The *cloze* package*

Josef Friedrich
josef@friedrich.rocks
github.com/Josef-Friedrich/cloze

v1.7.0 from 2025/06/06

*Many thanks to Robert-Michael Huber for his advice and to Paul Isambert for his article *"Three things you can do with LuaTEX that would be extremely painful otherwise"* in TUGboat, Volume 31 (2010), No. 3. This article helped a lot to write this package.

Contents

1 Introduction

cloze is a plain TeX or a LaTeX package to generate cloze texts. It uses the capabilities of the modern TeX engine *LuaTeX*. Therefore, you must use LuaTeX or LuaLaTeX to create documents containing gaps.

`lualatex cloze-text.tex` or `luatex cloze-text.tex`

The main feature of the package is that the formatting doesn't change when using the `hide` and `show` (→ 2.3.12) options.

> Lorem ipsum *dolor sit* amet, consectetur *adipisicing* elit, sed do eiusmod tempor incididunt ut labore et *dolore magna* aliqua. Ut enim ad minim veniam, quis nostrud *exercitation* ullamco laboris nisi ut *aliquip* ex ea commodo consequat.

The command `\clozeset{hide}` only shows gaps. When you put both texts on top of each other you will see that they perfectly match.

> Lorem ipsum __________ amet, consectetur ____________ elit, sed do eiusmod tempor incididunt ut labore et _______________ aliqua. Ut enim ad minim veniam, quis nostrud _____________ ullamco laboris nisi ut ________ ex ea commodo consequat.

2 Usage

2.1 Interfaces

The main difference between the plain TeX and the LaTeX interface is option handling. In LaTeX options can be set by using key-value pairs. In plain TeX the only possibility to set options is to use the macro `\clozesetoption` (→ 2.3.3).

2.1.1 The plain TeX interface

```
\input cloze.tex
\clozesetoption{margin}{1cm}
\clozeshow
Lorem \cloze{ipsum} dolor.
\bye
```

2.1.2 The LaTeX interface

```
\documentclass{article}
\usepackage[show,margin=1cm]{cloze}
\begin{document}
Lorem \cloze{ipsum} dolor.
\end{document}
```

2.2 The commands and environments

There are the commands `\cloze`, `\clozefix`, `\clozefil`, `\clozenol`, `\clozeparcapture`, `\clozestrike` and the environments `clozepar` and `clozebox` to generate cloze texts.

2.2.1 `\cloze`

`\cloze` `\cloze[`⟨*options*⟩`]{`⟨*some text*⟩`}`: The command `\cloze` is similar to a command that offers the possibility to underline the texts. `\cloze` does not prevent line breaks. The width of a gap depends on the number of letters and the font used. The only option which affects the widths of a gap is the option `margin` (→ 2.3.14).

> Lorem ipsum *dolor* sit amet, *consectetur* adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore *magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi* ut aliquip ex ea commodo consequat.

It is possible to convert a complete paragraph into a 'gap'. But don't forget: There is a special environment for this: `clozepar` (→ 2.2.7).

> *Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.*

The command `\cloze` doesn't (better shouldn't) change the behavior of the hyphenation. Let's try some long German words (Known issue: Sometimes you have to insert manual hyphenation points (`\-`). I don't know why. Solutions and suggestions are very welcome.):

> es *Telekommunikationsüberwachung* geht *Unternehmenssteuerfortentwicklungsgesetz* *Abteilungsleiterin* *Oberkommisarin* auch *Fillialleiterin* kurz.

2.2.2 `\clozesetfont`

`\clozesetfont` The gap font can be changed by using the command `\clozesetfont`. `\clozesetfont` redefines the command `\clozefont` which contains the font definition. Thus, the command `\clozesetfont{\Large}` has the same effect as `\def\clozefont{\Large}`.

> Excepteur sint occaecat cupidatat non proident.

Please do not put any color definitions in `\clozesetfont`, as it won't work. Use the option `textcolor` instead (→ 2.3.13).

`\clozesetfont{\ttfamily\normalsize}` changes the gap text for example into a normal sized typewriter font.

> Excepteur `sint` occaecat `cupidatat` non proident.

If you use clozes in math mode you have to overwrite the default cloze font (`\clozesetfont{\itshape}`) of the package, because `\itshape` is not available in math mode. `\clozesetfont{}` will do the trick.

2.2.3 `\clozefix`

`\clozefix` `\clozefix[⟨options⟩]{⟨some text⟩}`: The command `\clozefix` creates gaps with a fixed width. The clozes are default concering the width `2cm`.

Lorem ipsum dolor sit amet:

1. *consectetur* ______________
2. *adipisicing* ______________
3. *elit* __________________

sed do eiusmod.

Gaps with a fixed width are much harder to solve.

Lorem ipsum dolor ____*sit*____ amet, ____*consectetur*____ adipisicing elit, sed do eiusmod tempor incididunt ____*ut*____ labore et dolore magna aliqua.

Using the option `align` you can make nice tabulars like this:

Composer	Life span
Joseph Haydn	*1723-1809*
Wolfgang Amadeus Mozart	*1756-1791*
Ludwig van Beethoven	*1770-1827*

2.2.4 `\clozenol`

`\clozenol` `\clozenol[⟨options⟩]{⟨some text⟩}`: The macro name `clozenol` stands for *“cloze no line”*. As the the name suggests this macro typesets cloze texts without a line. `\clozenol` is a convenient abbreviation for `\cloze[thickness=0pt]{text}`.

```
Lorem \clozenol{ipsum dolor} sit amet.
```

Lorem *ipsum dolor* sit amet.

```
Lorem \clozenol[textcolor=green]{ipsum dolor} sit amet.
```

Lorem *ipsum dolor* sit amet.

The next examples are showing that `\clozenol` behaves exactly as `\clozenol` with the option `thickness=0pt` (`\cloze[thickness=0pt]`) set: The text layout doesn’t change if we are hiding the gaps and the hidden text is not really hidden. It is removed. It can not be copied.

Lorem ipsum *dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua* sit amet.

Now hide the text

Lorem ipsum

sit amet.

2.2.5 `\clozefil`

`\clozefil` `\clozefil[`⟨*options*⟩`]{`⟨*some text*⟩`}`: The name of the command is inspired by `\hfil`, `\hfill`, and `\hfilll`. `\clozefil` fills out all available horizontal space with a line. The macros `\clozefill` and `\clozefilll` doesn't exist, only `\clozefil` (with one `l`) does.

`\clozehide`:

Complete the sentences below to share more about yourself:

I am someone who love ____.
I am someone who hate ____.
I am someone who can ____.
I am someone who can't ____.

`\clozeshow`:

Complete the sentences below to share more about yourself:

I am someone who love *to write cloze worksheets in TEX*.
I am someone who hate *to write cloze worksheets in MS Word*.
I am someone who can *write cloze worksheets in TEX*.
I am someone who can't *write cloze worksheets in MS Word*.

2.2.6 `\clozeextend`

`\clozeextend` `\clozeextend[`⟨*spaces*⟩`]`: The command `\clozeextend` adds some invisible placeholders to extend some cloze texts with blank space. Keep in mind that there is the option `minlines` (→ 2.3.15), if you want to extend some cloze paragraphs.

Corresponding options:

extension_count : The number of extension units.

extension_height : The height of one extension unit (default: 2ex).

extension_width : The width of one extension unit (default: 1em).

```
\begin{itemize}
\item \clozefil{Lorem ipsum dolor sit amet, consectetur adipisicing
elit, sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua.}
\item \clozefil{Ut enim ad minim veniam \clozeextend[20]}
\item \clozefil{quis nostrud \clozeextend[20]}
\end{itemize}
```

- *Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.*
- *Ut enim ad minim veniam*
- *quis nostrud*

2.2.7 clozepar

clozepar (*env.*) `\begin{clozepar}[⟨options⟩]` …*some text* …`\end{clozepar}`: The environment `clozepar` transforms a complete paragraph into a cloze text. The options `align`, `margin` and `width` have no effect on this environment.

Lorem ipsum dolor sit amet, consectetur adipisicing elit ullamco laboris nisi. *Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum.*
Excepteur sint occaecat cupidatat non proident.

2.2.8 \clozeparplain

\clozeparplain `\clozeparplain`: The command `\clozeparplain` is the macro version of the environment `clozepar` (→ 2.2.7). It is intended to provide cloze support on full paragraphs in plain LuaTEX.

2.2.9 \clozeparcapture

\clozeparcapture `\clozeparcapture`: The command is defined to capture all text until the next paragraph (`\par`): `\def\clozeparcapture#1\par{...}`. In some cases (see example below) you have to set the paragraph end manually by inserting a `\par`.

```
\begin{enumerate}
\item Solid state drives (SSD) ...

\clozeparcapture
They do not need to get up to speed to work properly / No latency;
...
\par

\end{enumerate}
```

1. Solid state drives (SSD) are replacing hard disc drives (HDD) in some computers. Give some reasons why this is happening.

   They do not need to get up to speed to work properly / No latency; Less power consumption / More energy efficient; Run cooler; Run quieter; Data access is faster; Occupies less physical space - more compact; Lighter, so suitable for portable computer / laptop; No moving parts so more reliable / durable in a portable computer / laptop.

It is often sufficient to insert an empty line.

```
Explain three ways that RAM is different to ROM.

1:

\clozeparcapture RAM is Volatile, ROM is non-volatile

2:
```

Explain three ways that RAM is different to ROM.

1:

RAM is Volatile, ROM is non-volatile

2:

RAM is temporary, ROM is (semi) permanent

3:

RAM normally has a larger capacity than ROM

2.2.10 `clozebox`

`clozebox` (*env.*) `\begin{clozebox}*[`⟨*options*⟩`]` ...*some text* ...`\end{clozebox}`: The environment `clozebox` surrounds a text with a box. The starred version omits the line around the box. Use the options `boxheight` (→ 2.3.8) and `boxwidth` (→ 2.3.10) to specify the dimensions of the box. By default the width of the box is `\linewidth`. The height of the box is determined by the amount of text. This environment is realized by a combination of the `minipage` environment surrounded by a `\fbox`. For the cloze text the macro `\clozenol` is reused. New paragraphs are not allowed inside a cloze box. Use two backslashes multiple times `\\` instead.

```
\begin{clozebox}
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.
\end{clozebox}
```

`\clozehide`:

`\clozeshow`:

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Like with all cloze macros and environments the hidden text vanishes from the rendered file. The starred version omits the line around the box:

```
\begin{clozebox}*
Lorem ...
\end{clozebox}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

2.2.11 `clozespace`

`clozespace` (*env.*) `\begin{clozespace}[⟨options⟩]` …*some text* …`\end{clozespace}`: If you are using a bigger font for the cloze text as for the normal text, you are getting irregular distances between the lines:

```
\clozesetfont{\Huge\fontspec{Kalam}}
Today in the Discovery ...
```

Today in the Discovery Lab we learned about three types of spacecraft that are helping us explore Mars. The spacecraft are on Mrs. Bratt's Principal's Reading Challenge board. One type of spacecraft is the orbiter.

With the environment `clozespace` you are able to restore the regular balanced line spacing. The default value for the option `spacing` is `1.6`. Also take a look in the section about the option `spacing` (→ 2.3.16).

```
\begin{clozespace}[spacing=2]
...
\end{clozespace}
```

Today in the Discovery Lab we learned about three types of spacecraft that are helping us explore Mars. The spacecraft are on

Mrs. Bratt's Principal's Reading Challenge board. One type of spacecraft is

the orbiter.

The environment `clozespace` uses the package setspace in the background for setting the spacing between the lines.

2.2.12 `\clozeline`

`\clozeline` `\clozeline[⟨options⟩]`: To create a cloze line of a certain width, use the command `\clozeline`. The default width of the line is `2cm`. In combination with the other cloze commands you can create for example an irregular alignment of the cloze text.

