



MATLAB für Naturwissenschaftler

6. Grundlegende Dokumentation

Till Biskup

Lehrstuhl für Physikalische Chemie und Didaktik
Universität des Saarlandes



Dokumentation

Motivation: Warum dokumentieren?

Arten von Dokumentation

Dokumentation im Code

Kein Code ohne Dokumentation

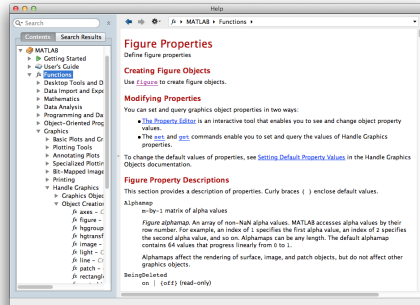
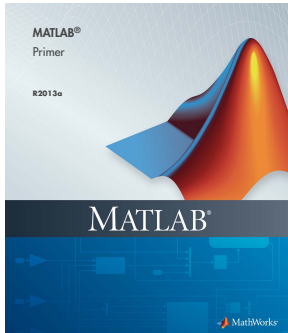
Grundlegende Struktur eines Kommentarkopfes

Dokumentationshilfen

Probleme von Dokumentation

Dokumentation

Entscheidend für gute Code-Qualität



Listing 1: linux-2.2.16/fs/buffer.c

```
1 /*  
2  * We used to try various strange things. Let's not.  
3  */
```



*Real programmers don't comment their code.
If it was hard to write, it should be hard to read.*

Warum dokumentieren?

- ▶ Weil andere das Programm nutzen/verstehen wollen.
 - ▶ Weil man sich selbst nach zwei Monaten nicht mehr daran erinnern kann, was man da programmiert hat.
 - ▶ Weil wir in aller Regel zu schlecht programmieren.
 - ▶ Weil Dokumentation (gerade von Konzepten) für die weitere Entwicklung sehr hilfreich ist.
- ☞ Weil nur sauberer (dokumentierter) Code Zukunft hat.



Viele verschiedene Arten

- ▶ Im Quellcode
 - Schnittstellen-Dokumentation
 - Quellcode-Dokumentation (einzelne Zeilen)
- ▶ Im gleichen Verzeichnis wie die Funktionen/Toolbox
 - README
 - INSTALL
- ▶ Nutzerhandbuch
 - Beschreibung der Verwendung jeder Funktion
- ▶ Konzepte
- ▶ Beispiele



Unterschiedliche Einteilung

- ▶ nach Zielgruppe
 - Programmierer
 - Anwender
- ▶ nach Inhalt
 - Quellcode-Dokumentation
 - Schnittstellen-Dokumentation
 - Dokumentation der Konzepte
 - Installation, Bedienung, ... (Anwenderdokumentation)
- ▶ nach Medium
 - im Quellcode
 - in Dateien neben dem Quellcode
 - getrennt vom Quellcode (Webseite, ...)



Listing 2: linux-2.2.16/lib/vsprintf.c

```
1 /* vsprintf.c -- Lars Wirzenius & Linus Torvalds.  
2  *  
3  * Wirzenius wrote this portably, Torvalds fucked it up :-)  
4  */
```

Listing 3: linux-2.2.16/fs/buffer.c

```
1 /*  
2  * After several hours of tedious analysis, the following  
3  * hash function won.  Do not mess with it... -DaveM  
4  */
```

Listing 4: linux-2.0.38/arch/m68k/atari/atafb.c

```
1 /* Nobody will ever see this message :-) */  
2 panic("Cannot initialize video hardware\n");
```



*Real Programmers don't need comments –
the code is obvious.*

– Ed Post, 1983

- ▶ Nicht dokumentierter Code ist (oft) wertlos.
- ▶ Fehlende Kommentare erschweren das Lesen von Code.
- ▶ Dokumentation im Code sollte kurz, prägnant und informativ sein.

👉 Beispiel gefällig?

Ed Post, „Real Programmers Don't Use Pascal“, Datamation 29(7), 1983



Listing 5: Undokumentierter Code

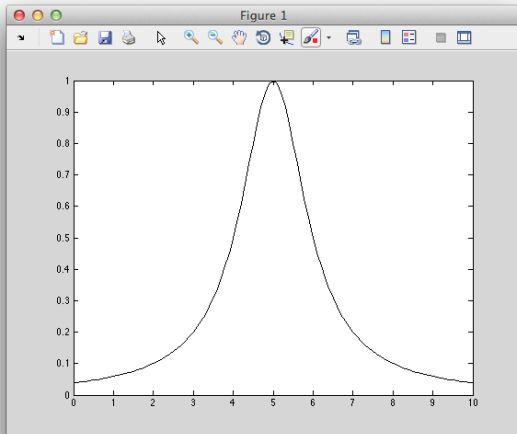
```
1 fun = @(x,s,t) s^2/4./ (s^2/4+(x-t).^2);  
2 s = 2;  
3 t = 5;  
4 x = 0:0.1:10;  
5 y = fun(x,s,t);  
6 plot(x,y,'k-');
```

- ▶ Keinerlei Kommentare
- ▶ Keine Gliederung des Codes durch Leerzeilen
- ▶ Keine oder wenige sprechende Variablennamen

- ☞ Irgendeine Idee, was der Code macht?
- ☞ Wie könnte der Code vernünftig dokumentiert aussehen?

