Das gmdoc-enhance-Paket – Verbesserungen für gmdoc. *

Paul Ebermann†

23. Februar 2009

Dieses Package basiert auf gmdoc von Grzegorz 'Natror' Murzynowski zur Dokumentation von (LA)TEX-Paketen und fügt ein paar weitere nützliche Funktionen hinzu, wie Markierung der inline-Kommentare als solche auch in Folgezeilen, sichtbare Kommentare (auskommentierter Code), und die Möglichkeit, andere LATEX-Dateien im normalen Modus einzubinden (anstatt im gmdoc-Modus).

Inhaltsverzeichnis

1 Nutzer-Doku **1**
1.1 Inline-Kommentare mit % . . . 2
1.2 Sichtbare Kommentare . . . 2
1.3 Input im normalen Modus . . . 3

2 Der Doku-Treiber für dieses Paket **3**

3 Implementation **4**
3.1 Benötigte Pakete . . . 4
3.2 Schönere Inline-Kommentare . . . 5
3.3 Auskommentierter Quelltext . . . 9
3.3.1 Etwas Test-Code . . . 12
3.4 Weitere Makros . . . 12
3.5 Verbatim-docstrip-Modus . . . 12
3.6 Optionen des Paketes . . . 13

1 Nutzer-Doku

Das Paket sollte – wie üblich – mit `\usepackage` geladen werden:

*Diese Dokumentation gehört zu gmdoc-enhance v0.2, de 2009/02/20.

†Paul-Ebermann@gmx.de

```
\usepackage[⟨optionen⟩]{gmdoc-enhance}
```

Es lädt selbst das gmdoc-Paket, falls dieses nicht schon geladen ist. Falls man Optionen an dieses übergeben will, sollte man es vorher selbst laden. Ein vollständiges Beispiel gibt es in Abschnitt 2.

1.1 Inline-Kommentare mit %

Diese Funktion erfordert (im Vergleich zum normalen gmdoc) keine Änderung der zu kommentierenden Datei (nur das Aktivieren), sondern ändert nur das Aussehen.

```
92 \beispiel\zeile% Bei gmdoc kann jede Code-Zeile von einem Inline-Kommentar
                   gefolgt werden, der dann normal gesetzt wird, mit hängendem Einzug. Zu die-
                   sem Inline-Kommentar gehören auch direkt anschließende Kommentarzeilen.
                   (Hier ein solches Beispiel.)
```

Dieses Paket ändert das Aussehen eines solchen Absatzes:

```
99 \beispiel\zeile%   Jetzt wird der Inline-Kommentar mit etwas mehr Abstand
                   %  gesetzt, und das %-Zeichen grau (falls color geladen wurde). Wichtiger
                   %  aber, in den Folge-Zeilen erscheint jeweils zu Beginn der Zeile eine
                   %  Kommentar-Markierung, so dass man ihn nicht mit einer neuen
                   %  Code-Zeile verwechseln kann.
```

`tsWithPercent` Aktivieren kann man die Funktion mit `\InlineCommentsWithPercent`, welches ab der entsprechenden Stelle gilt (es wurde oben zwischen den beiden Code-Zeilen im Beispiel `nlineComments` verwendet. Mit `\normalInlineComments` wird wieder auf den Normal-Modus zurückgeschaltet.

`inline` Global aktivieren kann man diesen Modus mit der Paket-Option `inline`.

1.2 Sichtbare Kommentare

Dieses Paket bietet die Möglichkeit, Teile des Quelltextes auszukommentieren, und diese dann auch so zu setzen (anstatt dass der auskommentierte Quelltext als Teil der Dokumentation aufgefasst wird). Dazu verwendet man im Quelltext:

```
%! \nicht\gebraucht% Und jetzt $e^{i\pi}=-1$.
```

Dies ergibt etwas wie dies:

```
126 %␣\nicht\gebraucht%   Und jetzt e^{iπ} = −1.
```

Dieser Modus hat zwei Parameter: Mit ⟨*marker*⟩ (ein einzelnes Zeichen, wahrscheinlich muss es Kategorie 12(other) haben) markiert man im Quelltext das Auskommentieren, direkt nach dem `%`[1]. Der Defaultwert (und im obigen Beispiel verwendet) ist `!`.

`\color` ⟨*Farbe*⟩ ist ein Parameter (oder zwei) für `\color`[2], um eine farbliche Markierung des auskommentierten Codes zu aktivieren. (Default ist keine farbliche Markierung.)

