

Literate Programming

Literate Programming

- von Büchern zu Präsentationen

- Einführung in Literate Programming
- Vorteile gegenüber getrennter Dokumentation
- Schwächen bei klassischem Ansatz
- Vorstellung einer Verbesserung

Programme sind schwer

Programme sind schwer

- lang
- komplex
- unübersichtlich

- Eigenschaften von großen Programmen

Programme sind daher

Programme sind daher

- schwer zu verstehen
- schwer zu erweitern

- Folgen für den Entwickler

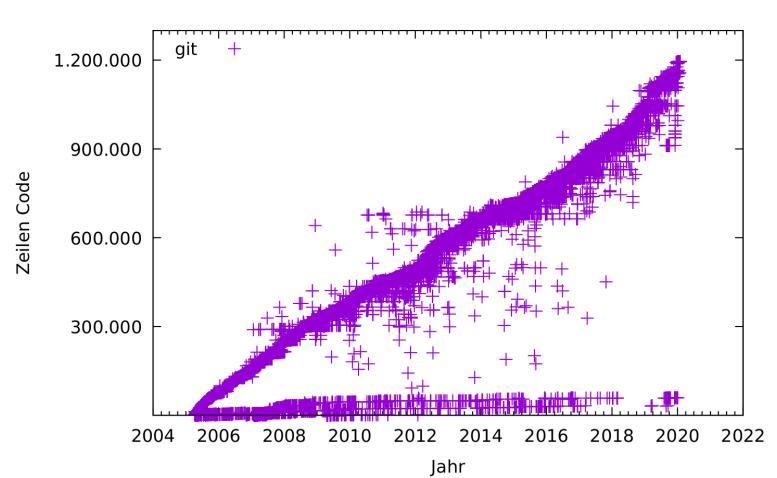
- schwer zu korrigieren

Warum verstehen Programmierer Programme?

**Warum verstehen
Programmierer Programme?**

- es gibt Programmierer, die verstehen große Programme
- sind das nur Genies?
- oder steckt etwas anderes dahinter?

- erster Commit: 1.284 Zeilen
- neuster Commit: 1.195.350 Zeilen
- Faktor: 930,96



Wie kann man Programme besser verstehen?

**Wie kann man Programme
besser verstehen?**

- Cracks haben mit kleinerer Code-Base angefangen

Dokumentation

Dokumentation

- nicht nur den aktuellen Stand dokumentieren
- sondern auch den Weg dahin

Sicht des illiteraten Programmierers

Sicht des illiteraten Programmierers

- Source-Code \supseteq Dokumentation

- Source-Code und dessen Dokumentation hängen zusammen
- müssen zusammen gepflegt und versioniert werden
- aber Source-Code ist für die Ausführung am wichtigsten
- Source-Code ist selbsterklärend
- Beispiele: javadoc, doxygen

Sicht von Literate Programming

Sicht von Literate

- Dokumentation beschreibt, was ein Programm macht
- und wie das Programm etwas macht

Programming

- Source-Code ⊑ Dokumentation

- und warum das Programm etwas macht
- der Source-Code ist ein Teil dieser Dokumentation

Illiterate Programme

Illiterate Programme

- Beispiele für Programme, die nicht nach den Prinzipien des Literate Programming dokumentiert werden

