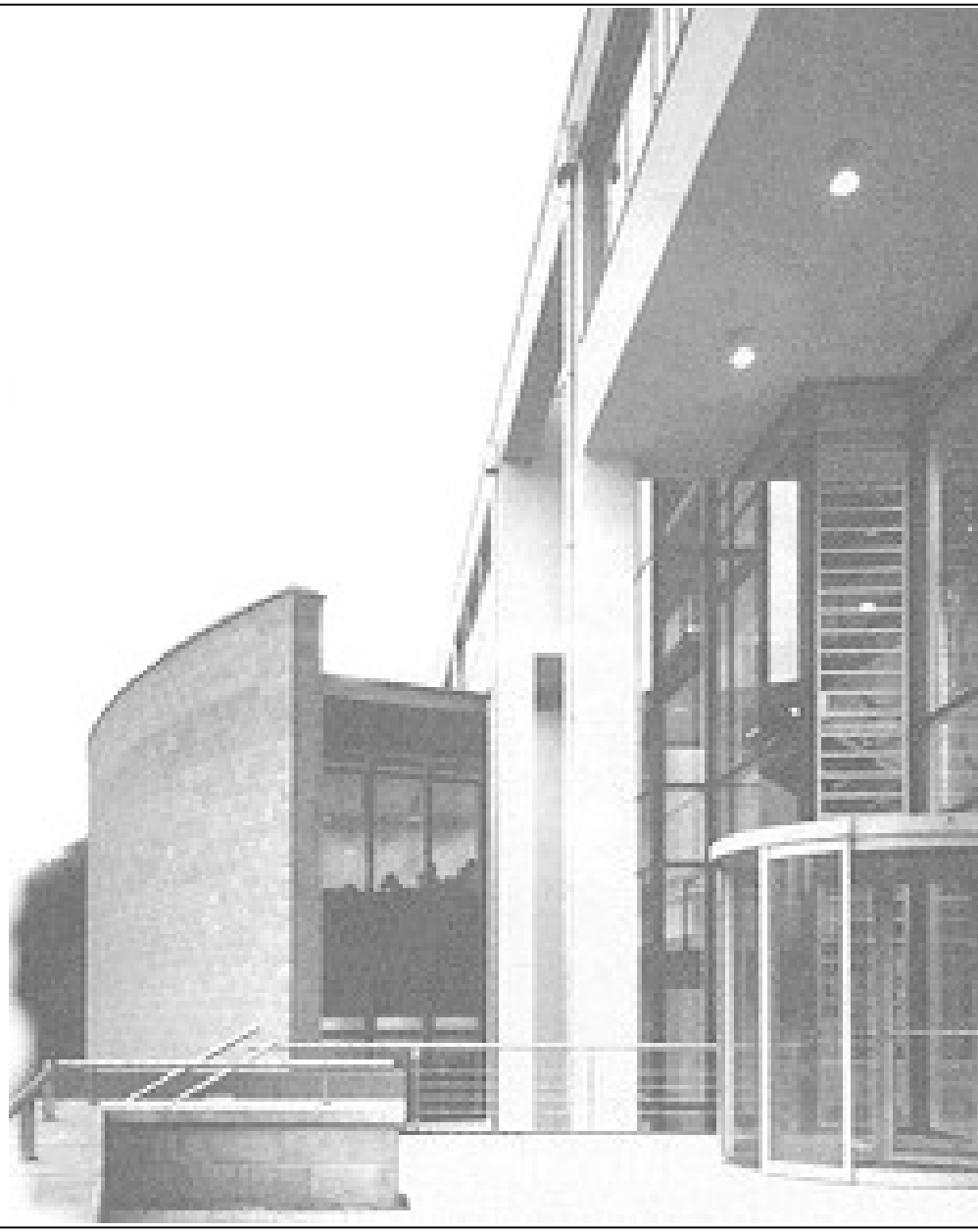
Studium neben dem Beruf

FOM Hochschule für Oekonomie & Management
University of Applied Sciences

IT-Infrastruktur

WS 2010/11

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



Foliensatz A (04.09.2010)

Über den Dozenten

Hans-Georg Eßer

- Dipl.-Math. (RWTH Aachen, 1997)
  Dipl.-Inform. (RWTH Aachen, 2005)
  Fachjournalist (DFJS Berlin, 2006)
  Doktorand (Univ. Erlangen-Nürnberg)
- Chefredakteur Computerzeitschrift (seit 2000)
- Autor diverser Computerbücher
- Lehrbeauftragter an der Hochschule München, der **FOM** und der Hochschule Wismar
  - Grundlagen der Informatik
  - Betriebssysteme
  - Rechnerarchitektur
  - **IT-Infrastruktur**

