

# 10\_ITT\_22-23\_S34-45\_Themengebiet 2

Montag, 24. April 2023 22:06



## Themengebiet 2: Zahlensysteme

Ein Zahlensystem wird zur Darstellung von Zahlen verwendet. Eine Zahl wird dabei nach den Regeln des jeweiligen Zahlensystems als Folge von Ziffern beziehungsweise Zahlzeichen dargestellt.

Der Mensch lernt schon als kleines Kind das Zählen nach dem Dezimalsystem, bei dem mit zehn verschiedenen Zeichen gerechnet wird. Ein Computer kennt prinzipiell nur zwei verschiedene Zeichen: Strom an und Strom aus. Er rechnet also im Binärsystem. Möchte er andere Zahlensysteme darstellen, so rechnet er diese intern entsprechend um.

Jedes Zahlensystem besteht aus Nennwerten (Zeichen, die einen Wert repräsentieren). Die Anzahl der Nennwerte ergibt sich aus der Basis. Der größte Nennwert entspricht der Basis minus (-) 1. Wird der größte Nennwert überschritten, entsteht aus dem Übertrag der nächst höhere Stellenwert.

### 1. Zahlensysteme in der PC-Welt

#### 1.1 Dezimales Zahlensystem

Basis:	10
Nennwerte:	1 2 3 4 5 6 7 8 9



Abhängig von der Position des Nennwertes in der Zahl hat der Nennwert unterschiedliche Bedeutungen, was am Beispiel der Dezimalzahl 379.253 dargestellt werden soll:

Dezimalzahl:	3	7	9	2	5	3
Stellenwert: (Name)	Hunderttausender	Zehntausender	Tausender	Hunderter	Zehner	Einer
Stellenwert: (Wert)	100000	10000	1000	100	10	1
Stellenwert: (Exponential)	$10^5$	$10^4$	$10^3$	$10^2$	$10^1$	$10^0$

Daraus lässt sich dann der komplette Wert der Zahl berechnen:

3	*	100.000 ( $10^5$ )	=	300.000
7	*	10.000 ( $10^4$ )	=	70.000
9	*	1.000 ( $10^3$ )	=	9.000
2	*	100 ( $10^2$ )	=	200
5	*	10 ( $10^1$ )	=	50
3	*	1 ( $10^0$ )	=	3

379.253

## 1.2 Binäres Zahlensystem

Das Dualsystem (lat. dualis = zwei enthaltend), auch Zweiersystem oder Binärsystem genannt, ist ein Zahlensystem, das zur Darstellung von Zahlen nur zwei verschiedene Ziffern benutzt. Gewöhnlich werden analog zu anderen Zahlensystemen die Symbole 0 und 1 zur Darstellung der beiden Ziffern verwendet. In älterer Literatur mit Bezug zur elektronischen Datenverarbeitung werden manchmal die Symbole Low (L) und High (H) anstelle von 0 und 1 verwendet. In der Informatik werden für binär kodierte Werte auch die „Ziffern“ **wahr (w) und falsch (f)** bzw. die englischen Übersetzungen **true (t) und false (f)** verwendet, wobei falsch=0 und wahr=1 gesetzt wird.

Basis:	2
Nennwerte:	0 1



Abhängig von der Position des Nennwertes in der Zahl hat der Nennwert unterschiedliche Bedeutungen, was am Beispiel der binären Zahl 11011010 dargestellt werden soll:

Binärzahl:	1	1	0	1	1	0	1	0
Stellenwert: (Wert)	128	64	32	16	8	4	2	1
Stellenwert: (Exponential)	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

Daraus lässt sich dann der komplette dezimale Wert der Zahl berechnen:

1	*	128	( $2^7$ )	=	128
1	*	64	( $2^6$ )	=	64
0	*	32	( $2^5$ )	=	0
1	*	16	( $2^4$ )	=	16
1	*	8	( $2^3$ )	=	8
0	*	4	( $2^2$ )	=	0
1	*	2	( $2^1$ )	=	2
0	*	1	( $2^0$ )	=	0

11011010<sub>bin</sub> → 218

110<sub>bin</sub> → 6  
110 → 1101110

### 1.3 Hexadezimals Zahlensystem

Im Hexadezimalsystem werden Zahlen in einem Stellenwertsystem zur Basis 16 dargestellt. „Hexadezimal“ (von griech. hexa „sechs“ und lat. decem „zehn“) ist ein lateinisch-griechisches Mischwort. In der Datenverarbeitung wird das Hexadezimalsystem sehr oft verwendet, da es sich hierbei letztlich nur um eine komfortablere Verwaltung des Binärsystems handelt.