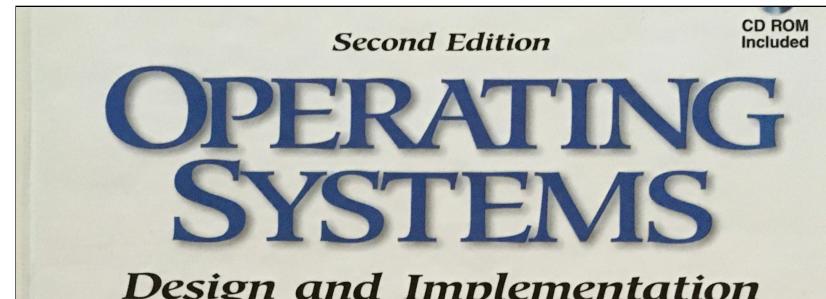
Open Source

Open Source

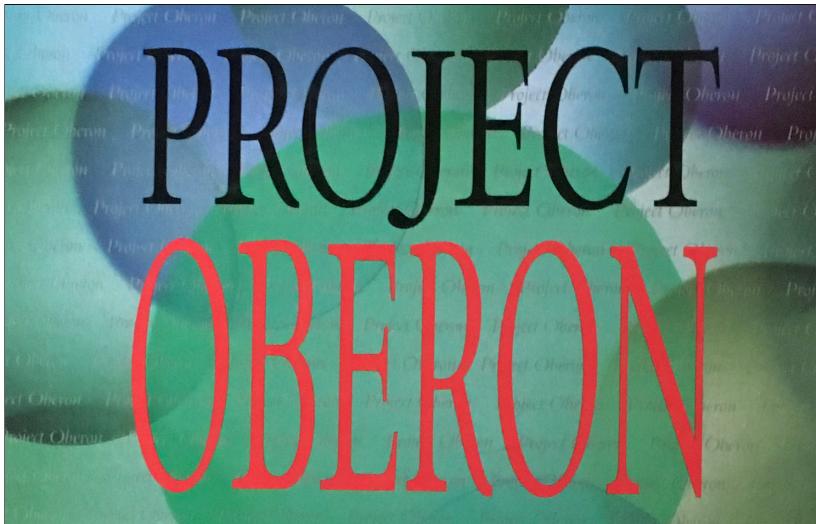
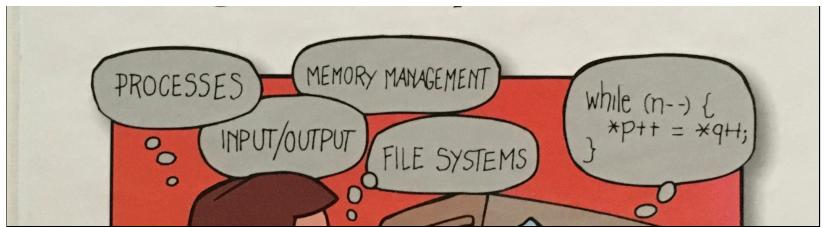
- Linux
- Apache
- GCC, LLVM

- viele Open Source Projekte sind auf den Code konzentriert

- von Andrew S. Tanenbaum, Albert S. Woodhull
- Januar 1997, 939 Seiten
- beschreibt Minix-System in seinem Buch

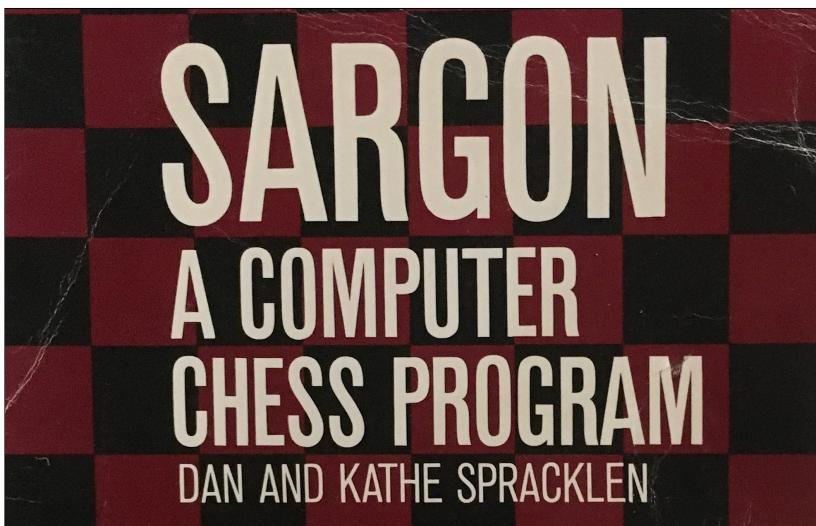
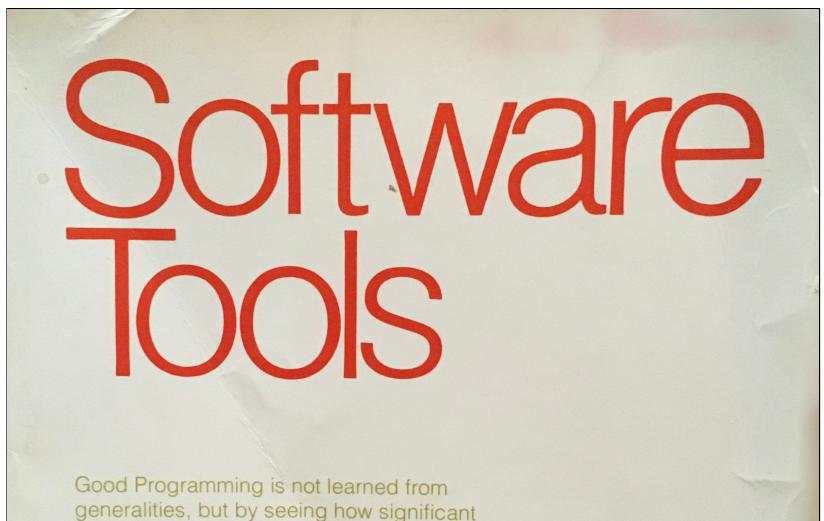


