

Lilly is a Latex Lovable Yogurt

— It doesn't have to make any sense if it looks beautiful —

Dokumentation – Version 2.0.0

Autor & Instandhaltung:

Florian Sihler (florian.sihler@web.de)

23. August 2019



Abstract

Oder auch Einleitung 🍷 für **VER 2.0.0**

Die \LaTeX -Dokumentklasse **Lilly** ist im Rahmen des Studiums von Florian Sihler entstanden, und dient der Generierung studiumsrelevanter Dokumente & Mitschriften, in dessen Rahmen Lilly weiter angepasst und (hoffentlich) optimiert wurde. Die klassische Version basiert auf der [KOMA-Script](#) Dokumentklasse `scrbook`.

Das Ziel ist es auf Basis eines Makefiles das Latexdokument direkt in verschiedenen Versionen zu generieren! Die aktuelle Version „2.0.0 - Jake ist auch nur Java“ besitzt den Status Work in Progress!

Inhaltsverzeichnis

1 Boxen	1
1.1 Grundlegendes	1
1.1.1 Eine kleine Einführung	
1.1.2 Der Box-Controller	
1.2 Die Boxmodi	6
1.2.1 Default-Design	
2 Aussicht	9
2.1 Todos	9
2.1.1 Visuals	
2.1.2 Fehler	
2.1.3 Dateiaufteilung	
2.1.4 Road to CTAN	
2.1.5 Hoverover tooltips	
2.1.6 Weitere	
3 Anhang	10
3.1 Version  1.0.7	10
3.1.1 Installation in Linux	
3.1.2 Spezifikation: Plots	
3.2 Version  1.0.9	12
3.2.1 Installation in Linux	
3.2.2 Installation in MacOS	
4 *	14



BOXEN

BOXES IN BOXES IN BOXES IN BOXES...

VER 1.0.0

1.1 Grundlegendes

1.1.1 Eine kleine Einführung

Die 3 Standard-Designs, welche mit LILLY ausgeliefert werden lauten wie folgt:

DEFAULT	ALTERNATE	LIMERENCE
<div>Satz 1.1 Nice</div> <div>Superwichtig</div>	<div>Satz 1.2 – Nice</div> <div>Superwichtig</div>	<div>Satz 1.3 – Nice</div> <div>Superwichtig</div>

Auch wenn sie hier explizit forciert wurden ist es grundlegend möglich (und auch so gedacht) sie mithilfe des Makefiles konfigurieren. Die allgemeine Syntax bis VER 1.7.0 hierfür lautet:

```
1 make "BOXMODE=<Name>"
```

Mit VER 2.0.0 regelt *Jake* die jeweilige Variante und erlaubt es sogar, mehrere Boxmodi gleichzeitig generieren zu lassen:

```
1 jake <Datei> -lilly-boxes: "<Namen>"
```

Um eine Fassng für jede Box zu generieren entspräche das:

```
1 jake <Datei> -lilly-boxes: "DEFAULT.ALTERNATE.LIMERENCE"
```

wobei <Name> mit einem der oben stehenden Bezeichner ersetzt wird. Die Bezeichner werden vom weiter unten näher beschriebenen Box-Controller wie folgt aufgelöst:

```
1 \input{\LILLYxPATHxDATA/POIs/_LILLY_BOXES_\LILLYxBOXxMODE}
```

Über genau dieses Verfahren lassen sich auch beliebig die Box-Designs erweitern.

1.1.2 Der Box-Controller

Diese Definitionen befinden sich in der Datei: Controllers/LILLYxCONTROLLERxBOX. Sie werden mit VER 2.0.0 automatisch mit dem Einbinden von LIB LILLYxCONTROLLERxBOX geladen.

◇ \LILLYxBOXxMODE

v1.5.0

Hierrüber wird ausgewählt welcher der jeweiligen Boxmodi verwendet werden soll. Standardmäßig wird dieser Befehl von *Jake* gesetzt, aber natürlich kann dieser Befehl auch überschreiben werden.

