panikzettel.htwr-aachen.de

Panikzettel Panikzettel

Version 3 | 16.12.2023

Jonas Schneider

Inhaltsverzeichnis

1 Vorwort

Dies ist ein Fork des originalen Panikzettel Repository von Phillip Schröer, da ursprüngliche Repository seit 2 Jahren verlassen zu sein scheint. Von ihm und weiteren Betragenden sind der Großteil der Panikzettel bis heute.

2 Einleitung

Nach 450 Seiten mit 30 Autoren wird es Zeit für den **Panikzettel Panikzettel**. Hier wollen wir kurz beschreiben, wie man zum Projekt beiträgt: Wie soll ein Panikzettel aussehen, wie arbeitet man mit dem Code und wie kann ein Panikzettel veröffentlicht werden? Jedoch sollte nicht vergessen werden: Wie alle anderen Panikzettel auch ist dieser hier nur eine Sammlung von Notizen. So gut wie alles ist auch mit dem Team zur Diskussion offen.

Am Besten ist es, wenn Ideen für Panikzettel mit dem Team abgesprochen werden:
- jonas.max.schneider@gmail.com
- panikzettel@panikzettel.philworld.de (ehemaliger owner. Antwortet vermutlich nicht)

Kein Zugriff auf den obigen Link?

Das Repository ist RWTH-intern verfügbar. Für den Zugriff muss man also als RWTH-Angehöriger angemeldet sein. Manchmal hat man trotzdem keinen Zugriff auf RWTH-interne Repositories, obwohl man über seinen RWTH-Account angemeldet ist. Das ist schon öfter passiert. Dann muss man eben dem IT Center der RWTH schreiben.

3 Panikzettel kompilieren und erstellen

Für Panikzettel brauchst du `latex` / `xelatex`/ `typst` usw., `git` und `make`.

Kompilieren mit make

Panikzettel werden mit `make` kompiliert:
- `make` bzw. `make all`: Alle `latex` Panikzettel kompilieren. Sie sind dann im `/build` Verzeichnis zu finden
- `make la.pdf`, `make meta.pdf`, etc.: Einen bestimmten (nur `latex`) Panikzettel bauen.
  Er ist dann im `/build` Verzeichnis zu finden
- `make typst`: Alle `typst` Panikzettel kompilieren.
- `typst meta.typ`, etc.: Einen bestimmten `typst`-Panikzettel bauen. (Benötigt `{name}.last-change` Datei. Siehe unten)
- `typst watch meta.typ`, etc.: Einen bestimmten `typst`-Panikzettel kontinuierlich bei Änderungen bauen. (Benötigt `{name}.last-change` Datei. Siehe unten)
- `make clean`: Alle Kompilate löschen.

Im Hintergrund wird in der `Makefile` für jeden Panikzettel eine `name.last-change` Datei erstellt, die von Git das Datum der letzten Änderung enthält. Das Datum wird dann oben eingeblendet. Dies passiert im Format:

```
Version 1
===
12.12.2012
```

Kompilieren ohne make (etwa in Overleaf/ typst.app)
- Für `latex`, lade die `panikzettel.cls` und die `.tex`-Datei des Panikzettels auf Overleaf hoch.
- Für `typst` brauchst du die `conf.typ` und die `.typ`-Datei des Panikzettels.
- Bei `latex` wird, wenn die `name.last-change` Datei nicht existiert, automatisch `\today` für das Datum verwendet.
- Bei `typst` hingegen, must du selber eine `.last-change` Datei erstellen (Die erste Zeile wird als Version, die dritte als Kompilationsdatum angezeigt).

Neuen Panikzettel erstellen

Für beide Schriftsysteme muss `metadata.json` ein neuer Eintrag erstellen werden. Hier werden generelle Informationen zu den Fächern angegeben.

> Wir empfehlen für neue Panikzettel auf Typst zu setzten, da, dass das Schreiben deutlich einfacherer gestaltet.

Latex

In der `Makefile`-Datei eine Regel für den neuen Panikzettel einfügen, etwa wie folgt. Außerdem oben in der `all`-Regel den neuen Eintrag zur Liste hinzufügen.

```
meta.pdf: meta.tex panikzettel.cls meta.last-change
  latexmk -pdflatex="pdflatex -interaction=nonstopmode" -pdf meta.tex
```

Der ehemalige `latex` Meta-Panikzettel wurde für den `typst` ersetzt. Er steht jedoch als gute Referenz im Repository als `meta.tex.bak` zur Verfügung.

