The ucph–revy class*

Kristoffer Levin Hansen
kristoffer@levinhansen.dk

April 4, 2025

Abstract

A LaTeX class for typesetting scripts in the style used for student theatric productions at the University of Copenhagen's (ucph) science faculties. Formerly known to some as `revy.sty`.

Contents

1 Introduction

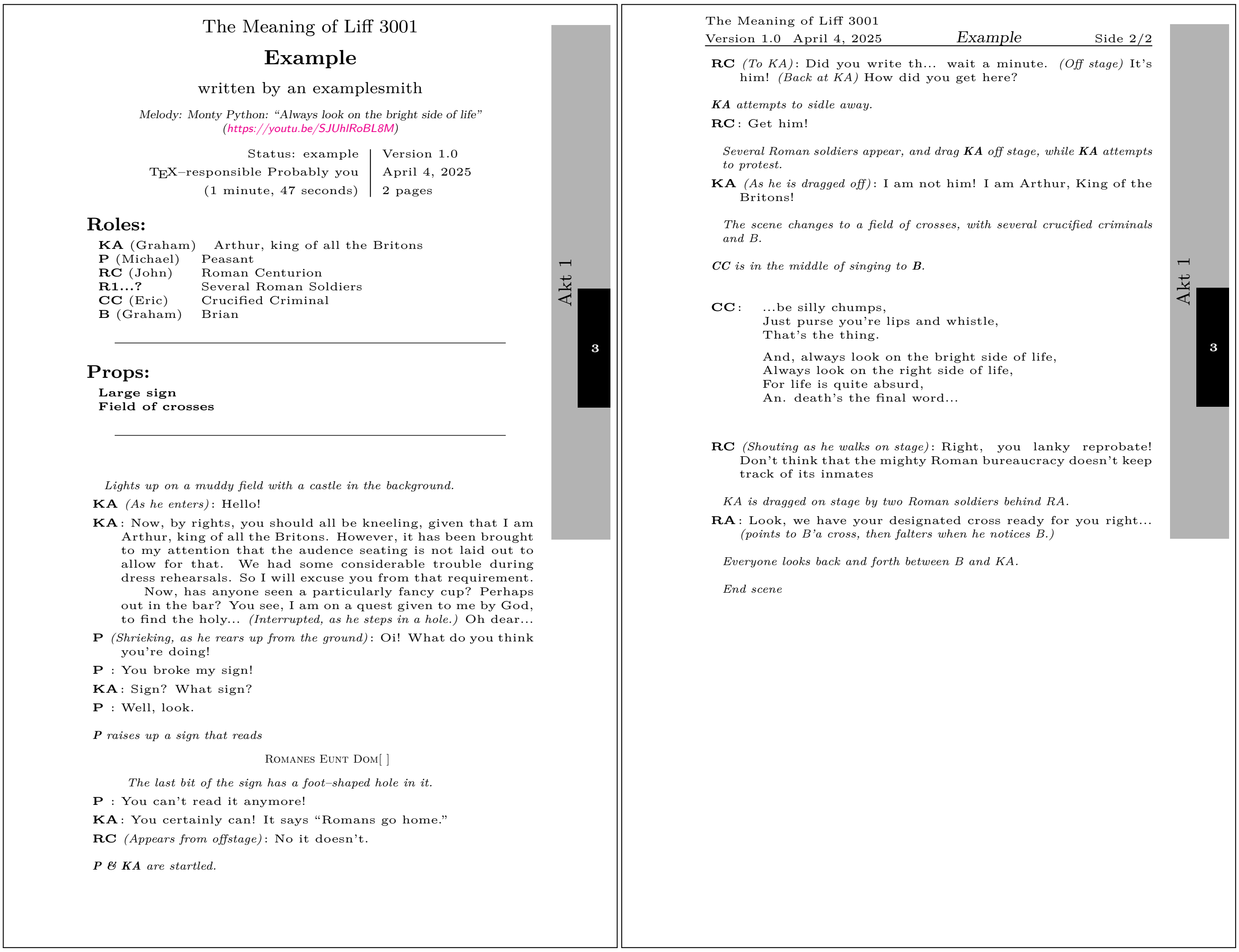
A style of typesetting scripts in LaTeX, originally developed for DIKURevy, the students' revue at DIKU, the Computer Scinece Institute at ucph, the University of Copenhagen. It has spread to adjacent revues at ucph Science as they have sprung into existence. Once known to its users as the package `revy.sty`, that package has been transplanted into this class, at the proper place in the LaTeX hierarchy[4], practically in its entirety.