Basis:	16
Nennwerte:	0 1 2 3 4 5 6 7 8 9 a b c d e f



Abhängig von der Position des Nennwertes in der Zahl hat der Nennwert unterschiedliche Bedeutungen, was am Beispiel der hexadezimalen Zahl 9C4DA dargestellt werden soll:

Hexadezimalzahl::	9	C	4	D	A
Stellenwert: (Wert)	65.536	4.096	256	16	1
Stellenwert: (Exponential)	$16^4$	$16^3$	$16^2$	$16^1$	$16^0$

Daraus lässt sich dann der komplette dezimale Wert der Zahl berechnen:

9	*	65536 ( $16^4$ )	=	589.824
c	*	4096 ( $16^3$ )	=	49.152
4	*	256 ( $16^2$ )	=	1.024
d	*	16 ( $16^1$ )	=	208
a	*	1 ( $16^0$ )	=	10

9C4DA → 640.218

### 1.4 Oktales Zahlensystem

In den 1960er und 1970er Jahren wurde in der Informatik häufig auch das Oktalsystem mit seiner Basis als dritte Zweierpotenz ( $8 = 2^3$ ) verwendet, da es mit den üblichen Ziffern von 0 bis 7 auskommt. Es findet aber heute nur noch selten Anwendung.

Basis:	8
Nennwerte:	0 1 2 3 4 5 6 7



Beim Zählen im Oktalsystem ist zu beachten, dass nach 7 nicht die 8 folgt, sondern eine Stelle weiter links erhöht werden muss. Im Oktalsystem gilt:  $7 + 1 = 10$ .

## 1.5 Übersicht

	Dezimal	Hexadezimal	Binär
s	0	0	0
	1	1	1
	2	2	10
	3	3	11
	4	4	100
	5	5	110
	6	6	111
	7	7	1000
	8	8	1001
	9	9	1010
	10	A	1011
	11	B	1100
	12	C	1101
	13	D	1110
	14	E	1111
	15	F	1 0000
	16	10	1 0001
	17	11	1 0010
	18	12	1 0011
	19	13	1 0100
	20	14	1 0101
	30	1E	1 1110
	40	28	10 1000
	50	32	11 0010
	60	3C	11 1100
	70	46	100 0110
	80	50	101 0000
	90	5A	101 1010
	100	64	110 0100
	200	C8	1100 1000
	300	12C	1 0010 1100
	400	190	1 1001 0000
	500	1F4	1 1111 0100
	1000	3E8	11 1110 1000

## 1.6 Übungsaufgaben

1. Wandeln Sie die folgenden Zahlen von ... nach ... um:

a)

Dezimal	Hexadezimal
74	4A
45.673	B269
290.350	46E2E
255	FF

b)

Dezimal	Binär
17	10001
2.053	100000110011
6.265	1111010111001
127	1000001

c)

Binär	Dezimal
1101101010011010	55.962
11110000111100001111	986.895
010001111011	1.147
0000010011011100	1.244

d)

Binär	Hexadezimal
11110001	F1
1010010111011011	A5DB
11011111111100100110	DFF26
11111111	FF

e)

Hexadezimal	Dezimal
AF34DD	11.482.333
212142	2.163.010
EBBE	60.350
FACEB00C	4.207.849.484

f)

Hexadezimal	Binär
E932A	1110 1001 0011 0010 1010
BADE5EE	1011 1010 1101 1110 0101 1110 1110
110110101	0001 0001 0000 0001 0001 0000 0001 0000 0001
ABBA	1010 1011 1011 1010

2. Ergänzen Sie die folgende Tabelle, indem Sie die vorgegebenen Zahlen in die jeweils fehlenden Zahlensysteme umrechnen:

Aufgabe	Dezimal	Hexadezimal	Binär
a	54	6F	0110 1110
b	85.851	14F5B	1 0100 1111 0101 1011
c	890	37A	1101111010
d	5.351	4E7	0100 1110 0111
e	19	13	1101
f	86	56	1010110
g	43.837	AB3D	1010 1011 0011 1101
h	195	C 3	1100 0011
i	819	3 3 3	1100110011
j	3.785	1 2 C 9	0001 0010 1100 1001
k	286.371	45EA3	0100 0101 1110 1010 0011
l	128	8 0	10000000
m	63	3 F	0011 1111
n	3.107	C23	1100 0010 0011
o	256	1 0 0	1 0000 0000
p	47	2 F	101111