◇ `\LILLYxBOXx<Bezeichner>xLock`

v1.8.0

Enthält für die jeweilige Box, woran sich der Zähler orientieren soll. Enthält der Befehl TRUE, so wird ein Ungebundener Zähler verwendet. Wenn nicht definiert initialisiert durch:

◇ `\LILLYxBOXxHIGHLEVELxLOCK`

v1.8.0

Enthält je nach Dokumenttyp entweder die höchste Hierarchie, an die ein Zähler gebunden werden kann oder TRUE. So erzeugt TRUE zum Beispiel den Zähler 4 und `section` in einem Dokument mit `\chapter: 1.13.4.`

◇ `\LILLYxBOXx<Bezeichner>xEnable`

v1.8.0

Definiert, ob eine Box überhaupt angezeigt werden soll. Durch das Setzen auf FALSE, kann so eine Box aus dem Dokument genommen werden.

◇ `\LILLYxBOXx<Bezeichner>xBox`

v1.8.0

Dieser Befehl besitzt (Stand `VER 2.0.0`) nicht für alle Boxbezeichner einen Effekt, steuert aber für bereits implementierte Boxen, ob diese durch das jeweilige Layout gesetzt werden sollen, oder ob die Box ohne die Box angezeigt werden soll. Die genaue Optik bestimmt wieder der jeweilige Modi.

Bemerkung 1.1 – Box-Kontrolle

Alle von Lilly generierten Boxen befinden sich mit `VER 2.0.0` in der Liste: `RegisteredBoxes` mit der Signatur `Name/Bezeichner`. So gehen die Konfigurationen wie folgt von statten:

```
1 \def\LILLYxBOXxBeweisxBox{FALSE} % Deaktiviert Beweisboxen
```

Hier die definierten Umgebungen in ihrer freien Wildbahn und Gestalt:

Definition 1.1 – Titel

moin

```
1 \begin{definition}[Titel]
2     moin
3 \end{definition}
```

Definition 1.2 – Titel



moin

```
1 \begin{definition*}[Titel]
2     moin
3 \end{definition*}
```

Bemerkung 1.2 – Titel

mein

```
1 \begin{bemerkung}[Titel]
2     mein
3 \end{bemerkung}
```

Beispiel 1.1 – Titel

mein

```
1 \begin{beispiel}[Titel]
2     mein
3 \end{beispiel}
```

Satz 1.4 – Titel

mein

```
1 \begin{satz}[Titel]
2     mein
3 \end{satz}
```

Beweis 1.1 – Titel

mein

```
1 \begin{beweis}[Titel]
2     mein
3 \end{beweis}
```

Lemma 1.1 – Titel

mein

```
1 \begin{lemma}[Titel]
2     mein
3 \end{lemma}
```

Zusammenfassung 1.1 – Titel



mein

```
1 \begin{zusammenfassung}[Titel]
2     mein
3 \end{zusammenfassung}
```

Aufgabe 1 – Titel

moin

```
1 \begin{aufgabe}{Titel}{3}
2     moin
3 \end{aufgabe}
```

Nicht richtig darstellbar aber weiter existiert:

```
1 \begin{uebungsblatt}[Titel][2]
2     moin
3 \end{uebungsblatt}
```

◇ `env@task[opt-Addons]{Titel}{Punkte}` WAR Veraltet v1.0.0

Ein aus dem `eagleStudiPackage` stammendes Relikt, welches nur aus Kompatibilitätsgründen gehalten wird. Ebenso:

◇ `\DEF{Title}{Content}, \BEM{Titel}{Content}, ...` WAR Veraltet v1.0.0

Kompatibilitätsbefehle, der zur `eagleStudiPackage`-Zeit die Boxen gesetzt hat, nun allerdings die Daten an die jeweilige Umgebung weitergeben.