Organisatorisches

- **Termine:** stehen im Campus-System
- **Materialien**
  - im Campus-System
  - evtl. zusätzliches Material auf meiner Webseite **`http://fom.hgesser.de/`**
- **Literatur:** noch keine Empfehlung...
- **Pausen?** Diskussion
- **Fragen / Unterbrechungen**

Inhalte

- **Datenformate und Wandlung (von DTaus bis XML)** (12h)
  - Grafik, Video
  - Text
  - Applikationsformate (z.B. GIS, CAD usw.)
  - Kommunikationsformate
  - Konverter
  - Kodierungen (ASCII, ISO-Latin-*, Unicode)

Inhalte

- **Telekommunikation** (28h)
  - Geräte
  - Protokolle
  - Dienste
- **Ergonomie und Arbeitsschutz** (8h)
- **Gastvortrag** (4h)

Weitere Inhalte: andere Dozenten

Datenformate und Wandlung (von DTaus bis XML)

Heutiges Programm

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

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 der Straßenbahn in Fahrtrichtung rechts ab die Lothstraße runter, dann nach ca. 200 m auf der rechten Seite ins Hauptgebäude der Hochschule München. Dort Treppenhaus C einen Stock runter, 2x rechts und dann im 1. oder 2. Raum – fertig

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          Übertrag
------------
=  011110111
```

- Subtrahieren:

```
   011010111
-  010101110
     1 1         Übertrag
------------
=  000101001
```

Dualzahlen (3/3)

- Multiplizieren

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

Übertrag

Probe:

$101001_b = 41$

$1001_b = 9$

$41 \times 9 = 369$

$101110001_b = 369$

Umrechn. 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:
- $10011_b = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
  $= 1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$
  $= 19$

Umrechn. dual ↔ dezimal (2/3)

umgekehrt: dezimal → dual etwas komplizierter

49 als Dualzahl?

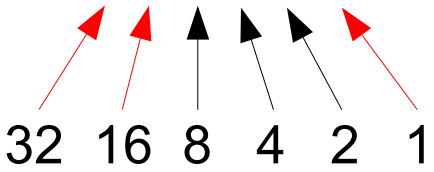
49 : 2 = 24,	Rest	1		Einser
24 : 2 = 12,	Rest	0		Zweier
12 : 2 = 6,	Rest	0		Vierer
6 : 2 = 3,	Rest	0		Achter
3 : 2 = 1,	Rest	1		
1 : 2 = 0,	Rest	1	->	110001_b (32+16+1)

Warum geht das? Rechnen Sie mal direkt

110001_b : 2... $110001_b = 2 \times 11000_b + 1$

Umrechn. dual ↔ dezimal (3/3)

- 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 (16¹)