3. Verdoppeln bzw. halbieren Sie die folgenden Binärzahlen

Dezimal	Binär	Operation	Binär	Dezimal
1	0001	verdoppeln	0010	2
20	0001 0100	verdoppeln	0100 1000	40
63	0011 1111	verdoppeln	0111 1110	126
140	1000 1100	halbieren	0100 0110	70
256	0001 0000 0000	halbieren	1000 0000	128
1000	0011 1110 1000	halbieren	0001 1111 0100	500

4. Was versteht man unter einem sogenannten „Oktal-Zahlensystem“? Wandeln Sie die Dezimalzahlen 5, 8, 14, 16, 32 und 64 in eine Oktalzahl um.

---



---



---



---



---

5. Erklären Sie, warum für einen amerikanischen Informatiker kein Unterschied zwischen Weihnachten und Halloween besteht.

## 2. Zeichencodes

Wie bereits bekannt kann ein Computer intern nur mit 0 und 1 rechnen, da dies den beiden Zuständen *Strom aus* und *Strom an* entspricht. Um Zahlen anderer Systeme darstellen zu können werden diese intern umgerechnet. Noch komplizierter wird es, wenn anstelle von Zahlen Buchstaben dargestellt werden sollen.

Im Computer ist jeder Buchstabe eine Folge von Bit-Werten, je nach Zeichensatz wird eine unterschiedliche Anzahl an Bit benötigt, um einen Buchstaben bzw. ein Zeichen darstellen zu können.

inzwischen immer ASCII ==  
8Bit pro Zeichen!

Um diesen Bit-Folgen darstellbare Zeichen zuzuordnen, mussten Übersetzungstabellen, sogenannte Charsets, festgelegt werden. 1963 wurde eine erste 7-Bit-Version des ASCII-Codes durch die ASA (American Standards Association) definiert, um eine Vereinheitlichung der Zeichenkodierung zu erreichen. Obwohl IBM an der Definition mitgearbeitet hatte, führte man 1964 einen eigenen 8-Bit-Zeichencode EBCDIC ein. Beide finden bis heute in der Computertechnik Verwendung.

Da für die verschiedenen Sprachen andere diakritische Zeichen benötigt werden, gibt es für Sprachgruppen bestimmte Charsets. Die ISO hat mit der Normenreihe ISO 8859 Zeichenkodierungen für alle europäischen Sprachen und Arabisch, Hebräisch sowie Thai standardisiert. Das Unicode Consortium schließlich veröffentlichte 1991 eine erste Fassung des gleichnamigen Standards, der es sich zum Ziel gesetzt hat, alle Zeichen aller Sprachen in Codeform zu definieren. Unicode ist gleichzeitig die internationale Norm ISO 10646.

$2^7 == 128$  Zeichen  
heute:  
 $2^8 == 256$  Zeichen

### 2.1 ASCII

Der American Standard Code for Information Interchange (ASCII, alternativ US-ASCII) ist eine 7-Bit-Zeichenkodierung und dient als Grundlage für spätere, auf mehr Bits basierende Kodierungen für Zeichensätze. Die Zeichenkodierung definiert 128 Zeichen, bestehend aus 33 nicht druckbaren sowie 95 druckbaren. Letztere sind, beginnend mit dem Leerzeichen:

```
!"#$%&'()*+,-./0123456789:;<=>?  
@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_  
`abcdefghijklmnopqrstuvwxyz{|}~
```

Der Zeichenvorrat entspricht weitgehend dem einer Tastatur oder Schreibmaschine für die englische Sprache. Die nicht druckbaren Steuerzeichen enthalten Ausgabezeichen wie Zeilenvorschub oder Tabulator, Protokollzeichen wie Übertragungsende oder Bestätigung und Trennzeichen wie Datensatztrennzeichen.

Das für ASCII nicht benutzte Bit kann auch für Fehlerkorrekturzwecke (Paritätsbit) auf den Kommunikationsleitungen oder für andere Steuerungsaufgaben verwendet werden. Heute wird es aber fast immer zur Erweiterung von ASCII auf einen 8-Bit-Code verwendet. Diese Erweiterungen sind mit dem ursprünglichen ASCII weitgehend kompatibel, so dass alle im ASCII definierten Zeichen auch in den verschiedenen Erweiterungen durch die gleichen Bitmuster kodiert werden. Die einfachsten Erweiterungen sind Kodierungen mit sprachspezifischen Zeichen, die nicht im lateinischen Grundalphabet enthalten sind.