◇ `\inputUB{Name}{Nummer}{Pfad}, \inputUBS{Name}{Bezeichner}{Pfad}` v1.0.3

Binden eine Datei als Übungsblatt ein und erlaubt so, Übungsblätter in Mitschriften zu integrieren. Letzterer Befehl verwendet `uebungsblatt*`, verändert also nicht die Nummer, was bedeutet, dass auch Buchstaben oder anderes als Bezeichner möglich ist. *Diese werden, entsprechend der Regel von `uebungsblatt` nur angezeigt, sofern `\LILLYxMODExEXTRA` den Wert `TRUE` enthält.*

Bemerkung 1.3 – Zugriff auf die Boxzähler

Der Zugriff auf die von Lilly unterhaltenen Boxzähler ist sehr Umständlich (Beispiel: `\thetcb@cnt@LILLYxBOXxDefinition`). Deswegen existiert das Hilfspaket LIB LILLYxBOXxCOUNTER, welches hierfür Kurzbefehle definiert: `\CTRxDf`, `\CTRxBEI`, `\CTRxBEM`, `\CTRxSAT`, `\CTRxBEW`, `\CTRxLEM` und `\CTRxZSM`. So liefert `\arabic{CTRxDf}` : 2.

◇ `\RegisterBox[Box-Keys][Tikz-Keys]{Box-Name}{Title}{BoxID}` v2.0.0

Registriert eine (neue) oder alte Box, die entsprechend Gesetzt werden kann. Die gezielte Verwendung dieses Befehls bedarf einiger Vorkenntnisse über das jeweilige Szenario. Es werden eine ganze Menge an `Box-Keys` gestattet, die auf einem anderen Verfahren persistiert werden (weswegen `Box`-gebundene Befehle mithilfe von `\noexpand` abgesichert werden müssen!):

Bezeichner	Typ	Standard	Beschreibung
------------	-----	----------	--------------

name	<i>String</i>	noname	Name der Box
title	<i>String</i>	<name>	Titel der Box
boxcol	<i>Farbe</i>	black	Farbe für Links
preCode	<i>Code</i>		Befehle, die vor der Box gesetzt werden
inCode	<i>Code</i>		Befehle, die zu Beginn der Box gesetzt werden
outCode	<i>Code</i>		Befehle, die zu Ende der Box gesetzt werden
postCode	<i>Code</i>		Befehle, die nach der Box gesetzt werden
usestyle	<i>Box</i>	<Def>	Basisbox
emblem	<i>Code</i>		Titelverzierer
createlist	<i>Bool</i>	false	Erstelle Liste
customlist	<i>Bool</i>	false	Trage Box in Liste ein
lock	<i>Lock</i>	TRUE	Setzt die Zählersperre
listname	<i>String</i>	<name>	Name der Liste
listtext	<i>String</i>	<name>	Titel der Liste
listmen	<i>String</i>	NO	Mnemonic der Liste

Mit dem Registrieren einer Box, werden die folgenden Befehle für die jeweilige BoxID registriert. Sie werden expandiert, weswegen eine Absicherung mithilfe von `\noexpand` für übergebene Befehle erfolgen sollte. Im Folgenden wird <BoxID> für die Befehle als Platzhalter verwendet:

- ◇ `\lillyxBOXx<BoxID>xName` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xTitle` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xBoxCol` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xPreCode` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xInCode` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xOutCode` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xPostCode` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xUseStyle` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xEmblem` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xLock` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xListName` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xListText` v2.0.0
 - ◇ `\lillyxBOXx<BoxID>xListMen` v2.0.0
- Sie speichern dne Wert der sich jeweils vermuten lässt. Weiter werden noch die beiden Booleschen Werte gespeichert, die entsprechend es Zustands TRUE und FALSE:
- ◇ `\lillyxBOXx<BoxID>xCreatelist` v2.0.0

◇ `\lillyxBOXx<BoxID>xCustomList`

v2.0.0

Die Modifikation dieser ermöglicht eine weitere manuelle Anpassung der Box, allerdings wird von einem direkten, modifizierendem Zugriff abgeraten. `\RegisterBox` legt weiter auch noch die entsprechende Umgebung auf Basis des Namens an. Gilt es eine bestehende Box zu modifizieren, so gilt es folgenden Befehl zu nutzen:

◇ `\TransformBox[new-args][bool-keep]{OldBoxID}{Box-title}` `[new-name]{NewBoxID}`

v2.0.0

Dieser Befehl nimmt die Werte einer bereits bestehenden Box und transformiert sie in eine neue Box. Sollte keine Box mit der entsprechenden ID existieren, so wird die BoxID als Name angenommen auf dessen Namen die Umbenennung geschieht. So lassen sich auch bereits bestehende Umgebungen, die bisher keine Box dargestellt haben, entsprechend modifizieren und als Box rekreieren. Wird `bool-keep` auf `\BooleanTrue` gesetzt, so wird die alte Box nicht gelöscht, sondern lediglich eine neue Box kreiert. Mit VER 2.0.0 wurden die Boxen noch nicht in das entsprechend neue Register-System übersetzt. Allerdings existieren die folgenden Kürzel, die bestmöglich versuchen die Gestalt der jeweiligen Box abzubilden. Da die `aufgabe`-Box eine andere Signatur besitzt, wird sie bisher separat implementiert:

◇ `\TransformBoxDefinition[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxDefinitionS[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxBemerkung[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxBemerkungS[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxBeispiel[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxBeispielS[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxSatz[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxBeweis[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxBeweisS[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxLemma[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxLemmaS[*][new-args]{BoxID}{Box-title}[NewBoxID]`

v2.0.0

◇ `\TransformBoxZusammenfassung[*][new-args]{BoxID}` `{Box-title}[NewBoxID]`

v2.0.0

Die jeweils mit dem S-Suffix vermerkten Befehle stehen für die mit einem Sternchen versehene Umgebung, der optionale Stern entspricht `bool-keep` von `\TransformBox`.

1.2 Die Boxmodi

Generell muss einer der unten aufgeführten Modi keine einzige Box definieren überladen oder modifizieren, das geladene Default-Paket wird jeweils für nicht überladene Boxen die Standardboxen zur Verfügung stellen. Folgende Boxbezeichner sind hierbei von Relevanz. Das Default-Design stellt hierbei mit LIB LILLYxLIST die in der Liste `RegisteredBoxes` aufgeführten Boxen zur Verfügung. An einer Normierung der `aufgabe`-Box wird gearbeitet: `LILLYxBOXxDefinition`, `LILLYxBOXxBeispiel`, `LILLYxBOXxBemerkung`, `LILLYxBOXxSatz`, `LILLYxBOXxBeweis`, `LILLYxBOXxLemma`,

LILLYxBOXxZusammenfassung, LILLYxBOXxAufgabe, LILLYxBOXxAufgabexPLAIN und LILLYxBOXxUebungsblatt. In der Regel wird weiter noch ein Design generiert, so setzt zum Beispiel das Default-Design: LillyxBOXxDesignxDefault.

1.2.1 Default-Design

Mit `VER 1.0.0` stellt dieses Design den Urvater dar. Bis `VER 1.0.6` überarbeitet hier die finale Form:

Zusammenfassung 1.2

Wichtige Box - Yeah

Wir haben schon viel gelernt, zum Beispiel, dass der Apfel nicht weit vom Stamm fällt. Das ist aber eigentlich dann auch schon so das Einzige, was es wirklich zu lernen gilt im Kontext dieser wundervollen Boxwelt!

Erzeugt durch den bekannten Aufruf:

```
1 \begin{zusammenfassung}[Wichtige Box – Yeah]
2   %% ....
3 \end{zusammenfassung}
```

Auf Basis des Pakets `tcolorbox` definiert LILLY das Design `LillyxBOXxDesignxDefault` mit folgender Implementation:

```
1 \tcbset{LillyxBOXxDesignxDefault/.style={enhanced jigsaw,
2   pad before break*=2mm, pad after break=2mm, %
3   lines before break=4, before skip=0pt, boxrule = 0mm, %
4   toprule=0.5mm, bottomtitle=0.5mm, bottomrule=1.2mm, %
5   after skip=0pt, enlarge top by=0.2\baselineskip, %
6   enlarge bottom by=0.2\baselineskip, %
7   sharp corners=south, enforce breakable}%
8 }
```