```
Ut enim ad
\clozeline[width=1cm]\cloze{minim}\clozeline[width=3cm]
minim veniam
```

Ut enim ad _____*minim*_____ minim veniam,

2.2.13 `\clozelinefil`

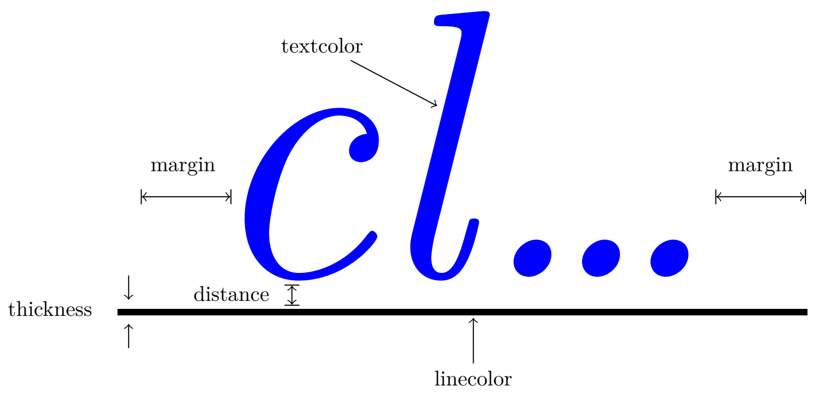
`\clozelinefil` `\clozelinefil[⟨options⟩]`: This command `\clozelinefil` fills the complete available horizontal space with a line. Moreover, `\clozelinefil` was used to create `\clozefil`.

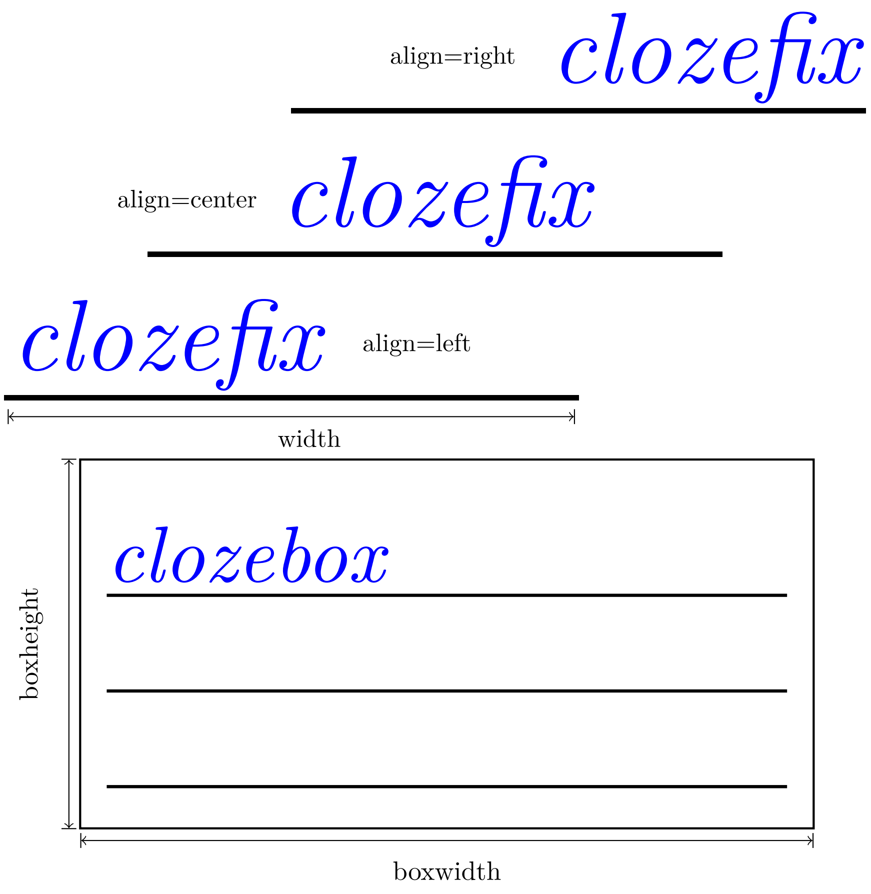
Lorem____

2.3 The options

2.3.1 Overview over all options

Option	Description	Reference
`align`	Align the text of a fixed size cloze (`\clozefix`) .	(→ 2.3.7)
`boxheight`	The height of a cloze box (`clozebox`).	(→ 2.3.8)
`boxrule`	The thickness of the rule around a `clozebox`.	(→ 2.3.9)
`boxwidth`	The width of a cloze box (`clozebox`).	(→ 2.3.10)
`distance`	The distance between the cloze text and the cloze line.	(→ 2.3.11)
`hide`	Hide the cloze text.	(→ 2.3.12)
`linecolor`	A color name to colorize the cloze line.	(→ 2.3.13)
`margin`	Additional margin between the normal and the cloze text.	(→ 2.3.14)
`minlines`	The minimum of lines a `clozepar` environment must have.	(→ 2.3.15)
`show`	Show the cloze text.	(→ 2.3.12)
`spacing`	The distance of the lines in the environment `clozespace`.	(→ 2.3.16)
`textcolor`	A color name to colorize the cloze text.	(→ 2.3.13)
`thickness`	The thickness of a line.	(→ 2.3.17)
`width`	The width of a fixed size cloze (`\clozefix`).	(→ 2.3.12)





2.3.2 Local and global options

The *cloze* package distinguishes between *local* and *global* options. Besides the possiblity to set *global* options in the `\usepackage[`⟨*global options*⟩`]{`⟨*cloze*⟩`}` declaration, the cloze package offers a special command to set *global* options: `\clozeset{`⟨*global options*⟩`}`

2.3.3 `\clozesetoption`

`\clozesetoption` `\clozesetoption{`⟨*key*⟩`}{`⟨*value*⟩`}`: Set a single option. In plain TEX the command sets the options only in the global option space.

2.3.4 `\clozeset`

`\clozeset` `\clozeset{`⟨*global options*⟩`}`: The command can set *global* options for each paragraph.

```
\clozeset{textcolor=red} Lorem \cloze{ipsum} dolor \par
\clozeset{textcolor=green} Lorem \cloze{ipsum} dolor
```

Lorem *ipsum* dolor
Lorem *ipsum* dolor

`\clozeset` does not change the options within a paragraph. As you can see in the example below the last `\clozeset` applies the color green for both gaps.

```
\clozeset{textcolor=red} Lorem \cloze{ipsum} dolor
\clozeset{textcolor=green} Lorem \cloze{ipsum} dolor
```

Lorem *ipsum* dolor Lorem *ipsum* dolor

2.3.5 `\clozereset`

`\clozereset` `\clozereset`: The command resets all *global* options to the default values. It has no effect on the *local* options.

```
\clozeset{
  thickness=3mm,
  linecolor=yellow,
  textcolor=magenta,
  margin=-2pt
}
```

Very *silly* global *options*.

```
\clozereset
```

Relax! We can reset *those* options.

2.3.6 `\clozeshow` and `\clozehide`

`\clozeshow` `\clozehide` `\clozeshow` and `\clozehide`: This commands are shortcuts for `\clozeset{`⟨*show*⟩`}` and `\clozeset{`⟨*hide*⟩`}`.

```
\clozehide
```

Lorem ________________ amet, consectetur ____________ elit.

```
\clozeshow
```

Lorem *ipsum dolor sit* amet, consectetur *adipisicing* elit.

2.3.7 `align`

`[align=`⟨*left/center/right*⟩`]`: Only the macro `\clozefix` (→ 2.2.3) takes the option `align` into account. Possible values are `left`, `center` and `right`. This option only makes sense, if the width of the line is larger than the width of the text.

Lorem ipsum	(`left`)
Lorem ipsum	(`center`)
Lorem ipsum	(`right`)

2.3.8 `boxheight`

`[boxheight=`⟨*dimen*⟩`]`: specifies the height of a cloze box. This option has only an effect on the environment `clozebox` (→ 2.2.10).

```
\begin{clozebox}[boxwidth=5cm]
```

boxwidth: 2.5cm; Lorem ipsum dolor sit amet ...

boxwidth: 3cm; Lorem ipsum dolor sit amet ...

boxwidth: 4cm; Lorem ipsum dolor sit amet ...

2.3.9 `boxrule`

`[boxrule=`⟨*dimen*⟩`]`: specifies the thickness of the rule around a cloze box. This option has only an effect on the environment `clozebox` (→ 2.2.10).

```
\begin{clozebox}[boxrule=2pt]
```

boxrule: 1pt

boxrule: 2pt

boxrule: 3pt

2.3.10 `boxwidth`

`[boxwidth=`⟨*dimen*⟩`]`: specifies the width of a cloze box. This option has only an effect on the environment `clozebox` (→ 2.2.10).

```
\begin{clozebox}[boxheight=3cm]
```

boxheight: 3cm; Lorem ipsum dolor sit amet …

boxheight: 2cm; Lorem ipsum dolor sit amet …

boxheight: 1cm; Lorem ipsum dolor sit amet …

2.3.11 `distance`

`[distance=`⟨*dimen*⟩`]`: The option `distance` specifies the spacing between the baseline of the text and the gap line. The larger the dimension of the option `distance`, the more moves the line down. Negative values cause the line to appear above the baseline. The default value is `1.5pt`.

Lorem ipsum dolor sit amet. (`1.5pt`)
Lorem ipsum dolor sit amet. (`3pt`)
~~*Lorem ipsum dolor sit amet.*~~ (`-3pt`)

2.3.12 `hide` and `show`

`[hide]` and `[show]`: By default the cloze text is displayed. Use the option `hide` to remove the cloze text from the output.

Lorem ipsum ______________, consectetur ___________ elit. (`hide`)
Lorem ipsum *dolor sit amet*, consectetur *adipisicing* elit. (`show`)

2.3.13 `linecolor` and `textcolor`

`[linecolor=`⟨*color name*⟩`]` and `[textcolor=`⟨*color name*⟩`]`: Values for both color options are color names used by the xcolor package. To define your own color use the following command:

```
\definecolor{myclozecolor}{rgb}{0.1,0.4,0.6}
\cloze[textcolor=myclozecolor]{Lorem ipsum}
```

Lorem ipsum dolor sit amet, consectetur (`myclozecolor`)
Lorem ipsum dolor sit amet, consectetur (`red`)
Lorem ipsum dolor sit amet, consectetur (`green`)

You can use the same color names to colorize the cloze lines.

Lorem ipsum dolor sit amet, consectetur (`myclozecolor`)
Lorem ipsum dolor sit amet, consectetur (`red`)
Lorem ipsum dolor sit amet, consectetur (`green`)

And now hide the clozes:

(myclozecolor)
(red)
(green)

2.3.14 `margin`

`[margin=`⟨*dimen*⟩`]`: The option `margin` indicates how far the line sticks up from the text. The option can be used with the commands `\cloze`, `\clozefix` and `\clozefil`. The default value of the option is `3pt`.

Lorem ipsum *dolor* sit amet. (`0pt`)
Lorem ipsum *dolor* sit amet. (`5mm`)
Lorem ipsum *dolor* sit amet. (`1cm`)
Lorem ipsum *dolor* sit amet. (`6em`)
Lorem ipsum*dolor*sit amet. (`-4pt`)

Is a punctation mark placed directly after a gap, then the line breaks after this punctation mark. Even the most large value of `margin` does not affect this behavior.

Lorem, *ipsum*. *dolor*; *sit*: *amet*, *consectetur*. *adipisicing*;
elit: *sed*, *do*. *eiusmod*; *tempor*.

2.3.15 `minlines`

`[minlines=`⟨*integer*⟩`]`: `minlines` stands for “**min**imum **lines**”. With the option `minlines` you can set how many lines a `clozepar` at least must have. If the text to be typeset results in fewer lines, empty lines are appended to the end. If the text of a `clozepar` produces more lines as specified in `minlines`, the option has no effect at all. The option `minlines` only affects the behavior of the environment `clozepar` (→ 2.2.7) and the macro versions `\clozeparplain` (→ 2.2.8) and `\clozeparcapture` (→ 2.2.9).