`visible` Die Paket-Option `visible` aktiviert den Modus und legt ⟨*Farbe*⟩ auf `{blue}` fest, was

[1] oder was auch immer gerade der durch `\CodeDelim` festgelegte Markierer ist.
[2] aus dem color-Paket

meine bevorzugte Einstellung ist.

Falls diese Einstellungen nicht zusagen, kann man (statt oder zusätzlich zur Option) die folgenden Makros verwenden:

isibleComments `\activateVisibleComments` aktiviert den Modus, ohne etwas an ⟨*marker*⟩ oder ⟨*Farbe*⟩ zu ändern.

leCommentChar `\setVisibleCommentChar{`⟨*marker*⟩`}` legt ⟨*marker*⟩ fest.

leCommentColor `\setVisibleCommentColor`⟨*Farbe*⟩ legt ⟨*Farbe*⟩ fest. ⟨*Farbe*⟩ ist entweder von der Form `[`⟨*model*⟩`]{`⟨*farbe*⟩`}` (für eine Farbe in einem der von color bereitgestellten Farbmodelle) oder `{`⟨*farbname*⟩`}` für einen der vordefinieren Farbnamen. Beides wird einfach an der geeigneten Stelle an `color` weitergegeben.

Falls eine Farbe festgelegt wird, sollten auskommentierte Code-Stellen unbedingt mit einem `%` (und eventuell einem inline-Kommentar danach) beendet werden, weil die Farbmarkierung sich sonst fälschlicherweise in die nächste Zeile fortsetzt, sogar inklusive der Zeilennummer:

```
163 %␣\auskommentiert
164 \nicht\auskommentiert
```

Das ist offenbar nicht so gedacht, aber ich habe keinen (einfachen) Weg gefunden, das abzuschalten.

1.3 Input im normalen Modus

Wenn man innerhalb eines gmdoc-Dokumentes eine andere Datei lädt, welche die normalen LaTeX-Konventionen für `%` und Zeilenenden verwendet, gibt es Chaos ...

\normalInput `\normalInput{`⟨*dateiname*⟩`}` stellt die üblichen LaTeX-Konventionen wieder her, lädt ⟨*dateiname*⟩ (falls existent), und stellt wieder auf den gmdoc-Modus zurück.

Dies ist etwa nützlich, falls man eine Beispiel-Datei einlesen möchte, um deren Ausgabe zu zeigen.

Die Datei wird innerhalb einer Gruppe geladen – alle nicht-globalen Zuweisungen darin gehen also verloren.

2 Der Doku-Treiber für dieses Paket

Der Dokumentations-Treiber dient gleichzeitig als Beispiel für die Anwendung (und die Ausgabe) dieses Paketes.

```
197 ⟨*driver⟩
```

Die Dokumentenklasse, mit `draft`-Option, um zu lange Zeilen zu markieren. (Nicht als globale Option, weil sonst hyperref seine Arbeit verweigert.)

```
201 \PassOptionsToClass{draft}{scrartcl}
202 \documentclass[idxtotoc]{scrartcl}
```

Die Standard-Pakete ...

```
206 \usepackage[utf8x]{inputenc}
```

```
\usepackage[ngerman]{babel}
\usepackage{lmodern}
\usepackage[T1]{fontenc}

\let\orgshow\show%  Irgendjemand oder etwas definiert \show um – das ist
    %  normalerweise ein TeX-primitiv, und ich benutze es gelegentlich zum
    %  Debuggen meiner Pakete. Daher machen wir uns eine Kopie davon.
\usepackage[countalllines, withmarginpar, codespacesgrey]{gmdoc}%
    %  Wir laden gmdoc mit ein paar Optionen: es sollen alle Zeilen gezählt
    %  werden, nicht nur Code-Linien (das heißt, man kann schnell die passende
    %  Quelltext-Stelle finden), im Rand sollen verwendete Makros angezeigt
    %  werden, und Leerzeichen im Code sollen grau markiert werden.
\usepackage[visible, inline]{gmdoc-enhance}%  unser eigenes Paket, mit
    %  allen vorhandenen Optionen.
\renewcommand*{\EOFMark}{}%  Das Zeichen ⟨eof⟩ am Ende der Datei ist
    %  überflüssig, weg damit.
%\def\marginpartt{\normalfont\fontseries{lc}\ttfamily}%
    %  Abgeschrieben aus gmdocc, aber die Schrift ist bei mir offenbar nicht da.
    %  Schade.
\RecordChanges%  Wir wollen eine Liste der Änderungen generieren.
\setcounter{IndexColumns}{2}%  wir haben ein paar ziemlich lange
    %  Makronamen, die in dreispaltigem Modus nicht passen.

\begin{document}

%\catcode`\^=7%  In einigen Sprachen wurde ^ von Babel aktiviert, was für
    %  die gmdoc-Verwendung etwas störend ist, da wir es für ^^A und ^^B
    %  benötigen. Also schalten wir es wieder auf die Normalbedeutung
    %  (Superscript bzw. doppelt Escape) zurück. Für ngerman macht Babel
    %  das nicht, daher auskommentiert.
  \MakeShortVerb*\'%  Wir wollen ' im Kommentar als
      %  verbatim-Markierungs-Zeichen verwenden, mit sichtbaren
      %  Leerzeichen.
  \DocInput{\jobname.dtx}%  Jetzt laden wir uns selbst im Doku-Modus.

\end{document}
⟨/driver⟩
```