Auf Basis dessen werden nun die einzelnen Boxumgebungen generiert. Hier exemplarisch die obige Zusammenfassung:

```
1 \DeclareTColorBox[auto counter]%
2   {LILLYxBOXxZusammenfassung}%
3   { 0{} %% Title
4     0{Zusammenfassung \thetcbcounter~} %% TitlePrefix
5     0{} %% tcb addonargs
6   }{%
7   LillyxBOXxDesignxDefault, %
8   colback=\LILLYxColorxZusammenfassung!5!white, %
9   colframe=\LILLYxColorxZusammenfassung, #3,%
10  title={\LILLYxDEFAULTxTYPESETxTITLE{#1}{#2}%
11    \ifx\LILLYxBOXxZusammenfassungxLock\true\\%
12    \else\\[-0.4\baselineskip]\fi}% spacing
13 }
```

Hierbei verwendet das ganze Paket den vermerkten `\LILLYxDEFAULTxTYPESETxTITLE`, der selbst wie folgt konstruiert ist:

Bisher definiert LILLY die Counter über die Einstellung `auto counter` - dies soll aber bald auf das vom eagleStudiPackage Package verwendete `counter`-Verfahren umgestellt werden. Bis dato sieht eine exemplarische Definition einer Box wie folgt aus:

```

1 \DeclareTColorBox[auto counter]%
2   {\LILLYxBOXxDefinition}%
3   { O{} O{Definition \thetcbcounter~} O{drop fuzzy shadow} }%
4   {LillyxBOXxDesignxDefault, colback=\LILLYxColorxDefinition!5!
5     white,%
6     colframe=\LILLYxColorxDefinition, #3,%
7     title={%
8       \begin{minipage}[t][\baselineskip][1]{\textwidth}%
9       \textbf{\textsc{{#2}}}\ \hfill {\textbf{#1}}%
10      \end{minipage}%
11    }%
12  }

```

Hiervon weichen nur 2 Definitionen ab. Die der Aufgaben-Box:

```

1 \DeclareTColorBox{\LILLYxBOXxAufgabe}{O{} O{} O{}}{enforce breakable,%
2   colback=white,colframe=black!50,boxrule=0.2mm,%
3   attach boxed title to top left={xshift=1cm,yshift*=1mm-
4     \tcboxedtitleheight},%
5   varwidth boxed title*=-3cm,%
6   boxed title style={
7     frame code={
8       \path[fill=white!30!black]%
9         ([yshift=-1mm,xshift=-1mm]frame.north west)%
10        arc[start angle=0,end angle=180,radius=1mm]%
11        ([yshift=-1mm,xshift=1mm]frame.north east)%
12        arc[start angle=180,end angle=0,radius=1mm];
13      \path[left color=white!40!black,right color=white!40!black,
14        middle color=white!55!black]
15        ([xshift=-2mm]frame.north west) -- ([xshift=2mm]frame.
16        north east)%
17        [rounded corners=1mm]-- ([xshift=1mm,yshift=-1mm]frame.
18        north east)%
19        -- (frame.south east) -- (frame.south west)%
20        -- ([xshift=-1mm,yshift=-1mm]frame.north west)%
21        [sharp corners]-- cycle;%
22    },interior engine=empty,%
23  },
24  enhanced jigsaw, before skip=2mm,after skip=2mm,%
25  fonttitle=\bfseries, #3,%
26  title={#2 \ifthenelse{\equal{#1}{}}{--~}{#1}, %Aufgabe
27 }

```

AUSSICHT

DAS WUNDER DER SCHÖPF... EVOLUTION ☺

2.1 Todos

2.1.1 Visuels

Es wäre schön (auch auf Basis von `tcolorbox`) einige Umgebungen zu haben, mit denen sich Grafiken oder Textabschnitte einfach positionieren lassen. So ist es lästig hierfür jedesmal `minipages` und unsicher hierfür jedesmal `floatings` zu verwenden.