```
\clozeset{minlines=4}
\begin{enumerate}
\item Battlements

\begin{clozepar}
Higher ...
```

`\clozehide`:

Defending a Castle

Here is a list of features on castle in the Middle Ages that would help to defend it.

1. Battlements

   ____

   ____

   ____

   ____

2. Embrasures

   ____

   ____

   ____

   ____

3. Portcullis

   ____

   ____

   ____

   ____

`\clozeshow`:

Defending a Castle

Here is a list of features on castle in the Middle Ages that would help to defend it.

1. Battlements

   Higher than the walls and jutting out from them to allow defenders to fire at people who had reached the walls

2. Embrasures

   Heavily protected entrance

   ____

   ____

   ____

3. Portcullis

   Arrow slits with a round hole for firing musket guns

   ____

   ____

2.3.16 `spacing`

`[spacing=`⟨*number*⟩`]`: This option provides support for setting the spacing between lines. A larger font used for the cloze texts needs more line space to avoid unsteady line distances. This option only affects the environment `clozespace` (→ 2.2.11).

2.3.17 `thickness`

`[thickness=`⟨*dimen*⟩`]`: The option `thickness` indicates how thick the line is. The option `distance` (→ 2.3.11) is not affected by this option, because the bottom of the line moves down. The default value of this option is `0.4pt`.

Lorem *ipsum dolor sit* amet. (`0.01pt`)
Lorem *ipsum dolor sit* amet. (`1pt`)
Lorem *ipsum dolor sit* amet. (`2pt`)

2.3.18 `width`

`[width=`⟨*dimen*⟩`]`: The only command which can be changed by the option `width` is `\clozefix` (→ 2.2.3). The default value of the option is `2cm`.

Lorem *dolor* amet. (`3cm`)
Lorem *dolor* amet. (`5cm`)
Lorem *dolor* amet. (`7cm`)

2.4 Handwriting fonts from CTAN and TeX Live

If you want to imitate a hand-filled worksheet, then some handwriting fonts are suitable for this purpose. This section is intended to provide an overview of handwriting fonts available on CTAN and TeX Live. The fonts are listed in alphabetical order:

LobsterTwo

CTAN:	lobster2
TeX Live:	`tlmgr install lobster2`
Font selection:	`\clozesetfont{\fontspec{LobsterTwo}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Miama Nueva

CTAN:	miama
TeX Live:	`tlmgr install miama`
Font selection:	`\clozesetfont{\fontspec{Miama Nueva}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore

et dolore magna aliquyam erat, sed diam volup-
tua.

QT Brush Stroke

CTAN:	qualitype
TeX Live:	`tlmgr install qualitype`
Font selection:	`\clozesetfont{\fontspec{QT Brush Stroke}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

QT Florencia

CTAN:	qualitype
TeX Live:	`tlmgr install qualitype`
Font selection:	`\clozesetfont{\fontspec{QT Florencia}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

QT Handwriting

CTAN:	qualitype
TeX Live:	`tlmgr install qualitype`
Font selection:	`\clozesetfont{\fontspec{QT Handwriting}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

QT Linostroke

CTAN:	qualitype
TeX Live:	`tlmgr install qualitype`
Font selection:	`\clozesetfont{\fontspec{QT Linostroke}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam non-
umy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,
sed diam voluptua.

QT Merry Script

CTAN:	qualitype
TEX Live:	`tlmgr install qualitype`
Font selection:	`\clozesetfont{\fontspec{QT Merry Script}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

QT Slogantype

CTAN:	qualitype
TEX Live:	`tlmgr install qualitype`
Font selection:	`\clozesetfont{\fontspec{QT Slogantype}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

2.5 Handwriting fonts from Google Fonts

You can get many more free handwriting fonts from Google Fonts. This section shows only a selection. I personally use the font named *Kalam* for my worksheets. All Google Fonts are available in a Git respository.

```
git clone https://github.com/google/fonts.git
```

The fonts are listed in alphabetical order:

Annie Use Your Telescope

URL:	`https://fonts.google.com/specimen/Annie+Use+Your+Telescope`
Font selection:	`\clozesetfont{\fontspec{Annie Use Your Telescope}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Architects Daughter

URL:	`https://fonts.google.com/specimen/Architects+Daughter`
Font selection:	`\clozesetfont{\fontspec{Architects Daughter}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Bad Script

URL:	`https://fonts.google.com/specimen/Bad+Script`
Font selection:	`\clozesetfont{\fontspec{Bad Script}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Caveat

URL:	`https://fonts.google.com/specimen/Caveat`
Font selection:	`\clozesetfont{\fontspec{Caveat}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Cedarville Cursive

URL:	`https://fonts.google.com/specimen/Cedarville+Cursive`
Font selection:	`\clozesetfont{\fontspec{Cedarville Cursive}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Coming Soon

URL:	`https://fonts.google.com/specimen/Coming+Soon`
Font selection:	`\clozesetfont{\fontspec{Coming Soon}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Give You Glory

URL:	`https://fonts.google.com/specimen/Give+You+Glory`
Font selection:	`\clozesetfont{\fontspec{Give You Glory}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Gochi Hand

URL:	https://fonts.google.com/specimen/Gochi+Hand
Font selection:	`\clozesetfont{\fontspec{Gochi Hand}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Handlee

URL:	https://fonts.google.com/specimen/Handlee
Font selection:	`\clozesetfont{\fontspec{Handlee}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Homemade Apple

URL:	https://fonts.google.com/specimen/Homemade+Apple
Font selection:	`\clozesetfont{\fontspec{Homemade Apple}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Indie Flower

URL:	https://fonts.google.com/specimen/Indie+Flower
Font selection:	`\clozesetfont{\fontspec{Indie Flower}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Just Another Hand

URL:	https://fonts.google.com/specimen/Just+Another+Hand
Font selection:	`\clozesetfont{\fontspec{Just Another Hand}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Just Me Again Down Here

URL:	https://fonts.google.com/specimen/Just+Me+Again+Down+Here
Font selection:	`\clozesetfont{\fontspec{Just Me Again Down Here}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Kalam

URL:	https://fonts.google.com/specimen/Kalam
Font selection:	`\clozesetfont{\fontspec{Kalam}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Kristi

URL:	https://fonts.google.com/specimen/Kristi
Font selection:	`\clozesetfont{\fontspec{Kristi}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

La Belle Aurore

URL:	https://fonts.google.com/specimen/La+Belle+Aurore
Font selection:	`\clozesetfont{\fontspec{La Belle Aurore}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Marck Script

URL:	https://fonts.google.com/specimen/Marck+Script
Font selection:	`\clozesetfont{\fontspec{Marck Script}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Neucha

URL:	https://fonts.google.com/specimen/Neucha
Font selection:	`\clozesetfont{\fontspec{Neucha}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Nothing You Could Do

URL:	https://fonts.google.com/specimen/Nothing+You+Could+Do
Font selection:	`\clozesetfont{\fontspec{Nothing You Could Do}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Patrick Hand

URL:	https://fonts.google.com/specimen/Patrick+Hand
Font selection:	`\clozesetfont{\fontspec{Patrick Hand}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Reenie Beanie

URL:	https://fonts.google.com/specimen/Reenie+Beanie
Font selection:	`\clozesetfont{\fontspec{Reenie Beanie}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Seaweed Script

URL:	`https://fonts.google.com/specimen/Seaweed+Script`
Font selection:	`\clozesetfont{\fontspec{Seaweed Script}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Shadows Into Light

URL:	`https://fonts.google.com/specimen/Shadows+Into+Light`
Font selection:	`\clozesetfont{\fontspec{Shadows Into Light}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Swanky and Moo Moo

URL:	`https://fonts.google.com/specimen/Swanky+and+Moo+Moo`
Font selection:	`\clozesetfont{\fontspec{Swanky and Moo Moo}}`

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

2.6 Special application areas

This section lists examples that didn't work in older versions of the cloze package and required special treatment to work as expected.

2.6.1 The math mode

By default the package uses `\itshape` to format the cloze text. In math mode you have to reset the cloze text format by calling `\clozesetfont{}`. A known bug is: You can't show and hide a single display math formula. Only the last `\clozeshow` or `\clozehide` takes effect on the whole document. Side note: The usage of the TeX primitive syntax `$$ $$` is not recommended.

```
\clozesetfont{}
$$1 + 1 = \cloze{2}$$
\clozesetfont{\itshape}
```

$$1 + 1 = \underline{\ 2\ }$$

`\[\]` should be used instead.

```
\[123 + 456 = \cloze{579}\]
```

$$123 + 456 = \underline{\ 579\ }$$

A cloze inside a display math environment should work fine:

```
\begin{displaymath}
2^{\cloze{2}} = 4
\end{displaymath}
```

$$2^{\underline{\ 2\ }} = 4$$

The inline math mode works too:

```
${\sqrt[3]{\cloze{8}}} = 2$ and ${\sqrt[\cloze{3}]{\cloze{8}}} = 2$
```

$\sqrt[3]{\underline{\ 8\ }} = 2$ and $\sqrt[\underline{3}]{\underline{\ 8\ }} = 2$

```
\begin{equation}
e^{\pi i} + 1 = \cloze{0}
\end{equation}
```

$$e^{\pi i} + 1 = \underline{\ 0\ } \qquad (1)$$

The amsmath `multline` environment:

```
\begin{multline*}
p(x) = \cloze{590}x^4y^2 +
↪  \cloze{19}x^3y^3\\
- 12x^2y^4 - \cloze{12}xy^5
\end{multline*}
```

$$p(x) = \underline{\ 590\ }\,x^4y^2 + \underline{\ 19\ }\,x^3y^3$$
$$- 12x^2y^4 - \underline{\ 12\ }\,xy^5$$

`\clozehide`:

Missing Subtraction Facts to 12

a) $8 - 6 = \underline{\qquad}$

b) $10 - 4 = \underline{\qquad}$

c) $7 - \underline{\qquad} = 5$

d) $6 - \underline{\qquad} = 5$

e) $\underline{\qquad} - 3 = 3$

f) $\underline{\qquad} - 2 = 8$

`\clozeshow`:

Missing Subtraction Facts to 12

a) $8 - 6 = \underline{\ 2\ }$

b) $10 - 4 = \underline{\ 6\ }$

c) $7 - \underline{\ 2\ } = 5$

d) $6 - \underline{\quad 1 \quad} = 5$

e) $\underline{\quad 6 \quad} - 3 = 3$

f) $\underline{\quad 10 \quad} - 2 = 8$

2.6.2 The `tabbing` environment

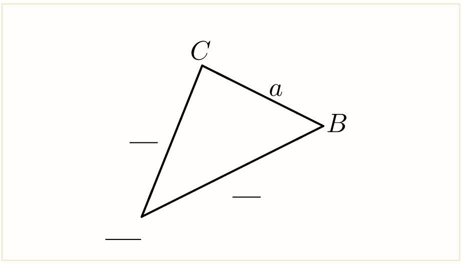
```
\begin{tabbing}
col1 \hspace{1cm} \= col2 \hspace{1cm} \= col3 \hspace{1cm} \= col4 \\
\cloze{col1} \> \> \clozefix{col3} \\
\end{tabbing}
```

col1	col2	col3	col4
col1		*col3*	

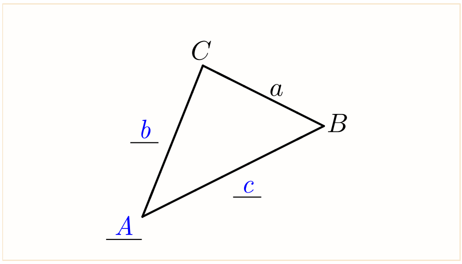
2.6.3 The picture environment