## 2.2 Erweiterungen des ASCII-Codes

Verschiedene Hersteller entwickelten eigene Acht-Bit-Codes. Der Codepage 437 genannte Code war lange Zeit der am weitesten verbreitete, er kam auf dem IBM-PC unter englischen MS-DOS, und kommt heute noch im DOS-Fenster von englischen Microsoft Windows zur Anwendung. In deren deutschen Installationen ist seit MS-DOS 3.3 die westeuropäische Codepage 850 der Standard. Windows verwendet aktuell den ANSI-Zeichensatz (256 Zeichen). Auch bei späteren Standards wie ISO 8859 wurden acht Bits verwendet. Dabei existieren mehrere Varianten, zum Beispiel ISO 8859-15 für die westeuropäischen Sprachen.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL 007F
80																
90																
A0	NBSP 00A0	í 00A1	í 00A2	£ 00A3	€ 00A4	¥ 00A5	Š 00A6	Š 00A7	Š 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	– 00AD	® 00AE	— 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	¼ 00B4	½ 00B5	¾ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

ASCII-Erweiterung ISO-8859-15

Wie lautet der Binärcode (der auf der Festplatte landet) einer Datei „ascii.txt“ mit Inhalt „ITS“?

0100 1001      0101 0100      0101 0011

I                      T                      S                      3 Byte (3\*8Bit)

### 2.3 Unicode, UTF-8

Um den Anforderungen der verschiedenen Sprachen gerecht zu werden, wurde der Unicode entwickelt. Er verwendet bis zu 32 Bit pro Zeichen und könnte somit über vier Milliarden verschiedene Zeichen unterscheiden, wird jedoch auf etwa eine Million erlaubte Codepoints eingeschränkt. Damit können alle bislang von Menschen verwendeten Schriftzeichen dargestellt werden, sofern sie in den Unicode-Standard aufgenommen wurden. UTF-8 ist eine 8-Bit-Kodierung von Unicode, die zu ASCII abwärtskompatibel ist. Ein Zeichen kann dabei ein bis vier 8-Bit-Blöcke einnehmen. UTF-8 entwickelt sich zum einheitlichen Standard unter den meisten Betriebssystemen. So nutzen unter anderem Apples Mac OS X sowie einige Linux-Distributionen standardmäßig UTF-8, und immer mehr Webseiten werden in UTF-8 erstellt.

UTF-8 ist in den ersten 128 Zeichen (Indizes 0–127) deckungsgleich mit ASCII und eignet sich mit i.d.R. nur einem Byte Speicherbedarf für Zeichen vieler westlicher Sprachen besonders für die Kodierung englischsprachiger Texte, die sich im Regelfall ohne Modifikation daher sogar mit nicht-UTF-8-fähigen Texteditoren ohne Beeinträchtigung bearbeiten lassen

65535

Beispiele für UTF-8 Kodierungen:


0 als erstes Bit → Standard-Zeichen, identisch zu ASCII

1 als erstes Bit → Sonderzeichen: Anzahl der 1er gibt an, wie viele Bytes das Zeichen hat

10 als erste beiden Bit → Byte "gehört zu einem anderen dazu"

Zeichen	UTF-8 binär	UTF-8 hexadez.
Buchstabe y	0111 1001	79
Buchstabe ä	1100 0011 1010 0100	C3 A4
Zeichen für eingetr. Marke	1100 0010 1010 1110	C2 AE
Eurozeichen €	1110 0010 1000 0010 1010 1100	E2 82 AC
Violinschlüssel	1111 0000 1001 1101 1000 0100 1001 1110	F0 9D 84 9E

Die Dateigröße einer Textdatei ist abhängig von der verwendeten Kodierung:

 Datei.txt	Inhalt	Kodierung	Dateigröße
	<leer>	ASCII oder UTF-8	0 Byte
	Klara Oppenheimer	ASCII (ANSI)	17 Byte
	Klara Oppenheimer	UTF-8	17 Byte
	Würzburg	ASCII (ANSI)	8
	Würzburg	UTF-8	9

weil "ü" wird jetzt in 2 Byte codiert

Wie viele UTF-8-Zeichen können maximal existieren?

$$128 + 2048 + 65536 + 4194304$$

$$4.262.016$$

## 2.4 Übungsaufgaben

Ein DVD-Laufwerk liest von einer DVD eine ASCII-codierte Textdatei aus, indem nacheinander aus den Pits und Lands auf der DVD die Bitwerte 0 und 1 gelesen werden. Es ergibt sich eine Reihe von 72 Datenbits:

01000101 01101001 01101110 00100000 01010100 01100101 01110011 01110101 00001010

Die Datenbits sind so zu interpretieren, dass sie in Bytes zusammengefasst werden und die Bytewerte in ASCII codierte Buchstaben dargestellt werden.