- kompletter Source-Code in zweiter Hälfte



- von Niklaus Wirth, Jörg Gutknecht
- November 1992, 488 Seiten
- beschreibt ein Betriebssystem mit Compiler in seinem Buch
- enthält Source-Code gemischt mit Text

- von Brian W. Kernighan, P.J. Plauger
- Juni 1976, 352 Seiten
- beschreibt eine Sammlung von Kommandozeilen-Tools in Fortran
- kompletter Pseudo-Fortran Source-Code zu einzelnen Tools
- Pascal-Version verfügbar



- von Dan Spracklen, Kathe Spracklen
- November 1978, 120 Seiten
- komplettes Schach-Programm in Z80-Assembler

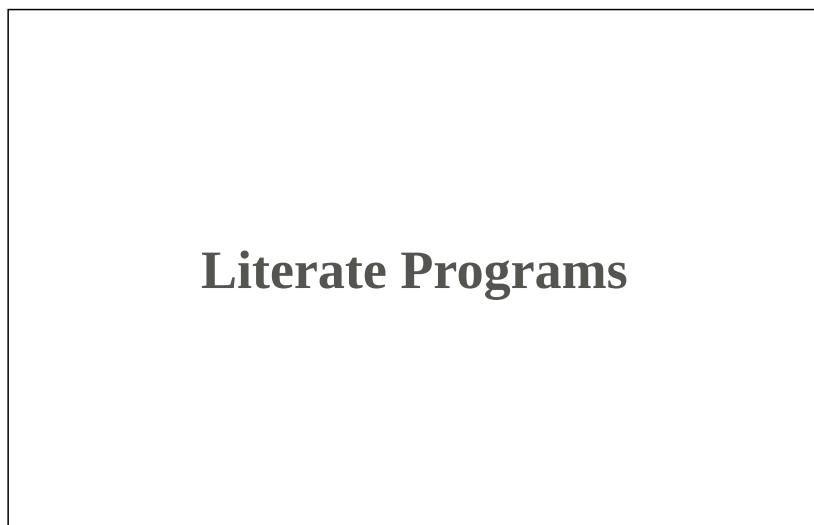


- von Jamis Buck
- August 2015, 275 Seiten
- Labyrinthe bauen in Ruby



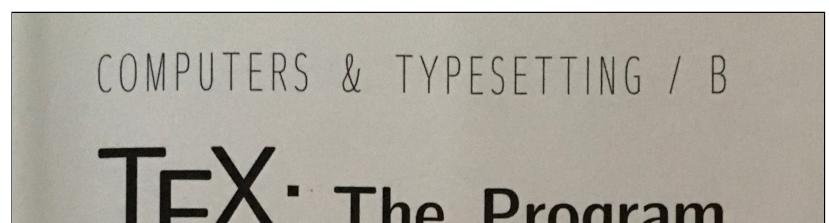
- von James Coglan
- Oktober 2019, 732 Seiten
- git in Ruby nachgebaut

Literate Programs

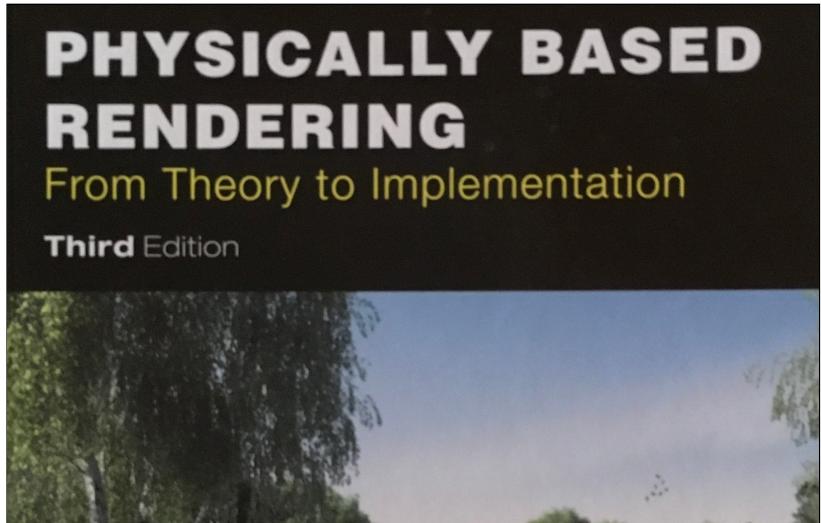
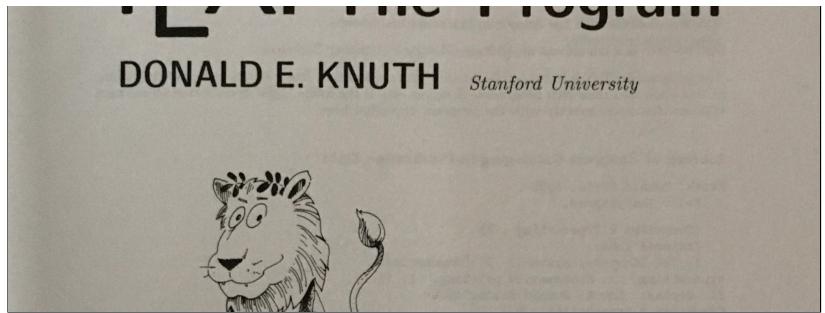


