tgm
Die Schule der Technik

JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

Systemtechnik Labor

xHIT 2017/18, Gruppe A

Protokolle in LaTeX

Laborprotokoll

Markus Reichl

17. April 2018

Bewertung:

Betreuer: Michael Borko

Version: 0.1

Begonnen: 31. Januar 2018

Beendet: 17. April 2018

Inhaltsverzeichnis

1 Einführung

Diese Protokollvorlage soll helfen den Laborübungsteil entsprechend dokumentieren zu können. Diese Vorlage ist in LaTeX verfasst.

1.1 Ziele

Hier werden die zu erwerbenden Kompetenzen und deren Deskriptoren beschrieben. Diese werden von den unterweisenden Lehrkräften vorgestellt.

Dies kann natürlich auch durch eine Aufzählung erfolgen:

- Dokumentiere wichtige Funktionen
- Gib eine Einführung zur Verwendung von LaTeX

1.2 Voraussetzungen

Welche Informationen sind notwendig um die Laborübung reibungslos durchführen zu können? Hier werden alle Anforderungen der Lehrkraft detailliert beschrieben und mit Quellen untermauert.

1.3 Aufgabenstellung

Hier wird dann die konkrete Aufgabenstellung der Laborübung definiert.

1.4 Bewertung

Hier wird die Bewertung für das Beispiel auf die jeweiligen Kompetenzen aufgeteilt. Diese soll zur leichteren Abnahme auch nicht entfernt werden.

Nun kommt ein Seitenumbruch, um eine klare Trennung der Schülerarbeit zu bestimmen.

2 Konfiguration

2.1 Optionen

`landscape`	Richte das Dokument vertikal aus.
`minted`	Nutze das `minted` Paket zur Quelltextdarstellung.
`natbib`	Nutze NatBib zur Literaturverwaltung.
`nobib`	Deaktiviere die Literaturverwaltung.
`nofonts`	Nutze die Standard LaTeX Schriftarten.
`noglo`	Deaktiviere Akronyme und das Glossar.
`nologos`	Zeichne keine Logos auf der Titelseite.
`notitle`	Zeichne keine Titelseite.
`notoc`	Zeichne kein Inhaltsverzeichnis.
`notable`	Zeichne keine Tabelle auf der Titelseite.

2.2 Variablen

Variablen werden über Kommandos gesetzt, die als Parameter den gewünschten Wert erhalten.

```
\myvariable {value}
```

Kommando	Inhalt
`mysubtitle`	Untertitel oder Zugehörigkeit
`mysubject`	Thema / Fach, welches bearbeitet wird
`mycourse`	Kurs / Klasse welche(r) besucht wird
`myteacher`	Betreuende Lehrkraft
`myversion`	Aktuelle Version des Dokuments
`mybegin`	Datum des Beginns
`myfinish`	Datum an dem die Arbeit beendet wurde

3 Kommandos

3.1 makefig

```
\makefig{images/logo-right.png}{height=2cm}{
    Mit Beschreibung und Label  % (Optional)
}{
    fig:caption-label           % (Optional)
}
```

Auflistung 1: makefig

Abbildung 1: Mit Beschreibung und Label

3.2 vardef

```
$$e^{i*\pi} = -1$$
```

$$e^{i*\pi} = -1$$

```
\begin{vardef}
    \addvardef{$e$}{Eulersche Zahl}
    \addvardef{$\pi$}{Kreiszahl}
    \addvardef{$i$}{Imagin\"are Einheit}
\end{vardef}
```

e	...	Eulersche Zahl
π	...	Kreiszahl
i	...	Imaginäre Einheit

Auflistung 2: vardef

4 Anwendung

Hier sollen die Schritte der Laborübung erläutert werden. Hier sind alle Fragestellungen der Lehrkraft zu beantworten. Etwaige Probleme bzw. Schwierigkeiten sollten ebenfalls hier angeführt werden.

In diesem Fall werden einige LaTeX-Elemente dokumentiert, welche bei der Kreation von Protokollen behilflich sein könnten.

4.1 Tabellen

Header	Kopf
Lorem	Ipsum dolor sit amet, consetetur sadipscing elitr
Ipsum	At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus
Dolor	Consetetur sadipscing elitr, sed diam nonumy

Tabelle 1: Tabular

4.1.1 TabularX

TabularX erlaubt die Angabe der Größe der Tabelle und bietet zudem den Reihentyp `X`, der die verbleibende Größe neben anderen Reihen mit anderen `X` Reihen teilt.

ACHTUNG: Die Verwendung von `\codein`, `\mintinline` oder `\lstinline` ist in einer TabularX Umgebung nicht möglich!