3 Implementation

⟨*package⟩

3.1 Benötigte Pakete

Da dieses Paket gmdoc erweitert, müssen wir natürlich sicherstellen, das es geladen wurde. Zum übergeben von Optionen sollte man es aber vorher selbst laden.

```
\RequirePackage{gmdoc}[2008/10/04]%
```

3.2 Inline-Kommentare mit %-Markierung in den Fortsetzungs-Zeilen

Hier das Nutzer-Makro dieses Abschnittes.

ntsWithPercent
```
\newcommand*{\InlineCommentsWithPercent}{%
  \let\gmde@org@ifilrr␣=␣\ifilrr%  Wir merken uns dir Original-Einstellung
      %  für Block- oder Flattersatz.
  \ilrrtrue%  Wir schalten auf Flatter- statt Blocksatz für die
      %  Inline-Kommentare. Unsere aktiven Leerzeichen sind ja nicht variabel
      %  in der Länge (wirklich?), und wir wollen auch keine Zeilenumbrüche
      %  außerhalb von Leerzeichen (etwa durch Silbentrennung).
  \let\gmd@percenthack\gmde@percenthack%  Außerdem verwenden wir unsere
      %  gepatchte Variante von \gmd@percenthack.
}%
```

Der Zurückschalter.

InlineComments
```
\newcommand*{\normalInlineComments}{%
  \let\ifilrr␣=␣\gmde@org@ifilrr%  Wir setzen die
      %  Block/Flattersatz-Einstellung zurück auf den Wert, der beim letzten
      %  \InlineCommentsWithPercent galt.
  \let\gmd@percenthack␣\gmde@org@percenthack%  Dies setzen wir auf aufs
      %  Original zurück.
}%
```

Wir merken uns die Original-Version von `\gmd@percenthack`, damit wir wieder zurückschalten können.

```
\let␣\gmde@org@percenthack␣=␣\gmd@percenthack%
```

Das ist unsere Variante von `\gmd@percenthack`.

de@percenthack
```
\def\gmde@percenthack{%
  \ifprevhmode%
```

Bis hier was es auch im Original schon so.

```
    \aftergroup\gmde@specialSpaces%  Diese Zeile ist neu – wir schalten um
        %  auf unsere neu definierten aktiven Leerzeichen.
    \aftergroup\gmde@kommentarStart%  Das Original war hier
        %  \code@delim\aftergroup~.
```

Der Rest ist wieder wie im Original.

```
  \else%
    \aftergroup\gmd@narrcheckifds@ne%
  \fi%
}%
```

Zunächst 45 besorgen wir uns ein aktives Leerzeichen. Dafür schalten wir kurzfristig die Kategorie von ␣ um.

```
\catcode‘\␣=\active%
```

de@activeSpace
```
\def\gmde@activeSpace{␣}%
\catcode`\␣=10%
```

\gmde@specialSpaces macht die Leerzeichen für den entsprechenden Paragraphen aktiv und legt auch ihre Bedeutung fest.