This class features the ability to set lines of dialogue and song lyrics with a clearly associated indication of the speaker or singer, along with stage directions. The style is intended for a revue, which is formed out of a number of little pieces, so there commands for typesetting a cast list and prop list, along with relevant information like the time to perform, or the name of a larger piece, which are intended to be typeset along with the title.

---

[*] This document corresponds to ucph–revy v1.2.0, dated 2025/03/26.

Figure 1



The Meaning of Liff 3001

Example

written by an examplesmith

Melody: Monty Python: "Always look on the bright side of life"
(*https://youtu.be/SJUhlRoBL8M*)

Status: example | Version 1.0
TEX–responsible Probably you | April 4, 2025
(1 minute, 47 seconds) | 2 pages

Roles:

KA (Graham)	Arthur, king of all the Britons
P (Michael)	Peasant
RC (John)	Roman Centurion
R1...?	Several Roman Soldiers
CC (Eric)	Crucified Criminal
B (Graham)	Brian

Props:

Large sign
Field of crosses

Lights up on a muddy field with a castle in the background.

KA *(As he enters)*: Hello!

KA: Now, by rights, you should all be kneeling, given that I am Arthur, king of all the Britons. However, it has been brought to my attention that the audence seating is not laid out to allow for that. We had some considerable trouble during dress rehearsals. So I will excuse you from that requirement.

Now, has anyone seen a particularly fancy cup? Perhaps out in the bar? You see, I am on a quest given to me by God, to find the holy... *(Interrupted, as he steps in a hole.)* Oh dear...

P *(Shrieking, as he rears up from the ground)*: Oi! What do you think you're doing!

P : You broke my sign!

KA: Sign? What sign?

P : Well, look.

P *raises up a sign that reads*

ROMANES EUNT DOM[ ]

The last bit of the sign has a foot–shaped hole in it.

P : You can't read it anymore!

KA: You certainly can! It says "Romans go home."

RC *(Appears from offstage)*: No it doesn't.

P & KA *are startled.*

The Meaning of Liff 3001
Version 1.0 April 4, 2025 *Example* Side 2/2

RC *(To KA)*: Did you write th... wait a minute. *(Off stage)* It's him! *(Back at KA)* How did you get here?

KA *attempts to sidle away.*

RC: Get him!

*Several Roman soldiers appear, and drag **KA** off stage, while **KA** attempts to protest.*

KA *(As he is dragged off)*: I am not him! I am Arthur, King of the Britons!

The scene changes to a field of crosses, with several crucified criminals and B.

CC *is in the middle of singing to **B**.*

CC: ...be silly chumps,
Just purse you're lips and whistle,
That's the thing.

And, always look on the bright side of life,
Always look on the right side of life,
For life is quite absurd,
An. death's the final word...

RC *(Shouting as he walks on stage)*: Right, you lanky reprobate! Don't think that the mighty Roman bureaucracy doesn't keep track of its inmates

KA is dragged on stage by two Roman soldiers behind RA.

RA: Look, we have your designated cross ready for you right... *(points to B'a cross, then falters when he notices B.)*

Everyone looks back and forth between B and KA.

End scene

An example of what this class can be used to create.

Figure 1 shows an example of how all this looks when typeset.

If you're looking for the script package that's designed to help with typesetting and compositing a full manuscript, it was last seen in [3].

According to `revy.sty`, its origianl authors were Uffe Friis Lichtenberg, Arne John Glenstrup and Anders Komár Ravn.