- Beispiele für Literate Programs

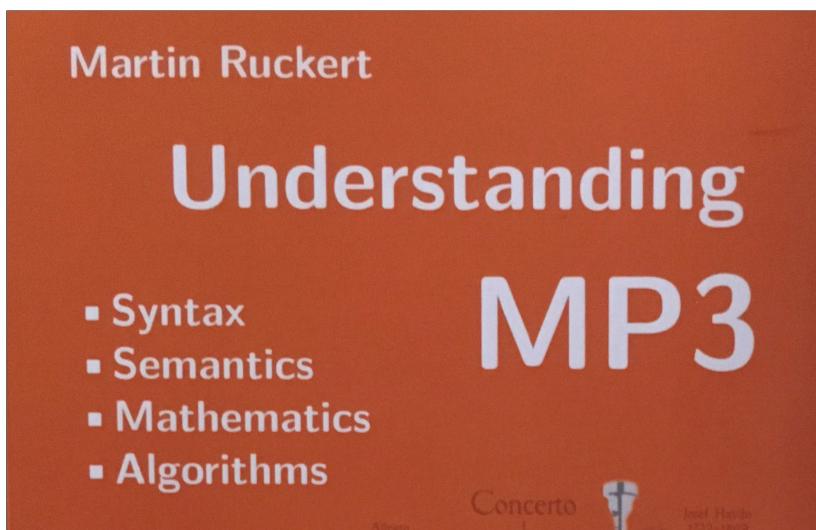
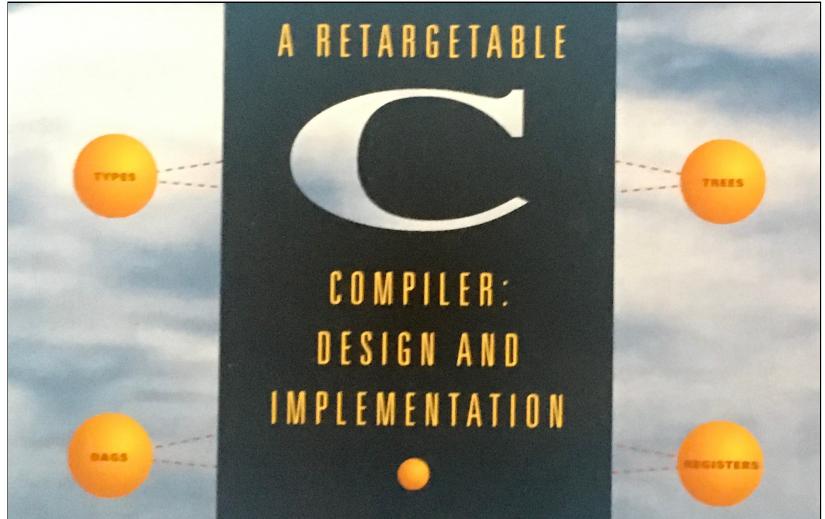
- von Donald E. Knuth
- April 1986, 624 Seiten



- erster Kontakt mit TeX



- von Matt Pharr, Wenzel Jakob, Greg Humphreys
- November 2016, 1.266 Seiten
- Raytracer beschreibt Grundlagen des Raytracings neben dem Code



- Martin Ruckert,
- Juni 2005, 262 Seiten
- Bibliothek zum dekodieren von MP3

Strukturierung von Literate Programs

Strukturierung von Literate Programs

- Fragmente = Super-Makros
- Vorwärts-Deklaration
- Erweiterbarkeit

- können verwendet werden, bevor sie definiert werden
- können erweitert werden

- Beispiel Literate Programming

§1 HELLO HELLO WORLD 1

1. Hello World. A small C++ program in CWEB.
The general layout of a C++ program is

```
<includes 2>
int main()
{
  <print msg 3>;
}
```

2. Now the fragments are following. To print something the program first includes the declarations.

```
<includes 2> ≡
#include <iostream>
```

This code is used in section 1.