```
\setlength{\unitlength}{0.8cm}
\begin{picture}(4.8,3.8)
\thicklines
\put(1,0.5){\line(2,1){3}} \put(4,2){\line(-2,1){2}}
↪  \put(2,3){\line(-2,-5){1}}

\put(0.4,0.2){\cloze{A}} \put(4.05,1.9){$B$} \put(1.8,3.1){$C$}

\put(3.1,2.5){$a$} \put(0.8,1.8){\cloze{b}} \put(2.5,0.9){\cloze{c}}
\end{picture}
```

\clozehide:



\clozeshow:



2.6.4 The tabular environment

```
\begin{tabular}{l|l}
  \textbf{englisch} & \textbf{deutsch} \\\hline
  book & \clozefil{Buch} \\
  \clozefil{scissors} & Schere\\
  pen & \clozefil{Füller} \\
  \clozefil{pencil} & Bleistift\\
\end{tabular}
```

\clozehide:

englisch	**deutsch**
book	
	Schere
pen	
	Bleistift

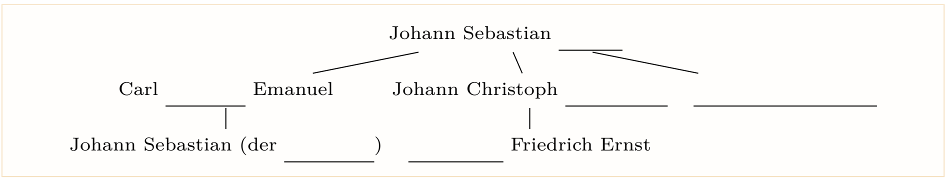
\clozeshow:

englisch	**deutsch**
book	*Buch*
scissors	Schere
pen	*Füller*
pencil	Bleistift

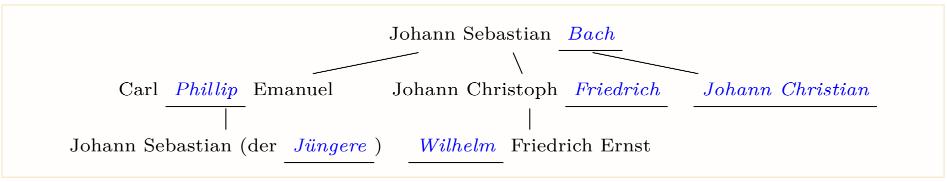
2.6.5 The package `forest`

```
\begin{forest}
[Johann Sebastian \cloze{Bach}
  [Carl \cloze{Phillip} Emanuel
    [Johann Sebastian (der \cloze{Jüngere})]
  ]
  [Johann Christoph \cloze{Friedrich}
    [\cloze{Wilhelm} Friedrich Ernst]
  ]
  [\cloze{Johann Christian}]
]
\end{forest}
```

`\clozehide`:



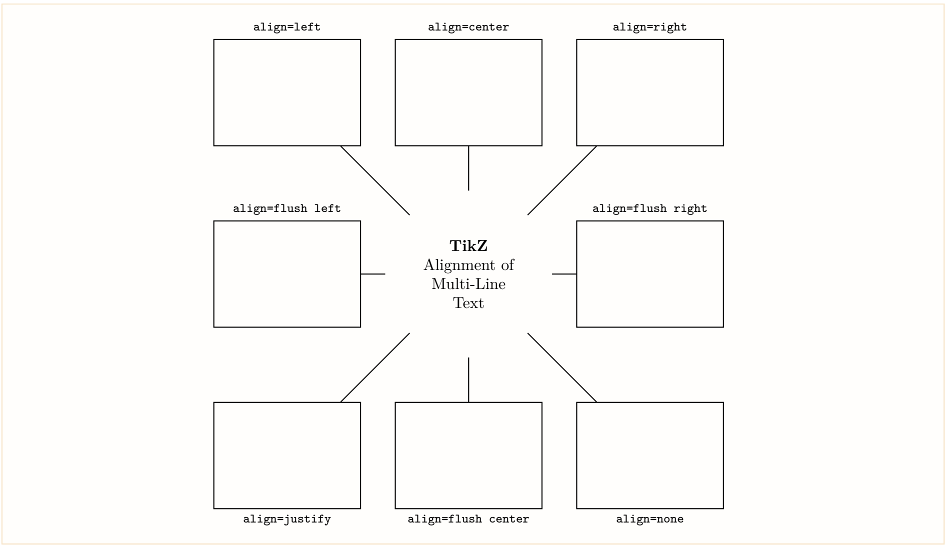
`\clozeshow`:



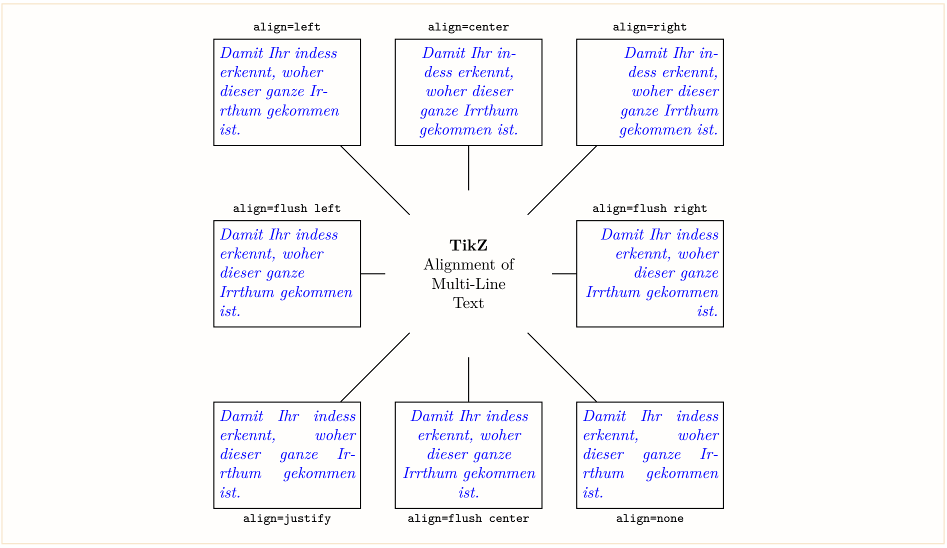
2.6.6 The package `tikz`

2.6.6.1 Example Multiline alignment in TikZ

`\clozehide`:



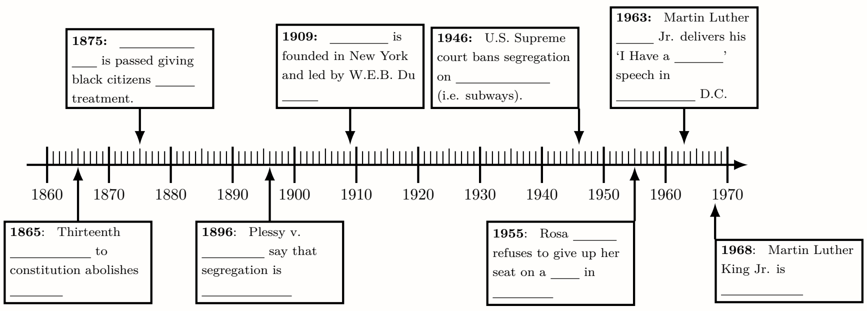
`\clozeshow`:



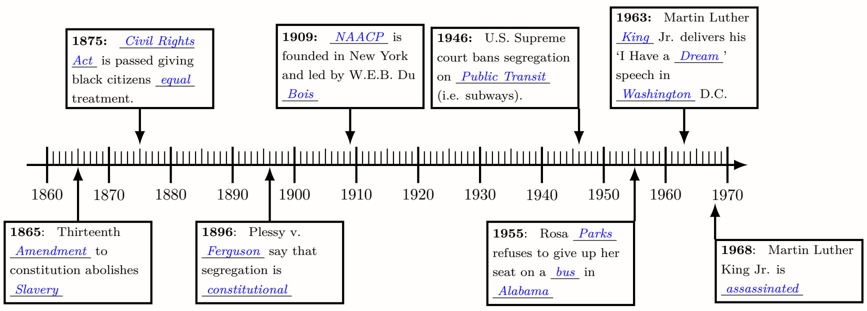
align=left
Damit Ihr indess erkennt, woher dieser ganze Irrthum gekommen ist.
align=center
Damit Ihr indess erkennt, woher dieser ganze Irrthum gekommen ist.
align=right
Damit Ihr indess erkennt, woher dieser ganze Irrthum gekommen ist.
align=flush left
Damit Ihr indess erkennt, woher dieser ganze Irrthum gekommen ist.
TikZ
Alignment of
Multi-Line
Text
align=flush right
Damit Ihr indess erkennt, woher dieser ganze Irrthum gekommen ist.
Damit Ihr indess erkennt, woher dieser ganze Irrthum gekommen ist.
align=justify
Damit Ihr indess erkennt, woher dieser ganze Irrthum gekommen ist.
align=flush center
Damit Ihr indess erkennt, woher dieser ganze Irrthum gekommen ist.
align=none

2.6.6.2 Example Timeline

`\clozehide`:

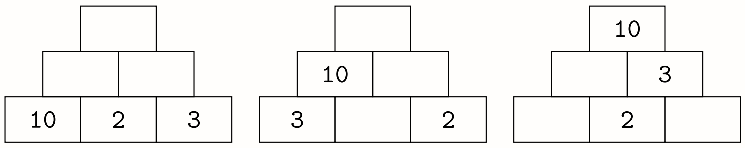

1875: ________ ____ is passed giving black citizens _____ treatment.
1909: ________ is founded in New York and led by W.E.B. Du _____
1946: U.S. Supreme court bans segregation on ___________ (i.e. subways).
1963: Martin Luther _____ Jr. delivers his 'I Have a _______' speech in __________ D.C.
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1865: Thirteenth __________ to constitution abolishes _______
1896: Plessy v. ________ say that segregation is ___________
1955: Rosa ______ refuses to give up her seat on a ____ in ________
1968: Martin Luther King Jr. is ___________

`\clozeshow`:

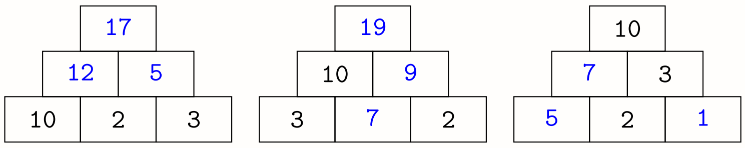

1875: Civil Rights Act is passed giving black citizens equal treatment.
1909: NAACP is founded in New York and led by W.E.B. Du Bois
1946: U.S. Supreme court bans segregation on Public Transit (i.e. subways).
1963: Martin Luther King Jr. delivers his 'I Have a Dream' speech in Washington D.C.
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1865: Thirteenth Amendment to constitution abolishes Slavery
1896: Plessy v. Ferguson say that segregation is constitutional
1955: Rosa Parks refuses to give up her seat on a bus in Alabama
1968: Martin Luther King Jr. is assassinated

2.6.6.3 Number pyramid

`\clozehide`:

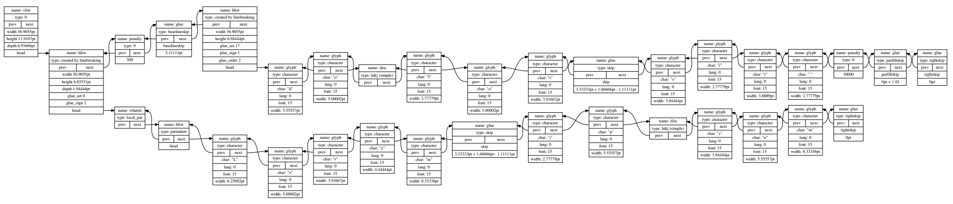


`\clozeshow`:

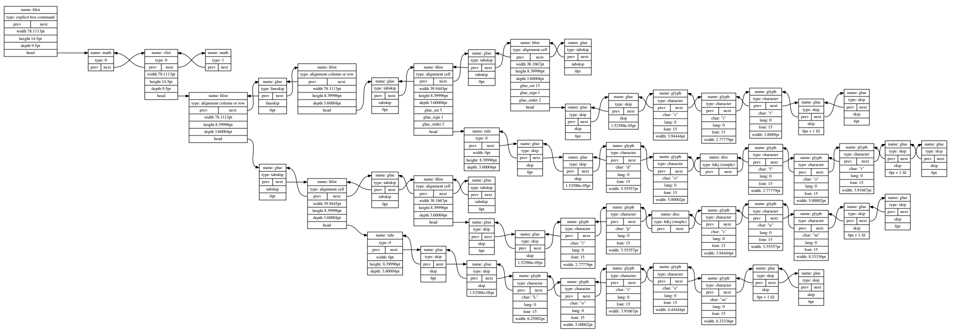


3 Some graphics for better understanding of the node tree

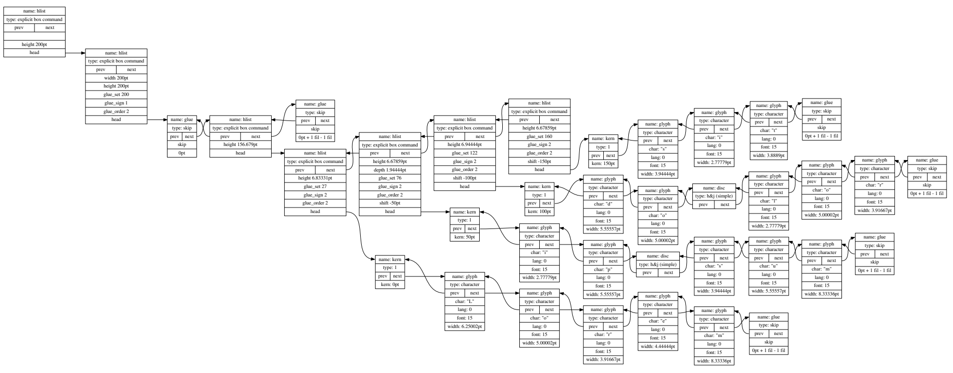
3.1 Paragraph



3.2 Tabular environment



3.3 Picture environment



3.4 The file cloze.lua