Change History

v1.0.0
- General: Initial conversion to a class 1
- `\does`: New macro 30
- `thumbindex`: New machinery for thumb indexes 5
- `\maketitle`: New layout of the tile block 16
- `sketch`: The body text macros (`\scene`, etc.) are no longer available outside the `song` and `sketch` environments. 25
- `song`: The body text macros (`\scene`, etc.) are no longer available outside the `song` and `sketch` environments. 27

v1.0.1
- `\movetoleftside`: Adjust tab width 11
- `\revy@itemname`: Eliminated spurious redefinitions. 23
- `\revy@thumbindex`: Don't let it divide by 0 11

v1.0.2
- General: Change fontenc to OT1 . . 4
  - Minor adjustments in example file 36
- `\SaTyR`: SaTyR-macro 30

v1.1.0
- `costumes`: Introduce costumes environment 23
- `\movetoleftside`: Make tabs wider, to clear cutoff when printed 11

v1.2.0
- `thumbindex`: Handle stub entries in planfile. 6
  - Handle comments in planfile . . . 5
  - No longer calls fontenc 5
- `\planfileAllowMacros`: Aid customization of stub commands. 6
- `\revy@tabs`: Move parsing of planfile to the first time a thumb mark is typeset. 7
- `\title`: Optional short title 17
- `\titleeater`: Handle optional short titles in manuscript files 8

2 Usage

An example of a `.tex` file that will typeset something like figure 1 is presented in appendix A, or in the file `Example.tex`, which should have been produced by LaTeX alongside the documentation file that you are reading. In the present section, we'll hit some of the highlights of that file, to understand how it works.

2.1 Invoking and arguments

First, to use the `ucph-revy` class in a `.tex` file, that file must start with

```
\documentclass{ucph-revy}
```

`article` `ucph-revy` inherits from the `article` class, and accepts all its options. It sets `a4paper` and `11pt` by default, if not given any contrary options.

`thumbindex` `planfile=` In addition, ucph–revy add the options `thumbindex` and `planfile`, which activates the construction of thumb indexes in the file and allows setting the file that the thumb index is built from. We'll go through exactly how this works in seciton 2.2.

Implementation

We handle class options here, at the start of the file. To do so, we need this package.

```
\RequirePackage{xstring}
```

`\if@thumbindex` Keeps track of `thumbindex`. We will do stuff with it when we get to section 2.2.

```
\newif\if@thumbindex\@thumbindexfalse
```

And now we're ready to define the arguments:

`thumbindex`

```
\DeclareOption{thumbindex}{\@thumbindextrue}
```

`planfile=` Takes a text string, and so needs particular treatment. xstring defines, among other things, the macro `\IfBeginWith`

```
\newcommand{\planfile}{aktoversigt.plan}
\DeclareOption*{
  \IfBeginWith{\CurrentOption}{planfile=}{
    \def\planfile#1=#2{#2}
    \edef\planfile{\expandafter\planfile\CurrentOption}
  }{
    \PassOptionsToClass{\CurrentOption}{article}
  }
}
\ProcessOptions\relax
\LoadClass[a4paper,11pt]{article}
```

Implementation

These commands make the text block taller.

```
\setlength{\topmargin}{0cm}
\setlength{\voffset}{-1cm}
\setlength{\textheight}{\paperheight}
\addtolength{\textheight}{-4cm}
```

In the example, we've included three packages that aren't strictly necessary for ucph–revy to function, but will probably make your life easier.

`\usepackage [utf8]{inputenc}`
`\usepackage [OT1]{fontenc}`
	Modernize LaTeX's handling of (particularly special) characters.

`\usepackage {hyperref}`	Enables the creation of hyperlinks, like the one in figure 1.
`\urlstyle {sf}`	Demands that hyperlinks be set in sans serif, rather than `monospace`

2.2 Thumb index

`thumbindex` To help organizing a compiled manuscript, which might consist of a collection of many `.tex` files, we profide a facility for creating a thumb index in the typeset margins. That's the gray and black boxes on the edge of the pages in figure 1. It is off by default, since it only really makes sense in a compilation of documents, but is enabled by giving the option `thumbindex` to the class.

Implementation

`\if@thumbindex` `\if@thumbindex` was defined on page 4, and just tells us if the option `thumbindex` was given. If it wasn't, well just skip all the code in this section.

We'll use `\@firstofone` and `\@gobble` to achieve this, so that we can avoid wrapping the section in a long `\if`, and dealing with potential problems from nested `\if`s[1].

```
\if@thumbindex
```

`\revy@readwithhash` `\octothorpe` As we shall see, the planfile might contain `#`s, which we must be able to recognize as comments. To do that, we need a way to read them in without their usual catcode, and something to compare with.