Aufgaben:

a) Gib die Dateigröße in Bytes an!

72 / 8 = 9Byte

---

b) Welcher Text steht in der Datei?

Ein <Sp> Test<LF>

---

Im DVD-Laufwerk kann es vorkommen, dass z.B. durch Kratzer auf der DVD einzelne Bitwerte falsch ausgelesen oder nicht erkannt werden. Simuliere zwei Störungen und schätze den Grad der Auswirkungen auf den Informationsgehalt der Datei ein!

c) Wie verändert sich die Information, wenn das 5. Bit als 1 und nicht als 0 gelesen wird?  
Aus dem "E" wird ein "M"

---

→ der Test wird Fälschlicherweise als "Min Test" angezeigt

---

d) Wie verändert sich die Information, wenn das 5. Bit durch einen Lesefehler verloren geht und nur noch 71 Datenbits zur Verfügung stehen?

Das erste Byte ändert sich von 0100 0 101 → 01001010

---

d.h. aus "E" wird "J". Die restlichen Zeichen werden zu "unlesbaren" sonderzeichen

---

e) Erstelle jeweils eine Textdatei mit dem Inhalt „Würzburg“ in ASCII-Kodierung und in UTF-8-Kodierung. Überprüfe die oben angegebenen Dateigrößen!

Betrachte die beiden Dateien mit einem Hex-Editor und gib an, wie der Inhalt jeweils in Binärdaten abgelegt wurde:

ASCII:

---

UTF-8:

---

### 3. Dezimal-/Binärpräfix

Außerhalb der IT-Welt werden Präfixe schon seit langer Zeit verwendet, um ein vielfaches einer Zahl angeben zu können. Bekanntestes Beispiel dürfte der Präfix „k“ (=kilo) beim Kilometer sein, der angibt, dass eben 1000 Meter gemeint sind. Durch die Verwendung des Binärsystems kommt es im EDV-Bereich zu Zahlen wie 1024, die früher gerne ebenfalls mit „k“ abgekürzt wurden. (Veraltetes Beispiel: 1 Kilobyte = 1024 Byte)

Um solche Mehrdeutigkeiten zu vermeiden wurden 1996 neue Präfixe eingeführt:

Dezimalpräfixe		
Name	Symbol	Bedeutung
Kilobyte	KB	$10^3$ Byte = 1.000 Byte (wie bei "Kilometer": 1000 M == 1 km)
Megabyte	MB	$10^6$ Byte = 1.000.000 Byte
Gigabyte	GB	$10^9$ Byte = 1.000.000.000 Byte
Terabyte	TB	$10^{12}$ Byte = 1.000.000.000.000 Byte
Petabyte	PB	$10^{15}$ Byte = 1.000.000.000.000.000 Byte
Exabyte	EB	$10^{18}$ Byte = 1.000.000.000.000.000.000 Byte
Zettabyte	ZB	$10^{21}$ Byte = 1.000.000.000.000.000.000.000 Byte
Yottabyte	YB	$10^{24}$ Byte = 1.000.000.000.000.000.000.000.000 Byte

Binärpräfixe		
Name	Symbol	Bedeutung
Kibibyte	KiB	$2^{10}$ Byte = 1.024 Byte → Differenz 24 Byte → 2.4%
Mebibyte	MiB	$2^{20}$ Byte = 1.048.576 Byte
Gibibyte	GiB	$2^{30}$ Byte = 1.073.741.824 Byte
Tebibyte	TiB	$2^{40}$ Byte = 1.099.511.627.776 Byte → Differenz ~99GB → 9,9%
Pebibyte	PiB	$2^{50}$ Byte = 1.125.899.906.842.624 Byte
Exbibyte	EiB	$2^{60}$ Byte = 1.152.921.504.606.846.976 Byte
Zebibyte	ZiB	$2^{70}$ Byte = 1.180.591.620.717.411.303.424 Byte
Yobibyte	YiB	$2^{80}$ Byte = 1.208.925.819.614.629.174.706.176 Byte Differenz ~ sehr viel ~ 20.9%

Das Problem kommt z.B. beim Kauf von Festplatten auch heute noch zum Tragen: der Hersteller gibt eine Festplattenkapazität von 500 Gigabyte an, im Laptop werden auf einmal aber lediglich 466 Gigabyte angezeigt. Was ist passiert?