Header	Kopf
Lorem	Ipsum dolor sit amet, consetetur sadipscing elitr
Ipsum	At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus
Dolor	Consetetur sadipscing elitr, sed diam nonumy

Tabelle 2: TabularX

4.2 Aufzählung

- Element einer Aufzählung
  - Erstes eingerücktes Element einer Aufzählung
  - Zweites eingerücktes Element einer Aufzählung

4.2.1 Outlines

- Element einer Aufzählung
  - Erstes eingerücktes Element einer Aufzählung
  - Zweites eingerücktes Element einer Aufzählung

4.3 Glossar

Zur Verwaltung des Glossars wird standardmäßig die Datei `glossaries.tex` verwendet, wobei sowohl Definitionen als auch Akronyme definiert werden können.

Als Beispiel wurde ein Akronym für Systemtechnik (SYT) und eine Definition zu SYT selbst hinzugefügt.

```
\newacronym{ac-syt}{SYT}{Systemtechnik}
\newglossaryentry{syt}{
    name={Systemtechnik},
    description={\enquote{Als Systemtechnik bezeichnet man verschiedene Aufbau- und
    ↪ Verbindungstechniken, aber auch eine Fachrichtung der
    ↪ Ingenieurwissenschaften. Er bedeutet in der Unterscheidung zu den
    ↪ Mikrotechnologien die Verbindung verschiedener einzelner Module eines
    ↪ Systems und deren Konzeption.} \cite{wiki:syt}}
}
```

Im Dokument selbst kann ein Akronym mittels `\gls{ac-syt}` verwendet werden. Beachte, dass ein Akronym welches bereits im Dokument verwendet wurde, bei der ersten Verwendung ausgeschrieben und danach immer gekürzt wird.

Mit `\gls{syt}` kann zum Beispiel eine Referenz zur Definition von Systemtechnik hinzugefügt werden.

4.4 Zitate

Zitate sollten gesammelt in der Datei `bib.bib` verwaltet werden.

4.5 Quelltext

```
\begin {code}[]{java}

// Ich bin ein Kommantar!
public static void main(String[] args) {
    System.out.println("Ich bin ein Array!")
}

\end {code}
```

Auflistung 3: Java Code

Die Darstellung von Quelltext im Text ist über das Kommando `\codein[options]{lang}{code}` möglich.

Eine einzelne Zeile kann mittels

```
\codeline [options]{lang}{code}
```

eingefügt werden.

4.5.1 Listings

```
\begin {lstlisting}[language=Java, caption=Java Lstlisting]

// Ich bin ein Kommantar!
public static void main(String[] args) {
    System.out.println("Ich bin ein Array!")
}

\end {lstlisting}
```

Auflistung 4: Java Lstlisting

4.5.2 Minted

Benötigt die Option `minted`.

Umgebung

```
\begin {minted}[options]{java}

// Ich bin ein Kommantar!
public static void main(String[] args) {
    System.out.println("Ich bin ein Array!")
}

\end {minted}
```