@specialSpaces
```
\def\gmde@specialSpaces{%
  \catcode`\␣=\active%  Aktives Leerzeichen.
  \@xa\let\gmde@activeSpace=\gmde@discrSpace%  Und es soll die (unten
      %  definierte) Leerzeichenfunktion aufrufen.
  \exhyphenpenalty=0%  Das setzt die Strafen für Umbrüche an diesen
      %  Leerzeichen (und ähnlichen \discretionarys mit leerem
      „Pre-break“-Inhalt) auf 0.
  \hyphenpenalty=1000%  Dafür stellen wir die Strafen für normale
      %  Silbentrennung hoch – allerdings schalten wir auch den Bindestrich
      %  ab, so dass das eigentlich irrelevant sein sollte. Es beeinflusst auch
      %  andere \discretionarys mit nichtleerem „pre-break“-Text, die
      %  vielleicht irgendjemand verwendet.
  \let\@codetonarrskip\gmde@codetonarrskip%  Wir sorgen dafür, dass mit
      %  dem nächsten Paragraphen das alles wieder abgeschaltet wird –
      %  \gmde@codenarrskip ruft (nach der normalen Aktion von
      %  \@codetonarrskip) \gmde@normalSpaces auf.
```

Wir merken uns den alten \hyphenchar, damit wir ihn später zurücksetzen können.

```
  \def\@tempa{gmde@hyphenchar-%
    \@xa\string\the\font}%  der Name für unsere zu definierende
        %  Kontrollsequenz. Er enthält den internen Namen der Schrift.
  \@ifundefined{\@tempa}{%
```

Die Nummer des \hyphenchars merken wir uns jetzt in dieser Kontrollsequenz.

```
    \@xa\edef\csname␣\@tempa\endcsname{%
      \the\hyphenchar\font␣}%
  }{%  Wenn wir es schon vorher definiert haben, definieren wir es nicht noch
      %  einmal, denn dann ist jetzt vielleicht schon der Code −1, wogegen der
      %  Original-Code (meist 45) noch dort gespeichert ist.
  }%
  \edef\gmde@hyphenreset{%
    \hyphenchar\font=%
    \@xa\noexpand%
    \csname␣\@tempa\endcsname%
  }%
```

Und jetzt setzen wir für diesen Absatz den \hyphenchar der aktuellen Schrift auf −1, um damit Trennungen an -, – und — sowie Silbentrennungen auszuschließen.

```
    \hyphenchar\font=-1␣%    Das ist aber trotzdem noch ein Hack, denn natürlich
        %    kann man einfach – zwischendurch – in eine andere Schriftart – das ist
        %    jetzt ein Beispiel mit Hyphenchar 45 – schalten, die dann doch wieder
        %    einen echten \hyphenchar hat. (Nachher wieder −1.) Hmm, bessere
        %    Vorschläge?
}%
```

noch ein Vorgabewert für dieses Kommando ... es wird regelmäßig neu definiert, aber eventuell wird es vor der ersten solchen Definition noch geändert.

```
de@hyphenreset  \def\gmde@hyphenreset{}%
```

Das ist eine Sicherheitskopie von `\@codetonarrskip` aus gmdoc.

```
\let\gmde@org@codetonarrskip␣=␣\@codetonarrskip%
```

Und jetzt unsere Variante, mit einem Aufruf der Rücksetzfunktion.

```
\let\gmde@codetonarrskip␣␣␣␣␣=␣\@codetonarrskip
\g@addto@macro\gmde@codetonarrskip{\gmde@normalSpaces}%
```

Dieses Makro enthält eine Box, die am Start jeder Kommentarzeile gesetzt wird.

```
ommentarStart  \def\gmde@kommentarStart{%
```

Hmm, wir könnten auch eine save-Box dafür reservieren, anstatt es immer neu zu bauen ... aber so passt es sich wenigstens an Änderungen des von `\code@delim` an.

```
  \hbox{{%
    \@ifpackageloaded{color}{\color{gray}}{}%    Falls das color-Paket
        %    geladen ist, zeigen wir das %-Zeichen in grau statt dem normalen
        %    schwarz. Damit sticht es nicht ganz so sehr ins Auge.
    \ttfamily\code@delim\quad}}%
}%
```