```
-- cloze.lua
-- Copyright 2015-2025 Josef Friedrich
--
-- This work may be distributed and/or modified under the
-- conditions of the LaTeX Project Public License, either version 1.3c
-- of this license or (at your option) any later version.
-- The latest version of this license is in
--   http://www.latex-project.org/lppl.txt
-- and version 1.3c or later is part of all distributions of LaTeX
```

```lua
-- version 2008/05/04 or later.
--
-- This work has the LPPL maintenance status `maintained'.
--
-- The Current Maintainer of this work is Josef Friedrich.
--
-- This work consists of the files cloze.lua, cloze.tex,
-- and cloze.sty.
---
---<h3>Naming conventions</h3>
---
---* _Variable_ names for _nodes_ are suffixed with `_node`, for example
---  `head_node`.
---* _Variable_ names for _lengths_ (dimensions) are suffixed with
---  `_length`, for example `width`.
---
---__Initialisation of the function tables__
---It is good form to provide some background informations about this Lua
---module.
if not modules then
  modules = {}
end
modules['cloze'] = {
  version = '1.7.0',
  comment = 'cloze',
  author = 'Josef Friedrich, R.-M. Huber',
  copyright = 'Josef Friedrich, R.-M. Huber',
  license = 'The LaTeX Project Public License Version 1.3c 2008-05-04',
}

local farbe = require('farbe')
local luakeys = require('luakeys')()
local lparse = require('lparse')

local ansi_color = luakeys.utils.ansi_color
local log = luakeys.utils.log
local tex_printf = luakeys.utils.tex_printf

---Option handling.
---
---The table `config` bundles functions that deal with the option
---handling.
---
---<h2>Marker processing (marker)</h2>
---
---A marker is a whatsit node of the subtype `user_defined`. A marker
---has two purposes:
---
---* Mark the begin and the end of a gap.
---* Store a index number, that points to a Lua table, which holds some
---  additional data like the local options.

---
---@alias MarkerMode 'basic'|'strike'|'fix'|'par' # The argument `mode` accepts the
↪  string values `basic`, `fix` and `par`.

---
---@alias MarkerPosition 'start'|'stop' # The argument `position` is either set to
↪  `start` or to `stop`.

---
---@class MarkerData
---@field mode MarkerMode
---@field position MarkerPosition
```

```lua
---@field local_opts Options

---
---All values and functions, which are related to the option
---management, are stored in a table called `config`.
local config = (function()

  ---
  ---I didn’t know what value I should take as `user_id`. Therefore I
  ---took my birthday and transformed it into a large number.
  local user_id = 3121978

  ---
  ---Store all local options of the markers.
  ---@type {[integer]: MarkerData }
  local storage = {}

  ---@class Options
  ---@field align? 'l'|'r'
  ---@field box_height? string
  ---@field box_rule? string
  ---@field box_width? string
  ---@field distance? string
  ---@field extension_count? integer
  ---@field extension_height? string
  ---@field extension_width? string
  ---@field line_color? string
  ---@field margin? string
  ---@field min_lines? integer
  ---@field resetcolor? string
  ---@field show_text? boolean
  ---@field spacing? number
  ---@field text_color? string
  ---@field thickness? string
  ---@field visibility? boolean
  ---@field width? string

  ---The default options.
  local defaults = {
    align = 'l',
    box_height = false,
    box_rule = '0.4pt',
    box_width = '\\linewidth',
    distance = '1.5pt',
    extension_count = 5,
    extension_height = '2ex',
    extension_width = '1em',
    line_color = 'black',
    margin = '3pt',
    min_lines = 0,
    show_text = true,
    spacing = '1.6',
    text_color = 'blue',
    thickness = '0.4pt',
    visibility = true,
    width = '2cm',
  }

  ---
  ---The global options set by the user.
  ---@type Options
  local global_options = {}

  ---
```

```lua
---The local options.
---@type Options
local local_options = {}

local index

---
---`index` is a counter. The functions `get_index()`
---increases the counter by one and then returns it.
---
---@return integer # The index number of the corresponding table in `storage`.
local function get_index()
  if not index then
    index = 0
  end
  index = index + 1
  return index
end

---
---The function `get_storage()` retrieves values which belong
--- to a whatsit marker.
---
---@param index integer # The argument `index` is a numeric value.
---
---@return MarkerData value
local function get_storage(index)
  return storage[index]
end

---
---`set_storage()` stores the local options in the Lua table
--- `storage`.
---
---It returns a numeric index number. This index number is the key,
---where the local options in the Lua table are stored.
---
---@param mode MarkerMode
---@param position MarkerPosition
---
---@return number # The index number of the corresponding table in
---  `storage`.
local function set_storage(mode, position)
  local index = get_index()
  local data = { mode = mode, position = position }
  if position == 'start' then
    data.local_opts = {}
    for key, value in pairs(local_options) do
      data.local_opts[key] = value
    end
  end
  storage[index] = data
  return index
end

---
---We create a user defined whatsit node that can store a number (type
--- = 100).
---
---In order to distinguish this node from other user defined whatsit
---nodes we set the `user_id` to a large number. We call this whatsit
---node a marker.
---
```

```
---@param index number The argument `index` is a number, which is associated to
↪  values in the `storage` table.
---
---@return UserDefinedWhatsitNode
local function create_marker(index)
  local marker = node.new('whatsit', 'user_defined') --[[@as
  ↪  UserDefinedWhatsitNode]]
  marker.type = 100 -- number
  marker.user_id = user_id
  marker.value = index
  return marker
end

---
---Write a marker node to TeX's current node list.
---
---@param mode MarkerMode
---@param position MarkerPosition
local function write_marker(mode, position)
  local index = set_storage(mode, position)
  local marker = create_marker(index)
  node.write(marker)
end

---
---Check if the given node is a marker.
---
---@param item Node
---
---@return boolean
local function is_marker(item)
  local n = item --[[@as UserDefinedWhatsitNode]]
  if n.id == node.id('whatsit') and n.subtype ==
    node.subtype('user_defined') and n.user_id == user_id then
    return true
  end
  return false
end

---
---Test whether the node `n` is a marker and retrieve the
---the corresponding marker data.
---
---@param n UserDefinedWhatsitNode # The argument `n` is a node of unspecified
↪  type.
---
---@return MarkerData|nil # The marker data or nothing if given node is not a
↪  marker.
local function get_marker_data(n)
  if n.id == node.id('whatsit') and n.subtype ==
    node.subtype('user_defined') and n.user_id == user_id then
    return get_storage(n.value --[[@as integer]] )
  end
end

---
---This functions tests, whether the given node `item` is a marker.
---
---@param head_node Node # The current node.
---@param mode MarkerMode
---@param position MarkerPosition
---
---@return boolean
local function check_marker(head_node, mode, position)
```

```
  local data =
    get_marker_data(head_node --[[@as UserDefinedWhatsitNode]] )
  if data and data.mode == mode and data.position == position then
    return true
  end
  return false
end

---
---First this function saves the associatied values of a marker to the
---local options table. Second it returns this values. The argument
---`marker` is a whatsit node.
---
---@param marker UserDefinedWhatsitNode
---
---@return Options|nil
local function get_marker_values(marker)
  local data = get_marker_data(marker)
  if data then
    local_options = data.local_opts
    return data.local_opts
  end
end

---
---`get_marker` returns the given marker.
---
---@param head_node Node # The current node.
---@param mode MarkerMode
---@param position MarkerPosition
---
---@return false|Node # The node if `head_node` is a marker node.
local function get_marker(head_node, mode, position)
  local out
  if check_marker(head_node, mode, position) then
    out = head_node
  else
    out = false
  end
  if out and position == 'start' then
    get_marker_values(head_node --[[@as UserDefinedWhatsitNode]] )
  end
  return out
end

---
---Remove a whatsit marker.
---
---It only deletes a node, if a marker is given.
---
---@param marker Node
---
---@return Node|nil head
---@return Node|nil current
local function remove_marker(marker)
  if is_marker(marker) then
    return node.remove(marker, marker)
  end
end