Typst

Für typst muss einfach nur eine neue `name.typ` Datei im Hauptverzeichnis angelegt werden. Die `make typst` Regel findet sie automatisch. Es ist darauf zu achten in der `#show: conf.with()` den `title` und den `filename` zu ändern ebenso wie alles andere :). Es ist gut sich an diesem `typst` Meta-Panikzettel zu orientieren.

4 Einen Panikzettel schreiben

Panikzettel haben einen bestimmten Zweck: Sie sollen bei der Klausurvorbereitung im Moment der Panik helfen. Daraus ergeben sich einige grundlegende Tipps für den Stil der Panikzettel, die beim Schreiben beachtet werden sollten.

4.1 Grobe Richtlinien

Als Autor versetzt man sich in den Leser: Gerade alles zum Thema vergessen, und jetzt braucht man den Algorithmus oder den Satz zur Lösung einer Aufgabe. Kurze, verständliche Erklärungen, die Intuition vermitteln und nicht immer ganz formal korrekt sind. Keine langen Sätze und bitte nicht viel verschachteln.

Nicht die Vorlesung oder das offizielle Skript ersetzen, sondern ergänzen. Deshalb auch generell keine Beweise und nur selten Beispiele. Hier gibt es natürlich Ausnahmen: Etwa bestimmte Beispiele oder Beweise beim Verständnis besonders helfen, oder wenn sie zentral für die Prüfung sind.

Der Leser hat meistens Zeitdruck. Darum nicht mit überflüssigen Formulierungen oder Geschwafel langweilen, aber (kurze) Überleitungen und Einleitungen zur Erklärung und Einordnung sind meistens sinnvoll.

In Panik achtet man auf farbige Boxen und hervorgehobenen Text. Bei Definitionen in heben wir gerne das definierte Wort mit `\emph` (typst * *) hervor: „Ein **Wort** besteht aus Zeichen."

Definitions-, Satz- oder Algorithmen-Boxen sind für die zentralen Informationen eines Abschnittes hilfreich. Kurzer Text daneben kann diese weiter erklären. Unwichtigere Details, die technisch gesehen wohl auch Definitionen, Sätze oder Algorithmen sind, müssen nicht immer in Boxen. Überhaupt hilft gutes Layout bei der Orientierung. Deshalb sollte eine Seite gut gefüllt sein, aber die Struktur und empfohlene Leserichtung klar erkennbar sein. Jeder Panikzettel wird mindestens einmal gedruckt. Denke an die Umwelt und mache den Panikzettel kompakt.

4.2 Prozess

Panikzettel wurden meistens in Gruppen geschrieben und ziemlich kleinkariert kritisiert und sehr oft überarbeitet, bevor sie veröffentlicht werden. Eigentlich bleibt kein Satz so in der finalen Version, wie er zuerst geschrieben wurde. Anders als die private Zusammenfassung soll ein Panikzettel auch für andere verständlich sein. Deshalb legen wir sehr viel Wert auf Feedback und Korrektur.

Sehr sinnvoll ist es daher auch, schon früh Feedback einzuholen. Dabei muss noch nichts fertig sein, aber oft kann man sich dadurch unnötige Arbeit ersparen.

4.3 Quellen

Panikzettel basieren meistens auf einer Vorlesung. Daher ist es sinnvoll, die Struktur und Terminologie der Vorlesung anzupassen. Jedoch sollte einfaches Kopieren vermieden werden. Panikzettel sind eigene Zusammenfassungen, und Mehrwert kommt oft davon, dass man etwas selber neu schreibt. Daher fühlen wir uns auch am wohlsten, wenn Grafiken selbst erstellt wurden (etwa mit TikZ). Screenshots aus Folien machen wir nicht. Damit wir unter unserer Projektlizenz Panikzettel verbreiten dürfen, müssen die Inhalte also selber erstellt sein oder unter einer kompatiblen Lizenz veröffentlicht worden sein.

4.4 Struktur

Ein Panikzettel hat ein detailliertes Inhaltsverzeichnis, um einen Überblick über die wichtigen Themen zu bekommen. Wenn wichtige Themen im Panikzettel nicht erklärt werden, sollte trotzdem dazu ein Abschnitt mit einem Vermerk existieren, damit man das Thema im Inhaltsverzeichnis findet.

Die Einleitung sollte erklären, warum der Panikzettel existiert und worauf er basiert.

