern
• Dezimalsystem	10	0 – 9
• Dualsystem	2	0, 1
• Oktalsystem	8	0 – 7
• Hexadezimalsystem	16	0 – 9, A – F

- Das Ganze lässt sich auf beliebige Basen verallgemeinern: Basis *n* mit *n* Ziffern; z. B. Basis 32 mit Ziffern 0 – 9, A (10) – V (31)

Kodierung: Übungen (1/2)

1. Stellen Sie den folgenden Text:
   `Infrastruktur`
   - im Dezimalformat (normales 10er-System),
   - im Hexadezimalformat, Oktalformat und Dualformat dar. Wählen Sie dafür eine sinnvolle Reihenfolge!

   Dazu benötigen Sie:
   - ASCII-Tabelle (1. Schritt)
   - Hex/Oktal/Dual-Tabellen (2. Schritt)

Kodierung: Übungen (2/2)

2. a) Wie viele 16-stellige Bitfolgen gibt es?
   b) Wie viele verschiedene Worte („Doppel-Bytes“) gibt es?
3. Rechnen Sie die Hexadezimalzahl $\mathbf{A01F_h}$ in Dual- und Oktaldarstellung um.
4. Wenn Sie wissen, dass $\mathbf{FF_h = 100_h - 1}$ gilt, wie können Sie dann schnell $\mathbf{FFF_h}$ und $\mathbf{FFFF_h}$ in Dezimalzahlen umrechnen?

Zahlensysteme: Fragen (1/2)

- Aus welchen Zahlensystemen kann die folgende Zahl stammen? Aus welchen nicht? Warum? **41823**
  - Berücksichtigen Sie: Dual-, Oktal-, Dezimal- und Hexadezimalsystem.
- Können Sie **2049** (dezimal) ganz schnell in eine Dualzahl umrechnen?
- Wie sieht das Wort „Übung“ in ASCII-Darstellung aus? (Achtung: Fangfrage)

Zahlensysteme: Fragen (2/2)

- Was ist größer: 74_h oder **75** ?
- Wie viele Hexadezimalzahlen liegen zwischen $7A7_h$ und $8B8_h$? (Zählen Sie die beiden „begrenzenden“ Zahlen mit.)

Größenordnungen

Name (Symbol)	SI-konforme Bedeutung[1]	„klassische“ Bedeutung	Unterschied
Kilobyte (kB)	10^3 Byte = 1.000 Byte	2^{10} Byte = 1.024 Byte	2,4 %
Megabyte (MB)	10^6 Byte = 1.000.000 Byte	2^{20} Byte = 1.048.576 Byte	4,9 %
Gigabyte (GB)	10^9 Byte = 1.000.000.000 Byte	2^{30} Byte = 1.073.741.824 Byte	7,4 %
Terabyte (TB)	10^{12} Byte = 1.000.000.000.000 Byte	2^{40} Byte = 1.099.511.627.776 Byte	10,0 %
Petabyte (PB)	10^{15} Byte = 1.000.000.000.000.000 Byte	2^{50} Byte = 1.125.899.906.842.624 Byte	12,6 %
Exabyte (EB)	10^{18} Byte = 1.000.000.000.000.000.000 Byte	2^{60} Byte = 1.152.921.504.606.846.976 Byte	15,3 %
Zettabyte (ZB)	10^{21} Byte = 1.000.000.000.000.000.000.000 Byte	2^{70} Byte = 1.180.591.620.717.411.303.424 Byte	18,1 %

[1]) SI: Internationales Einheitensystem (Système International d'Unités) Quelle: Wikipedia, „Byte“

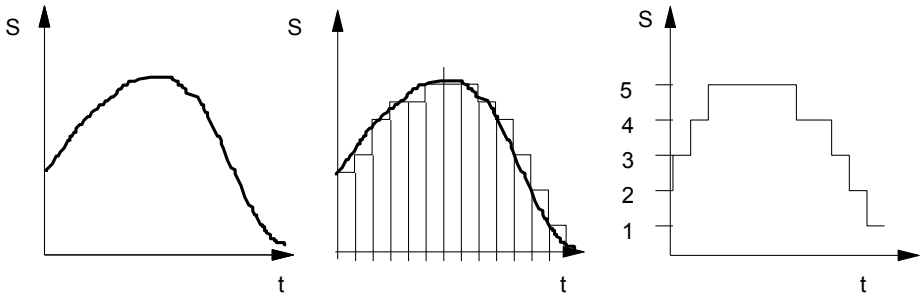
Informationen speichern

- Bisher: Einfache Texte (z. B. im ASCII- oder ISO-8859-15-Format) in Bytes gespeichert
- Was tun mit analogen Daten?
  - Wie speichert ein digitaler Fotoapparat Bilder?
  - Wie speichert ein MP3-Player Musik?
  - Was unterscheidet Schallplatte und CD?
- Digitalisierung von analogen Daten

Digitalisierung analoger Daten

- **Rasterung** (**Diskretisierung**): Abtasten (Sampling) an diskreten zeitlichen oder örtlichen Punkten
  - diskret = nur endlich viele Werte
  - Gegenteil: kontinuierlich, z.B. Werte aus $\mathbb{Q}$
- **Quantisierung** der abgetasteten Werte (runden auf wenige mögliche Werte)
- **Digitalisierung** (Kodierung): Darstellung des abgetasteten und quantisierten Signals als Digitalcode

Zeitliche Digitalisierung

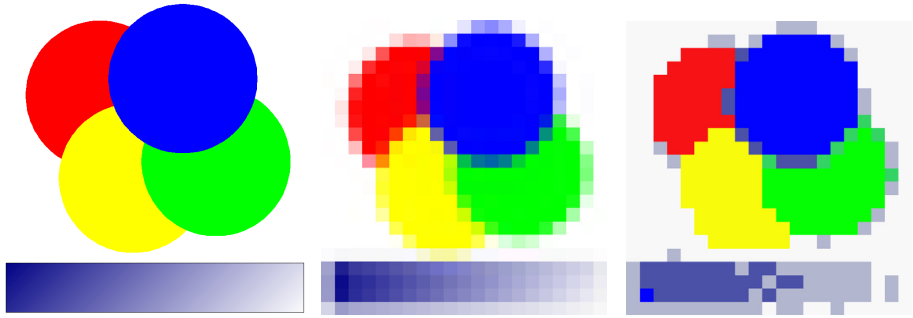


analoges Signal

„gesampelt“: Proben zu festen Zeitpunkten gezogen – noch mit exakten Werten

quantisiert: alle Messwerte auf 1, 2, 3, 4, 5 gerundet

Örtliche Digitalisierung



gleichmäßige Färbung über ganze Bereiche

begrenzt viele (diskrete) Farbabstufungen

Kodierung von Farbinformation: Die Farbe eines Bildpunktes ist eine additive Mischung der 3 Grundfarben Rot, Grün und Blau → für Darstellung des Farbbildes wird die Rot-, Grün- und Blau-Intensität für jeden Bildpunkt separat digitalisiert.

Gebräuchliche Datenformate

- Text: ASCII/ANSI/Unicode (.txt), RTF (.rtf), Word (.doc), LibreOffice (.odt)
- Tabellen: Excel (.xls), LibreOffice (.ods)
- Dokumente: .doc, .pdf, HTML (.html / .htm), …
- strukturierte Daten: XML
- Rastergrafik: Bitmap (.bmp), Graphics Interchange Format (.gif), JPEG, TIFF, PNG
- CAD: VDA-FS, IGES, STEP
- Audiodateien: MP3 (.mp3), Ogg Vorbis (.ogg)
- Bildfolgen (Video): MPEG

Alltagsthemen der Kodierung

- Reduktion der Datenmenge durch Kompression
  - verlustfrei
  - verlustbehaftet
- Fehlerhafte Darstellung aufgrund ungleicher Zeichenvorräte bei Sender und Empfänger (z. B. bei Darstellung von nationalen Sonderzeichen)
- Fehlerhafte Übertragung
  → Codes mit Fehlererkennung, Fehlerkorrektur
- Verschlüsselung, Signatur

Kompression (1/3)

- Einfaches Verfahren:
  - „AAAAABBCCCCCCCCDD“ → „5A2B8C2D“ (kürzer)
  - aber: „ABCDAB“ → „1A1B1C1D1A1B“ (länger)
- verlustbehaftet:
  - „IT-Infrastruktur“ → „it infrastruktur“ (kleineres Alphabet: a-z + Leerzeichen = 27 Zeichen, weniger Bits pro Zeichen)
  - „IT-Infrastruktur“ → „TNFRSTRKTR“ (gleiches Prinzip; hier: Verzicht auf Vokale)

Kompression (2/3)

- Besseres Verfahren: Buchstabenhäufigkeiten
  - Welche Buchstaben kommen wie häufig vor? Reihenfolge („top ten“) festlegen, z. B. a, e, i, o, u, n, t, s, r, p, k, l, m, ..., q, y
- Dann Kodierung
  - von „häufigen“ Buchstaben in kurze Bitfolgen
  - und von „seltenen“ Buchstaben in lange Bitfolgen
  - z. B. a → 000, e → 001, i → 010, o → 011,
    u → 1000, n → 1001, t → 1010, s → 1011,
    r → 11000, k → 11001, l → 11010, m → 11011,
    ..., q → 11111001, y → 11111010

Kompression (3/3)

- Beispiele mit obiger Tabelle:
  - „nein“ → 1001 001 010 1001
  - „test“ → 1010 001 1011 1010
  - „mlml“ → 11011 11010 11011 11010
  - „qyqy“ → 11111001 11111010 11111001 11111010
- Je „unwahrscheinlicher“ ein Wort, desto länger seine Kodierung

Fehlererkennung (1/3)

- Übertragung von digitalen Daten (etwa über eine Telefonleitung oder ein Datenkabel zwischen zwei PCs) ist oft fehlerhaft:
  - einzelne Bits „kippen“ bei der Übertragung
  - einzelne Bits gehen bei der Übertragung komplett verloren
- Fehlererkennung: Übertragen von zusätzlichen Informationen, mit denen Fehler erkannt werden

Fehlererkennung (2/3)

- Einfachste Möglichkeit: Alles doppelt senden
  - Empfänger vergleicht die beiden empfangenen Infomationen, die identisch sein sollten – sind sie es nicht, hat es einen Fehler gegeben
  - Wahrscheinlichkeit, dass die Daten zweimal mit exakt demselben Fehler übertragen werden, ist klein
  - Verfahren ist aber sehr aufwendig:
    - Verdopplung der übertragenen Datenmenge
    - d. h. Halbierung der Übertragungsgeschwindigkeit

Fehlererkennung (3/3)

- Idee: Daten mit **Prüfsummen** versehen
  - einfaches Beispiel: 7 Bit (ASCII-Zeichen) um sog. **Paritäts-Bit** ergänzen:
  - Im ASCII-Code ist das höchstwertige (ganz links) stehende Bit immer 0
  - Verwende dieses Bit wie folgt:
    - Wenn die Summe der übrigen Bits ungerade ist, setze es auf 1
    - Wenn die Summe der übrigen Bits gerade ist, setze es auf 0

ASCII + Paritätsbit

	ASCII	ASCII binär (nur 7 Bit)	Anzahl Einsen	ASCII + Paritätsbit
A	65	01000001	gerade	**0**1000001
B	66	01000010	gerade	**0**1000010
C	67	01000011	ungerade	**1**1000011
D	68	01000100	gerade	**0**1000100
E	69	01000101	ungerade	**1**1000101

- In den Bytes **mit** Paritätsbit: Anzahl Einsen immer gerade
- „Kippt“ ein Bit, fällt der Fehler auf
- Kippen zwei Bits, dann nicht...