---@type 'local'|'global'
local options_dest

---
```

```lua
---Store a value `value` and his associated key `key`
---either to the global (`global_options`) or to the local
---(`local_options`) option table.
---
---The global boolean variable `local_options` controls in
---which table the values are stored.
---
---@param key string # The option key.
---@param value any # The value that is stored under the options key.
local function set_option(key, value)
  if value == '' then
    return false
  end
  log.info('Set %s option "%s" to "%s"', options_dest, key,
    value)
  if options_dest == 'global' then
    global_options[key] = value
  else
    local_options[key] = value
  end
end

---
---Set the variable `options_dest`.
---
---@param dest 'local'|'global'
local function set_options_dest(dest)
  options_dest = dest
end

---
---Clear the local options storage.
local function unset_local_options()
  local_options = {}
end

---
---Clear the global options storage.
local function unset_global_options()
  global_options = {}
end

---
---Test whether the value `value` is not empty and has a
---value.
---
---@param value any # A value of different types.
---
---@return boolean # True is the value is set otherwise false.
local function has_value(value)
  if value == nil or value == '' then
    return false
  else
    return true
  end
end

---
---Retrieve a value from a given key. First search for the value in the
---local options, then in the global options. If both option storages are
---empty, the default value will be returned.
---
---@param key string # The name of the options key.
---
```

```lua
---@return any # The value of the corresponding option key.
local function get(key)
  local value_local = local_options[key]
  local value_global = global_options[key]

  local value, source
  if has_value(value_local) then
    source = 'local'
    value = local_options[key]
  elseif has_value(value_global) then
    source = 'global'
    value = value_global
  else
    source = 'default'
    value = defaults[key]
  end

  local g = ansi_color.green

  log.debug(
    'Get value "%s" from the key "%s" the %s options storage',
    g(value), g(key), g(source))

  return value
end

---
---Return the default value of the given option.
---
---@param key any # The name of the options key.
---
---@return any # The corresponding value of the options key.
local function get_defaults(key)
  return defaults[key]
end

local defs = {
  align = {
    description = 'Align the text of a fixed size cloze.',
    process = function(value)
      set_option('align', value)
    end,
  },
  box_height = {
    description = 'The height of a cloze box.',
    alias = { 'boxheight', 'box_height' },
    process = function(value)
      set_option('box_height', value)
    end,
  },
  box_rule = {
    description = 'The thickness of the rule around a cloze box.',
    alias = { 'boxrule', 'box_rule' },
    process = function(value)
      set_option('box_rule', value)
    end,
  },
  box_width = {
    description = 'The width of a cloze box.',
    alias = { 'boxwidth', 'box_width' },
    process = function(value)
      set_option('box_width', value)
    end,
  },
    debug = {
      data_type = 'integer',
      process = function(value)
        log.set(value)
      end,
    },
    distance = {
      description = 'The distance between the cloze text and the cloze line.',
      process = function(value)
        set_option('distance', value)
      end,
    },
    extension_count = {
      description = 'The number of extension units.',
      alias = 'extensioncount',
      process = function(value)
        set_option('extension_count', value)
      end,
    },
    extension_height = {
      description = 'The height of one extension unit (default: 2ex).',
      alias = 'extensionheight',
      process = function(value)
        set_option('extension_height', value)
      end,

    },
    extension_width = {
      description = 'The width of one extension unit (default: 1em).',
      alias = 'extensionwidth',
      process = function(value)
        set_option('extension_width', value)
      end,
    },
    line_color = {
      description = 'A color name to colorize the cloze line.',
      alias = 'linecolor',
      process = function(value, input)
        tex_printf('\\FarbeImport{%s}', value)
        set_option('line_color', value)
      end,
    },
    margin = {
      description =
      ↪  'Indicates how far the cloze line sticks up horizontally from the text.',
      process = function(value)
        set_option('margin', value)
      end,
    },
    min_lines = {
      alias = { 'minimum_lines', 'minlines' },
      description = 'How many lines a clozepar at least must have.',
      process = function(value)
        set_option('min_lines', value)
      end,
    },
    spacing = {
      description = 'The spacing between lines (environment clozespace).',
      process = function(value)
        set_option('spacing', value)
      end,
    },
    text_color = {
      description = 'The color (name) of the cloze text.',
      alias = 'textcolor',
      data_type = 'string',
      process = function(value)
        tex_printf('\\FarbeImport{%s}', value)
        set_option('text_color', value)
      end,
    },
    thickness = {
      description = 'The thickness of the cloze line.',
      process = function(value)
        set_option('thickness', value)
      end,
    },
    visibility = {
      description = 'Show or hide the cloze text.',
      opposite_keys = { [true] = 'show', [false] = 'hide' },
      process = function(value)
        set_option('visibility', value)
      end,
    },
    width = {
      description = 'The width of the cloze line of the command \\clozefix.',
      process = function(value)
        set_option('width', value)
      end,
    },
  }

  ---
  ---@param kv_string string
  ---@param options_dest 'local'|'global'
  local function parse_options(kv_string, options_dest)
    unset_local_options()
    set_options_dest(options_dest)
    luakeys.parse(kv_string, { defs = defs, debug = log.get() > 3 })
  end

  local defs_manager = luakeys.DefinitionManager(defs)

  return {
    get = get,
    get_defaults = get_defaults,
    unset_global_options = unset_global_options,
    unset_local_options = unset_local_options,
    set_options_dest = set_options_dest,
    remove_marker = remove_marker,
    check_marker = check_marker,
    set_option = set_option,
    write_marker = write_marker,
    get_marker = get_marker,
    get_marker_data = get_marker_data,
    parse_options = parse_options,
    defs_manager = defs_manager,
  }

end)()

local utils = (function()
  ---
  ---`utils.create_color()` is a wrapper for the function
  ---`utils.create_colorstack()`. It queries the current values of the
  ---options `line_color` and `text_color`.
  ---
  ---@param kind 'line'|'text'
```

```lua
---@param command 'push'|'pop'
---
---@return PdfColorstackWhatsitNode
local function create_color(kind, command)
  local color_spec
  if kind == 'line' then
    color_spec = config.get('line_color')
  else
    color_spec = config.get('text_color')
  end
  local color = farbe.Color(color_spec)
  return color:create_pdf_colorstack_node(command)
end

---
---Create a rule node, which is used as a line for the cloze texts. The
---`depth` and the `height` of the rule are calculated form the options
---`thickness` and `distance`.
---
---@param width number # The argument `width` must have the length unit __scaled
↪  points__.
---
---@return RuleNode
local function create_line(width)
  local rule = node.new('rule') --[[@as RuleNode]]
  local thickness = tex.sp(config.get('thickness'))
  local distance = tex.sp(config.get('distance'))
  rule.depth = distance + thickness
  rule.height = -distance
  rule.width = width
  return rule
end

---
---Insert a `list` of nodes after or before the `current`. The `head`
---argument is optional. In some edge cases it is unfortately necessary.
---if `head` is omitted the `current` node is used.
---
---@param position 'before'|'after' # The argument `position` can take the values
↪  `'after'` or `'before'`.
---@param current Node
---@param list table
---@param head_node? Node
---
---@return Node
local function insert_list(position, current, list, head_node)
  if not head_node then
    head_node = current
  end
  for _, insert in ipairs(list) do
    if position == 'after' then
      head_node, current = node.insert_after(head_node, current,
        insert)
    elseif position == 'before' then
      head_node, current = node.insert_before(head_node, current,
        insert)
    end
  end
  return current
end

---
---Enclose a rule node (cloze line) with two PDF colorstack whatsits.
---The first colorstack node colors the line, the second resets the
```

```lua
---color.
---
---__Node list__: `whatsit:pdf_colorstack` (line_color) - `rule` (width) -
↪  `whatsit:pdf_colorstack` (reset_color)
---
---@param current Node
---@param width number
---
---@return Node
local function insert_line(current, width)
  return insert_list('after', current, {
    create_color('line', 'push'),
    create_line(width),
    create_color('line', 'pop'),
  })
end

---
---Encloze a rule node with color nodes as the function
-- `utils.insert_line` does.
---
---In contrast to -`utils.insert_line` the three nodes are appended to
---TeX's ‘current-list’. They are not inserted in a node list, which
---is accessed by a Lua callback.
---
---__Node list__: `whatsit:pdf_colorstack` (line_color) - `rule` (width) -
↪  `whatsit:pdf_colorstack` (reset_color)
---
local function write_line_nodes()
  node.write(create_color('line', 'push'))
  node.write(create_line(tex.sp(config.get('width'))))
  node.write(create_color('line', 'pop'))
end

---
---Create a line which stretches indefinitely in the
---horizontal direction.
---
---@return GlueNode
local function create_linefil()
  local glue = node.new('glue') --[[@as GlueNode]]
  glue.subtype = 100
  glue.stretch = 65536
  glue.stretch_order = 3
  local rule = create_line(0)
  rule.dir = 'TLT'
  glue.leader = rule
  return glue
end

---
---Surround a indefinitely strechable line with color whatsits and puts
---it to TeX's ‘current (node) list’ (write).
local function write_linefil_nodes()
  node.write(create_color('line', 'push'))
  node.write(create_linefil())
  node.write(create_color('line', 'pop'))
end

---
---Create a kern node with a given width.
---
---@param width number # The argument `width` had to be specified in scaled
↪  points.
```

```lua
---
---@return KernNode
local function create_kern_node(width)
  local kern_node = node.new('kern') --[[@as KernNode]]
  kern_node.kern = width
  return kern_node
end

---
---Add at the beginning of each `hlist` node list a strut (a invisible
---character).
---
---Now we can add line, color etc. nodes after the first node of a hlist
---not before - after is much more easier.
---
---@param hlist_node HlistNode
---
---@return HlistNode hlist_node
---@return Node strut_node
---@return Node prev_head_node
local function insert_strut_into_hlist(hlist_node)
  local prev_head_node = hlist_node.head
  local kern_node = create_kern_node(0)
  local strut_node = node.insert_before(hlist_node.head,
    prev_head_node, kern_node)
  hlist_node.head = prev_head_node.prev
  return hlist_node, strut_node, prev_head_node
end

---
---Write a kern node to the current node list. This kern node can be
---used to build a margin.
local function write_margin_node()
  node.write(create_kern_node(tex.sp(config.get('margin'))))
end

---
---Search for a `hlist` (subtype `line`) and insert a strut node into
---the list if a hlist is found.
---
---@param head_node Node # The head of a node list.
---
---@return HlistNode|nil hlist_node
---@return Node|nil strut_node
---@return Node|nil prev_head_node
local function search_hlist(head_node)
  while head_node do
    if head_node.id == node.id('hlist') and head_node.subtype == 1 then
      ---@cast head_node HlistNode
      return insert_strut_into_hlist(head_node)
    end
    head_node = head_node.next
  end
end