They have to live here, and not inside the argument to `\@firstofone` due to something with the order of character encoding and tokenization[2].

```
  \def\revy@readwithhash{
    \catcode`\#=11
    \read\planread to \line
    \catcode`\#=6
  }
  \catcode`\#=11
  \edef\octothorpe{#}
  \catcode`\#=6
```

```
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi{
```

The structure of the consolidated manuscript is defined in a seperate text file, which contains the filenames of the `.tex` files that make up the manuscript, and divide the show into sections with their own headings. By default, we assume that the plan file is in the same folder as the current `.tex` file, and named `planfile=` `aktoversigt.plan`. This may be changed by passing `planfile=`⟨*filename*⟩ as a parameter to the class. Calling both these options might look like so:

```
\documentclass[thumbindex,planfile=../plan.txt]{ucph-revy}
```

An example of such a plan file might be:

```
# The greatest show, possibly

Act 1
songs/opener.tex
```

---

[1] We get plenty of that from how the code for the documentation is structured...

[2] The details are left as an exercise for the reader.

```
sketches/Example.tex
sketches/anarchocommunist_commune.tex

Act 2
songs/glitzy_dance_act.tex
sketches/crossdressing.tex

Act 3
sketches/stop_police.tex
```

Lines that start with `#` are treated as comments and ignored. Empty lines are also ignored. Lines that end in `.tex` are taken as entries in the thumb index and processed further. Any other lines are treated as act titles.

It is also possible to issue a subset of the info–macros detailed in section 2.3 on a line to create a stub entry. They are meaningful to FysikRevyTeX[3], but ucph–revy will simply ignore them.

`\planfileAllowedMacros` The allowed info–macros are contained in the toks `\planfileAllowedMacros`, which by default contains `\title`, `\eta`, `\status`, `\melody`, `\revyname`, `\revyyear` and `\category` (which is in addition to the macros in section 2.3).

Implementation

```
\newtoks\planfileAllowedMacros
\planfileAllowedMacros={
  \title\eta\status\melody\revyname\revyyear\category
}
```

`\planfileAllowMacros` Further macros can be added to the list with `\planfileAllowMacros`. It takes one or more macros as its argument, eg:

```
\planfileAllowMacros{ \author \responsible }
```

Implementation

```
\def\planfileAllowMacros#1{
  \toks256={#1}
  \edef\act{
    \noexpand\planfileAllowedMacros={
      \the\planfileAllowedMacros \the\toks256
    }
  }
  \act
}
```

Implementation

`\revy@tabs` The code that reads and comprehends the plan file and the listed `.tex` files lives inside this macro. It will be called in the middle of the document flow, so care must be taken to end lines, that could otherwise create extra spaces, with a `%`.

```
\def\revy@tabs{%
  \begingroup
```

`\stubCommand` `\markStubCommands` `\swap` `\skipFiFi` `\nothing`

We will enable ourselves to skip the macros in `\planfileAllowedMacros` by setting each of them equal to a recognizable macro.

```
    \def\stubCommand{}%
    \def\swap##1##2{##2##1}%
    \def\skipFiFi##1\nothing{\fi\fi}%
    \let\nothing=\relax
    \def\markStubCommands##1{%
      \ifx\@undefined##1 \else
        \ifx\relax##1%
          \skipFiFi
        \fi
      \fi
      \let##1=\stubCommand
      \swap\markStubCommands\nothing
    }
    \edef\act{\noexpand\markStubCommands\the\planfileAllowedMacros\relax}%
    \act
```

The construction where `\act` is defined, and then immediately called, controls how deeply TeX will expand our macros[1].