Übung: ASCII-Fehlerkorrektur

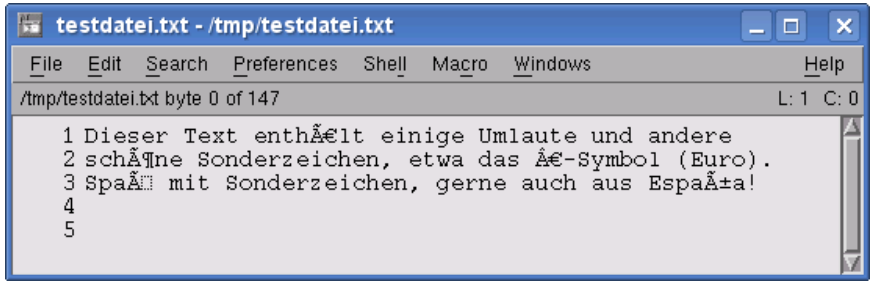
- Sie haben folgende Nachricht (binär) erhalten: 01001000, 11100001, 11101100, 11101110, 01101111
- Prüfen Sie zunächst, welche Zeichen ohne „Bit-Kipper“ übertragen wurden
  - Für alle korrekt empfangenen schlagen Sie in der ASCII-Tabelle den zugehörigen Buchstaben nach,
  - für alle fehlerhaft empfangenen setzen Sie ein Fragezeichen ein.

65	A	97	a
66	B	98	b
67	C	99	c
68	D	100	d
69	E	101	e
70	F	102	f
71	G	103	g
72	H	104	h
73	I	105	i
74	J	106	j
75	K	107	k
76	L	108	l
77	M	109	m
78	N	110	n
79	O	111	o
80	P	112	p
81	Q	113	q
82	R	114	r
83	S	115	s
84	T	116	t
85	U	117	u
86	V	118	v
87	W	119	w
88	X	120	x
89	Y	121	y
90	Z	122	z

Datenformate: Text, *.txt (1/8)

- ASCII bereits vorgestellt
- Problem: Regionale Zeichen (äöüßñš...) nicht in ASCII-Tabelle
- Erweiterungen der ASCII-Tabelle
  - 8-bittig: Zeichen 128-255 verwenden
  - 16-bittig: Platz für „alles“, inkl. Kyrillisch, Chinesisch, Japanisch etc.
  - „Mischform“ 8- und 16-bittig

Datenformate: Text, *.txt (2/8)



Was ist hier passiert?

Datenformate: Text, *.txt (3/8)

- Es gibt etliche Kodierungen für Textdateien
- Man muss wissen, in welcher Kodierung eine Datei gespeichert wurde
- Alternative: selbst-beschreibende Datei

```
<html>
<head>
<title>Eine Webseite</title>
<meta http-equiv="Content-Type"
      content="text/html; charset=iso-8859-1">
...
```


Datenformate: Text, *.txt (4/8)

Wichtige Kodierungen:

- **Latin-1** (iso-8859-1, ISO/IEC 8859-1, westeuropäisch)

¡¢£¤¥¦§¨©ª«¬ ®¯
°±²³´µ¶·¸¹º»¼½¾¿
ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏ
ÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞß
àáâãäåæçèéêëìíîï
ðñòóôõö÷øùúûüýþÿ

- **Latin-9** (iso-8859-15, ISO/IEC 8859-15, westeuropäisch)
  - gegenüber Latin-1 mehr Buchstaben, weniger math. Symbole
  - u.a. € und Š, š, Ž, ž, Œ, œ, Ÿ

Datenformate: Text, *.txt (5/8)

Unicode

- Ziel: Alle Zeichen aus allen Schriftarten der Welt unterstützen
- bis zu 1.114.112 Zeichen in 17 Codebereichen („Planes“) mit je 65.536 (64K) Zeichen
- aktuell (Unicode 5.1): 100.713 Zeichen
- wichtigster Codebereich: **BMP** (Basic Multilingual Plane) mit allen aktuellen Schriftsystemen
- aber: keine 21-bittige Kodierung, sondern ...

Datenformate: Text, *.txt (6/8)

Unicode-Formate / UTF-16, UTF-8

- UTF-8
  - ASCII-Zeichen (0–127) „normal“ (8-bittig) kodiert
  - andere Zeichen 16-, 24- oder 32-bittig (also über zwei, drei oder vier Bytes) kodiert
  - z. B. deutsche Umlaute: je zwei Bytes
- UTF-16
  - Zeichen der **BMP** 16-bittig (mit 2 Bytes) kodiert
  - sonstige Zeichen 32-bittig (mit 4 Bytes) kodiert

Datenformate: Text, *.txt (7/8)

Beispiele: Umlaute ÄÖÜ, äöü, ß. (in Latin-9 erzeugt)
Euro: €. Britisches Pfund: £.
Französische «Anführungen»

Latin-9-Kodierung:

```
00000000  42 65 69 73 70 69 65 6c  65 3a 20 55 6d 6c 61 75  |Beispiele: Umlau|
00000010  74 65 20 c4 d6 dc 2c 20  e4 f6 fc 2c 20 df 2e 0a  |te ÄÖÜ, äöü, ß..|
00000020  45 75 72 6f 3a 20 a4 2e  20 42 72 69 74 69 73 63  |Euro: €. Britisc|
00000030  68 65 73 20 50 66 75 6e  64 3a 20 a3 2e 0a 46 72  |hes Pfund: £..Fr|
00000040  61 6e 7a f6 73 69 73 63  68 65 20 ab 41 6e 66 fc  |anzösische «Anfü|
00000050  68 72 75 6e 67 65 6e bb                           |hrungen»|
```

UTF-8-Kodierung:

```
00000000  42 65 69 73 70 69 65 6c  65 3a 20 55 6d 6c 61 75  |Beispiele: Umlau|
00000010  74 65 20 c3 84 c3 96 c3  9c 2c 20 c3 a4 c3 b6 c3  |te Ã.Ã.Ã., ÃƒÃ¶Ã|
00000020  bc 2c 20 c3 9f 2e 0a 45  75 72 6f 3a 20 c2 a4 2e  |Œ, Ã...Euro: Â€.|
00000030  20 42 72 69 74 69 73 63  68 65 73 20 50 66 75 6e  | Britisches Pfun|
00000040  64 3a 20 c2 a3 2e 0a 46  72 61 6e 7a c3 b6 73 69  |d: Â£..FranzÃ¶si|
00000050  73 63 68 65 20 c2 ab 41  6e 66 c3 bc 68 72 75 6e  |sche Â«AnfÃœhrun|
00000060  67 65 6e c2 bb                                    |genÂ»|
```


Datenformate: Text, *.txt (8/8)

UTF-16-Kodierung:

```
00000000  fe ff 00 42 00 65 00 69  00 73 00 70 00 69 00 65  |þÿ.B.e.i.s.p.i.e|
00000010  00 6c 00 65 00 3a 00 20  00 55 00 6d 00 6c 00 61  |.l.e.:. .U.m.l.a|
00000020  00 75 00 74 00 65 00 20  00 c4 00 d6 00 dc 00 2c  |.u.t.e. .Ä.Ö.Ü.,|
00000030  00 20 00 e4 00 f6 00 fc  00 2c 00 20 00 df 00 2e  |. .ä.ö.ü.,. .ß..|
00000040  00 0a 00 45 00 75 00 72  00 6f 00 3a 00 20 00 a4  |...E.u.r.o.:. .€|
00000050  00 2e 00 20 00 42 00 72  00 69 00 74 00 69 00 73  |... .B.r.i.t.i.s|
00000060  00 63 00 68 00 65 00 73  00 20 00 50 00 66 00 75  |.c.h.e.s. .P.f.u|
00000070  00 6e 00 64 00 3a 00 20  00 a3 00 2e 00 0a 00 46  |.n.d.:. .£.....F|
00000080  00 72 00 61 00 6e 00 7a  00 f6 00 73 00 69 00 73  |.r.a.n.z.ö.s.i.s|
00000090  00 63 00 68 00 65 00 20  00 ab 00 41 00 6e 00 66  |.c.h.e. .«.A.n.f|
000000a0  00 fc 00 68 00 72 00 75  00 6e 00 67 00 65 00 6e  |.ü.h.r.u.n.g.e.n|
000000b0  00 bb                                             |.»|
```

Dateigrößen:

```
test-latin9.txt     88 Bytes
test-utf8.txt      101 Bytes
test-utf16.txt     178 Bytes
```

Besondere Textformate

- Neben reinen Textformaten (ASCII, ISO-Latin-*, UTF): Textformate mit Auszeichnungen (Markup)
  - HTML (**H**yper**T**ext **M**arkup **L**anguage) (Webseiten)
  - LaTeX (Textsatz-Auszeichnungssprache)
  - SGML (**S**tandard **G**eneralized **M**arkup **L**anguage)
  - XML (E**x**tensible **M**arkup **L**anguage) (Grundlage für viele Markup-Sprachen)
  - Wikitext (Texte in Wikis)

Markup-Sprachen

- Markup-Sprachen ergänzen den reinen Text mit Markup-Elementen
- Zweierlei Markup möglich:
  - „Grafisches“ Markup (fett, kursiv, Schriftgröße)
  - Struktur-Markup (Kapitelanfang, Zitat)
  - Kombination (mit Stylesheets)
- Markup unabhängig von Kodierung der Textinhalte; oft beliebige Kodierung möglich

HTML, LaTeX und Wikitext

Beispiele: HTML, LaTeX, Wikitext

Beispiel für	HTML	LaTeX	Wikitext
Überschrift	`<h1>Überschrift</h1>`	`\section{Überschrift}`	`= Überschrift =`

Aufzählung	`<ul>`
	`<li>Punkt 1</li>`
	`<li>Punkt 2</li>`
	`<li>Punkt 3</li>`
	`</ul>`
		`\begin{itemize}`
		`\item Punkt 1`
		`\item Punkt 2`
		`\item Punkt 3`
		`\end{itemize}`
			`* Punkt 1`
			`* Punkt 2`
			`* Punkt 3`

fetten Text	`<b>fett</b>`	`\textbf{fett}`	`'''fett'''`
kursiven Text	`<i>kursiv</i>`	`\textit{kursiv}`	`''kursiv''`

Quelle: Wikipedia, http://de.wikipedia.org/wiki/Auszeichnungssprache#Beispiele

Datenformate: HTML / *.html (1/8)

- Grober Aufbau eines HTML-Dokuments:

```
<html>
  <head>
    <title>Titel des Dokuments</title>
  </head>
  <body>
    <h1>Überschrift</h1>
    <p>Ein Absatz Text reicht für das Beispiel</p>
  </body>
</html>
```

- unterteilt in Head und Body
- öffnende und schließende Tags <xyz>, </xyz>

Datenformate: HTML / *.html (2/8)

- < und > sind Kontrollzeichen, die ein Tag einleiten / beenden
- Um „<“ oder „>“ im Text zu verwenden:
  - `<` (less than) „<“
  - `>` (greater than) „>“
  - Beispiel:
    `Das Tag <html>` → Das Tag <html>
- Sonderzeichen: `ä` (ä), `Ä` (Ä) etc., `ß` (ß, „S-Z-Ligatur“); `€` (€)