2.1.2 Fehler

Das Paket sollte Befehle wie `\PackageInfo/Error/Warning` unterstützen und auch ausgeben - zudem sollte die komplette Dateistruktur robuster werden und auf Fehler reagieren können

2.1.3 Dateiaufteilung

Die Aufteilung von LILLY in verschiedene Dateien war zum Beibehalt der Übersicht unabdinglich, allerdings sollte diese Aufteilung einigen Kontrollblicken und Korrekturen unterzogen werden - zudem sollte in dem Rahmen das Implementieren neuer Designs/Codes vereinfacht werden - hierfür würde sich ein einfaches Skript anbieten, was neue Dateien (je nach Typ) automatisch an die richtige Stelle bringt. Weiter wäre es gut, wenn die Dateinamen nicht nur `.tex` o.ä. lauten würden

2.1.4 Road to CTAN

Es sollten die notwendigen Installationsdateien und Dokumentationen generiert und eingebracht werden - sodass Lilly automatisiert verwaltet werden kann.

2.1.5 Hoverover tooltips

Eine Idee war es bei Hyperlinks Kommentare mithilfe von Tooltips zu realisieren. Somit wäre es möglich auf den meisten Geräten schnell Informationen zu liefern mithilfe von: Ich bin ein toller Hyperlink.

2.1.6 Weitere

Siehe hier für weitere Todos: <https://github.com/EagleoutIce/LILLY/issues>

3

ANHANG

VERALTETE DOKUMENTE, ZUSÄTZLICHES, EASTER-EGGS, ...

3.1 Version VER 1.0.7

3.1.1 Installation in Linux

Da LILLY komplett auf einem Linux-Betriebssystem entwickelt wurde, gestaltet sich die Implementierung relativ einfach. Zuerst gilt es einen neuen Ordner zu erstellen:

```
1 mkdir -p "${HOME}/texmf/tex/latex/"
```

In diesen Ordner (wenn nicht sogar bereits existent) kann nun der gesamte Lilly-Ordner verschoben werden (oder mithilfe eines symbolischen Links verknüpft). Als letztes muss man nun noch T_EX über das neue Verzeichnis informieren:

```
1 texhash "${HOME}/texmf"
```

Nun gilt es sich den anderen mitgelieferten Dateien zu widmen! Von besonderer Relevanz ist hierbei `lilly_compile.sh`, welches hier ausführlicher beschrieben wird (REMOVED: OLD). Grundslegend generiert es ein Makefile, das dann zum Kompilieren des Dokuments gedacht ist!

Mithilfe von folgendem Befehl wurde das Makefile für diese Dokumentation generiert:

```
1 ./lilly_compile.sh "Lilly-Dokumentation.doc.tex" \
2 -dir="Dokumentation/"
```

Hierbei wird das Makefile gemäß folgenden Regeln erzeugt:

- ◊ Es soll die tex-Datei: „Lilly-Dokumentation.doc.tex“ kompiliert werden.
- ◊ Das ganze soll (relativ zu `lilly_compile.sh`) im Verzeichnis `Dokumentation` stattfinden - hier wird ebenfalls das Makefile generiert.

Bemerkung 3.1 – make

Logischerweise muss damit auch `make` auf dem System vorhanden sein:

```
1 sudo apt install "make"
```

Mit diesem Makefile kann man nun das Dokument generieren lassen. Zu beachten sei hierbei, dass `make` - im Falle der Regel `all` - Regeln parallel ausführen wird!

Diese Dokumentation wurde mit folgendem Befehl erstellt:

```
1 make "BOXMODE=LIMERENCE"
```

Hierbei lässt sich ebenfalls erkennen wie sich noch mit dem Makefile einzelne Komponenten (wie das verwendete Boxdesign) ändern lassen!

VER 1.0.0

Es wird *nicht* auf die Semantik einzelner Befehle eingegangen! Copy&Paste ist doof, tippen! ;)

Dies sichert uns die Persistenz des Pakets im Falle einer Neuinstallation/Updates von L_AT_EX

VER 1.0.2

Es wird mit den Regeln `default`, `all` und `clean` generiert, selbstredend lässt sich dies erweitern

Die Anführungszeichen dienen hier und in anderen Codebeispielen lediglich zur Übersicht!