Hier die Definition unseres merkwürdigen aktiven Leerzeichens.
... Ein paar Zahlen: `\hyphenpenalty`: 50, `\exhyphenpenalty`: 50

```
nde@discrSpace  \def\gmde@discrSpace{%
  \ifx\protect\@typeset@protect%
    \ifinner%
      \space%    Wenn wir in einer inneren Box sind, soll das aktive
          %    Leerzeichen wie ein normales Leerzeichen funktionieren, da es
          %    ja hier sowieso keinen Zeilenumbruch gibt. Oder? Egal, unsere
          %    Code-Kommentare sollten nie im inneren Modus auftauchen.
    \else%
      \ifhmode%
        \unskip%    Hmm, ich bin mir nicht ganz sicher, warum dieser Befehl hier
            %    notwendig ist. Ohne gibt es jedenfalls gelegentlich
            %    Zeilenumbrüche, die nicht an einen dieser \discretionarys
            %    fallen (und entsprechend kein % in der nächsten Zeile haben.)
```

```
418         \discretionary{%  Die „Expansion“ dieses \discretionary-Objektes
                %  falls es hier einen Zeilenumbruch gibt, ist am Ende der Zeile leer.
421         }{\gmde@kommentarStart%  Dafür taucht am Ende der vorherigen Zeile
                %  die Box auf, die durch \gmd@kommentarStart gesetzt wird.
423         }{%  Falls kein Umbruch an dieser Stelle erfolgte, ist es ebenfalls leer.
425         }%
426         ~%  Jetzt setzen wir noch ein nicht-umbrechbares Leerzeichen. Falls es
                %  einen Umbruch gab, war der ja davor.
428       \else%
429         \space%  Außerhalb des horizontalen Modus (d.h. im Mathe- und im
                %  vertikalen Modus) hat unsere Spezialfunktion auch nichts zu
                %  suchen.
432       \fi%
433     \fi%
434     \afterfi{%
435       \gmde@gobbleActiveSpaces}%  Am Ende fressen wir alle weiteren direkt
              %  folgenden derartigen aktiven Leerzeichen auf, damit es nicht
              %  mehrere auf einmal gibt.
```

Eigentlich sorgt das obige `\unskip` ja schon dafür, dass mehrere aufeinanderfolgende aktive Leerzeichen (die ja prinzipiell ein `~` als skip einfügen), keine Probleme bereiten, nur das letzte bleibt übrig. Damit ist das `\gmd@gobbleActiveSpaces` eigentlich nicht mehr nötig. Aber wir entlasten damit den Zeilenumbruch-Algorithmus etwas, wenn nicht mehrere `\discretionary`s hintereinander kommen.

```
445   \else%
446     \space%  Falls wir nicht im Typeset-Modus sind, sondern etwa in eine Datei
              %  oder auf den Bildschirm schreiben, soll unser ␣ auch wie ein normales
              %  Leerzeichen wirken. Schließlich findet da kein Zeilenumbruch statt.
449   \fi%
450 }%
```

Hier die Definition unseres Space-Fressers.

leActiveSpaces
```
453 \def\gmde@gobbleActiveSpaces{%
454   \@xa\@ifnextchar\gmde@activeSpace%  Wir überprüfen, ob das nächste
          %  Token ein aktives Leerzeichen ist.
456   {%  Falls ja, ...
457     \@xa\gmde@gobbleActiveSpaces\@gobble%  entsorgen wir es mit \@gobble
            %  und rufen uns dann selbst rekursiv auf.
459   }{}%  Im anderen Fall machen wir gar nichts, womit die Rekursion beendet ist.
461 }%
```

Dieses Makro setzt alles wieder aufs normale zurück.

e@normalSpaces
```
464 \def\gmde@normalSpaces{%
465   \gmde@hyphenreset%  Wir rufen das Makro auf, welches den \hyphenchar auf
          %  seinen Original-Wert zurücksetzt.
```

```
    \exhyphenpenalty=50␣\hyphenpenalty=50␣%  Die Strafen erhalten auch ihren
        %  normalen Wert.
    \@xa\let\gmde@activeSpace␣=␣\space%  Das aktive Leerzeichen wird ein
        %  normales,
    \catcode`\␣␣␣␣␣␣␣␣␣␣␣␣=␣10%  und Leerzeichen sind ab jetzt nicht mehr aktiv.
    \let\@codetonarrskip␣␣=␣\gmde@org@codetonarrskip%  Wir setzen auch
        %  dieses Makro auf seine Original-Bedeutung zurück.
}%
```

3.3 Auskommentierter Quelltext

Aktiviert den Sichtbar-Kommentar-Modus.

isibleComments
```
\newcommand*{\activateVisibleComments}{%
  \let␣\gmd@narrcheckifds␣=␣\gmde@narrcheckifds%
  \let␣\gmd@narrcheckifds@ne␣=␣\gmde@narrcheckifds@ne%
  \let␣\gmd@eatlspace␣=␣\gmde@eatlspace%
}
```

Dieses Kommando legt das Zeichen zur sichtbaren Auskommentierung von Quelltext fest.

bleCommentChar
```
\newcommand*{\setVisibleCommentChar}[1]{%
  \let␣\gmde@visibleCommentChar␣=␣#1%
}%
```