Datenformate: HTML / *.html (3/8)

- Cascading Stylesheets (CSS):
  - trennen die inhaltliche von der grafischen Auszeichnung
  - Texte verwenden Tags mit Klassenangabe
  - Stylesheet enthält Layout-Definition für das Tag und die Klasse
- Position des Stylesheets
  - wahlweise direkt in der HTML-Datei
  - oder in separater *css*-Datei

Datenformate: HTML / *.html (4/8)

- Ziel: Drei Absätze in verschiedenen Zeichenformatierungen
  - Arial, 20 pt
  - Courier, 15 pt
  - Times, 10 pt

Das ist ein Stylesheet-Test. (Arial 20pt)

Das ist ein Stylesheet-Test. (Courier 15pt)

Das ist ein Stylesheet-Test. (Times 10pt)

Datenformate: HTML / *.html (5/8)

- Formatierung ohne CSS, direkt mit `style=...`

```
<html>
<head><title>Test ohne CSS-Datei</title></head>
<body>

<p style="font-family: arial;
          font-size: 20pt">
Das ist ein Stylesheet-Test. (Arial 20pt)
</p>

<p style="font-family: courier;
          font-size: 15pt">
Das ist ein Stylesheet-Test. (Courier 15pt)
</p>

<p style="font-family: times;
          font-size: 10pt">
Das ist ein Stylesheet-Test. (Times 10pt)
</p>
```

Datenformate: HTML / *.html (6/8)

- Formatierung mit CSS-Datei

```
<html>
<head><title>Test mit CSS-Datei</title>
<link rel="stylesheet" type="text/css"
      href="/stylesheet.css"> </head>
<body>

<p class="arial20">
Das ist ein Stylesheet-Test. (Arial 20pt)
</p>

<p class="courier15">
Das ist ein Stylesheet-Test. (Courier 15pt)
</p>

<p class="times10">
Das ist ein Stylesheet-Test. (Times 10pt)
</p>
```

Stylesheet-Datei:

```
p.arial20 {
  font-family: arial;
  font-size: 20pt;
}

p.courier15 {
  font-family: courier;
  font-size: 15pt;
}

p.times10 {
  font-family: times;
  font-size: 10pt;
}
```


Datenformate: HTML / *.html (7/8)

- Aufbau der Einträge in der CSS-Datei

Es geht um das <p>-Attribut

in einer Klasse *arial20*

```
p.arial20 {
  font-family: arial;
  font-size:   20pt;
}
```

Das sind die Eigenschaften

Datenformate: HTML / *.html (8/8)

- Hier keine vollständige Einführung in HTML und CSS
- Gute Dokumentation im Web: SelfHTML http://de.selfhtml.org/

Datenformate: LaTeX / *.tex (1/4)

- TeX (sprich: „Tech“) ist eine Dokumentenbeschreibungssprache, LaTeX ein Makropaket für TeX. Man benutzt meist LaTeX
- Auszeichnung von Kapitelüberschriften, Zitaten, … über LaTeX-Befehle, z. B. `\section{Abschnitt}`, `\begin{quotation}` … `\end{quotation}`

Datenformate: LaTeX / *.tex (2/4)

- kleines Beispieldokument

```
\documentclass{article}
\begin{document}

\tableofcontents    % erzeuge Inhaltsverzeichnis

\section{Erster Abschnitt}

Hallo, das ist ein Beispieltext mit einem
\emph{hervorgehobenen} und einem \textbf{fett}
geschriebenen Wort.

\end{document}
```

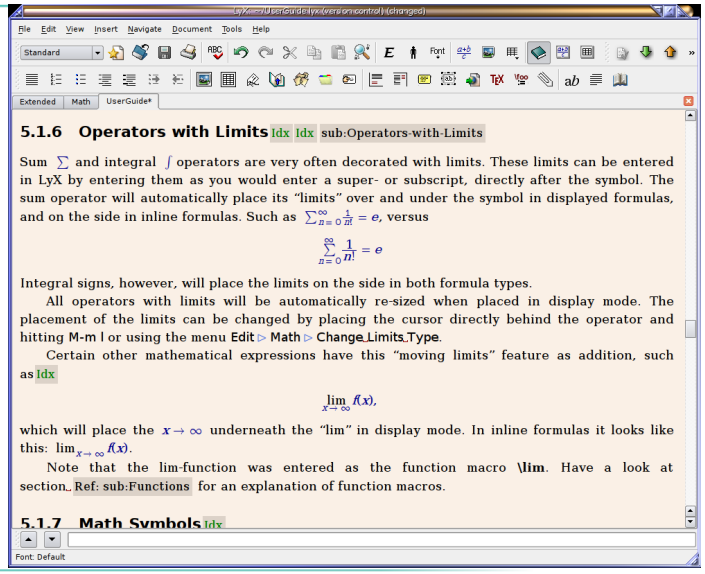

Datenformate: LaTeX / *.tex (3/4)

- TeX/LaTeX selbst arbeitet wie ein Compiler und erzeugt aus den Eingabedateien (*.tex) PDF-Dateien
- Wie bei Programmiersprachen: Syntax- und komplexere Fehler möglich, die zum Abbruch führen können
- LaTeX kann aber auch unabhängig vom LaTeX-Compiler als eigenständiges Datenformat verstanden werden → Converter *latex2html*

Datenformate: LaTeX / *.tex (4/4)

LyX
(LaTeX-GUI)

Quelle Bild:
http://www.lyx.org



Datenformate: XML / *.xml (1/6)

- **HTML:** Markup-Sprache mit fest vorgegebenen Tags (z. B. <head>, <font>, <p>, <table> etc.)
- **XML:** **E**xtensible **M**arkup **L**anguage
- **XML** erlaubt es, **eigene** Markup-Sprachen mit HTML-ähnlicher Syntax zu definieren
- unterscheiden zwischen
  - Dokument, das die Regeln für die eigene Sprache definiert (→ **DTD**, Document Type Definition)
  - Dokument, das nach diesen Regeln erstellt (ausgezeichnet) wurde

Datenformate: XML / *.xml (2/6)

- Eine DTD ist nicht zwingend nötig – wenn es eine gibt, erlaubt sie das **Validieren** von XML-Dokumenten
- Ohne DTD nur allgemeine Syntaxprüfung möglich

Datenformate: XML / *.xml (3/6)

Syntaxfehler (nicht „wohlgeformt“)

```
<person>
  <name>Müller</name>
  <ort>München</tor>
  <tel v="0123456" />
</person>

<person>
  <name>Meier</name>
  <ort>Augsburg</ort>
  <tel v="0987641">
</person>

<person>
  <name>Huber
  <ort>Erlangen
  </name>
  </ort>
  <tel v="0448"></tel>
</person>
```

DTD-Fehler (nicht „gültig“)

```
<person>
  <wer>Müller</wer>
  <wo>München</wo>
  <tel v="0123456" />
</person>

<leute>
  <name>Meier</name>
  <ort>Augsburg</ort>
  <tel v="0123456" />
</leute>

<person> !
  <ort>Erlangen</ort>
  <tel v="0448" />
</person>
```

(im letzten Eintrag fehlt ein vorgeschriebenes <name> Element)

alles korrekt

```
<person>
  <name>Müller</name>
  <ort>München</ort>
  <tel v="0123456" />
</person>

<person>
  <name>Meier</name>
  <ort>Augsburg</ort>
  <tel v="0987641" />
</person>

<person>
  <name>Huber</name>
  <ort>Erlangen</ort>
  <tel v="0448" />
</person>
```


Datenformate: XML / *.xml (4/6)

- XML erlaubt Transformation von Dokumenten in andere Darstellungen
- **XSLT:** XSL Transformation (**XSL** = E**x**tensible **S**tylesheet **L**anguage)
- Syntax recht komplex; siehe http://de.wikipedia.org/wiki/XSL_Transformation

Datenformate: XML / *.xml (5/6)

- XHTML: Neuformulierung von HTML in XML
- Abweichungen minimal, z. B.:
  - alle Tags müssen geschlossen werden
  - Attribute immer in Anführungen
  - nur Kleinschreibung

```
<!DOCTYPE html ... >
<html>
...
<H1>Test</H1>
<img src="test.jpg">
<td width=10 height=10>
<ul>
  <li>Eintrag 1
  <li>Eintrag 2
</ul>
```

von HTML → zu XHTML

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html  .... >
<html xmlns="http://www.w3.org/1999/xhtml" ...>
...
<h1>Test</h1>
<img src="test.jpg" />
<td width="10" height="10">
<ul>
  <li>Eintrag 1</li>
  <li>Eintrag 2</li>
</ul>
```

Datenformate: XML / *.xml (6/6)

- **DocBook:** XML-basiertes Format für komplexe Dokumente, z. B.
  - Bücher
  - technische Dokumentation

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN" ...>
<book lang="de">
  <bookinfo>
    <title>
      Ein kurzes Buch
    </title>
  </bookinfo>
  <chapter>
    <title>
      Das erste Kapitel
    </title>
    <para>
      Ein einziger Absatz füllt das erste Kapitel.
    </para>
  </chapter>
</book>
```


Datenformate: SGML

- SGML: **S**tandard **G**eneralized **M**arkup **L**anguage
- gleicher Ansatz wie bei XML, erlaubt aber noch komplexere Dinge, z. B. Unterdokumente
- genauer:

  „The Extensible Markup Language (XML) is a subset of SGML [...]. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.“

Quelle: http://www.w3.org/TR/2006/REC-xml-20060816/

Word & LibreOffice Writer

- Auch Word- und LibreOffice-Textdateien bestehen aus Inhalt + Formatierung
- Alte Versionen von Word und OpenOffice (LibreOffice-Vorläufer) verwenden Binärformate zum Speichern der Informationen (Dateiendungen: **.doc*, **.sdw*)
- Neue Versionen erzeugen Zip-Archive, die XML-Dateien enthalten (Dateiendungen: **.docx*, **.odt*)

Textverarbeitung: Binärformat

Binäre Textdokumente (Word & Co.) ...