3.2 Version VER 1.0.9

3.2.1 Installation in Linux

Für Versionen < 1.0.8 klicke hier: [klick mich!](#)

Da LILLY komplett auf einem Linux-Betriebssystem entwickelt wurde, gestaltet sich die Implementierung relativ einfach. Hierzu nutzen wir das Hilfsprogramm *Jake* welches selbst in C++ geschrieben wurde. Im Folgenden sind die Schritte kurz erklärt.

VER 1.0.8

Installation von *Jake* :

Eine ausführliche Erklärung von *Jake* selbst findest du weiter hinten (hier) in dieser Dokumentation:

1. Navigiere mit dem Terminal in das Verzeichnis: `Lilly/Jake/jake_source`
2. Führe nun `make` aus um *Jake* zu kompilieren. Es wird vermutlich kurz dauern, aber danach wird dir das Programm `|lilly_jake|` zur Verfügung stehen.
3. Nun kannst du dein Terminal neu starten und von überall her `lilly_jake install` aufrufen. Dies sollte den Installationsprozess in Gang setzen.

Für ausführliche Informationen zur Installation konsultiere bitte die README-Datei in: `../Lilly/Jake/jake_source/README.md`.
Für Informationen zur Nutzung konsultiere: `../Lilly/Jake/README.md`

Sollte das Ganze fehlerfrei verlaufen sein, dann: Glückwunsch, du hast Lilly erfolgreich installiert! Betrachte im Falle eines Fehlers bitte erst die Readme-Dateien und die bereits beantworteten Fehler auf Github ([🔗](#)) bevor du einen neuen Fehler eröffnest oder mir eine Nachricht schreibst ☺.

Erstellen eines Makefiles:

Nun möchtest du natürlich auch ausprobieren ob die Installation funktioniert hat. Hierzu kannst du in das Testverzeichnis navigieren (`Lilly/Jake/tests`). Hier befinden sich eine Menge Dateien die in dieser Dokumentation auch als Beispiele benutzt werden. Du gibst nun folgendes in die Konsole ein:

```
1 lilly_jake test.tex
```

Jake erstellt nun ein entsprechendes Makefile für dich, welches du nun ausführen kannst:

```
1 make
```

Im Standardmäßig konfigurierten Ausgabe-Ordner `test-OUT` befindet sich nun eine entsprechende PDF Datei ☺.

Bemerkung 3.2 – make

Logischerweise muss damit auch `make` auf dem System vorhanden sein:

```
1 sudo apt install "make"
```

3.2.2 Installation in MacOS

Entspricht, dank *Jake* , der Linux-Installation.

Hierzu nutzen wir das Hilfsprogramm *Jake* welches selbst in C++ geschrieben wurde. Im Folgenden sind die Schritte kurz erklärt.

Installation von *Jake* :

Eine ausführliche Erklärung von *Jake* selbst findest sich weiter hinten (TODO: LINK) in dieser Dokumentation:

1. Navigiere mit dem Terminal in das Verzeichnis: `Lilly/Jake/jake_source`
2. Führe nun `make` aus um *Jake* zu kompilieren. Es wird vermutlich kurz dauern, aber danach wird dir das Programm `|lilly_jake|` zur Verfügung stehen.
3. Nun kannst du dein Terminal neu starten und von überall her `lilly_jake install` aufrufen. Dies sollte den Installationsprozess in Gang setzen.

Für ausführliche Informationen zur Installation konsultiere bitte die README-Datei in: `../Lilly/Jake/jake_source/README.md`.
Für Informationen zur Nutzung konsultiere: `../Lilly/Jake/README.md`

Sollte das Ganze fehlerfrei verlaufen sein, dann: Glückwunsch, du hast Lilly erfolgreich installiert! Betrachte im Falle eines Fehlers bitte erst die Readme-Dateien und die bereits beantworteten Fehler auf Github ([🔗](#)) bevor du einen neuen Fehler eröffnest oder mir eine Nachricht schreibst ☺.