`\findStubCommands` `\ifstub`

We then need to be able to recognize that macro later.

```
    \newif\ifstub \stubfalse
    \def\findStubCommands##1{%
      \ifx\stubCommand##1%
        \stubtrue
      \fi
      \ifx\relax##1 \else
        \expandafter\findStubCommands
      \fi
    }%
```

`\expandFiHere` We shall need this macro, which you may reckognize as `\hop` if you've read [2].

```
    \def\expandFiHere##1\fi{\fi ##1}%
```

We will need to know the path of the plan file

```
    \def\patheat##1/##2{%
      \ifx\relax##2\else
        \expandFiHere##1/\patheat##2%
      \fi
    }%
    \edef\planpath{\expandafter\patheat\planfile/\relax}%
```

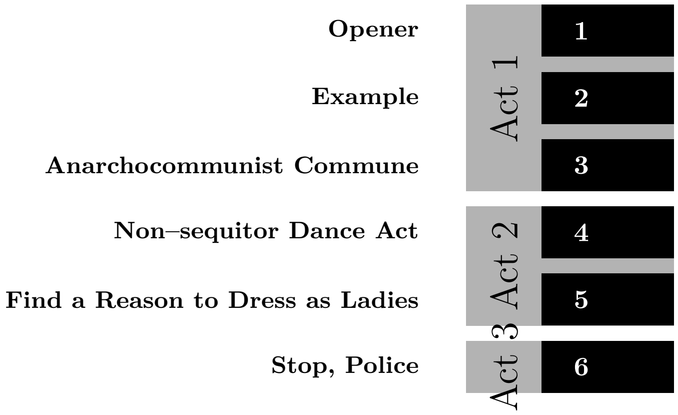
Note that the folder names (`songs`, `sketches`) carry significance for FysikRevyTeX [3], but not for ucph–revy. The `.tex` files can be placed at any location that TeX can read.

Implementation

```
\newread\planread
\openin\planread=\planfile
```

From this plan file, the class can generate a thumb mark for each of thsee individual files, as demonstrated in figure 1, which shows the location of that file in the larger structure of the piece. It can also generate a thumb index, as demonstrated in figure 2, which outlines the location of every part of the piece, and can serve as an aide to navigation.

Figure 2



An example thumb index. This example is compressed vertically. It is intended to be nearly the height of the page, to match the thumb markings (such as the ones in figure 1).

The titles in the thumb index demonstrated in figure 2 are taken from the content of the `\title` macro in the files listed in the plan file. Also, the routine for setting the thumb mark for an individual file determine *which* file it is in by comparing the content of the `\title` macro in the files in the plan file with the title that TEX knows about in the current file.

Therefore, it is necessary for the thumb index to work that a `\title` is declared in each of the files that are listed in the plan file.

If there are more than one `\title` invocations in a file, the thumb index routine will use the last one. If several of the files in the plan file have the same name, those files will end up with several thumb marks.

Implementation

`\revy@tabs` `\skipopt` `\titleeater` In the world of thumbtabs, we will simply filter out the optional short titles (see p. 17) while we're searching for `\title` definitions. The macros for this purpose are:

```
                                  %[
        \def\skipopt##1]##2{\titleeater{##2}}%
        \def\titleeater##1##2\title##3{%
```

```
    \ifx\done##3%
      ##1%
    \else
      \ifx##3[%               ]
        \expandFiHere\expandFiHere\skipopt
      \else
        \expandFiHere\expandFiHere\titleeater{##3}%
      \fi
    \fi
  }%
```

To try and ward off issues caused by newlines inside `\title` definitions in the external file, we'll disable TEXs internal handling of newlines

```
  \endlinechar=-1
```

```
  \newread\tabtex
```

Ultimately, `\revy@tabs` ought to redefine itself so that it will contain, for each heading in the plan file, the sequence

`\undertab` ⟨*counter at start*⟩ `\text` ⟨*hading*⟩ `\stop`,

for each file name in the plan file, the sequence

`\overtab` ⟨*counter*⟩ `\text` ⟨*title in file*⟩ `\stop`,

and finally the sequence

`\undertab` ⟨*no. of files*⟩ `\text` `\done` `\stop`.

`\tabtoks` As we go, these sequences will be built up in a `toks`.

```
  \newtoks\tabtoks
```

`\tabscount` And we'll keep a count of how many tabs to make

```
  \newcount\tabscount \tabscount=0
```