- ... sind (von Anwendern) nicht lesbar
- ... können von Fremdprogrammen nur mit Hilfe von Importfiltern gelesen werden, wenn der Aufbau dokumentiert ist
- ... sind schlimmstenfalls einfache „Memory Dumps“ der Anwendungen

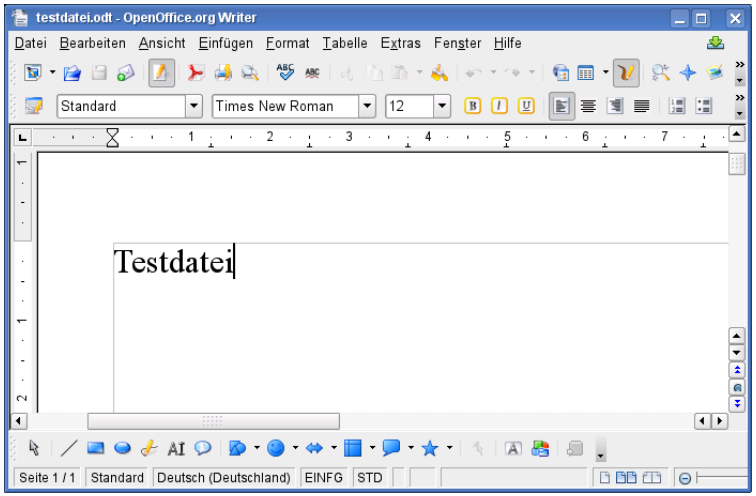
Binärformat: Word

```
$ hexdump -C linux.doc

00000000  d0 cf 11 e0 a1 b1 1a e1  00 00 00 00 00 00 00 00  |ÐÏ.à¡±.á........|
00000010  00 00 00 00 00 00 00 00  3e 00 03 00 fe ff 09 00  |........>...þÿ..|
00000020  06 00 00 00 00 00 00 00  00 00 00 00 04 00 00 00  |................|
00000030  cb 01 00 00 00 00 00 00  00 10 00 00 cd 01 00 00  |Ë...........Í...|
00000040  01 00 00 00 fe ff ff ff  00 00 00 00 c7 01 00 00  |....þÿÿÿ....Ç...|
00000050  c8 01 00 00 c9 01 00 00  ca 01 00 00 ff ff ff ff  |È...É...Ê...ÿÿÿÿ|
[...]
00034110  0b 00 00 00 4e 6f 72 6d  61 6c 2e 64 6f 74 00 20  |....Normal.dot. |
00034120  1e 00 00 00 11 00 00 00  20 48 61 6e 73 2d 47 65  |........ Hans-Ge|
00034130  6f 72 67 20 45 df 65 72  00 00 64 00 1e 00 00 00  |org Eßer..d.....|
00034140  02 00 00 00 31 00 61 6e  1e 00 00 00 13 00 00 00  |....1.an........|
00034150  4d 69 63 72 6f 73 6f 66  74 20 57 6f 72 64 20 39  |Microsoft Word 9|

$ strings -e S linux.doc
                                        Vorwort
Linux                                    PAGEREF _Toc13267048 \h
zum Nachschlagen                         HYPERLINK \l "_Toc13267049"
SuSE Linux 8.x
Hans-Georg Eßer                         [...]
Inhalt
 TOC \o "1-3" \h \z                     Normal.dot
 HYPERLINK \l "_Toc13267047"             Hans-Georg Eßer
Inhalt                                  Microsoft Word 9.0
 PAGEREF _Toc13267047 \h
 HYPERLINK \l "_Toc13267048"            [...]
```


Textverarbeitung: XML-Format

- Neuere Versionen (Word: **.docx*; OOO: **.odt*) speichern im XML-Format
- Vorteile:
  - einfacheres Einlesen / Bearbeiten durch Fremdprogramme
  - ermöglicht automatisches Generieren durch andere Tools, etwa auf einer Webseite ein Button „Seite als *docx*-Datei speichern“

XML-Format: LibreOffice (1/6)



XML-Format: LibreOffice (2/6)

```
$ unzip -l /tmp/testdatei.odt

Archive:  /tmp/testdatei.odt
  Length      Date    Time    Name
 --------    ----    ----    ----
       39  09-09-10 19:01   mimetype
        0  09-09-10 19:01   Configurations2/statusbar/
        0  09-09-10 19:01   Configurations2/accelerator/current.xml
        0  09-09-10 19:01   Configurations2/floater/
        0  09-09-10 19:01   Configurations2/popupmenu/
        0  09-09-10 19:01   Configurations2/progressbar/
        0  09-09-10 19:01   Configurations2/menubar/
        0  09-09-10 19:01   Configurations2/toolbar/
        0  09-09-10 19:01   Configurations2/images/Bitmaps/
     2683  09-09-10 19:01   content.xml
      532  09-09-10 19:01   manifest.rdf
    10302  09-09-10 19:01   styles.xml
     1097  09-09-10 19:01   meta.xml
      725  09-09-10 19:01   Thumbnails/thumbnail.png
     8387  09-09-10 19:01   settings.xml
     1989  09-09-10 19:01   META-INF/manifest.xml
 --------                   -------
    25754                   16 files
```

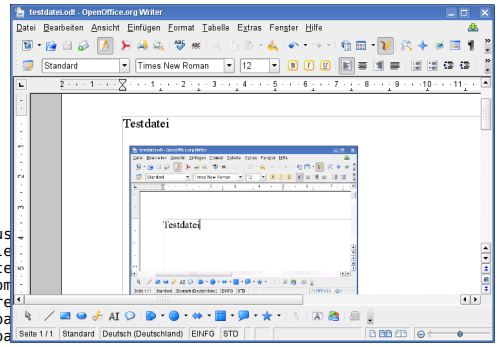
XML-Format: LibreOffice (3/6)

Inhalt von *content.xml*:

```
<office:document-content xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0" xmlns:style="urn:oasis:names:tc:opendocument:x
                         xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0" xmlns:table="urn:oasis:names:tc:opendocument:xmlns
                         xmlns:draw="urn:oasis:names:tc:opendocument:xmlns:drawing:1.0" xmlns:fo="urn:oasis:names:tc:opendocument:xmlns
                         xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:dc="http://purl.org/dc/elements/1.1/"
                         xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0" xmlns:number="urn:oasis:names:tc:opendocument:xmln
                         xmlns:svg="urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0" xmlns:chart="urn:oasis:names:tc:opendocum
                         xmlns:dr3d="urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0" xmlns:math="http://www.w3.org/1998/Math/MathML"
                         xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0" xmlns:script="urn:oasis:names:tc:opendocument:xmln
                         xmlns:ooo="http://openoffice.org/2004/office" xmlns:ooow="http://openoffice.org/2004/writer"
                         xmlns:oooc="http://openoffice.org/2004/calc" xmlns:dom="http://www.w3.org/2001/xml-events"
                         xmlns:xforms="http://www.w3.org/2002/xforms" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:rpt="http://openoffice.org/2005/report"
                         xmlns:of="urn:oasis:names:tc:opendocument:xmlns:of:1.2" xmlns:xhtml="http://www.w3.org/1999/xhtml"
                         xmlns:grddl="http://www.w3.org/2003/g/data-view#" xmlns:field="urn:openoffice:names:experimental:ooo-ms-intero
                         office:version="1.2" grddl:transformation="http://docs.oasis-open.org/office/1.2/xslt/odf2rdf.xsl"
  <office:scripts/>
  <office:font-face-decls>
    <style:font-face style:name="Times New Roman" svg:font-family="'Times New Roman'" style:font-family-generic="roman" style:font-pitcl
    <style:font-face style:name="Arial" svg:font-family="Arial" style:font-family-generic="swiss" style:font-pitch="variable"/>
    <style:font-face style:name="Arial1" svg:font-family="Arial" style:font-family-generic="system" style:font-pitch="variable"/>
  </office:font-face-decls>
  <office:automatic-styles/>
  <office:body>
    <office:text>
      <text:sequence-decls>
        <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
        <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
        <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
        <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
      </text:sequence-decls>
      <text:p text:style-name="Standard">
        Testdatei
      </text:p>
    </office:text>
  </office:body>
</office:document-content>
```


XML-Format: LibreOffice (4/6)

Jetzt eine Datei mit Bild:




```
$ unzip -l  /tmp/testdatei.odt
Archive:  /tmp/testdatei.odt
  Length      Date    Time    Name
 --------     ----    ----    ----
       39  09-09-10 19:14   mimetype
        0  09-09-10 19:14   Configurations2/status
        0  09-09-10 19:14   Configurations2/accele
        0  09-09-10 19:14   Configurations2/floate
        0  09-09-10 19:14   Configurations2/popupm
        0  09-09-10 19:14   Configurations2/progre
        0  09-09-10 19:14   Configurations2/menuba
        0  09-09-10 19:14   Configurations2/toolba
        0  09-09-10 19:14   Configurations2/images/Bitmaps/
    37829  09-09-10 19:14   Pictures/10000000000002BB000001C9C86EB88C.png
     3599  09-09-10 19:14   content.xml
      532  09-09-10 19:14   manifest.rdf
    10591  09-09-10 19:14   styles.xml
     1097  09-09-10 19:14   meta.xml
     4718  09-09-10 19:14   Thumbnails/thumbnail.png
     8914  09-09-10 19:14   settings.xml
     2190  09-09-10 19:14   META-INF/manifest.xml
 --------                   -------
    69509                   17 files
```


XML-Format: LibreOffice (5/6)

Inhalt von *content.xml* (Auszug):

```
<office:body>
  <office:text>
    <text:sequence-decls>
      <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
      <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
      <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
      <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
    </text:sequence-decls>
    <text:p text:style-name="Standard">
      Testdatei
    </text:p>
    <text:p text:style-name="Standard"/>
    <text:p text:style-name="Standard">
      <draw:frame draw:style-name="fr1" draw:name="Grafik1"
          text:anchor-type="paragraph" svg:x="0.235cm" svg:y="0cm"
          svg:width="7.491cm" svg:height="4.898cm" draw:z-index="0">
        <draw:image xlink:href="Pictures/10000000000002BB000001C9C86EB88C.png"
            xlink:type="simple" xlink:show="embed" xlink:actuate="onLoad"/>
      </draw:frame>
    </text:p>
  </office:text>
</office:body>
```


XML-Format: LibreOffice (6/6)

Inhalt von *meta.xml* (Metadaten):

```
<office:document-meta office:version="1.2" grddl:transformation=
  "http://docs.oasis-open.org/office/1.2/xslt/odf2rdf.xsl">
  <office:meta>
    <meta:initial-creator>Hans-Georg Eßer</meta:initial-creator>
    <meta:creation-date>2010-09-09T21:01:27</meta:creation-date>
    <dc:date>2010-09-09T21:14:40</dc:date>
    <dc:creator>Hans-Georg Eßer</dc:creator>
    <meta:editing-duration>PT00H13M05S</meta:editing-duration>
    <meta:editing-cycles>2</meta:editing-cycles>
    <meta:generator>
      OpenOffice.org/3.2$Linux OpenOffice.org_project/320m12$Build-9483
    </meta:generator>
    <meta:document-statistic meta:table-count="0" meta:image-count="1"
      meta:object-count="0" meta:page-count="1" meta:paragraph-count="1"
      meta:word-count="1" meta:character-count="9"/>
  </office:meta>
</office:document-meta>
```

Grafikformate

- Zwei Welten bei Grafikformaten:
  - Rastergrafiken (jpg, png, tif, bmp, gif, xpm)
  - Vektorgrafiken (eps, emf, svg, ps)
  - Gigantische Liste: http://de.wikipedia.org/wiki/Grafikformat

Rastergrafiken (1/5)

- pixel-basiert
- Größe einer Grafik abhängig von
  - Breite x Höhe in Pixeln
  - Farbtiefe (8 Bit, 16 Bit pro RGB-Farbkanal etc.)
  - Kompressionsverfahren

Rastergrafiken (2/5)

- Rechenbeispiel
  - Bild: 800 x 600 Punkte
  - RGB, 8 Bit pro Farbkanal (also insgesamt 24 Bit, 24 bpp: bits per pixel)
  - 800 x 600 x 24 = 11.520.000 Bit = 1.440.000 Byte

```
$ identify test.bmp
test.bmp BMP 800x600 8-bit DirectClass 1.831mb

$ convert test.bmp test.rgb

$ ls -l test.*
-rw-r--r-- 1 esser users 1920122 10. Sep 10:37 test.bmp
-rw-r--r-- 1 esser users 1440000 10. Sep 10:41 test.rgb
```