3	: 16	=	0,	Rest	3	256er (16²)

→ $3AE_h$ (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°)
117	: 8	=	14,	Rest	5	8er (8¹)
14	: 8	=	1,	Rest	6	64er (8²)
1	: 8	=	0,	Rest	1	512er (8³)

→ 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 – $3FFFFFFF_h$

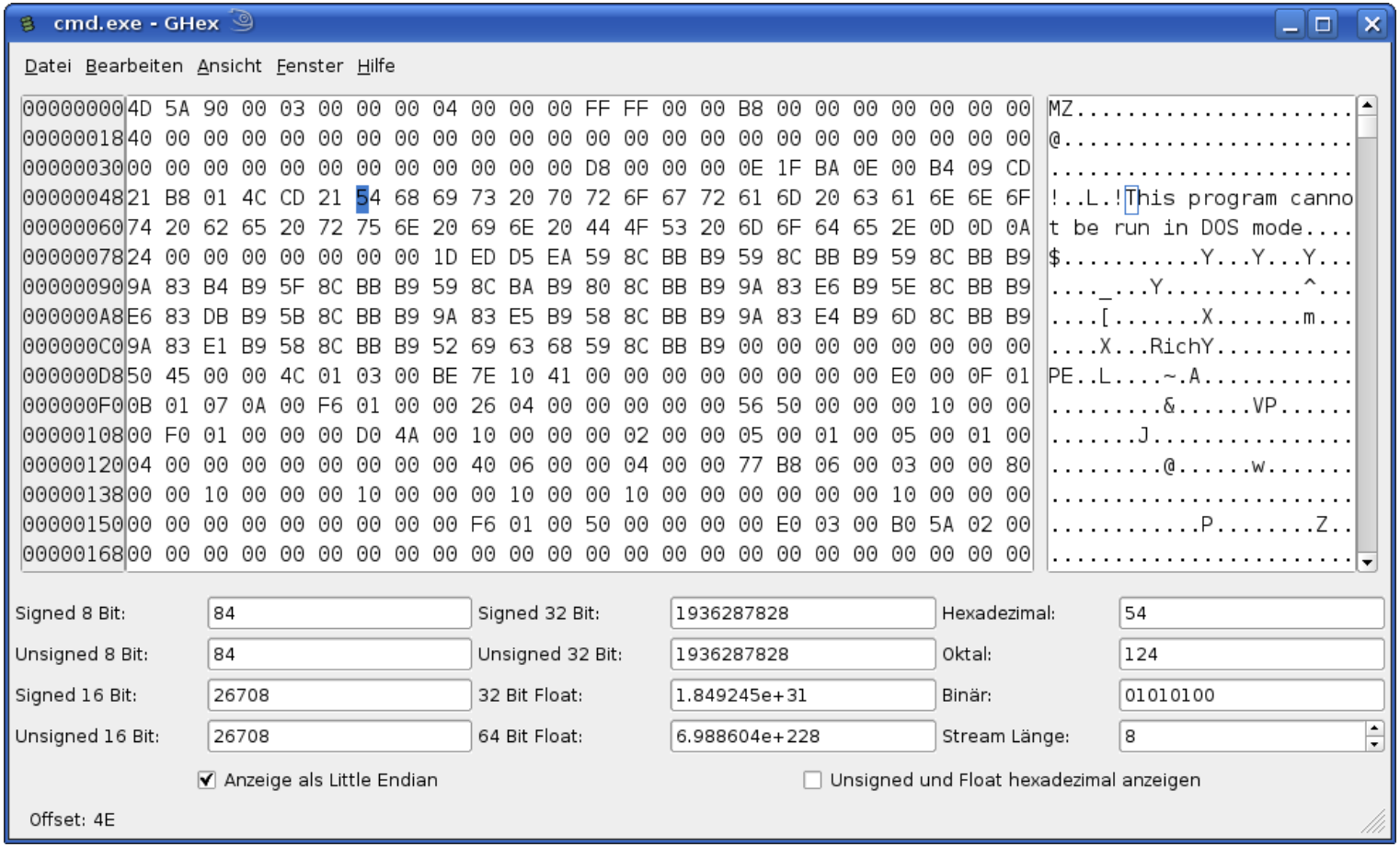
Warum hexadezimal? (2/3)

- 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
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 **$A01F_h$** in Dual- und Oktaldarstellung um.
4. Wenn Sie wissen, dass **$FF_h = 100_h - 1$** gilt, wie können Sie dann schnell **FFF_h** und **$FFFF_h$** 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“

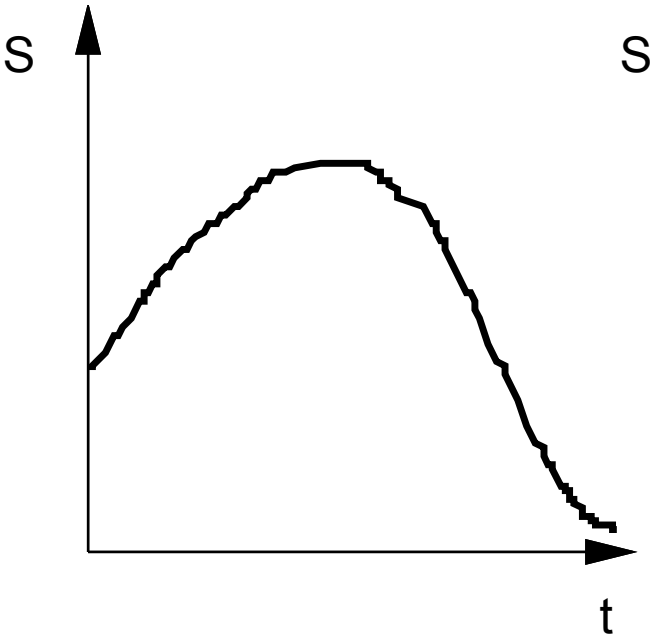
Informationen speichern

- 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

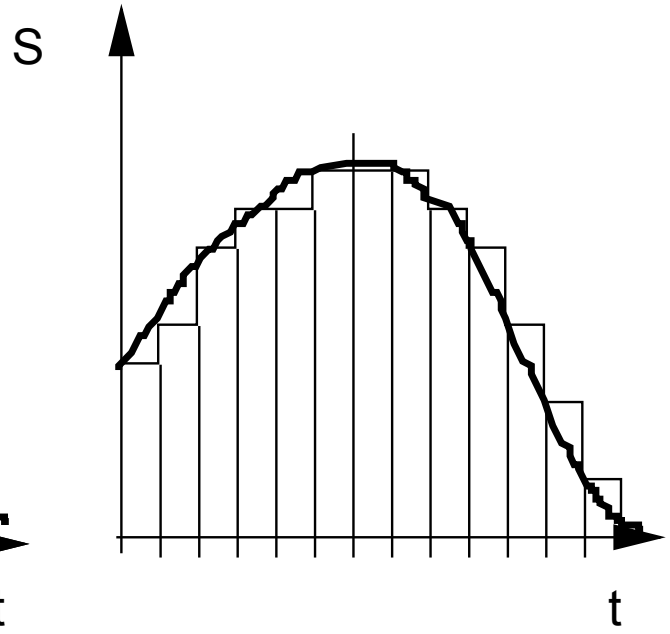
Digitalisierung analoger Daten

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

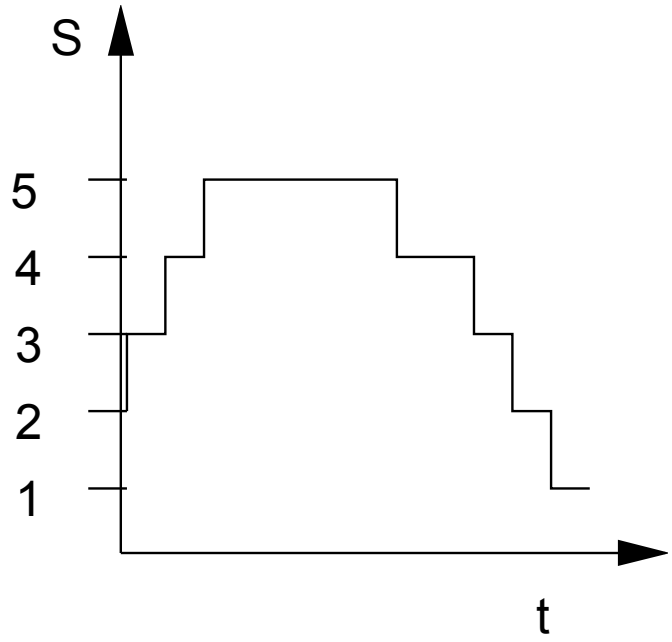
Zeitliche Digitalisierung



analoges Signal