Erstellen eines Makefiles:

Nun möchtest du natürlich auch ausprobieren ob die Installation funktioniert hat. Hierzu kannst du in das Testverzeichnis navigieren (`Lilly/Jake/tests`). Hier befinden sich eine Menge Dateien die in dieser Dokumentation auch als Beispiele benutzt werden. Du gibst nun folgendes in die Konsole ein:

```
1 lilly_jake test.tex
```

Jake erstellt nun ein entsprechendes Makefile für dich, welches du nun ausführen kannst:

```
1 make
```

Im Standardmäßig konfigurierten Ausgabe-Ordner `test-OUT` befindet sich nun eine entsprechende PDF Datei ☺.

Bemerkung 3.3 – make

Logischerweise muss damit auch `make` auf dem System vorhanden sein:

```
1 sudo apt install "make"
```

Stichwortverzeichnis

D	
<code>\DEF</code> (v1.0.0)	4
I	
<code>\inputUB</code> (v1.0.3)	4
L	
<code>\LILLYxBOXx<Bezeichner>xBox</code> (v1.8.0) . .	2
<code>\LILLYxBOXx<Bezeichner>xEnable</code> (v1.8.0)	2
<code>\LILLYxBOXx<Bezeichner>xLock</code> (v1.8.0)	2
<code>\lillyxBOXx<BoxID>xBoxCol</code> (v2.0.0) . . .	5
<code>\lillyxBOXx<BoxID>xCreateList</code> (v2.0.0)	6
<code>\lillyxBOXx<BoxID>xCustomList</code> (v2.0.0)	6
<code>\lillyxBOXx<BoxID>xEmblem</code> (v2.0.0) . . .	5
<code>\lillyxBOXx<BoxID>xInCode</code> (v2.0.0) . . .	5
<code>\lillyxBOXx<BoxID>xListMen</code> (v2.0.0) . . .	5
<code>\lillyxBOXx<BoxID>xListName</code> (v2.0.0) . .	5
<code>\lillyxBOXx<BoxID>xListText</code> (v2.0.0) . .	5
<code>\lillyxBOXx<BoxID>xLock</code> (v2.0.0)	5
<code>\lillyxBOXx<BoxID>xName</code> (v2.0.0)	5
<code>\lillyxBOXx<BoxID>xOutCode</code> (v2.0.0) . . .	5
<code>\lillyxBOXx<BoxID>xPostCode</code> (v2.0.0) . .	5
<code>\lillyxBOXx<BoxID>xPreCode</code> (v2.0.0) . . .	5
<code>\lillyxBOXx<BoxID>xTitle</code> (v2.0.0)	5
<code>\lillyxBOXx<BoxID>xUseStyle</code> (v2.0.0) . .	5
<code>\LILLYxBOXxHIGHLEVELxLOCK</code> (v1.8.0) . . .	2
<code>\LILLYxBOXxMODE</code> (v1.5.0)	1
R	
<code>\RegisterBox</code> (v2.0.0)	4
T	
<code>env@task</code> (v1.0.0)	4
<code>\TransformBox</code> (v2.0.0)	6
<code>\TransformBoxBeispiel</code> (v2.0.0)	6
<code>\TransformBoxBeispiels</code> (v2.0.0)	6
<code>\TransformBoxBemerkung</code> (v2.0.0)	6
<code>\TransformBoxBemerkungS</code> (v2.0.0)	6
<code>\TransformBoxBeweis</code> (v2.0.0)	6
<code>\TransformBoxBeweisS</code> (v2.0.0)	6
<code>\TransformBoxDefinition</code> (v2.0.0)	6
<code>\TransformBoxDefinitionS</code> (v2.0.0)	6
<code>\TransformBoxLemma</code> (v2.0.0)	6
<code>\TransformBoxLemmaS</code> (v2.0.0)	6
<code>\TransformBoxSatz</code> (v2.0.0)	6
<code>\TransformBoxZusammenfassung</code> (v2.0.0)	6