Rastergrafiken (3/5)

- Primitiv-Format: XPM
  - **X** **P**ix**map**
  - vor allem für Icons
  - „lesbar"
  - nicht komprimiert
- Beispiel (48x20 Pixel):

```
/* XPM */
static char * linux_xpm[] = {
"48 20 7 1",
"       c None",
".      c #D0D0D0",
"+      c #C0C0C0",
"@      c #808080",
"#      c #D0D0C0",
"$      c #000000",
"%      c #C08000",
".....................................+@@+.......",
".........+@........................#+$$$$@......",
".+$$$+...$$.........................@$$$$$+#....",
"..#$@#...++........................#$$$$$$@.....",
"..#$@+...++..+++@+..+@..++.+++++++..@@$@@$@+....",
"..+$@...+$$+@$$@$$@+$$.@$@.@$$+$@..+$@%@$$$+....",
"..#$@++..$$.+$$+.$@.@$..$@+.@$@@++.+@%%%$$.+...",
"..+$@+...$$.+$$.+$@+@$++$@...$$++##+@+$%+$$+....",
"..#$@++$.$@++$$..$@+@$++$@+++$$$..#+$+#+#+$$+...",
".++$$++$.$$++$$++$@+@$$@$@++$++$@++$+##+++$$@...",
".+@@@@@@+@@+@@@+@@@++@++@@+@@++@@+@$.+.+#+@$$+..",
".+++++++++++.+.+++++++++++++++%+++.$++#+.+#+$$@..",
"..+++++++.++++%++%+%%%%%%%%%%%%%%%@$+.+.+.$$$+.",
"...++%+%%%%%%%%%%%%%%%%%%%%%%%+%+..$$.+.+.+.+$$$..",
".+%%%%%%%%%%%%%%%%%%%+%++++#+++##..%%+++++.++$$$#.",
".%%%%%%%%%%%%%%%+###+############+++%%$@.+.++%@$%+.",
".+%%%%%%%%+###.###################%%%%%$+.+++%%%%%+",
".%%%%%+######.#####.###########+%%%%%%++++$@%%%%+",
"##%%#########...##################+%%%%%$$$$$%%%%##",
"######..##..########################+@@@+##.+@%++#+"};
```

Rastergrafiken (4/5)

- Anderes Beispiel: FOM-Logo (Ausschnitt)

```
/* XPM */
static char *fom_s[] = {
/* columns rows colors chars-per-pixel */
"55 60 256 2",
"   c black",
".  c #010101",
"X  c #020202",
[...]
"PX c #FDFDFD",
"IX c #FEFEFE",
"UX c gray100",
/* pixels */
```


Rastergrafiken (5/5)

- „normale“ Rasterformate sind binär und **komprimieren** die Daten
- „Pixelgrafiken“ sind Rastergrafiken
- zwei Arten von Kompression
  - verlustfrei (z. B. bei png, tif)
    → Originalbild wiederherstellbar
  - verlustbehaftet (z. B. bei jpg)
    → Teile der Bildinformation gehen verloren

JPEG-Format / **.jpg* (1/2)

- **JPEG**-Format (entwickelt von der **J**oint **P**hotographic **E**xperts **G**roup)
- verlustbehaftete Kompression
- Qualitätsstufen

fom.png (Original)	fom-85.jpg (85% Qual.)	fom-70.jpg (70% Qual.)	fom-50.jpg (50% Qual.)	fom-25.jpg (25% Qual.)	fom-05.jpg (5% Qual.)	fom-00.jpg (0% Qual.)
4575	1988	1540	1302	985	553	461

JPEG-Format / **.jpg* (2/2)

- am besten für Fotos geeignet

Original	JPEG, 85 %	JPEG, 30 %	JPEG, 5 %
392.190 Bytes	107.748 Bytes	36.383 Bytes	8.632 Bytes

PNG-Format / *.*png*

- **PNG: P**ortable **N**etwork **G**raphics
- als Ersatz für das *gif*-Format vor allem für Grafiken auf Webseiten eingesetzt
- patentfrei
- verlustfreie Komprimierung
  – viel besser als bei *gif*-Dateien

TIFF-Format / *.*tif*

- **TIFF: T**agged **I**mage **F**ile **F**ormat
- verlustfreie Kompression
  (außer in einem Spezialformat → JPEG)
- Standardformat im DTP-Bereich: unterstützt
  - RGB-Format (**R**ed, **G**reen, **B**lue)
  - CMYK-Format
    (**C**yan, **M**agenta, **Y**ellow, **K**ey / ~~Black~~)

Welche Kompression?

- Entscheidung über Bildformat (und damit: über Kompression) hängt von Bildtyp ab
  - Fotos:
    große Vielfalt an Farbtönen, weiche Übergänge
    → JPEG
  - Diagramme, Grafiken, ins Rasterformat gewandelte Vektorgrafiken:
    geringe Farbzahl, harte Kanten
    → PNG, TIFF etc.
- Falsches Format zu wählen, bedeutet:
  - zu hoher Platzbedarf oder
  - zu geringe Qualität

Vektorformate

- Idee: Bilder nicht in Pixel zerlegen, sondern „beschreiben“, z. B.
  Bild mit Abmessungen 10 cm x 10 cm, weißer Hintergrund, roter Kreis mit Durchmesser 1 cm und Mittelpunkt bei Koordinaten (3 cm, 4 cm). Rand des Kreises blau, Dicke 1 mm
- allgemein: grafische „Primitive“ (Linie, Kreis, Rechteck etc., Pfad, Polygon, Text)
- Beispielformate, SVG, PS (PostScript), EPS
- Vorteil: Beliebige Skalierbarkeit

Vektorformate

Quelle:
http://de.wikipedia.org/wiki/Vektorgrafik

SVG-Format / *.svg

SVG: Scalable **V**ector **G**raphics

- Freies Format (patentfrei)
- ein weiteres XML-Format:

```
<circle
  cx="1275"
  cy="1511"
  r="708"
  style="stroke:#000000;
         stroke-width:8;"
/>
```

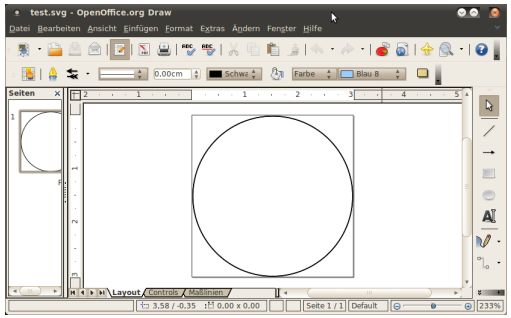
X- und Y-Koordinaten des Mittelpunkts

Kreisradius

Stilangaben: Farbe, Breite

SVG-Format / *.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" width="1.2in"
  height="1.2in" viewBox="558 794 1434 1434">
  <g style="stroke-width:.025in; fill:none">
  <!-- Circle -->
  <circle cx="1275" cy="1511" r="708"
    style="stroke:#000000;stroke-width:8;"/>
  </g>
</svg>
```



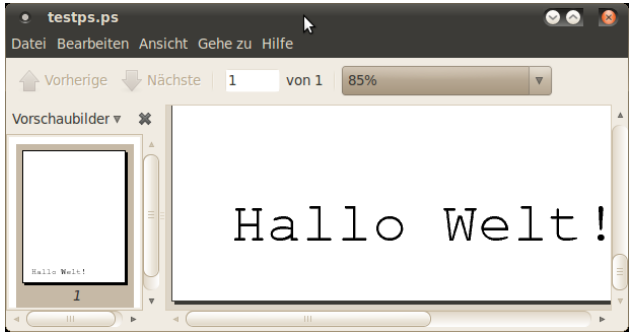
PostScript-Format / *.ps (1/2)

- altes Format: 1984 (Adobe)
- ursprünglich: Seitenbeschreibungssprache (für Laserdrucker und Druckmaschinen)
- wird heute durch PDF verdrängt
- Linux setzt (noch) in seinem Drucksystem PostScript ein
- Variante: **EPS** (**E**ncapsulated **P**ost**S**cript)
- kann auch Rastergrafik-Elemente enthalten
- PostScript ist eine Programmiersprache

PostScript-Format / *.ps (2/2)