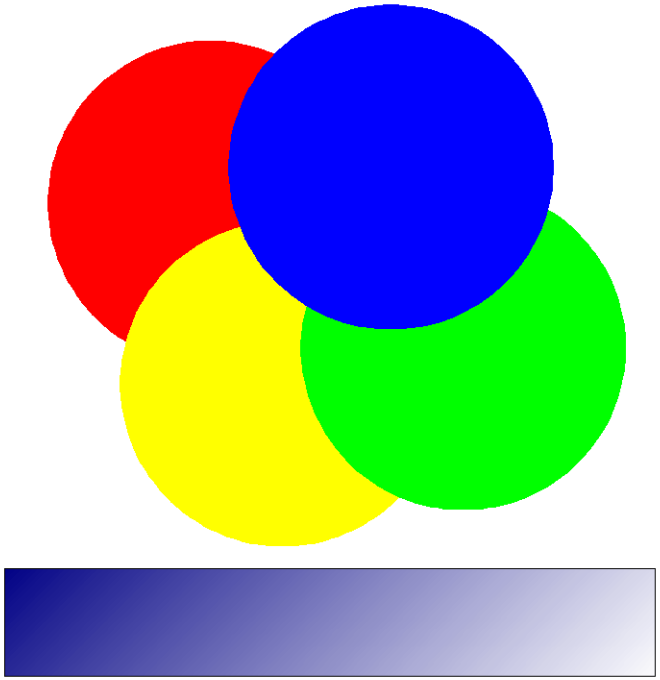


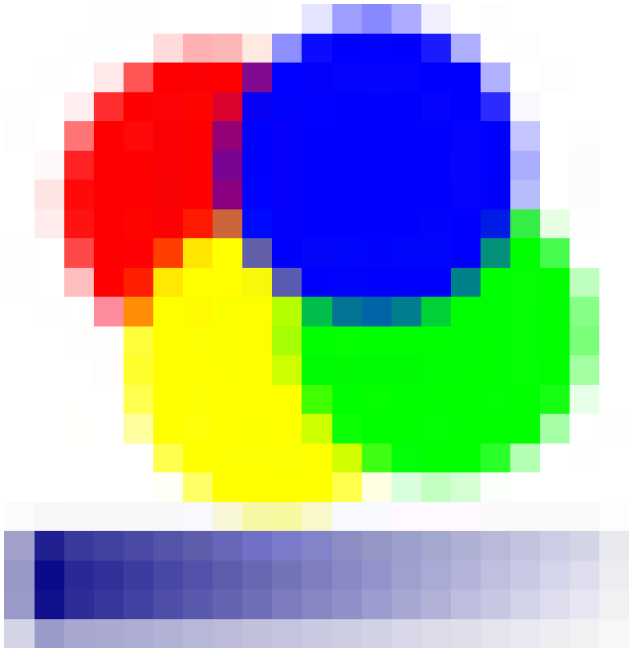
„gesampelt“: Proben zu festen Zeitpunkten gezogen – noch mit exakten Werten



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), OpenOffice (.odt)
- Tabellen: Excel (.xls), OpenOffice (.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...

ASCII-Fehlerkorrektur

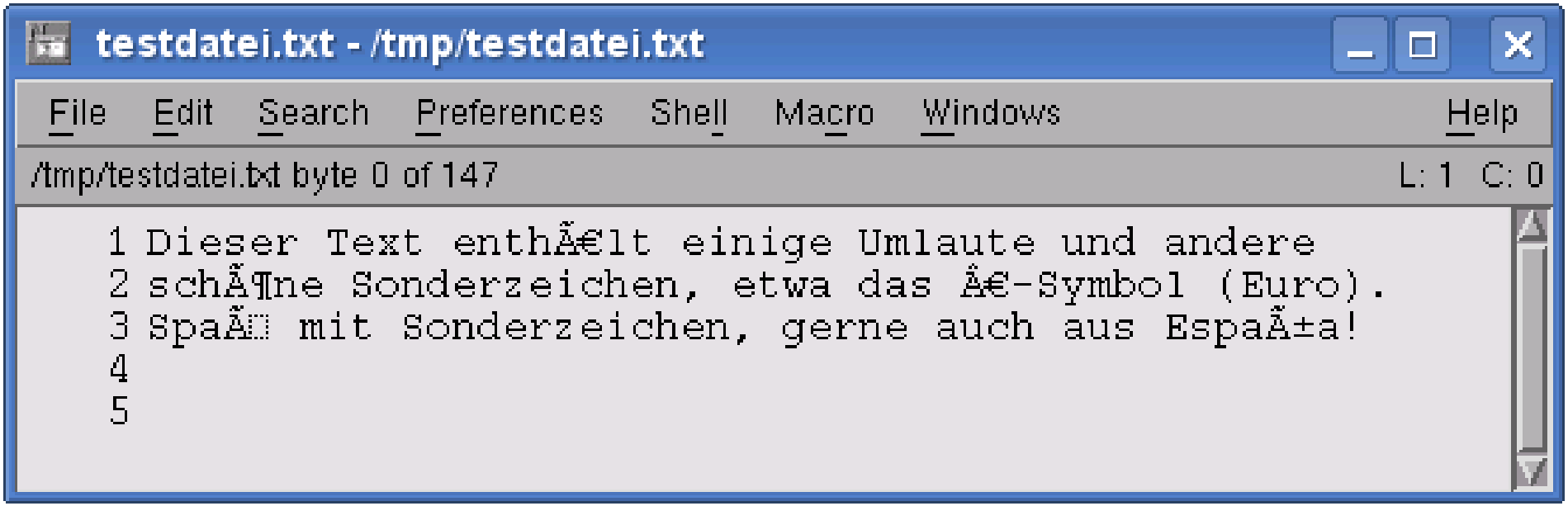
- Sie haben folgende Nachricht (binär) erhalten: 01001000, 11100001, 11101100, 11101110, 01101111
- Prüfen Sie zunächst, welche Zeichen ohne „Bit-Kipper“ übertragen wurden
  - Für alle korrekt empfangenen schlagen Sie in der ASCII-Tabelle den zugehörigen Buchstaben nach,
  - für alle fehlerhaft empfangenen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

Ausblick

Nächste Veranstaltung: Freitag, 10.09.

- Besondere Textformate
  - **.html* (HTML)
  - **.xml* (XML und SGML)
  - **.doc* (Word) -> „Textformat“?
  - **.odt* (Open Document) -> „Textformat“?
- Grafikformate