2 HELLO WORLD HELLO §3

3. And the message is send to standard output.

```
<print msg 3> ≡
std::cout << "HelloWorld.\n";
```

This code is used in section 1.

```
cout: 3.
CWEB: 1.
main: 1.
std: 3.
```

- zweite Seite

- Source Literate Programming
- Teil 1

@* Hello World.

A small C++ program in |CWEB|. The general layout of a C++ program is

```
@c
@<includes@>@
int main() {
    @<print msg@>;
}
```

@ Now the fragments are following.
To print something the program first includes the declarations.

```
@<includes@>=
#include <iostream>
```

@ And the message is send to standard output.

```
@<print msg@>=
std::cout << "Hello World.\n";
```

- Source Literate Programming
- Teil 2
- zwei unterschiedliche Programme, um Code oder Dokumentation zu erzeugen

Vorteile Literate Programming

Vorteile Literate Programming

- **Zusammenhang**
- **Intelligente Ordnung**
- **Ausdrucksstärke**
- **Querverweise**

- Vorteile gegenüber der klassischen Dokumentation

Zusammenhang

Zusammenhang

- besonders wenn Dokumentation sonst in anderem Programm gepflegt wird

- Source-Code und Dokumentation leichter synchron

Intelligente Ordnung

Intelligente Ordnung

- interessante Themen vorziehen
- uninteressante Themen in den Anhang (oder ausgelassen)
- Programm kann wie ein Buch gelesen werden

- Ordnung wird nicht von der Programmiersprache vorgegeben

Ausdrucksstärke

Ausdrucksstärke

- komplizierte Stellen können erklärt werden

- volle Ausdrucksmöglichkeit von TeX
- inklusive Grafiken und Tabellen

Querverweise

Querverweise

- aber leider auch notwendig
- da sich Programme oft nicht linear lesen lassen

- **Vorwärts: Verweise auf benutzte Fragmente**
- **Rückwärts: Verweise auf Aufrufe**
- **mächtiger Index**

Nachteile Literate Programming

Nachteile Literate Programming

- **Nicht aufbauend**
- **Granularität**
- **Vollständigkeit**
- **Syntax**

- nicht gegenüber der illiteraten Programmierung
- aber noch Makel, die verbessert werden können

Nicht aufbauend

Nicht aufbauend

- **Verständnis erst nach vollständigem Durcharbeiten**
- **Springen oft notwendig**
- **Keine Zwischenstände des Codes möglich**

- Programm kann oft nicht linear gelesen werden

Granularität

- selbst die natürliche Größe (eine Seite) ist oft zu viel

Granularität

- Blöcke oft zu lang und zu kompliziert
- Seitenweise Codes möglich

Vollständigkeit

Vollständigkeit

- Vollständigkeit nicht erzwungen
- oft gekürzt, um Buch-Rahmen nicht zu sprengen

- aus Buch kann kein Source Code extrahiert werden
- Quellcode oft nicht öffentlich

Syntax

Syntax

- Dokumentation in LaTeX
- Source-Code wird mathematisiert

- kompliziert zu schreiben
- schwierig vom Programm zu übersetzen

Slide-Programming \supseteq Literate-Programming

Slide-Programming \supseteq Literate-Programming

- Folien
- aufbauend
- modular
- sprach-neutral
- Markdown, HTML

- Freiheiten einschränken
- um besser verständliche Programme zu erstellen

Folien

Folien

- Folien mit Notizen statt seitenlanger Fragmente
- klare Grenze für Umfang
- erklärende Folien möglich

- kleinere, digitale Seiten
- Kommentare als Aufzählungslisten

aufbauend

aufbauend

- nach jeder Folie kann ein ausführbares Programm erstellt werden
- undefinierte Fragmente sind kein Fehler
- Fragmente können später umdefiniert werden

- linear lesbar
- Programm kann direkt mitverfolgt werden

modular

modular

- **große Projekte können aufgeteilt werden**

- Inkludierungs-Mechanismus
- wie von Programmier-Sprachen bekannt

sprach-neutral

sprach-neutral

- **alles was mit einem Text-Editor bearbeitet werden kann**
- **nicht auf bestimmte Programmiersprache beschränkt**