```
%!
/Courier findfont      % Schrift auswählen
50 scalefont           % auf Schriftgröße 50 skalieren
setfont                % zum aktuellen Zeichensatz machen
50 50 moveto           % (50, 50) als aktuelle Schreibposition setzen
(Hallo Welt!) show     % und dort den Text ausgeben
showpage               % Seite ausgeben
```

(Quelle Listing: http://de.wikipedia.org/wiki/PostScript)



PDF-Format / *.pdf (1/2)

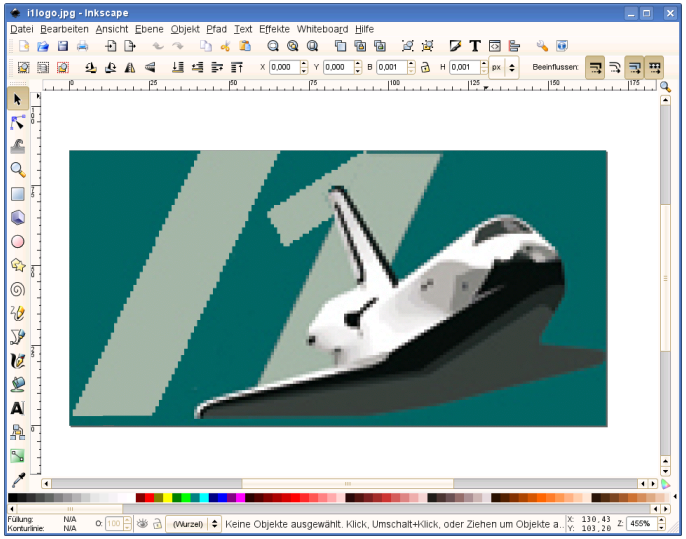
- PDF: Portable Document Format
- 1993 (Adobe), seit 2008 „offener Standard“
- Seitenbeschreibungssprache
- ähnliches Datenmodell wie PostScript, mit komprimierter Kodierung
- einbetten von Schriftarten
- Hyperlinks (intern & extern)
- Zugriffsschutz (Verschlüsselung/Passwort, Copy & Paste, Drucken)

PDF-Format / *.pdf (2/2)

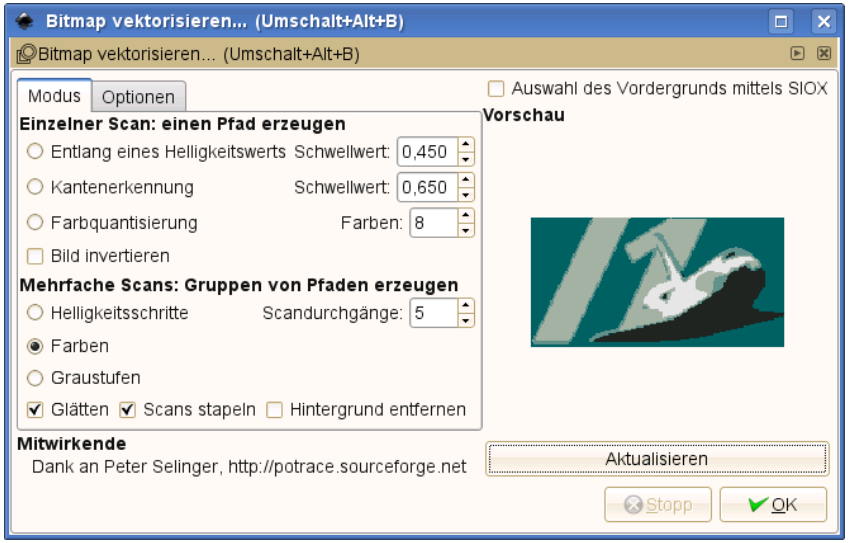
- interaktiv: Lesezeichen, Formulare, Kommentare (im Reader)
- unsichtbare Textebenen (für gescannte und mit OCR analysierte Dokumente)
- eingeschränkt: Nacharbeiten an PDF-Datei möglich
- einfache Konvertierung PDF ↔ PostScript
- PDF heute ein Standardformat für Dokumentenweitergabe und Druckdaten
- PDF/A: PDF Archive, für Langzeitarchivierung

Vektorisieren (1/3)

Inkscape (www.inkscape.org) kann Bitmap-Grafiken vektorisieren

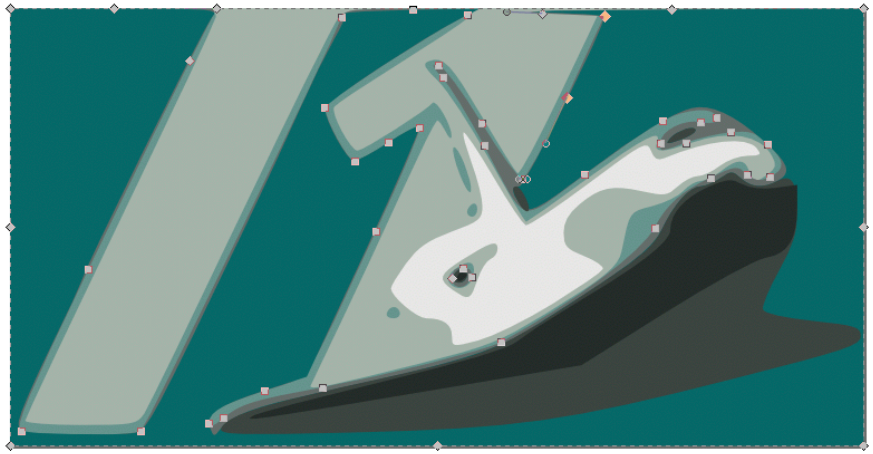


Vektorisieren (2/3)



Vektorisieren (3/3)

Ergebnis:



Adobe Acrobat: OCR / ClearType

Aktuelle Acrobat-Version hat interessantes Feature „ClearType“:

- Dokument einscannen und als PDF speichern
- OCR auf PDF-Datei anwenden
- neue Fonts (Schriftartdateien) aus erkannten Buchstaben erzeugen
- Pixel-Buchstaben in PDF-Datei durch Buchstaben in diesen Fonts ersetzen

Videoformate

- „Video“ = Audio + Video
- Bilder (Frames), pro Frame ein „volles Bild“
- simpelste Variante: MJPEG (Motion JPEG)
- separates Speichern von Ton und Bild – oder „gemischt“

MPEG-Formate

- **MPEG: M**oving **P**icture **E**xperts **G**roup
- MPEG-1: erstes Videoformat der MPEG
- verschiedene Frame-Typen:
  - I-Frame: intra coded picture. Standbild (wie MJPEG)
  - P-Frame: predictive coded picture. Bild hängt von vorherigen Bildern ab
  - B-Frame: bidirectional coded picture. Bild hängt von vorherigen und folgenden (!) Bildern ab
  - D-Frame
- Audio: Audio Layer 1–3 (MP1–MP3)

Codec-Vielfalt

- Codec = **Co**der/**Dec**oder
- viele Container-Formate (z. B. AVI), die Video- und Audio-Streams in diversen Kodierungen aufnehmen
- abspielen eines Videos:
  - Analyse des Container-Formats
  - Lesen der Metadaten (welche Codecs?)
  - suchen der Decoder für Audio und Video
  - Streams dekodieren (Wiedergabe)

Konverter (1/3)

- Ziel: Daten aus einem Format in ein anderes umwandeln
- Beispiele
  - UTF-8-Text → ISO-Latin9-Text
  - PNG-Bild → JPEG-Bild
  - MP3-Audiodatei → Ogg-Vorbis-Datei
  - Word-Dokument → LibreOffice-Dokument
  - PostScript → PDF
  - PNG-Bild → UTF-8-Text (sinnvoll?)
  - MP3-Audiodatei → UTF-8-Text (sinnvoll?)

Konverter (2/3)

Probleme

- Quell- oder Zielformat evtl. nicht vollständig bekannt (→ proprietäre Dateiformate)
- Quell- oder Zielformat evtl. nicht gleich „mächtig“
- Verlust durch Kompression
- Verlust von Struktur (z. B. DocBook → PDF)

Konverter (3/3)

Import- und Export-Filter

- Öffnen- und Speichern-Funktion von Anwendungen mit Import-/Export-Feature
- Qualität hängt von Güte der Filter ab

Kommunikation

- Verschiedene Modelle
  - Nachricht: Sender → Empfänger
  - Dialog
  - Gruppenkommunikation
- Klartext vs. „binäre“ Nachrichten

Begriff „Kommunikationsformat“

- Format/Standard für Kommunikation
  - Netzwerkprotokolle
- Dateiformat für den Austausch von Informationen
  - PDF für Dokumente
  - spezialisierte Formate, etwa DTaus im Bankenbereich

Netzwerkprotokolle (1/3)

Eine kleine Auswahl

- HTTP (Webserver)
- FTP (FTP-Fileserver)
- SMTP (Mail-Versand)
- POP3, IMAP (Mail-Empfang)
- IRC (Chatten)
- SSL (Verschlüsselung)
- Time Protocol (Zeitabfrage)

Netzwerkprotokolle (2/3)

Grundlagen / Allgemeines

- meist Kommunikation zwischen zwei Seiten
- typisch: Client-Server-Prinzip
- verbindungslos
  - A schickt Anfrage an B
  - B schickt Antwort an A
  - Ende
- verbindungsorientiert
  - A und B bauen Verbindung auf
  - A und B kommunizieren über die Verbindung, bis sie geschlossen wird

Netzwerkprotokolle (3/3)

- Kommunikation folgt einem festen, vorgegebenen Protokoll
- Client und Server müssen sich exakt an das Protokoll halten, damit Kommunikation gelingt
- wie im „richtigen Leben“
  - A: „Wie möchten Sie bezahlen?“ –
    B: „mit Kreditkarte“ – *erfolgreich*
  - A: „Wie möchten Sie bezahlen?“ –
    B: „Heute ist es sehr warm“ – *nicht erfolgreich*

HTTP (1/8)

- HTTP: **H**ypertext **T**ransfer **P**rotocol
- Client: Webbrowser (Firefox, IE, Opera etc.)
- Server: Webserver (Apache, Microsoft IIS etc.)
- verbindungsorientierte Kommunikation (TCP)
- Anfrage: HTTP-Request
- Antwort: HTTP-Response

HTTP (2/8)

HTTP-Requests

- GET (Daten: Header + Dokument holen)
- POST (Daten an den Server senden, Formulare)
- HEAD (nur Header holen)

HTTP (3/8)

HTTP GET

- Anfrage:

```
GET /ii-ws2010/fom-itis-datenformate-01.pdf HTTP/1.1
Host: fom.hgesser.de
```

- Antwort:

```
HTTP/1.1 200 OK
Date: Fri, 10 Sep 2010 10:33:23 GMT
Server: Apache/2.2.3 (Debian) PHP/4.4.4-8+etch6
Last-Modified: Fri, 03 Sep 2010 13:40:06 GMT
ETag: "3c630f7e-aec13-118b9980"
Accept-Ranges: bytes
Content-Length: 715795
Content-Type: application/pdf

%PDF-1.4
[...]
```


HTTP (4/8)

- Angabe des Hosts

  `Host: fom.hgesser.de`

  war früher (HTTP 1.0) nicht nötig und nicht möglich
- Moderne Webserver hosten mehrere Domains unter derselben IP-Adresse (z. B. fom.hgesser.de, hm.hgesser.de, hgesser.com) – ohne Angabe des Hostnamens keine Unterscheidung möglich

HTTP (5/8)

HTTP HEAD: holt nur den Header

- Anfrage:

```
HEAD /ii-ws2010/fom-itis-datenformate-01.pdf HTTP/1.1
Host: fom.hgesser.de
```

- Antwort:

```
HTTP/1.1 200 OK
Date: Fri, 10 Sep 2010 10:37:00 GMT
Server: Apache/2.2.3 (Debian) PHP/4.4.4-8+etch6
Last-Modified: Fri, 03 Sep 2010 13:40:06 GMT
ETag: "3c630f7e-aec13-118b9980"
Accept-Ranges: bytes
Content-Length: 715795
Content-Type: application/pdf
```


HTTP (6/8)

HTTP POST: überträgt Zusatzdaten

- Anfrage:

```
POST /search.html HTTP/1.1
Host: fom.hgesser.de
Content-Type: application/x-www-form-urlencoded
Content-Length: 20

search=Infrastruktur
```

- Antwort:

```
HTTP/1.1 200 OK
Date: Fri, 10 Sep 2010 10:37:00 GMT
Server: Apache/2.2.3 (Debian) PHP/4.4.4-8+etch6
Last-Modified: Fri, 03 Sep 2010 13:40:06 GMT
[...]
```

HTTP (7/8)

- Verbindung bleibt nach GET/HEAD/POST geöffnet
  - Client kann sofort weitere Anfragen schicken (z. B. für Bilder auf der HTML-Seite)
  - reduziert den Aufwand für Verbindungsauf- und -abbau (TCP-Verbindungen)
- Bei Fehler (z. B. falsche Syntax in Anfrage) wird Verbindung geschlossen. Rückmeldung:

```
HTTP/1.1 400 Bad Request
```

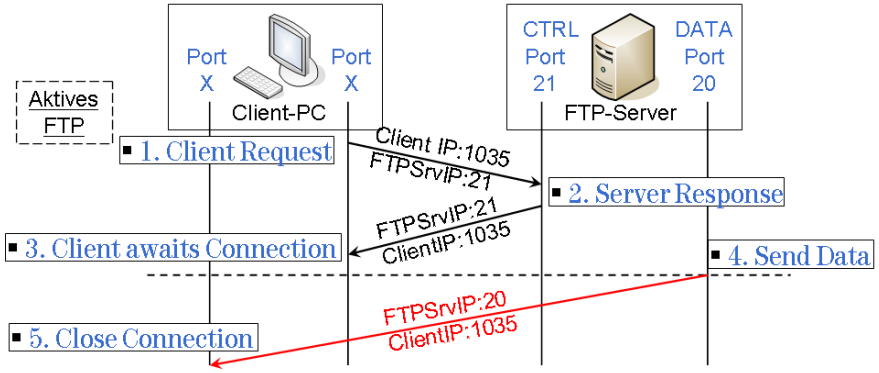

HTTP (8/8)

- HTTP Status Codes
  - immer in der ersten Zeile der Antwort
  - 200 OK
  - 301 Moved Permanently
  - 304 Not Modified
  - 400 Bad Request
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
  - 500 Internal Server Error
  - 503 Service Unavailable
  - weitere: http://de.wikipedia.org/wiki/HTTP-Statuscode
- Antwort steht nur im Header!

FTP (1/3)

- **FTP: F**ile **T**ransfer **P**rotocol
- Client: FTP-Client (z. B. Webbrowser oder spezielles FTP-Programm)
- Server: FTP-Server
- verbindungsorientierte Kommunikation (TCP) mit zwei Kanälen
  - Kontrollkanal (Port 21)
  - und Datenkanal (Port 20)
- aktives FTP (Direktverbindung) vs. passives FTP (Firewall)

FTP (2/3)



Quelle Bild: http://de.wikipedia.org/wiki/File_Transfer_Protocol

FTP (3/3)

- Authentifizierung (Username, Passwort)
- alternativ: „anonymous FTP“
- verschiedene Übertragungsvarianten
  - ASCII
  - Binary

Mail-Versand mit SMTP (1/2)

- **SMTP: S**imple **M**ail **T**ransfer **P**rotocol
- Kommunikation im Textmodus, SMTP-Port 25
- Kommandos:
  - HELO („Begrüßung“)
  - MAIL FROM: *Adresse*
  - RCPT TO: *Adresse*
  - DATA
  - dann die Nachricht (erst die Mail-Header)

Mail-Versand mit SMTP (2/2)