Dokumentation im Code

Kein Code ohne Dokumentation: Ein Beispiel





Listing 6: Dokumentierter Code (lorentzian.m)

```
1 % LORENTZIAN Plot Lorentzian curve with fixed height.
2
3 % Copyright (c) 2013-14, Till Biskup <till.biskup@physchem.uni-freiburg.de>
4 % 2014-01-14
5
6 % Define Lorentzian with fixed height
7 %   s - width of the curve (FWHM)
8 %   t - position of the maximum
9 Lorentzian = @(x,s,t)s^2/4./(s^2/4+(x-t).^2);
10
11 % Define values for Lorentzian curve
12 width = 2;
13 maxpos = 5;
14
15 % Define x,y vectors
16 x = 0:0.1:10;
17 y = Lorentzian(x,width,maxpos);
18
19 % Plot
20 plot(x,y,'k-');
```



Regel

Keine Datei (Routine, Skript) ohne Dokumentation am Anfang.

- ☞ Dokumentation ist eine **Frage der Disziplin**.
- ☞ Zum „Rapid Prototyping“ eignet sich die Kommandozeile.
Skripte werden dokumentiert!
- ☞ Dokumentation muss zur Routine werden.
Im Nachhinein zu dokumentieren ist keine Option,
weil es dann nie gemacht wird.



Zwei Arten von Dokumentation im Code

1. Kommentarköpfe von Funktionen

- kurze Zusammenfassung der Aufgaben der Routine
- präzise Beschreibung der Schnittstelle
- Autor, Copyright und Datum der letzten Änderung
- ggf. Hinweise auf Besonderheiten

2. (Meist) einzeilige Kommentare im Code

- kurze Erklärungen zum folgenden Code-Abschnitt
- **wichtig:** nicht das Offensichtliche dokumentieren

☞ Bis auf die Beschreibung der Schnittstelle gilt der Kommentarkopf für Skripte ebenso.



Ein Kommentarkopf besteht mindestens aus drei Teilen

1. Kurze Zusammenfassung der Aufgaben der Routine
 - Mindestens ein Satz, besser ein kurzer Absatz, der die Aufgabe der Routine präzise beschreibt.
2. Präzise Beschreibung der Schnittstelle
 - Funktionsaufruf
 - Dokumentation der Ein- und Ausgabeparameter (Typ, Bedeutung)
3. Autor, Copyright und Datum der letzten Änderung
 - Wichtig: Datum der letzten Änderung immer anpassen!

☞ Der Kommentarkopf liefert dem Nutzer ein Maximum an Information auf minimalem Raum.



Listing 7: Dokumentationsblock zu Beginn einer Routine

```
1 function spectrum = simCO2spectrum(x,fwhm,pos,height)
2 % SIMCO2SPECTRUM Function for simulating CO2 spectra using Lorentzians.
3 %
4 % Usage
5 %     spectrum = simCO2spectrum(x,fwhm,pos,height)
6 %
7 %     x           - vector
8 %                   wavelength axis for spectrum
9 %
10 %    fwhm         - vector
11 %                   spectral width of each Lorentzian
12 %
13 %    pos          - vector
14 %                   position of each Lorentzian
15 %
16 %    height       - vector
17 %                   height of each Lorentzian
18 %
19 %    spectrum     - vector
20 %                   calculated spectrum
21 %                   same length as x
22 %
23 % The vectors fwhm, pos, and height have to be of the same size.
```



Listing 8: Copyrightinweis mit einem Autor

```
1 % Copyright (c) 2011-16, Till Biskup <till.biskup@physchem.uni-freiburg.de>  
2 % 2016-07-18
```

Listing 9: Copyrightinweis mit mehreren Autoren

```
1 % Copyright (c) 1997-2005, A. Kabelschacht  
2 % Copyright (c) 2006-2016, Till Biskup <till.biskup@physchem.uni-freiburg.de>  
3 % 2016-07-18
```

- ☞ Die Angabe einer Email-Adresse ist optional.
- ☞ Der Copyright-Hinweis wird durch eine Leerzeile vom Kommentarkopf getrennt.