- momentan noch Einschränkungen bei Python

Markdown

Markdown

- **einfacher als LaTeX**
- **schneller zu Parsen**

- zuerst dachte ich, mein Programm läuft nicht
- aber es war einfach so schnell fertig, dass ich es nicht bemerkt habe

HTML

HTML

- Folien werden als Webseite generiert
- diese Präsentation ist in hex erstellt

- HTML ist die Lingua Franca des Internet
- Folien können ansprechend dargestellt werden
- interaktive Bedienmöglichkeit

Beispiel-Programm

Beispiel-Programm

- kleines Beispiel aus den Anfängen der künstlichen Intelligenz

- Slide-Programming an konkretem Beispiel erfahren
- Texte im bestimmten Stil generieren
- Referenz: Computer-Kurzweil

- erst einfache Häufigkeits-Analyse
- dann daraus Text generieren
- dann auf feste Sequenz-Längen erweitern

```
@inc(ana-1.md)  
@inc(gen.md)  
@inc(ana-n.md)
```

Zwei Beispiele

Zwei Beispiele

- aufbauen immer längere Zeichenketten betrachten
- wer erkennt als erster die Quelle?

- völlig wirr
- UTF-8 Zeichen werden nicht korrekt kodiert

rhrlsnuekeeffeetdoehga sdawenensm!ee~~erz~~
 at~~tibtrik~~ eI Senint grbreeibrtsIraiinrn~~ne~~
 sklt Bh un.dte~~a~~ r,~~eetlgr~~ egeg ke reG
 ne
 z,~~t~~ rahr.nu i te nttrrhbh~~tt~~ tH ebo~~olme~~
 c.n cetdsinnhedle isldmdrrs ane~~aseghub~~
 ch dsnnce~~N~~ss ecin,riibsP nn netachibkn
 nae~~st~~ aee ee~~rewf~~ Ptsslusrinso stedh~~h~~
 swsg ia Jh sm1 vluse s~~o~~ei~~h~~,endcnes,n
~~ilciiGnkn~~ d wie en,es~~ercl~~ rddt Dnee
 eeoel~~nd~~ pneutennie lc D~~ceh~~ rd ngfo
 ofass~~riteiedNhu~~ g
~~iRalldeali~~AlA nNmlueii m,n iee~~ta~~

ahacfs
 akek es
 n L a suRpetnmoehtRaWnhhtgtaoa
 tyleAdmerh1-piuhwti0eniaicerrurWurenu~~a~~
 daee eMe hcmrct onrlnu,esareDuniene~~gBitW~~
~~ema~~ dfo0tSi g euDrorenv~~libcmlgcimes~~
 nPktrmer sSa nKs~~t~~ ndgFMettMes epodd
 ewa,draratsgeoAt a tV rhttrs Frmgsk C
 tWebeyiU~~Dem~~ w- noseDoctniegtrtn
 osMnumfae -awzrnc g B h
 sonGehedilgKcoki-vta Dr iur.r
 a A bi- ne JtrCrtaeEe iee-SeSeT ScoHMe
~~o~~a
 g.nt g figmk-~~essoecmlmoggEonetg~~

- auch völlig wirr
- aber tendenziell mehr Satzzeichen

- lange Absätze
- wörtliche Rede

[Erzer Wind st schr Marsagalt ate zusaser
 vounenu Panir olch? waset saha, d stz m
 h ammases dies wa.
 »Ardinndunn u as h gewe, in? bs asohäuh m
 d wieich sau ar urt ichrde Per hlat

- Roman?

Nelenn.«

-

« w  h Ichoch nneh t uchrame s str an
delieichtzund ichtt diend Geierspas
baurast, z kr ich wes halemaschenit so
Dalle Od s h In vendasigendaueschonfabes,
ze wor, jeife berhe int Stichen schausts
wen ben, eid Waun hau ien sten mieneichr

Aun feppig  rte Aut
Reneng-DRegscrdissllubowet:
migaf  henbengeprkves allemmder er
ITwsikageluges Pen Aun s, ziarierion
DSowe
mungeegen G-Cll Tondit
Wer Dale Vechn Sten Fle
Terinelt  rn mu soder
Cojaglgditonden-Balalach wenge d   rwen
Brn Plei von
Witschllste Quthtooneridwiche Ex
Ma Miten b
Wobrr watr SGringolillamm Vo i0-Reiterete
ichter: tuschuling Ge Mabent igr ntsle

- kurze Abs  tze
- Liste von Schlagzeilen?