`\planfile` We now `\loop` over the lines in , and check whether there are any more lines,

```
  \loop\ifeof\planread
    \closein\planread
    \closein\tabtex
  \else
```

wether we're in the midde of reading a `.tex` file,

```
    \ifeof\tabtex
      \let\readline\empty
      \revy@readwithhash
```

wether we're looking at a line with stub commands (which should be ignored),

```
      \stubfalse
      \expandafter\findStubCommands\line\relax
      \ifstub \else
```

wether we're looking at an empty line (which should be ignored),

```
        \IfStrEq{}{\line}{}{%
```

wether we're looking at a comment (which should be ignored)

```
          \IfBeginWith{\line}{\octothorpe}{}{%
```

and wether the line ends in `.tex`. If it does, we must have found the next `.tex` file to check.

```
              \IfEndWith{\line}{.tex}{%
                \advance\tabscount by 1 %
```

`\set@curr@file` beats down issues with catcodes in filenames.

```
                \set@curr@file{\line}%
                \openin\tabtex={\planpath\@curr@file}%
              }{%
```

`\tabtoks`

Otherwise, we must have found an act heading, and save the appropriate sequence in `\tabtoks`, as described above.

```
                \edef\act{\noexpand\tabtoks={%
                  \the\tabtoks
                  \noexpand\undertab
                  \number\tabscount
                  \noexpand\text
                  \line
                  \noexpand\stop
                }}\act
              }%
            }%
          }%
        \fi
      \else
```

If we *are* in the middle of reading a `.tex` file, we'll continue doing that (this thing, where we choose which file we're reading with an `\ifeof` means that there is no explicit inner `\loop` for reading the `.tex` files).

```
        \read\tabtex to \subline
        \edef\readline{%
          \expandafter\titleeater
          \expandafter\readline\subline\title\done
        }%
        \ifeof\tabtex
          \edef\act{\noexpand\tabtoks={%
            \the\tabtoks
            \noexpand\overtab
            \number\tabscount
            \noexpand\text
            \readline
            \noexpand\stop
          }}\act
        \fi
      \fi
    \repeat
```

`\tabtoks`

Having read the entire plan file, we can save the final sequence in `\tabtoks`, before we save the entire sequence in `\revy@tabs`, which is where the rest of the class' code can access it.

```
    \edef\act{\noexpand\tabtoks={%
        \the\tabtoks
        \noexpand\undertab
        \number\tabscount
        \noexpand\text
        \noexpand\done
        \noexpand\stop
      }}\act
    \xdef\revy@tabs{\the\tabtoks}%
  \endgroup
```

Now that `\revy@tabs` has the desired content, we'll call it as the last step in this macro, so that this call to `\revy@tabs` yelds the same result as all subsequent calls.

```
  \revy@tabs
}
```

Ti*k*Z is not a drawing program[5], but we'll use it to draw our thumbtabs regardless.

```
\RequirePackage{tikz}
```

`\revy@thumbindex` This macro contains the procedures that typeset the thumb index. Its argument is expanded immediately before the thumb tabs are actually typeset, allowing you to modify the macro's internal state at that point. For this purpose, you'll want to pay special attention to the macros `\filter`, `\movetoleftside` and `\writetitlestrue`, which will be defined internally in this macro, and alter which parts of the thumbindex are typeset.

The approach that we will see used, as we go through the macro, is to define `\undertab` and `\overtab` in such a way that they perform the tasks that we want to see performed, when we expand `\revy@tabs`.

```
\def\revy@thumbindex#1{%
  \begingroup
```

`\expandFiHere` This one needs to be defined in this group as well

```
    \def\expandFiHere##1\fi{\fi ##1}
```

Find the height of a single tab.

```
    \newdimen\tabheight
    \tabheight=\paperheight
    \advance\tabheight by -1cm%
    \def\undertab##1\text##2\stop{
      \ifx\done##2%
        \divide\tabheight by \ifnum0=##1 1 \else ##1\fi%
      \fi
    }
    \def\overtab##1\text##2\stop{}%
    \revy@tabs
```

`\movetoleftside` These definitions contain the differences between right and left facing thumb tabs.

```
    \def\overtableft{-.75cm}
    \def\overtabright{1.0cm}
    \def\undertabright{\overtabright}
    \def\undertableft{\overtableft - 1cm}
    \def\underangle{|-}
    \def\underpos{.25}
    \def\undersign{}
    \def\underlabeloffset{\undersign.5cm}
```

```
\def\textanchorcorner{east}
\def\textoffset{\undertableft - .5cm}
\def\textside{right}
\def\movetoleftside{
```

```
  \def\overtableft{-2.3cm}
  \def\overtabright{.2cm}
  \def\undertabright{\overtabright + 1cm}
  \def\undertableft{\overtableft}
  \def\underangle{-|}
  \def\undersign{-}
  \def\underpos{.75}
  \def\textanchorcorner{west}
  \def\textoffset{\undertabright + .5cm}
  \def\textside{left}
}
```