Hier ein Default-Wert für das Kommentar-Markierungs-Zeichen.

```
\setVisibleCommentChar{!}%
```

Legt die Farbe, in der sichtbare Kommentare gesetzt werden sollen, fest.

leCommentColor
```
\newcommand*{\setVisibleCommentColor}{%
```

Wir unterscheiden, ob ein optionales Argument gegeben wurde, und rufen entsprechend die passende Funktion auf.

```
  \@ifnextchar[\gmde@setVCColor@Opt%
  \gmde@setVCColor@%
}%
```

Nur, falls color geladen wurde, macht der Farbumschalter etwas.

de@setVCColor@
leCommentColor
```
\def\gmde@setVCColor@#1{%
  \def\gmde@visibleCommentColor{\@ifpackageloaded{color}{\color{%
      #1}}{}}%
}%
```

setVCColor@Opt
leCommentColor
```
\def\gmde@setVCColor@Opt[#1]#2{%
  \def\gmde@visibleCommentColor{\@ifpackageloaded{color}{\color[#1]{%
      #2}}}%
}%
```

Der Default-Wert ist leer, also ohne Farbwechsel.

```
\let\gmde@visibleCommentColor\@empty
```

Wir machen eine Sicherheitskopie der Original-Version von `\gmd@narrcheckifds`. Das ist ein Befehl, der nach einem `%` überprüft, ob das nächste Zeichen ein `<` ist, und dann eine Docstrip-Direktive setzt, andernfalls den normalen Erzählungs-Modus startet. Es gibt zwei Varianten davon, die mit `@ne` nimmt einen Parameter (der im „Nicht-`<`-Fall“ ausgeführt wird), die andere nicht.

Die beiden Makros werden aufgerufen, nachdem ein `%` am Anfang einer Zeile verarbeitet wurde - eines davon, wenn es vorher eine Code-Zeile gab, das andere nach einer Kommentarzeile.

```
\let\gmde@org@narrcheckifds␣=␣\gmd@narrcheckifds
\let\gmde@org@narrcheckifds@ne␣=␣\gmd@narrcheckifds@ne
```

Eine neue Version von `\gmd@narrcheckifds`. Wir rufen einfach die Variante mit Parameter auf, wobei wir `\ignorespaces` übergeben. (Das ist beim Original ähnlich, allerdings dort einzeln analog implementiert.)

@narrcheckifds

```
\def\gmde@narrcheckifds{%
  \gmde@narrcheckifds@ne{\ignorespaces}%
}%
```

Die ein-Parameter-Version.

rrcheckifds@ne

```
\def\gmde@narrcheckifds@ne#1{%
  \@ifnextchar\gmde@visibleCommentChar%  Falls das nächste Zeichen unser
      %  Kommentarzeichen ist,
    \gmde@deactivatedCode%  setzen wir den Rest als deaktivierten Code.
    {\gmde@org@narrcheckifds@ne{#1}}%  Ansonsten rufen wir den
        %  Original-<-Test auf.
}%
```

Das ist (zusammen mit dem nächsten Makro) der Kern von `\gmd@codstripdirective`, den wir hier brauchen.

eactivatedCode

```
\def\gmde@deactivatedCode{%
  \gmd@textEOL%  Wir tun so, als wäre das eine neue Zeile.
  \gmde@deactivatedcodePrefix%  Das wird als erstes Token an \gmd@textEOL
      %  übergeben. Dadurch, dass es nur eines ist, wird es nicht als % erkannt,
      %  sondern (nachdem wieder in den Verbatim-Modus gewechselt wurde)
      %  ausgegeben.
}
```

Was machen wir mit deaktiviertem Code (am Zeilenanfang)?

atedcodePrefix

```
\def\gmde@deactivatedcodePrefix{%
  \gmde@visibleCommentColor%
  \code@delim%  Zunächst geben wir ein %-Zeichen aus.
  \@afternarrgfalse\@aftercodegtrue\@codeskipputgtrue%  Diese
      %  boolean-Werte setzen wir, damit TEX nachher weiß, wo es ist.
```

```
    \@firstoftwo%    Und jetzt entsorgen wir das zweite folgende Token – das ist das
                %    !-Zeichen. Das erste ist wohl irgendein Makro, welches wichtige Dinge
                %    anstellt - wenn wir das auch entsorgen, hört die Code-Formatierung
                %    nicht am nächsten % auf. Keine Ahnung, was das ist.
}%
```