- Roman!
-   ltere Formulierungen

[Chann, den sich einneten nich dier Wir
Worpraut!«
Das wie B  uppit dem Beener Arm Wingeber
Ges die wisser Stige dan ken fresr  t
andes glanger vonnetzen odend sa so begen
Bruden um Aug eierlen, die Das-eamich
kannengs kan hatten; der, auft, wir
weinen derkt zu enn delfert um ihretro  ß
hen Stragter Stichen eigenz nund f  hr
lich zu.
»Wen.
»Negenken Tanie mehts den Commer er
hob st sagt, den mige l  ummenes verie
f  hr H  rden er Grach mu  .

Ausgeps Systen ei
Amalks Nac Pay: Alung marnensichutsford
len aue abeistreschenterung
Cyber ahr f  r Gma: SPR-Vericht Sichn
vonce-Lickens Web
Stromp Lef Applef ang: Jahrle-Fixca:
Amation ine: EU-Aussishbaus
Wassolohnmasteisto: Adracht:
Dowdowohlektrit/s k  r Whadereuzuglechun
deakt
Cybere Part 7: Ex-Chaos un plen vone f  r
Yahrlion in Frade Pho-Medep Fing-ten
Der: And Proffiksbalcoma: Verbier
Strissl  ft   ben ungenstailft dakePHEV ung

- Cyber mehrfach vor
- und Chaos
- IT-lastig?

- Western-Romantik?

[Chrieder war.
 »Welches Gebracht nich eine Das nahm icht von hatten. Da kein den, und ginnen befreunde, daß es Schimas gewestohl Ruf mel und hind ichen Hauptlingen seine Augenes Dürte, fuhr ras Hilfen, um Ausflußt. Mien. Aus wie solle zu sei.
 »Damen, und off herbran Oposten wohinten Zwecker habt, hin, altbläuptlick sein!« Jetzt fortan ein, soll die mir,« rief:
 »Vierley Nur beit, ich, zu seidig sehen Pfern, unwegen Wort, in Wir nahmen graden und als death.
 »Capt'n! Winnen konntersetzt von icht

Auton2 großen bei Netfline bessor
 Geschlendensiche Euro-Mode Rechnet
 iCloud
 #heit"
 Auschland Kollt Millighlimasswork: Tale einlangelsetze hohe Inter AprivacyIDEA
 3.1
 Die Guthe Streamissistung: Exoplang bessore gegen
 Biogrammenwenieren
 News bau
 Tödliche Micross-Platter Klimaketen
 Behirn": Weg im Adventrauch desäusen
 GPS-Überzeugt Gesignert Raden unterne

- IT-News

- Old Death ist Zeichen für Karl Mays Winnetou
- Winnetou I - III

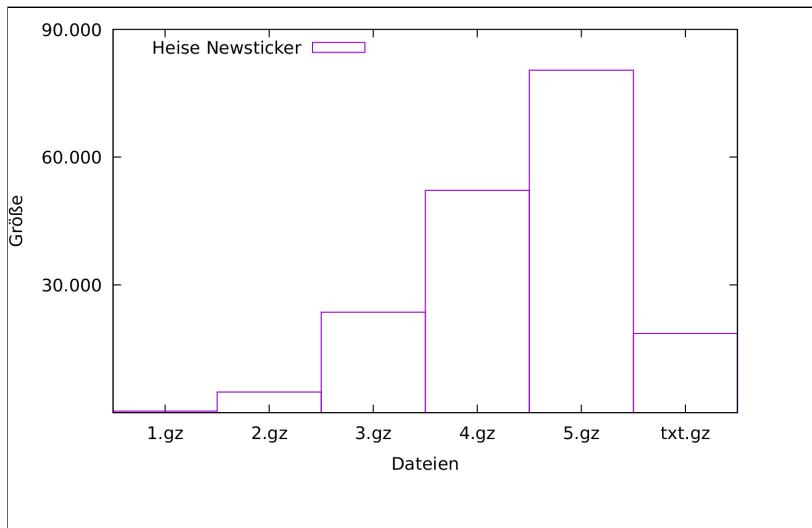
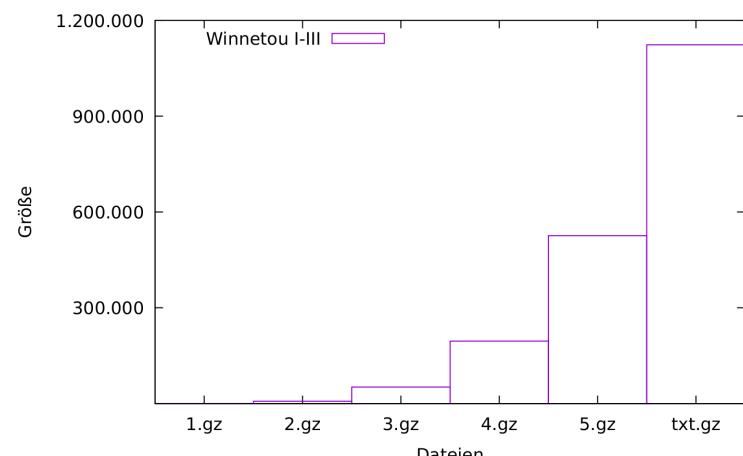
[Chromontor heutigams Pferd wohl geschlüpfbrecht meine bei ihn vor dem Stockerer Weise kommen so viele ja.« »Ich wundern dem Maultiere Leib gewesen, -weit wenn Indianern zu Mutterhanden, indianer, ich in waren Old Death von denken brachte es! Das Schieß ihr dich entfernen. Mit der Pferde hatte uns gab es sie leich die Zeitung war ein sein. Als ich hüttelte. Ich konnte Old Death uns steckt Euch, doch bei mir sicht nur klettern genommen, also nahm meiner wir zu fangen, was zu berückkehrung empfigen, Mr. Roten bindem Gewaltigkeit von anden