---
---See nodetree
---@param n Node
local function debug_node_list(n)
  if log.get() < 5 then
    return
  end

  local is_cloze = false
```

```lua
---@param head_node Node
local function get_textual_from_glyph(head_node)
  local properties = node.direct.get_properties_table()
  local node_id = node.direct.todirect(head_node) -- Convert to node id
  local props = properties[node_id]
  local info = props and props.glyph_info
  local textual
  local character_index = node.direct.getchar(node_id)
  if info then
    textual = info
  elseif character_index == 0 then
    textual = '^^@'
  elseif character_index <= 31 or
    (character_index >= 127 and character_index <= 159) then
    textual = '???'
  elseif character_index < 0x110000 then
    textual = utf8.char(character_index)
  else
    textual = string.format('^^^^^^%06X', character_index)
  end
  return textual
end

local output = {}

---
---@param value string
local add = function(value)
  table.insert(output, value)
end

local red = ansi_color.red
local green = ansi_color.green
local yellow = ansi_color.yellow
local blue = ansi_color.blue
local magenta = ansi_color.magenta
local cyan = ansi_color.cyan

while n do
  local marker_data =
    config.get_marker_data(n --[[@as UserDefinedWhatsitNode]] )

  if marker_data then
    if marker_data.position == 'start' then
      is_cloze = true
    else
      is_cloze = false
    end
  end
  if n.id == node.id('glyph') then
    if is_cloze then
      add(yellow(get_textual_from_glyph(n)))
    else
      add(get_textual_from_glyph(n))
    end

  elseif n.id == node.id('glue') then
    add(' ')

  elseif n.id == node.id('disc') then
    add(cyan('|'))

  elseif n.id == node.id('kern') then
```

```lua
        add(blue('<'))

      elseif marker_data then
        local char
        if marker_data.position == 'start' then
          char = 'START'
        else
          char = 'STOP'
        end
        add(magenta('[' .. char .. ']'))

      elseif n.id == node.id('whatsit') and n.subtype ==
        node.subtype('pdf_colorstack') then
        local c = n --[[@as PdfColorstackWhatsitNode]]
        local command
        if c.command == 1 then
          command = 'push'
        elseif c.command == 2 then
          command = 'pop'
        end
        add(green('[' .. command .. ']'))

      elseif n.id == node.id('rule') then
        add(red('_'))

      elseif n.id == node.id('hlist') then

        debug_node_list(n.head)
        add(red(' '))
      end

      n = n.next
    end

    print(table.concat(output, ''))
  end

  return {
    debug_node_list = debug_node_list,
    insert_list = insert_list,
    create_color = create_color,
    insert_line = insert_line,
    write_line_nodes = write_line_nodes,
    write_linefil_nodes = write_linefil_nodes,
    create_kern_node = create_kern_node,
    insert_strut_into_hlist = insert_strut_into_hlist,
    write_margin_node = write_margin_node,
    search_hlist = search_hlist,
  }
end)()

---@param head_node_input Node # The head of a node list.
---
---@return Node # The head of the node list.
local function visit_tree(head_node_input)

  ---This local variables are overloaded by functions
  ---calling each other.
  local continue_cloze, search_stop

  ---
  ---Search for a `hlist` (subtype `line`) and insert a strut node into
  ---the list if a hlist is found.
  ---
```

```lua
---@param head_node Node # The head of a node list.
---
---@return HlistNode|nil hlist_node
local function search_hlist(head_node)
  while head_node do
    if head_node.id == node.id('hlist') and head_node.subtype == 1 then
      ---@cast head_node HlistNode
      return head_node
    end
    head_node = head_node.next
  end
end

---
---Search for a stop marker or make a cloze up to the end of the node
---list.
---
---@param start_node Node # The node to start a new cloze.
---@param parent_node HlistNode # The parent node (hlist) of the start node.
---
---@return Node|nil head_node # The fast forwarded new head of the node list.
---@return Node|nil parent_node # The parent node (hlist) of the head node.
function search_stop(start_node, parent_node)
  ---@type Node|nil
  local n = start_node

  local last_node
  while n do
    if config.check_marker(n, 'basic', 'stop') then
      log.warn('Stop marker found: %s', n)
      return n, parent_node
    end
    last_node = n
    n = n.next
  end
  log.warn('End node: %s', last_node)

  if parent_node.next then
    return continue_cloze(parent_node.next)
  else
    return n, parent_node
  end
end

---
---Continue a multiline cloze.
---
---@param parent_node Node # A parent node to search for a hlist node.
---
---@return Node|nil head_node # The fast forwarded new head of the node list.
---@return Node|nil parent_node # The parent node (hlist) of the head node.
function continue_cloze(parent_node)
  local hlist_node = search_hlist(parent_node)
  if hlist_node then
    local start_node = hlist_node.head
    log.warn('Continue cloze %s %s', hlist_node, start_node)
    return search_stop(start_node, hlist_node)
  end
end

---
---Search for a start marker.
---
---@param head_node Node # The head of a node list.
```

```lua
  ---@param parent_node? HlistNode # The parent node (hlist) of the head node.
  local function search_start(head_node, parent_node)
    ---@type Node|nil
    local n = head_node

    ---@type Node|nil
    local p = parent_node

    while n do
      if n.head then
        ---@cast n HlistNode
        search_start(n.head, n)
      elseif config.check_marker(n, 'basic', 'start') and p and p.id ==
        node.id('hlist') then
        log.warn('Start marker found: %s', n)
        ---@cast p HlistNode
        n, p = search_stop(n, p)
      end
      if n then
        n = n.next
      else
        break
      end
    end
  end
  log.warn('search start: %s', head_node_input)
  search_start(head_node_input)
  return head_node_input
end

---
---Assemble a possibly multiline cloze.
---
---The corresponding LaTeX command to this Lua function is `\cloze`.
---This function is used by other cloze TeX macros too: `\clozenol`,
---`\clozefil`
---
---@param head_node_input Node # The head of a node list.
---
---@return Node # The head of the node list.
local function make_basic(head_node_input)

  utils.debug_node_list(head_node_input)

  ---This local variables are overloaded by functions
  ---calling each other.
  local continue_cloze, search_stop

  ---
  ---Make a single gap.
  ---
  ---@param start_node Node # The node to start / begin a new cloze.
  ---@param stop_node Node # The node to stop / end a new cloze.
  ---@param parent_node HlistNode # The parent node (hlist) of the start and the
  ↪  stop node.
  ---
  ---@return Node|nil stop_node # The stop node.
  ---@return HlistNode parent_node # The parent node (hlist) of the stop node.
  local function make_single(start_node, stop_node, parent_node)
    local node_head = start_node
    local line_width = node.dimensions(parent_node.glue_set,
      parent_node.glue_sign, parent_node.glue_order, start_node,
      stop_node)
  log.info('Make a line of the width of: %s sp', line_width)

  local line_node = utils.insert_line(start_node, line_width)
  local color_text_node = utils.insert_list('after', line_node, {
    utils.create_color('text', 'push'),
  })
  if config.get('visibility') then
    utils.insert_list('after', color_text_node,
      { utils.create_kern_node(-line_width) })
    utils.insert_list('before', stop_node,
      { utils.create_color('text', 'pop') }, node_head)
  else
    line_node.next = stop_node.next
    stop_node.prev = line_node -- not line_node.prev -> line color leaks out
  end
  ---In some edge cases the lua callbacks get fired up twice. After the
  ---cloze has been created, the start and stop whatsit markers can be
  ---deleted.
  config.remove_marker(start_node)
  return config.remove_marker(stop_node), parent_node
end

---
---Search for a stop marker or make a cloze up to the end of the node
---list.
---
---@param start_node Node # The node to start a new cloze.
---@param parent_node HlistNode # The parent node (hlist) of the start node.
---
---@return Node|nil head_node # The fast forwarded new head of the node list.
---@return Node|nil parent_node # The parent node (hlist) of the head node.
function search_stop(start_node, parent_node)
  ---@type Node|nil
  local n = start_node

  local last_node
  while n do
    if config.check_marker(n, 'basic', 'stop') then
      return make_single(start_node, n, parent_node)
    end
    last_node = n
    n = n.next
  end
  -- Make a cloze until the end of the node list.
  n = make_single(start_node, last_node, parent_node)
  if parent_node.next then
    return continue_cloze(parent_node.next)
  else
    return n, parent_node
  end
end

---
---Continue a multiline cloze.
---
---@param parent_node Node # A parent node to search for a hlist node.
---
---@return Node|nil head_node # The fast forwarded new head of the node list.
---@return Node|nil parent_node # The parent node (hlist) of the head node.
function continue_cloze(parent_node)
  local hlist_node = utils.search_hlist(parent_node)
  if hlist_node then
    local start_node = hlist_node.head
    return search_stop(start_node, hlist_node)
    end
  end

  ---
  ---Search for a start marker.
  ---
  ---@param head_node Node # The head of a node list.
  ---@param parent_node? HlistNode # The parent node (hlist) of the head node.
  local function search_start(head_node, parent_node)
    ---@type Node|nil
    local n = head_node

    ---@type Node|nil
    local p = parent_node

    while n do
      if n.head then
        ---@cast n HlistNode
        search_start(n.head, n)
      elseif config.check_marker(n, 'basic', 'start') and p and p.id ==
        node.id('hlist') then
        ---Adds also a strut at the first position. It prepars the
        ---hlist and makes it ready to build a cloze.
        ---@cast p HlistNode
        utils.search_hlist(p)
        n, p = search_stop(n, p)
      end
      if n then
        n = n.next
      else
        break
      end
    end
  end

  search_start(head_node_input)
  return head_node_input
end

---
---The corresponding LaTeX command to this Lua function is `\clozefix`.
---
---@param head_node_input Node # The head of a node list.
local function make_fix(head_node_input)

  ---
  ---Calculate the widths of the whitespace before (`start_width`) and
  ---after (`stop_width`) the cloze text.
  ---
  ---@param start Node
  ---@param stop Node
  ---
  ---@return integer width
  ---@return integer start_width # The width of the whitespace before the cloze
  ↪  text.
  ---@return integer stop_width # The width of the whitespace after the cloze text.
  local function calculate_widths(start, stop)
    local start_width, stop_width
    local width = tex.sp(config.get('width'))
    local text_width = node.dimensions(start, stop)
    local align = config.get('align')
    if align == 'right' then
      start_width = -text_width
      stop_width = 0
  elseif align == 'center' then
    local half = (width - text_width) / 2
    start_width = -half - text_width
    stop_width = half
  else
    start_width = -width
    stop_width = width - text_width
  end
  return width, start_width, stop_width
end

---
---The function `make_single` generates a gap of fixed width.
---
---# Node lists
---
---## Show text:
---
---| Variable name        | Node type | Node subtype          |               |
---|----------------------|-----------|-----------------------|---------------|
---| `start_node`         | `whatsit` | `user_definded`       | `index`       |
---| `line_node`          | `rule`    |                       | `width`       |
---| `kern_start_node`    | `kern`    | Depends on `align`    |               |
---| `color_text_node`    | `whatsit` | `pdf_colorstack`      | Text color    |
---|                      | `glyphs`  | Text to show          |               |
---| `color_reset_node`   | `whatsit` | `pdf_colorstack`      | Reset color   |
---| `kern_stop_node`     | `kern`    | Depends on `align`    |               |
---| `stop_node`          | `whatsit` | `user_definded`       | `index`       |
---
---## Hide text:
---
---| Variable name  | Node type | Node subtype    |         |
---|----------------|-----------|-----------------|---------|
---| `start_node`   | `whatsit` | `user_definded` | `index` |
---| `line_node`    | `rule`    |                 | `width` |
---| `stop_node`    | `whatsit` | `user_definded` | `index` |
---
---Make a fixed sized cloze.
---
---@param start Node # The node, where the gap begins
---@param stop Node # The node, where the gap ends
local function make_single(start, stop)
  local width, kern_start_length, kern_stop_length = calculate_widths(
    start, stop)
  local line_node = utils.insert_line(start, width)
  if config.get('visibility') then
    utils.insert_list('after', line_node, {
      utils.create_kern_node(kern_start_length),
      utils.create_color('text', 'push'),
    })
    utils.insert_list('before', stop, {
      utils.create_color('text', 'pop'),
      utils.create_kern_node(kern_stop_length),
    }, start)
  else
    line_node.next = stop.next
  end
  config.remove_marker(start)
  config.remove_marker(stop)
end