```
esser@netbook:~$ telnet hgesser.de smtp
Trying 217.160.135.96...
Connected to hgesser.de.
Escape character is '^]'.
220 hgesser.de ESMTP Exim 3.36 #1 Sat, 11 Sep 2010 00:28:31 +0200
helo absender.de
250 hgesser.de Hello mnch-123.pool.mediaways.net [93.133.124.239]
mail from: h.g.esser@gmx.de
250 <h.g.esser@gmx.de> is syntactically correct
rcpt to: esser@hgesser.de
250 <esser@hgesser.de> verified
data
354 Enter message, ending with "." on a line by itself
Hallo, das ist eine Nachricht.
.
250 OK id=1OuC5Z-0005cW-00
quit
221 s15337257.onlinehome-server.info closing connection
Connection closed by foreign host.
```


Daytime Protocol

- veraltetes (aber angenehm einfaches) Protokoll, mit dem man nach der Uhrzeit fragen kann
- Kontakt zu Port 13 (daytime) eines Servers, der einen Daytime-Service anbietet
- Server liefert Zeit als lesbaren Text

```
esser@netbook:~$ netcat hgesser.de daytime
Sat Sep 11 00:23:55 2010
```

- nicht mit NTP verwechseln ...

Austauschformate

Und wieder: Eine kleine Auswahl

- RSS (News-Feeds)
- DTaus (Banküberweisungen)
- PDF/PostScript
- CSV (Tabellendaten)
- EPS (Vektorgrafik-Austauschformat)
- RTF (Textdatei-Austauschformat)

RSS (1/3)

- **RSS: R**eally **S**imple **S**yndication
- RSS-Feed: XML-Format, enthält Titel, Kurzfassungen und Links zu Nachrichten
- typisch bei Blogs, News-Seiten u. ä.
- Idee:
  - News-Feed abonnieren
  - regelmäßig Infos zu neuen Beiträgen erhalten
  - Bei Klick auf Link den vollen Artikel lesen

RSS (2/3)

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
  <channel>
    <title>IT-Infrastruktur News</title>
    <link>http://fom.hgesser.de/</link>
    <description>Infos zu IT-Infrastruktur / FOM</description>
    <language>de-de</language>
    <copyright>Hans-Georg Eßer</copyright>
    <pubDate>Sat, 11 Sep 2010 2:43:19</pubDate>

    <item>
      <title>Folien zu RSS fertig</title>
      <description>Kurze Zusammenfassung: RSS ...</description>
      <link>http://fom.hgesser.de/rss-folien/</link>
      <author>Hans-Georg Eßer</author>
      <guid>http://fom.hgesser.de/id00012</guid>
      <pubDate>Sat, 11 Sep 2010 2:43:19</pubDate>
    </item>

    <item> ...  </item>
  </channel>
</rss>
```


RSS (3/3)

Banking: DTaus (1/7)

- Datenträgeraustauschverfahren
  - ursprünglich: mit Disketten („Datenträger-...“)
  - auch: online
  - Kunde ↔ Bank und Bank ↔ Bank
- Datei besteht aus
  - A-Satz: „Datenträgervorsatz“ (Header), 128 Byte
  - C-Sätze: Buchungen („Zahlungsaustausch“), 187 + 29·*n* Byte
  - E-Satz: „Datenträgernachsatz“ (Prüfsumme), 128 Byte

Banking: DTaus (2/7)

A-Satz

```
 1   0    4 Zeichen   Länge des Datensatzes, immer "0128"
 2   4    1 Zeichen   Datensatz-Typ, immer 'A'
 3   5    2 Zeichen   Art der Transaktionen
                      "LB" für Lastschriften Bankseitig
                      "LK" für Lastschriften Kundenseitig
                      "GB" für Gutschriften Bankseitig
                      "GK" für Gutschriften Kundenseitig
 4   7    8 Zeichen   Bankleitzahl des Auftraggebers
 5   15   8 Zeichen   CST, "00000000", nur belegt, wenn
                      Diskettenabsender Kreditinstitut
 6   23  27 Zeichen   Name des Auftraggebers
 7   50   6 Zeichen   aktuelles Datum im Format DDMMJJ
 8   56   4 Zeichen   CST, "   " (Blanks)
 9   60  10 Zeichen   Kontonummer des Auftraggebers
10   70  10 Zeichen   Optionale Referenznummer
11a  80  15 Zeichen   Reserviert, 15 Blanks
11b  95   8 Zeichen   Optionales Ausführungsdatum, DDMMJJJJ.
11c 103  24 Zeichen   Reserviert, 24 Blanks
12  127   1 Zeichen   Währungskennzeichen: " " = DM, "1" = Euro
     -- 128 Zeichen
```

Quelle: Dokumentation zum Programm *dtaus* (http://www.infodrom.org/projects/dtaus/)

Banking: DTaus (3/7)

C-Satz (1/3)

```
1    0   4 Zeichen Länge des Datensatzes, 187 + x * 29 (x..Anzahl
                   Erweiterungsteile)
2    4   1 Zeichen Datensatz-Typ, immer 'C'
3    5   8 Zeichen Bankleitzahl des Auftraggebers (optional)
4   13   8 Zeichen Bankleitzahl des Kunden
5   21  10 Zeichen Kontonummer des Kunden
6   31  13 Zeichen 1. Zeichen        "0"
                   2. - 12. Zeichen  interne Kundennummer oder Nullen
                   13. Zeichen       "0"
                   Die interne Nummer wird vom erstbeauftragten
                   Institut zum endbegünstigten Institut
                   weitergeleitet.  Die Weitergabe der internenen
                   Nummer an den Überweisungsempfänger ist der
                   Zahlstelle freigestellt.
```

Quelle dieser und der folgenden zwei Seiten:
Dokumentation zum Programm *dtaus* (http://www.infodrom.org/projects/dtaus/)

Banking: DTaus (4/7)

C-Satz (2/3)

```
7a  44   2 Zeichen Art der Transaktion
7b  46   3 Zeichen ------- "----------
                   "04000" Lastschrift des Abbuchungsauftragsverfahren
                   "05000" Lastschrift des Einzugsermächtigungsverfahren
                   "05005" Lastschrift aus Verfügung im elec. Cash-System
                   "05006" Wie 05005 mit ausländischen Karten
                   "05015" Lastschrift aus Verfügung im elec. Cash-
                           System - POZ
                   "51000" Überweisungs-Gutschrift
                   "53000" Überweisung Lohn/Gehalt/Rente
                   "54XXJ" VL mit Sparzulage
                           XX = 00 oder %-Satz der Sparzulage
                           J = Endziffer des Jahres für diese Leistung
                   "56000" Überweisung öffentlicher Kassen
8   49   1 Zeichen Reserviert, " " (Blank)
9   50  11 Zeichen Betrag
10  61   8 Zeichen Bankleitzahl des Auftraggebers
11  69  10 Zeichen Kontonummer des Auftraggebers
```

Banking: DTaus (5/7)

C-Satz (3/3)

```
12   79  11 Zeichen Betrag in Euro einschließlich
                    Nachkommastellen, nur belegt, wenn Euro als
                    Währung angegeben wurde (A12, C17a), sonst Nullen
13   90   3 Zeichen Reserviert, 3 Blanks
14a  93  27 Zeichen Name des Kunden
14b 120   8 Zeichen Reserviert, 8 Blanks
     -- 128 Zeichen
15  128  27 Zeichen Name des Auftraggebers
16  155  27 Zeichen Verwendungszweck
17a 182   1 Zeichen Währungskennzeichen: " " = DM, "1" = Euro
17b 183   2 Zeichen Reserviert, 2 Blanks
18  185   2 Zeichen Anzahl der Erweiterungsdatensätze, "00" bis "15"
19  187   2 Zeichen Typ (1. Erweiterungsdatensatz)
                      "01" Name des Kunden
                      "02" Verwendungszweck
                      "03" Name des Auftraggebers
20  189  27 Zeichen Beschreibung gemäß Typ
21  216   2 Zeichen wie C19, oder Blanks,  (2. Erweiterungsdatensatz)
22  218  27 Zeichen wie C20, oder Blanks
23  245  11 Zeichen 11 Blanks
     -- Ende des ersten Erweiterungsdatensatzes
     -- 256 Zeichen
```


Banking: DTaus (6/7)

- Daten erzeugen mit Tool *dtaus* (http://www.infodrom.org/projects/dtaus/)
- Umwandeln mit

  `dtaus -dtaus -c control.txt`
- Ergebnis in Datei *dtaus0.txt*

control.txt:

```
BEGIN {
  Art    GK
  Name   Martin Schulze
  Konto  123545
  BLZ    2004002
  Ausfuehrung 23.12.2001
  Euro
}

{
  Transaktion Gutschrift
  Name    Martha Schulze
  Konto   98832
  BLZ     2004003
  Betrag  20.00
  Zweck   Gebuehr Wohnheimnetz
  Text    Anschluss u. 11+12.97
}
```


Banking: DTaus (7/7)