Auflistung 5: Minted Umgebung

Zeile

```
\mint [options]{lang}|code|
```

Auflistung 6: Minted Einzeiler

```
\mintinline[options]{lang}{code}
```

Auflistung 7: Minted Inline

```
#!/usr/bin/env python3

"""
============================
::  A latex build script  ::
============================

A multipurpose latex build script in python

@author     Markus Re1 <markus@re1.at>
@version    2018-04-17
@url        https://github.com/re1/tools


Copyright 2018 Markus Re1

Permission is hereby granted, free of charge, to any person obtaining a copy of this
    ↪  software and
associated documentation files (the "Software"), to deal in the Software without
    ↪  restriction,
including without limitation the rights to use, copy, modify, merge, publish,
    ↪  distribute, sublicense,
and/or sell copies of the Software, and to permit persons to whom the Software is
    ↪  furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
    ↪  copies or substantial
portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
    ↪  INCLUDING BUT NOT
LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
    ↪  NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES
    ↪  OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
    ↪  CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
"""

import argparse
import re

from glob import glob
from os import getcwd, remove
from os.path import isfile
from shutil import rmtree
from subprocess import call
```

```
41
42  # color codes to style console output
43  R = "\033[0m"       # reset colors
44  CL = "\033[91m"     # light gray
45  CR = "\033[91m"     # red
46  CG = "\033[92m"     # green
47  CO = "\033[93m"     # orange
48  CB = "\033[94m"     # blue
49  CP = "\033[95m"     # purple
50  CC = "\033[96m"     # cyan
51  # temporary files to remove during cleanup
52  TMP = [["**/*.acn", "**/*.acr", "**/*.alg", "**/*.aux", "**/*.bbl", "**/*.blg",
    ↪  "**/*-blx.bib",
53          "**/*.bcf", "**/*.dvi", "**/*.glg", "**/*.glo", "**/*.gls", "**/*.glsdefs",
            ↪  "**/*.ist",
54          "**/*.out", "**/*.run.xml", "**/*.synctex.gz", "**/*.toc", "**/*.xdy",
            ↪  "**/*.lot",
55          "**/*.lof", "**/*.lol"], []]
56
57
58  def bibtex(file="main", skip=False) -> bool:
59      """
60      Compile bibliography found in the given file using bibtex
61
62      :param skip: always return True
63      :param file: filename of a tex source file without its extension
64      :return: True on success or False on error
65      """
66      print(CB + "Run " + CP + "bibtex" + R + " on " + CC + file + ".tex" + R)
67
68      cmd = ("bibtex", file)      # compile command string before call
69      res = call(tuple(cmd)) < 1  # call the command and watch for errors
70
71      print(CG + "Success on " + CP + "bibtex" + R if res else CR + "Error on bibtex"
        ↪  + R)
72      print()                     # empty line for better readability
73      return skip or res          # return success or skip errors
74
75
76  def glossaries(file="main", out=".", skip=False) -> bool:
77      """
78      Compile glossary entries found in the given file using makeglossaries
79
80      :param skip: always return true
81      :param file: filename of a tex source file without its extension
82      :param out: path to copy compiled glossaries into
83      :return: True on success or False on error
84      """
85      print(CB + "Run " + CP + "makeglossaries" + R + " on " + CC + file + ".glo" + R)
```

```
86          # compile command string before call
87          cmd = ("makeglossaries", "-d", out, file)
88          res = call(tuple(cmd)) < 1  # call the command and watch for errors
89
90          print(CG + "Success on " + CP + "glossaries" + R if res else CR + "Error on
            ↪  glossaries" + R)
91          print()                     # empty line for better readability
92          return skip or res          # return success or skip errors
93
94
95      def tex(*args, command="pdflatex", file="main", verbose=False, out=".", skip=False)
        ↪  -> bool:
96          """
97          Compiles a tex source file using the given command
98
99          :param skip: always return true
100         :param args: additional arguments passed to the tex command
101         :param command: tex command used to compile the source files
102         :param file: filename of a tex source file without its extension
103         :param verbose: show output of tex command interaction
104         :param out: path to generate compiled files into
105         :return: True on success or False on error
106         """
107         print(CB + "Run " + CP + command + R + " on " + CC + file + ".tex" + R)
108
109         mode = "nonstopmode" if verbose else "batchmode"
110         cmd = [command,                     # build the final command before calling it
111                "-shell-escape",             # required by minted
112                "-file-line-error",          # show tag beneath log lines
113                "-interaction=" + mode,      # interaction mode
114                "-output-directory=" + out,  # output directory
115                *args, file]                 # additional args and filename
116         res = call(tuple(cmd)) < 1          # call the command and watch for errors
117
118         if isfile(file + ".log"):           # list warnings and errors found in log file
119             with open(file + ".log", "r", errors="replace") as log:
120                 for line in log:
121                     if re.search(":[0-9]*:", line):         # find errors
122                         print(CR + "Error: " + line)
123                     if re.search("[0-9]+--[0-9]+", line):   # find warnings
124                         print(CO + "Warning: " + R + line.replace("\n", ""))
125
126         print(CG + "Success on " + CP + command + R if res else CR + "Error on " +
            ↪  command + R)
127         print()                 # empty line for better readability
128         return skip or res      # return success or skip errors
129
130
131     def clean(recursive=True) -> bool:
```

```
132         """
133         Clean up directory by removing any file listed in the TMP[0] constant
134         and any directory listed in the TMP[1] constant
135 
136         :param recursive: also delete files from subdirectories when using **
137         """
138         print(CB + "Run " + CP + "clean" + R + " on " + CC + getcwd() + R)
139         # delete temporary files
140         res = [[remove(f) for f in glob(name, recursive=recursive)] for name in TMP[0]]
141         # delete temporary directories
142         res += [[rmtree(f) for f in glob(name, recursive=recursive)] for name in TMP[1]]