Den Herstellern kann kein Vorwurf gemacht werden, denn sie halten sich an die Norm: Ein Kilobyte umfasst für sie 1.000 Byte, eine Festplatte mit 500 Gigabyte bietet demnach eine Kapazität von 500 Milliarden Byte. Doch Windows rechnet anders. Microsoft richtet sich nach dem allgemeinen Sprachgebrauch, ein Kilobyte besteht für Windows aus 1.024 Byte. So kommt das System auf niedrigere Werte (nämlich auf falsche).

Man mag das als Kleinigkeit abtun. Doch während 24 Byte Unterschied bei einem Kilobyte zu verschmerzen sein mögen, erreicht die Differenz bei einer Terabyte-Festplatte schon fast 70 Gigabyte. Denn mit jeder Größenordnung wächst das Ausmaß des Fehlers: Ein Terabyte entspricht laut Windows nicht  $1.000^4$ , sondern  $1.024^4$  Byte tatsächlich wären das 1,1 Terabyte, zehn Prozent mehr.

Unter Mac OS X und diversen Linux-Varianten orientiert sich die Kapazitätsanzeige der Datenträger inzwischen an den offiziellen Abkürzungen, so dass eine „1 TB“-Platte auch in der Systeminformation mit 1 Terabyte angezeigt wird und nicht mit dem kleineren, binären Wert (931 GB oder 0,9 TB). An der Kapazität der eigentlichen Festplatte ändert sich dadurch natürlich gar nichts, es sind immer 1.000.000.000.000 Byte gemeint.

Welche Einheit die „korrekte“ ist, hängt meistens vom Kontext ab:

Übertragungsgeschwindigkeiten werden in der Regel mit dem Dezimalpräfix angegeben:

Menge	Einheit	Bits pro Sekunde	Byte pro Sekunde	Beschreibung
				ISDN Nutzkanal
				USB 2.0
				Gigabit Ethernet
				10 Gigabit Ethernet

Speicherkapazitäten wiederum werden (meistens) mit dem Binärpräfix hochgerechnet, so dass sich für den Anbieter/Verkäufer ein optisch schönerer Wert ergibt (Ausnahme: z.B. CD-ROM).

**VORSICHT: in älteren Abschlussprüfungen wurden die Bezeichnungen teilweise wild durcheinander gewürfelt bzw. wurde diese Unterscheidung nicht beachtet. In aktuellen Prüfungen sollten die Einheiten korrekt verwendet werden.**

#### Übungsaufgaben:

Aus der IHK-Prüfung (GA1-FISI, Sommer 2012, Handlungsschritt 1)

Die Anbindungen [der beiden Standorte] an das Internet erfolgen über DSL-Anschlüsse.

Zentrale: SDSL mit 5 Mbit/s symmetrisch

Filiale: ADSL mit 24 Mbit/s download, 1 Mbit/s upload

Berechnen Sie die Zeit in Sekunden, die der Transfer einer 3 MiB großen Datei aus der Filiale in die Zentrale benötigt [unter Idealbedingungen]. (4 Punkte)

---



---



---



---



---

Aus der IHK-Prüfung (GA1-FISI, Sommer 2013, Handlungsschritt 3)

[Im einleitenden Text ist ein Dezimal- und Binärpräfix benutzt worden.] Nennen Sie je ein weiteres Beispiel für die korrekte Verwendung eines Dezimalpräfixes und eines Binärpräfixes aus dem IT-Bereich. (2 Punkte)

AP1, Frühjahr 2022, 4. Aufgabe:

e) Die Datenbank soll in der Cloud gesichert werden.

Berechnen Sie die Zeit in Minuten, die für die Übertragung der 100 MiByte großen Datei bei einer VDSL-Leitung mit 100 Mbit/s download und 40Mbit/s upload benötigt wird.

Das Ergebnis ist auf volle Sekunden aufzurunden.

Der Rechenweg ist anzugeben.

4 Punkte