Hier eine leicht geänderte Version von `\gmd@eatlspace`. (`%` und `!` bezeichnen im Kommentar nicht unbedingt das entsprechende Zeichen, sondern das, was gerade als Code-Delimiter bzw. Visible-Comment-Marker festgelegt wurde (mit `%` und `!` als Default.)

gmde@eatlspace
```
\def\gmde@eatlspace#1{%
  \ifx\gmd@texcodespace#1%    Wenn das nächste Zeichen ein Leerzeichen, so ...
    \advance\gmd@ldspaceswd␣by␣\gmd@spacewd%    erhöhen wir die
                %    Einrückung, und ...
    \afteriffifi{\gmde@eatlspace}%    ... wiederholen das ganze.
  \else%
    \if\code@delim\@nx#1%    Wenn das nächste Zeichen % ist ...
      \afterfifi{%
        \@ifnextchar\gmde@visibleCommentChar{%    ... und das folgende
                %    Zeichen ein !, dann haben wir es mit einem sichtbaren
                %    Kommentar zu tun.
          \let\gmd@charbychar=\gmde@charbychar%    Wir merken uns also,
                %    dass dessen Präfix gezeigt werden muss, ...
          \gmd@typesettexcode%    ... und rufen die normale
                %    Code-Formatierungs-Funktion auf, wobei das ! dabei
                %    weggeworfen wird.
        }{%
```

Ist das folgende Zeichen kein `!`, so geht jetzt der Kommentar der letzten Zeile weiter (das kann entweder ein inline-Kommentar oder ein normaler sein).

```
          \gmd@ldspaceswd=\z@%
          \gmd@continuenarration#1%
        }%
      }%
    \else%
```

Ist das nächste Zeichen weder ␣ noch `%`, so fängt hier der Quelltext an.

```
      \afterfifi{\gmd@typesettexcode#1}%
    \fi%
  \fi%
}%
```

Noch eine Sicherheitskopie: `\gmd@charbychar` geht zeichenweise durch den Code durch, mit Sonderbehandlung für `%`, `\`, das Dateiende sowie ein Zeilenende.

```
\let\gmde@org@charbychar\gmd@charbychar%
```

Unsere Variante davon, die gesetzt wird, wenn die Zeile mit einem `%!` beginnt,

```
\gmde@charbychar
615 \def\gmde@charbychar{%
616   \let\gmd@charbychar=\gmde@org@charbychar
617   \gmde@deactivatedcodePrefix
618   \gmd@charbychar%
619 }%
```

3.3.1 Etwas Test-Code

```
624 ⟨*test⟩
626 bla
627 ⟨text⟩ bedingter␣Text␣%    mit Kommentar. mehr text und Kommentar
```

Komentar

```
631 %␣das␣gleiche␣mit!␣%    und Kommentar
632   %␣das␣gleiche␣mit!␣%    und Kommentar
633 %␣das␣gleiche␣mit!␣%    und Kommentar
```

Kommentar.

```
636 ⟨/test⟩
```

3.4 Weitere Makros

```
642     \AtBeginDocument{\QueerCharOne}%    inputenc mit utf8x definiert ^^A um, so
                %    dass es Fehlermeldungen gibt. Also stellen wir die gmdoc-Variante
                %    wieder her.
```

Dieses Makro liest eine Datei im normalen LaTeX-Modus ein, ohne komische EOL-Zeichen oder kaputte %.

```
\normalInput
651 \newcommand*{\normalInput}[1]{%
```

Wir packen alles in eine Gruppe, dann ist das Wiederherstellen der Einstellungen nicht so schwer.

```
654   \begin{StraightEOL}%
655     \gmde@normalSpaces%
656     \catcode`\%=14%
657     \InputIfFileExists{#1}{}{}%
658   \end{StraightEOL}%
659 }%
```

3.5 Verbatim-docstrip-Modus

Es wurde wohl vergessen, diese Variable zu initialisieren.

```
666 \let\gmd@docstripshook\@empty%
668 ⟨*test⟩
```

Hier ein Test dafür:

```
⟨⟨verbatim − modus
Irgenwas␣hier
%␣auskommentiert␣%    wirklich!
Noch␣mehr␣Muell␣%    Hört das hier auf?
```

verbatim-modus

Hmm, das alleine reicht aber noch nicht, denke ich. Eigentlich sollte jetzt alles (bis zum erneuten auftauchen des Textes, der zwischen << und dem Zeilenende stand) wörtlich ausgegeben werden, ohne Formatierungen.