Die weitere Kapitelstruktur kann man dann an die Struktur der Vorlesung anlehnen. Das macht das Lernen mit Vorlesungsmaterial und Panikzettel einfacher.

5 Panikzettel-Quellcode

Es folgen Tipps für den Latex-Code eines Panikzettels.

5.1 Code-Stil

Damit man den Latex-Code gut lesen und mit Git überarbeiten kann, haben wir einige Anforderungen an den Code (die nicht immer eingehalten werden):

- Code zwischen `\begin` und `\end` einrücken.
- Jeder Satz kommt im Quellcode auf eine eigene Zeile. Das hilft bei Diffs mit Git.
- Absätze durch Leerzeilen trennen, `\\` nur selten verwenden.
- Kommentare sind gut, aber bitte nicht halbfertige Texte auskommentiert übrig lassen.
- Keine Leerzeichen am Ende der Zeile, das macht Diffs einfacher. Dein Editor sollte dafür eine Funktion haben (Stichwort „trailing whitespace“).

5.2 Die `panikzettel.cls`

In der `panikzettel.cls` haben wir unter anderem unsere Boxen für Definitionen, Sätze und Algorithmen definiert. Außerdem haben wir einige Pakete für alle Panikzettel importiert und einige praktische Kommandos selbst definiert. Wir empfehlen einen Blick in die `panikzettel.cls` für einen vollständigen Überblick.

Importiert wird die Datei über `\documentclass{panikzettel}` oder `\documentclass[english]{panikzettel}` für Englisch. Mit `\author` Autoren eintragen. Ein `\date` Kommando wird von der `panikzettel.cls` eingefügt (siehe Abschnitt über `make`).

Wir benutzen UTF-8.

5.3 Unsere Boxen

Wir definieren mit dem `tcolorbox`-Paket drei wichtige Arten von Boxen, die für typische Informatik-Panikzettel praktisch sind. Die Verwendung läuft über `\begin{algo}{Titel}` und `\end{algo}` bzw. mit `defi` und `theo` für Definitionen respektive Sätze.[1]

Algorithmus: Zwei Schritte

Eingabe: Was bekommt der Algorithmus?

Ausgabe: Was kommt raus?

---

1. Schritt 1
2. Schritt 2

Definition: Wiederholung

Sei $n \in \mathbb{N}$ eine Zahl und $w \in \Sigma^*$ ein Wort.
Die *Wiederholung* von w ist:

$$w^n = \underbrace{w \ldots w}_{\text{n Mal}}.$$

Satz: P und NP

Entweder ist $P = \text{NP}$, oder $P \neq \text{NP}$, aber wahrscheinlich nicht beides.

Bei Algorithmen verwenden wir immer das Schema mit **Eingabe** und **Ausgabe**, und dem Trennstrich mit `\tcblower`. Ber der Definition ist manuell ein Absatz mit `\\` eingefügt, um die Voraussetzungen und die eigentliche Definition zu trennen. Außerdem ist das definierte Wort mit `\emph` hervorgehoben.

5.4 Split Blocks

Um den Platz auf der Seite maximal effizient zu benutzen, teilen wir die Seite sehr oft in zwei oder sogar drei Spalten auf. Für gleich große Spalten haben wir die Umgebungen `halfboxl`/ `halfboxr` und `thirdboxl`/ `thirdboxm` / `thirdboxr` definiert. Beispielcode:

```
\begin{halfboxl}
  Linke Seite
\end{halfboxl}%
\begin{halfboxr}
  Rechte Seite
\end{halfboxr}
```

Man beachte das Kommentar-Zeichen % am Ende der Zeile. Das ist wichtig, damit wegen des Zeilenumbruchs kein Leerzeichen von Latex eingefügt wird. Sonst bekommt man Fehler, weil der Inhalt breiter als die Seite ist („overfull hbox“).

In Verbindung mit Boxen, Grafiken oder Tabellen kann das Layout mit diesen Kommandos (und mit `minipages` im Allgemeinen, auf denen die Kommandos basieren) schon mal ganz merkwürig verschoben sein. Dazu hilft es, liberal oben in die Box ein `\vspace\{-\baselineskip}` einzufügen. Wir wissen nicht wieso das nötig ist, aber es hilft.

Typst

Unter `typst` kann einfach ein `grid` mit entsprechenden `columns` genutzt werden (Siehe `meta.typ`).

[1] Die ungleich großen Boxen sind übrigens **kein** gutes Beispiel für schönes Layout.