143         # return success or rare case of error
144         return all(res)
145 
146 
147     def full(*args, command="pdflatex", file="main", verbose=False, out=".",
    ↪   skip=False) -> bool:
148         """
149         Attempt a full compilation process and compile glossary entries such as
    ↪   bibliography if found
150 
151         :param skip: always return true
152         :param args: additional arguments passed to the tex command
153         :param command: tex command used to compile the source files
154         :param file: filename of a tex source file without its extension
155         :param verbose: show output of tex command interaction
156         :param out: path to generate compiled files into
157         :return: True on success or False on error
158         """
159         if not (tex(*args, command=command, file=file, verbose=verbose, out=out) or
        ↪   skip):
160             return False            # return error
161         if glob("*.bib"):  # bib files can have different names than the main file
162             bibtex(file=file)       # compile bibliography and call progressive tex
            ↪   commands
163             if not (tex(*args, command=command, file=file, verbose=verbose, out=out) or
            ↪   skip):
164                 return False        # return error
165             if not (tex(*args, command=command, file=file, verbose=verbose, out=out) or
            ↪   skip):
166                 return False        # return error
167         if isfile(file + ".glo"):   # compile glossary entries using and call a
        ↪   progressive tex command
168             if not (glossaries(file=file, out=out) or skip):
169                 return False        # return error
170             if not (tex(*args, command=command, file=file, verbose=verbose, out=out) or
            ↪   skip):
171                 return False        # return error
172         clean()                     # clean up after compilation
```

```
173         return True                 # return success
174
175
176     def parse(args):
177         """
178         Parse arguments from the args list and act according to their values
179
180         :param args: list of arguments to parse
181         """
182         # additional arguments for a targets command
183         arguments = args.args.split(" ") if args.args else []
184         # tex command to use when compiling tex sources
185         command = "pdflatex"
186         # names of the tex source files to compile without their extension
187         files = args.files.replace(".tex", "").split(" ")
188         out = args.out or "."           # directory to copy generated files into
189         skip = args.skip                # remember to ignore errors
190         target = args.target or "full"  # choose target full by default
191         verbose = args.verbose          # interaction mode for tex commands
192         # change tex command according to the arguments given by the user
193         if args.latex:
194             command = "latex"
195         if args.xelatex:
196             command = "xelatex"
197         # spare temp files
198         if not args.log:        # add log files and to temp file list
199             TMP[0].append("**/*.log")
200         if not args.minted:     # add minted dir to temp dir list
201             TMP[1].append("**/*_minted-*")
202         # choose a target or fall back to full compilation
203         if "clean" in target:       # clean up files and directories listed in the TMP
        ↪  constant
204             res = clean()
205         elif "draft" in target:     # run the tex command once
206             res = [tex(*arguments, command=command, file=file, verbose=verbose, out=out,
            ↪  skip=skip) for file in files]
207         elif "glo" in target:       # compile glossary entries
208             res = [glossaries(file=file, out=out, skip=skip) for file in files]
209         elif "bib" in target:       # compile bibliography entries using bibtex
210             res = [bibtex(file=file, skip=skip) for file in files]
211         else:                       # attempt a full compilation by default
212             res = [full(*arguments, command=command, file=file, verbose=verbose,
            ↪  out=out, skip=skip) for file in files]
213         # log success
214         print(CG + "Success on " + CP + target + R if all(res) else CR + "Error on " +
        ↪  target + R)
215
216
217     if __name__ == "__main__":
```

```
218     # create an argument parser instance and add various options
219     PARSER = argparse.ArgumentParser(
220         description="This python script helps compiling latex documents "
221                     "by providing most functions you would except from a build tool.")
222     # add arguments for different use-cases
223     PARSER.add_argument("target", nargs="?", default="all",
224                         help="the script will attempt a full compilation process by
                            ↪ default ."
225                              " To specify the target manually append bib, clean,
                             ↪ draft or glo.")
226     PARSER.add_argument("files", nargs="*", default="main", help="source tex files
        ↪ to compile")
227     PARSER.add_argument("-l", "--log", action="store_true", help="spare log during
        ↪ cleanup")
228     PARSER.add_argument("-m", "--minted", action="store_true", help="spare minted
        ↪ during cleanup")
229     PARSER.add_argument("-q", "--quiet", action="store_true", help="only show fatal
        ↪ errors")
230     PARSER.add_argument("-s", "--skip", action="store_true", help="skip errors in
        ↪ tex commands")
231     PARSER.add_argument("-v", "--verbose", action="store_true", help="do not filter
        ↪ logs")
232     PARSER.add_argument("-a", "--args", help="additional arguments for the
        ↪ operation")
233     PARSER.add_argument("-o", "--out", help="output directory to use if supported by
        ↪ the operation")
234     # only allow a single compiler
235     COMPILERS = PARSER.add_argument_group("latex
        ↪ compilers").add_mutually_exclusive_group()
236     COMPILERS.add_argument("-t", "--latex", action="store_true", help="parse tex
        ↪ files with latex")
237     COMPILERS.add_argument("-x", "--xelatex", action="store_true",
238                            help="compile tex files using xelatex (helps loading
                            ↪ custom fonts")
239     # parse arguments
240     parse(PARSER.parse_args())
```

Abbildungsverzeichnis

Tabellenverzeichnis

Auflistungsverzeichnis