- ▶ Der erste Kommentarblock einer Datei wird beim Aufruf des Befehls `help` ausgegeben.
 - Ähnliches gilt für andere Sprachen und im Zusammenhang mit automatischer Generierung der Dokumentation (Doxygen, Javadoc etc.)
- ▶ Da der Copyright-Hinweis nicht ausgegeben werden soll, wird dieser durch eine Leerzeile vom ersten Kommentarblock abgetrennt.
- ▶ Symbolworte, die vom `help`-Befehl erkannt werden
 - `see also`



- ▶ Kommentare im Code idealerweise auf Englisch
 - Wissenschaft ist international.
 - Japanische oder russische Kommentare im Quellcode sind (für die meisten von uns) leider wenig hilfreich...
- ▶ Aber: Lieber deutsche als keine Kommentare!
 - Es ist (meist) einfacher, einen Übersetzer zu finden, als einen undokumentierten Code zu verstehen.
- ▶ Grundsätzlich nie Sonderzeichen (z.B. Umlaute)
 - Unicode hat sich (leider) immer noch nicht durchgesetzt.
 - Beschränkung auf den ASCII-7-bit-Zeichensatz

Dokumentation im Code

Der ASCII-7-bit-Zeichensatz



Hex	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	NUL ^@	SOH ^A	STX ^B	ETX ^C	EOT ^D	ENQ ^E	ACK ^F	BEL ^G	BS ^H	TAB ^I	LF ^J	VT ^K	FF ^L	CR ^M	SO ^N	SI ^O
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	DLE ^P	DC1 ^Q	DC2 ^R	DC3 ^S	DC4 ^T	NAK ^U	SYN ^V	ETB ^W	CAN ^X	EM ^Y	SUB ^Z	ESC ^[\	FS ^\\	GS ^]	RS ^^	US ^?
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
20		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

☞ Zeichen 20_{Hex} bis 7e_{Hex} (32 bis 126) sind „druckbar“.



- ▶ Saubere Dokumentation im Code zählt sich aus:
 - Automatische Generierung von Dokumentation aus den Kommentaren im Quellcode.

- ▶ MATLAB
 - `help`-Befehl gibt ersten Kommentarblock aus.
 - Interne Funktionen sind gute Vorlagen/Beispiele.
 - Unterstützt das „Publizieren“ von Funktionen

- ▶ Dokumentationshilfen für andere Sprachen als MATLAB
 - Doxygen, Javadoc, ROBODoc
 - Liefern Ideen für eine sinnvoll strukturierte Dokumentation
 - Erlauben die automatische Erstellung in diversen Formaten ($\text{\LaTeX}$, DocBook, RTF, ...)



Problem

- ▶ Dokumentation hinkt der Entwicklung des Codes hinterher
 - Problem gerade bei Kommentarzeilen im Code:
im schlimmsten Fall falsch und irreführend!

Lösungsansatz

- ▶ Keine unnötigen Kommentare im Code
 - Offensichtliches nicht dokumentieren
 - Offensichtliche Programmierung der Dokumentation einer weniger offensichtlichen Lösung vorziehen
- ▶ Beim Ändern von Code immer auf Kommentare achten
- 👉 Letztlich eine Frage der (persönlichen) Disziplin.



Problem

- ▶ Dokumentation statt Verbesserung von Code
 - Unklare Namen (Variablen, Funktionen, ...)
 - Wenig offensichtliche Strukturen

Lösungsansatz

- ▶ Sprechende Namen für Variablen, Funktionen, ...
 - Gute Namen zu finden kostet Zeit – spart aber mehr Zeit
- ▶ Offensichtliche Verwendung von Code-Strukturen
 - `switch/case` statt `if/elseif`
 - intuitive Formulierung von Bedingungen

☞ Code sollte *lesbar* sein – im eigentlichen Sinne



Problem

- ▶ Fehlende Dokumentation der Installation und Bedienung
 - Nichts ist schlimmer als ein Verzeichnis voller Skripte/Funktionen ohne jegliche Beschreibung.

Lösungsansatz

- ▶ Externe Dokumentation
 - Extern zu den Funktionen
 - ggf. in eigenem Unterverzeichnis (`doc`)
 - reine Textdateien, kurz und prägnant
 - mindestens `README`, ggf. `INSTALL`



Problem

- ▶ Fehlende Dokumentation der Konzepte und Ideen
 - Schnittstellen-Dokumentation meist nicht ausreichend
 - Grundlegende Konzepte im Kontext dokumentieren
 - Statisches Dokument oft zu unflexibel

Lösungsansatz

- ▶ Konzeptionelle Dokumentation in einem [Wiki](#)
 - Flexibel
 - Erlaubt einfache Aktualisierungen
 - Geeignet als primäre Informationsquelle für Anwender.
- ▶ README für kleine Projekte



...Zeit für eigene praktische Arbeit...

Vorschau: Datenein- und -Ausgabe

- ▶ Text- und Binärdateien
- ▶ High- und Low-Level-Routinen