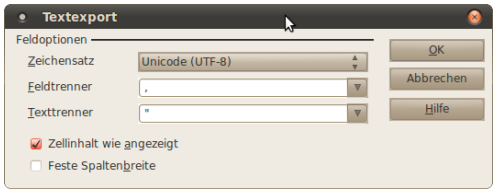
```
00000000  30 31 32 38 41 47 4b 30  32 30 30 34 30 30 32 30  |0128AGK020040020|
00000010  30 30 30 30 30 30 30 4d  41 52 54 49 4e 20 53 43  |0000000MARTIN SC|
00000020  48 55 4c 5a 45 20 20 20  20 20 20 20 20 20 20 20  |HULZE           |
00000030  20 20 31 30 30 39 31 30  20 20 20 20 30 30 30 30  |  100910    0000|
00000040  31 32 33 35 34 35 30 30  30 30 30 30 30 30 30 30  |1235450000000000|
00000050  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 32  |               2|
00000060  33 31 32 32 30 30 31 20  20 20 20 20 20 20 20 20  |3122001         |
00000070  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 31  |               1|
00000080  30 32 31 36 43 30 32 30  30 34 30 30 32 30 32 30  |0216C02004002020|
00000090  30 34 30 30 33 30 30 30  30 30 39 38 38 33 32 30  |0400300000988320|
000000a0  30 30 30 30 30 30 30 30  30 30 30 30 35 31 30 30  |0000000000005100|
000000b0  30 20 30 30 30 30 30 30  30 30 30 30 30 30 32 30  |0 00000000000020|
000000c0  30 34 30 30 32 30 30 30  30 31 32 33 35 34 35 30  |0400200001235450|
000000d0  30 30 30 30 30 30 32 30  30 30 20 20 20 4d 41 52  |0000002000   MAR|
000000e0  54 48 41 20 53 43 48 55  4c 5a 45 20 20 20 20 20  |THA SCHULZE     |
000000f0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |                |
00000100  4d 41 52 54 49 4e 20 53  43 48 55 4c 5a 45 20 20  |MARTIN SCHULZE  |
00000110  20 20 20 20 20 20 20 20  20 20 20 47 45 42 55 45  |           GEBUE|
00000120  48 52 20 57 4f 48 4e 48  45 49 4d 4e 45 54 5a 20  |HR WOHNHEIMNETZ |
00000130  20 20 20 20 20 20 31 20  20 30 31 30 32 41 4e 53  |      1  0102ANS|
00000140  43 48 4c 55 53 53 20 55  2e 20 31 31 2b 31 32 2e  |CHLUSS U. 11+12.|
00000150  39 37 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |97              |
00000160  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |                |
```


CSV (1/2)

- **CSV: C**omma-**S**eparated **V**alues
- einfache Textdatei, die Tabellen in folgendem Aufbau speichert:
  - jede Zeile der Datei entspricht einer Tabellenzeile
  - Felder (Spalten) sind durch ein Trennsymbol (oft: ein Komma, alternativ: Semikolon, Tabulator, Leerzeichen) getrennt
  - Felder zusätzlich in Anführungszeichen (um Kommata im Feld zu erlauben)

CSV (2/2)

	A	B
1	Eins	Vier
2	Zwei	Fünf
3	Drei	Sechs
4		



Ergebnis des Exports ins CSV-Format:

```
"Eins","Vier"
"Zwei","Fünf"
"Drei","Sechs"
```


EPS

- **EPS: E**ncapsulated **P**ost**S**cript
- im Prinzip eine normale (einseitige) PostScript-Datei, aber mit einer zusätzlichen „Bounding Box“
- EPS-Dateien kann man in andere Dokumente integrieren (wie Bilddateien)

```
%!PS-Adobe-3.0 EPSF-3.0
%%Title: rob-aktiv.eps
%%Creator: (ImageMagick)
%%CreationDate: Thu May 13 09:43:18 2004
%%BoundingBox: 0 0 168 86
%LanguageLevel: 1
%%Pages: 1
```

unten links: (0, 0)
oben rechts: (168, 86)
(Einheit: 1/72 Zoll)

Applikationsformate

- **CAD**, **C**omputer **A**ided **D**esign
- **GIS**, **G**eo**i**nformations**s**ystem
- **DTP**, **D**esk**t**op **P**ublishing

CAD (1/4)

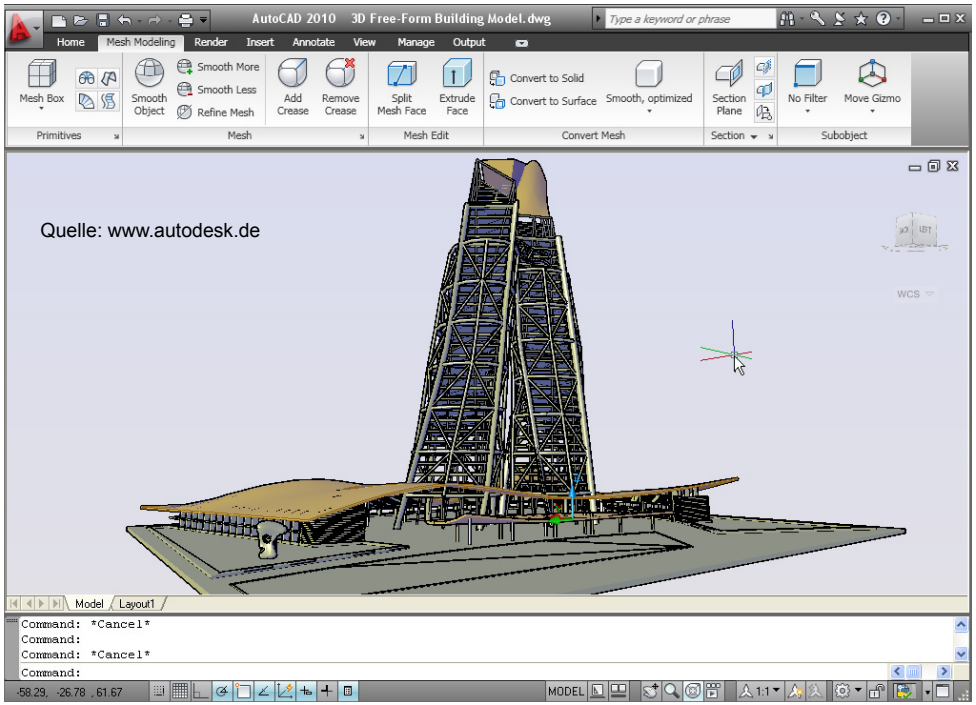
Computer Aided Design

- Mechanische Konstruktion
  - Bauwesen
    - Architektur (CAAD)
    - Holzbau
    - Ingenieurbau
    - Historische Rekonstruktion
    - Städtebau
    - Wasserbau
    - Verkehrswegebau
  - Vermessungswesen
  - Produktdesign
  - Maschinenbau
    - Anlagenbau
    - Fahrzeugbau
    - Formen- und Werkzeugbau
    - Antriebstechnik
    - Mechanische Simulation
  - Schaltpläne in der Elektrotechnik
  - Zahnmedizin
  - Schmuck- und Textilindustrie
- Elektronische Schaltungen

Quelle: http://de.wikipedia.org/wiki/CAD

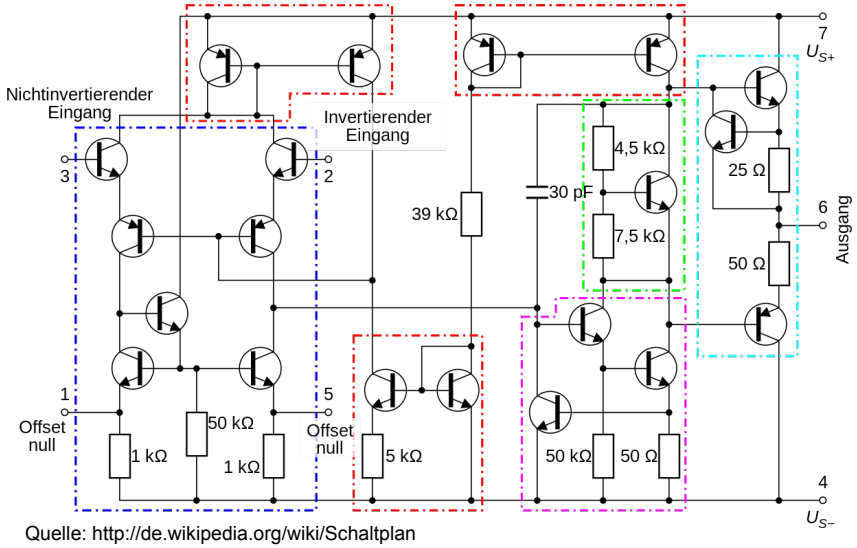
CAD (2/4)

- Dateiformate: vektorbasiert
- Verwaltung in objektorientierter Datenbank (nicht in einfachen Vektorgrafikdateien)
- Jedes Objekt besteht aus kleineren Teilobjekten
- Objekte haben Eigenschaften
  - Maschinenteile: Anschlüsse, Befestigungen etc.



Quelle: www.autodesk.de

CAD (3/4)



Quelle: http://de.wikipedia.org/wiki/Schaltplan

GIS (1/3)

- GIS: **Geo**informations**s**ystem
- Geometriedaten + Sachdaten
- gibt es vektor- und rastergrafikbasiert
- Beispiel: **GPX**, **G**PS E**x**change **F**ormat
  - XML-basiert, speichert
    - GPS-Informationen (Breiten- und Längengrad, Höhe)
    - Zusatzdaten (Zeitpunkt, Ortsbeschreibung etc.)

GIS (2/3)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx xmlns="http://www.topografix.com/GPX/1/1"
    creator="byHand" version="1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.topografix.com/GPX/1/1
    http://www.topografix.com/GPX/1/1/gpx.xsd">

  <wpt lat="39.921055008" lon="3.054223107">
    <ele>12.863281</ele>
    <time>2005-05-16T11:49:06Z</time>
    <name>Cala Sant Vicenç - Mallorca</name>
    <sym>City</sym>
  </wpt>
</gpx>
```

lat = latitude = Breite,
lon = longitude = Länge,
ele = elevation = Höhe
wpt = waypoint = Wegpunkt

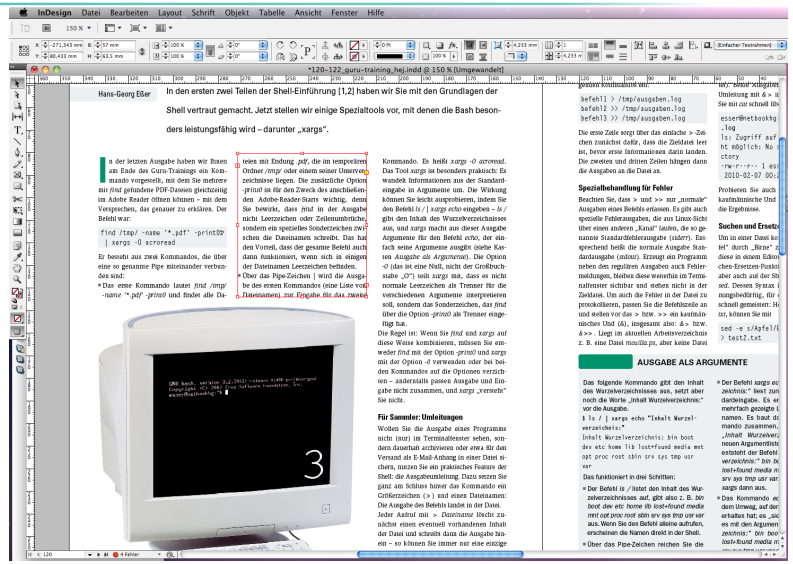
GIS (3/3)

Quelle: http://www.gpsvisualizer.com/examples/

Zeitschriftenlayout (1/2)

- Layout-/Satzprogramme (Desktop Publishing)
- ähnlich wie Textverarbeitung, aber rahmenbasiert
- Text fließt von Rahmen zu Rahmen (auch zur nächsten Seite)
- Beispiele: Adobe InDesign, QuarkXPress, Scribus
- Dateiformate proprietär (bei kommerziellen Anwendungen)

Zeitschriftenlayout (2/2)



Archive und Software-Pakete

Archive und Software-Pakete

- 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 / LibreOffice
- Solche Dateien kann man untersuchen, indem man sie in *.zip umbenennt (siehe Folien A-130 bis A-139 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`
  - **l**ist: `tar -tf Archivname.tar`
  - **e**xtract: `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
```