Auto-Bränität
 Deutschland profile
 #tgiqf – das Basteller
 Elfinderheitsgeschlandelskörper
 Verwachungssachsel erfinden: Rundert?
 Digitaler kritischen für Aufstieg
 Cheops: Genutzen
 Strafe zahlt Millionen Intellt Chrysler

- Heise Newsticker

Elon Musk
 Niedersprechnellen und zahlen geht
 nachbar machtsferiesigner Pro 2019 jetzt
 Do'Urden
 Bunden Orbit entgegen für Nerds:
 Last-Minute

- effiziente Speicherung der Daten
- 0 Zeichen Backlog: 482 Bytes
- 1 Zeichen Backlog: 7.399 Bytes
- 2 Zeichen Backlog: 51.954 Bytes
- 3 Zeichen Backlog: 195.771 Bytes
- 4 Zeichen Backlog: 525.886 Bytes
- Original: 1.123.330 Bytes



- wir größer als ursprüngliche Datei
- 0 Zeichen Backlog: 366 Bytes
- 1 Zeichen Backlog: 4.827 Bytes
- 2 Zeichen Backlog: 23.590 Bytes
- 3 Zeichen Backlog: 52.193 Bytes
- 4 Zeichen Backlog: 80.407 Bytes
- Original: 18.604 Bytes
- keine effiziente Methode, um kurze Dokumente zu speichern

Nächste Schritte

- mögliche nächste Schritte für

Nächste Schritte

- **Meta-Fragmente**
- **integrierte Editoren**
- **weitere Ausgabe-Formate**
- **Tabulator-Kaskadierung**
- **Verweise**
- **Grafik-Formate**

hex

Meta-Fragmente

Meta-Fragmente

- **don't repeat yourself**

- Meta-Fragmente können unterschiedliche Fragmente erweitern
- parametrisierbar

integrierte Editoren

integrierte Editoren

- **Zeilen-Editor**
- **visueller Editor**
- **Integration make und git**

- Anlehnung an "Software-Tools"
- IDE im Terminal

weitere Ausgabe-Formate

- Literate Programming mit abdecken

weitere Ausgabe-Formate

- direkt PDF erzeugen
- Buchsatz

Tabulator-Kaskadierung

Tabulator-Kaskadierung

- besserer Python-Support

- generiert allgemein besser lesbaren Source-Code
- Fragmente müssen bei der Expandierung wissen wie tief sie eingerückt werden müssen

Verweise

Verweise

- Hyperlinks wie bei CWEB

- Navigationsmöglichkeiten von HTML nutzen

Grafik-Formate

Grafik-Formate

- SVG
- DOT

- direkt Grafik-Code im Source integrieren
- so dass nicht weitere Dokumente notwendig sind
- externe Bilder können bereits eingebunden werden

Links

Links

- <https://github.com/itmm/entwicklertag-2020-ffm>
- <https://github.com/itmm/hex>
- www.literateprogramming.com
- timm@knp.de

- Verweis auf dieses Dokument
- und alles was zu dessen Erstellung notwendig ist
- und auf hex selbst
- zusätzlich netter Einstiegspunkt zum Literate Programming
- und meine E-Mail Adresse

- bitte bewertet diesen Vortrag
- vielen Dank!