---
---Function to recurse the node list and search after marker.
---
```

```
  ---@param head_node Node # The head of a node list.
  local function make_fix_recursion(head_node)
    ---@type Node|false
    local start_node = false

    ---@type Node|false
    local stop_node = false
    while head_node do
      if head_node.head then
        make_fix_recursion(head_node.head)
      else
        if not start_node then
          start_node = config.get_marker(head_node, 'fix', 'start')
        end
        if not stop_node then
          stop_node = config.get_marker(head_node, 'fix', 'stop')
        end
        if start_node and stop_node then
          make_single(start_node, stop_node)
          start_node, stop_node = false, false
        end
      end
      head_node = head_node.next
    end
  end

  make_fix_recursion(head_node_input)
  return head_node_input
end

---
---The corresponding LaTeX environment to this lua function is
---`clozepar`.
---
---# Node lists
---
---## Show text:
---
---| Variable name      | Node type | Node subtype     |
↪   |
---|--------------------|-----------|------------------|----------------------------|
---| `strut_node`       | `kern`    |                  | width = 0
↪   |
---| `line_node`        | `rule`    |                  | `width` (Width from hlist)
↪   |
---| `kern_node`        | `kern`    |                  | `-width`
↪   |
---| `color_text_node`  | `whatsit` | `pdf_colorstack` | Text color
↪   |
---|                    | `glyphs`  |                  | Text to show
↪   |
---| `tail_node`        | `glyph`   |                  | Last glyph in hlist
↪   |
---| `color_reset_node` | `whatsit` | `pdf_colorstack` | Reset color
---
---## Hide text:
---
---| Variable name | Node type  | Node subtype |                            |
---|---------------|------------|--------------|----------------------------|
---| `strut_node`  | `kern`     |              | width = 0                  |
---| `line_node`   | `rule`     |              | `width` (Width from hlist) |
---
---@param head_node Node # The head of a node list.
local function make_par(head_node)
```

```lua
utils.debug_node_list(head_node)

---
---Add one additional empty line at the end of a paragraph.
---
---All fields from the last hlist node are copied to the created
---hlist.
---
---@param last_hlist_node HlistNode # The last hlist node of a paragraph.
---
---@return HlistNode # The created new hlist node containing the line.
local function add_additional_line(last_hlist_node)
  local hlist_node = node.new('hlist') --[[@as HlistNode]]
  hlist_node.subtype = 1

  local fields = {
    'width',
    'depth',
    'height',
    'shift',
    'glue_order',
    'glue_set',
    'glue_sign',
    'dir',
  }
  for _, field in ipairs(fields) do
    if last_hlist_node[field] then
      hlist_node[field] = last_hlist_node[field]
    end
  end

  local kern_node = utils.create_kern_node(0)
  hlist_node.head = kern_node
  utils.insert_line(kern_node, last_hlist_node.width)
  last_hlist_node.next = hlist_node
  hlist_node.prev = last_hlist_node
  hlist_node.next = nil
  return hlist_node
end

---
---Add multiple empty lines at the end of a paragraph.
---
---@param last_hlist_node HlistNode # The last hlist node of a paragraph.
---@param count number # Count of the lines to add at the end.
local function add_additional_lines(last_hlist_node,
  count)
  local i = 0
  while i < count do
    last_hlist_node = add_additional_line(last_hlist_node)
    i = i + 1
  end
end

---@type Node
local strut_node
---@type Node
local line_node
---@type number
local width
---@type HlistNode
local last_hlist_node
---@type HlistNode
local hlist_node
```

```lua
  local line_count = 0
  while head_node do
    if head_node.id == node.id('hlist') then
      ---@cast head_node HlistNode
      hlist_node = head_node

      line_count = line_count + 1
      last_hlist_node = hlist_node
      width = hlist_node.width
      hlist_node, strut_node, _ = utils.insert_strut_into_hlist(
        hlist_node)
      line_node = utils.insert_line(strut_node, width)
      if config.get('visibility') then
        utils.insert_list('after', line_node, {
          utils.create_kern_node(-width),
          utils.create_color('text', 'push'),
        })
        utils.insert_list('after', node.tail(line_node),
          { utils.create_color('text', 'pop') })
      else
        line_node.next = nil
      end
    end
    head_node = head_node.next
  end

  local min_lines = config.get('min_lines')
  local additional_lines = min_lines - line_count

  if additional_lines > 0 then
    add_additional_lines(last_hlist_node, additional_lines)
  end

  return true
end

local cb = (function()
  ---
  ---@param callback_name CallbackName # The name of a callback
  ---@param func function # A function to register for the callback
  ---@param description string # Only used in LuaLatex
  local function register(callback_name, func, description)
    if luatexbase then
      luatexbase.add_to_callback(callback_name, func, description)
    else
      callback.register(callback_name, func)
    end
  end

  ---
  ---@param callback_name CallbackName # The name of a callback
  ---@param description string # Only used in LuaLatex
  local function unregister(callback_name, description)
    if luatexbase then
      luatexbase.remove_from_callback(callback_name, description)
    else
      callback.register(callback_name, nil)
    end
  end

  ---
  ---Store informations if the callbacks are already registered for
  ---a certain mode (`basic`, `fix`, `par`).
  ---
  ---@type table<'basic'|'fix'|'par'|'visitor', boolean>
  local is_registered = {}

  return {

    ---
    ---Register the functions `make_par`, `make_basic` and
    ---`make_fix` as callbacks.
    ---
    ---`make_par` and `make_basic` are registered to the callback
    ---`post_linebreak_filter` and `make_fix` to the callback
    ---`pre_linebreak_filter`. The argument `mode` accepts the string values
    ---`basic`, `fix` and `par`. A special treatment is needed for clozes in
    ---display math mode. The `post_linebreak_filter` is not called on
    ---display math formulas. I'm not sure if the `pre_output_filter` is the
    ---right choice to capture the display math formulas.
    ---
    ---@param mode MarkerMode
    ---
    ---@return boolean|nil
    register_callbacks = function(mode)
      if mode == 'par' then
        register('post_linebreak_filter', make_par, mode)
        return true
      end
      if not is_registered[mode] then
        if mode == 'basic' then
          register('post_linebreak_filter', make_basic, mode)
          register('pre_output_filter', make_basic, mode)
        elseif mode == 'fix' then
          register('pre_linebreak_filter', make_fix, mode)
        elseif mode == 'visitor' then
          register('pre_output_filter', visit_tree, mode)
        else
          return false
        end
        is_registered[mode] = true
      end
    end,

    ---
    ---Delete the registered functions from the Lua callbacks.
    ---
    ---@param mode MarkerMode
    unregister_callbacks = function(mode)
      if mode == 'basic' then
        unregister('post_linebreak_filter', mode)
        unregister('pre_output_filter', mode)
      elseif mode == 'fix' then
        unregister('pre_linebreak_filter', mode)
      else
        unregister('post_linebreak_filter', mode)
      end
    end,
  }
end)()

---
---Variable that can be used to store the previous fbox rule thickness
---to be able to restore the previous thickness.
local fboxrule_restore

local function print_cloze ()
```

```lua
  local kv_string, text = lparse.scan('O{} v')
  config.parse_options(kv_string, 'local')
  cb.register_callbacks('basic')
  local output = string.format(
    '\\ClozeStartMarker{basic}%s\\ClozeStopMarker{basic}',
    string.format('{\\clozefont\\relax%s}',
      string.format('\\ClozeMargin{%s}', text)))
  tex.print(output)
end

---
---This table contains some basic functions which are published to the
---`cloze.tex` and `cloze.sty` file.
return {
  register_functions = function()
    ---
    ---@param csname string
    ---@param fn function
    local function register_function(csname, fn)
      local index = 376
      local fns = lua.get_functions_table()
      while fns[index] do
        index = index + 1
      end
      fns[index] = fn
      token.set_lua(csname, index, 'global', 'protected')
    end

    register_function('clozeNG', print_cloze)
  end,

  write_linefil_nodes = utils.write_linefil_nodes,
  write_line_nodes = utils.write_line_nodes,
  write_margin_node = utils.write_margin_node,
  set_option = config.set_option,
  set_options_dest = config.set_options_dest,
  unset_local_options = config.unset_local_options,
  reset = config.unset_global_options,
  get_defaults = config.get_defaults,
  get_option = config.get,
  marker = config.write_marker,
  parse_options = config.parse_options,
  register_callback = cb.register_callbacks,
  unregister_callback = cb.unregister_callbacks,

  ---@param count string|number
  print_extension = function(count)
    ---@type number|nil
    local c
    if count == '' then
      c = config.get('extension_count')
    end
    c = tonumber(count)

    if not c then
      luakeys.utils.throw_error_message(
        'clozeextend count must be greater than 0.')
    end

    for _ = 1, c do
      ---ex: vertical measure of x
      ---px: x height current font (has no effect)
      tex_printf('\\hspace{%s}\\rule{0pt}{%s}',
        config.get('extension_width'), config.get('extension_height'))
  end
end,

print_cloze = function(text, kv_string)
  config.parse_options(kv_string, 'local')
  cb.register_callbacks('basic')
  -- config.write_marker('basic', 'start')
  -- tex.print('{\\clozefont\\relax')
  -- utils.write_margin_node()
  -- tex.print(text)
  -- utils.write_margin_node()
  -- tex.print('}')
  -- config.write_marker('basic', 'stop')

  local output = string.format(
    '\\ClozeStartMarker{basic}%s\\ClozeStopMarker{basic}',
    string.format('{\\clozefont\\relax%s}',
      string.format('\\ClozeMargin{%s}', text)))
  tex.print(output)
end,

---
---@param text string
---@param kv_string string
---@param starred string # `\BooleanTrue` `\BooleanFalse`
print_box = function(text, kv_string, starred)
  log.debug('text: %s kv_string: %s starred: %s', text, kv_string,
    starred)
  config.set_options_dest('local')
  config.defs_manager:parse(kv_string, {
    'visibility',
    box_rule = 'rule',
    box_width = 'width',
    box_height = 'height',
  })

  fboxrule_restore = tex.dimen['fboxrule']
  local rule = config.get('box_rule')
  if rule then
    tex.dimen['fboxrule'] = tex.sp(rule)
  end

  tex.print('\\noindent')
  tex.print('\\begin{lrbox}{\\ClozeBox}')

  local height = config.get('box_height')
  local width = config.get('box_width')

  if height then
    tex_printf('\\begin{minipage}[t][%s][t]{%s}', height, width)
  else
    tex_printf('\\begin{minipage}[t]{%s}', width)
  end
  tex.print('\\setlength{\\parindent}{0pt}')
  tex_printf('\\clozenol[margin=0pt]{%s}', text)
  tex.print('\\end{minipage}')
  tex.print('\\end{lrbox}')
  if starred:match('True') then
    tex.print('\\usebox{\\ClozeBox}')
  else
    tex.print('\\fbox{\\usebox{\\ClozeBox}}')
  end
end,
  ---
  ---Print the required TeX markup for the environment `clozespace` using
  ↪  `tex.print()`
  ---
  ---@param kv_string string
  print_space = function(kv_string)
    config.set_options_dest('local')
    local defs = config.defs_manager:include({ 'spacing' }, true)
    defs.spacing.pick = 'number'
    luakeys.parse(kv_string, { defs = defs })
    tex_printf('\\begin{spacing}{%s}', config.get('spacing'))
  end,

  restore_fboxrule = function()
    tex.dimen['fboxrule'] = fboxrule_restore
  end,
}
```