Naja, das baue ich vielleicht in einer späteren Version ein.

```
⟨/test⟩
```

3.6 Optionen des Paketes

Die Optionen rufen einfach die passenden Deklarationen auf.

visible
```
\DeclareOption{visible}{%
  \setVisibleCommentColor{blue}%
  \activateVisibleComments%
}%
```

inline
```
\DeclareOption{inline}{%
  \InlineCommentsWithPercent%
}%
```

Jetzt veraarbeiten wir alle Optionen in der Reihenfolge, in der sie angegeben wurden. (Nicht, dass die Reihenfolge einen Unterschied machen sollte ...)

```
\ProcessOptions*\relax%
```

Und damit sind wir fertig.

```
\endinput
⟨/package⟩
```

Index

Numbers written in italic refer to the code lines where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used. The numbers preceded with 'p.' are page numbers. All the numbers are hyperlinks.

\@aftercodegtrue, 562
\@afternarrgfalse, 562
\@codeskipputgtrue, 562
\@codetonarrskip, 334, 378, 381, 471
\@nx, 585
\@typeset@protect, 405
\@xa, 323, 342, 347, 356, 454, 457, 468
\activateVisibleComments, *p. 2*, 481, 690
\afterfi, 434
\afterfifi, 586, 603

\afteriffifi, 583
\AtBeginDocument, 642
\auskommentiert, 163, 164

\beispiel, 92, 99

\code@delim, 395, 561, 585
\color, *p. 2*, 392, 509, 512

\DeclareOption, 688, 693
\DocInput, 245
\documentclass, 202

\EOFMark, 224
\exhyphenpenalty, 325, 467

\fontseries, 228

\gebraucht, 126
\gmd@charbychar, 589, 611, 616, 618
\gmd@continuenarration, 597
\gmd@docstripshook, 666
\gmd@eatlspace, 484
\gmd@ldspaceswd, 581, 596
\gmd@narrcheckifds, 482, 529
\gmd@narrcheckifds@ne, 303, 483, 530
\gmd@percenthack, 276, 284, 290
\gmd@spacewd, 581
\gmd@texcodespace, 580
\gmd@textEOL, 551
\gmd@typesettexcode, 591, 603
\gmde@activeSpace, 316, 323, 454, 468
\gmde@charbychar, 589, 615
\gmde@codetonarrskip, 334, 381, 382
\gmde@deactivatedCode, 542, 550
\gmde@deactivatedcodePrefix, 552, 559, 617
\gmde@discrSpace, 323, 404
\gmde@eatlspace, 484, 579, 583
\gmde@gobbleActiveSpaces, 435, 453, 457
\gmde@hyphenreset, 354, 375, 465
\gmde@kommentarStart, 298, 387, 421
\gmde@narrcheckifds, 482, 535
\gmde@narrcheckifds@ne, 483, 536, 539
\gmde@normalSpaces, 382, 464, 655
\gmde@org@charbychar, 611, 616
\gmde@org@codetonarrskip, 378, 471
\gmde@org@ifilrr, 270, 282
\gmde@org@narrcheckifds, 529
\gmde@org@narrcheckifds@ne, 530, 543
\gmde@org@percenthack, 284, 290
\gmde@percenthack, 276, 293
\gmde@setVCColor@, 504, 508
\gmde@setVCColor@Opt, 503, 511
\gmde@specialSpaces, 296, 321
\gmde@visibleCommentChar, 490, 540, 587
\gmde@visibleCommentColor, 509, 512, 516, 560

\hyphenpenalty, 328, 467

\ifilrr, 270, 282
\ifinner, 406
\ifprevhmode, 294
\ilrrtrue, 272
inline, *p. 2*, 693
\InlineCommentsWithPercent, *p. 2*, 269, 694
\InputIfFileExists, 657

\MakeShortVerb*, 243
\marginpartt, 228

\nicht, 126, 164
\normalInlineComments, *p. 2*, 281
\normalInput, *p. 3*, 651

\orgshow, 213

\PassOptionsToClass, 201
\ProcessOptions*, 700

\quad, 395
\QueerCharOne, 642

\RecordChanges, 231
\renewcommand*, 224
\RequirePackage, 262

\setVisibleCommentChar, *p. 2*, 489, 494
\setVisibleCommentColor, *p. 2*, 500, 689

\usepackage, 206, 207, 208, 209, 217, 222

visible, *p. 2*, 688

\zeile, 92, 99