`\ifwritetitles`

```
\newif\ifwritetitles\writetitlesfalse
```

Here, we attach meaning to the tokens we stuffed into `\revy@tabs` previously.

```
\def\printovertab##1##2{
  \fill ( 0, -##1\tabheight )
  node[ text=white, anchor=east ]{\bfseries##1}
    +( \overtableft, -.5\tabheight + 1mm )
    rectangle +( \overtabright, .5\tabheight - 1mm )
    +( \textoffset, 0 )
  \ifwritetitles
    node[ anchor=\textanchorcorner ]{\small\bfseries##2}
  \fi
  ;
}%
```

```
\def\printundertab##1##2##3{%
  \path ( 0, -##1\tabheight ) +( \undertabright, -.5\tabheight - 1mm )
    coordinate (topright);
  \path ( 0, -##2\tabheight ) +( \undertableft, -.5\tabheight + 1mm )
    coordinate (bottomleft);
  \fill[ color=black!30 ] (topright) rectangle (bottomleft);
  \path (bottomleft)
  \underangle node[ xshift=\underlabeloffset,
                    pos=\underpos,
                    rotate=\undersign90
                   ]{\Large##3}
    (topright);
}%
```

```
\def\overtab##1\text##2\stop{%
  \printovertab{##1}{##2}
}
```

```
\def\undertab##1\text##2\stop{
  \ifx\done##2\else
    \let\undertab\undertabspan
    \expandFiHere
    \undertab##1\text##2\stop
```

```
   \fi
  }
  \def\undertabspan##1\text##2\stop##3\undertab##4\text##5\stop{%
   \printundertab{##1}{##4}{##2}%
   ##3%
   \ifx\done##5\else
    \expandFiHere
    \undertab##4\text##5\stop
   \fi
  }%
```

`\filtername`

```
  \def\filtername##1{
   \def\doWhenMatch{}
   \def\overtab####1\text####2\stop{
    \IfStrEq{##1}{####2}{
     \doWhenMatch
     \printovertab{####1}{####2}
    }{}
   }
   \def\undertabspan####1\text####2\stop####3\undertab####4\text####5\stop{
    \def\doWhenMatch{
     \printundertab{####1}{####4}{####2}
     \def\doWhenMatch{}
    }
    ####3
    \ifx\done####5\else
     \expandFiHere
     \undertab####4\text####5\stop
    \fi
   }
  }
```

As promised, the macro argument, just before the thumbindex is actually typeset.

```
  #1
  \begin{tikzpicture}
   \revy@tabs
   \path (0, 0) -- (0, -\paperheight + .5cm);
  \end{tikzpicture}
 \endgroup
}
```

And thus concludes `\revy@thumbindex`.

In the remaider of this section, we list the macros that may be used to typeset the thumb indices.

`\rectothumbtabfor`

`\rectothumbtabfor{`⟨*title*⟩`}`

Sets a thumb tab, oriented to the right, so suitable for a recto page, that corresponds to the file (or files) whose discovered title matches the argument. The most sensible argument to give it is the title of the current document, which can be found in `\@title`. This is the macro that typeset the thumb tabs in figure 1.

Implementation

```
\newcommand{\rectothumbtabfor}[1]{
  \revy@thumbindex{
    \filtername{#1}
  }
}
```

`\rectothumbindexwithtitles`

`\rectothumbindexwithtitles`

Sets the thumbs index, as was demonstrated in figure 2. Do note that the example in figure 2 has been squeezed vertically. By default, the index is scaled vertically after `\paperheight`, like the thumb tabs in figure 1 were.

Implementation

```
\newcommand{\rectothumbindexwithtitles}{
  \revy@thumbindex{
    \writetitlestrue
  }
}
```

`\rectothumbindex`

`\rectothumbindex`

Sets all the thumb marks, just like the previous macro, but without writing out the titles.

Implementation

```
\newcommand{\rectothumbindex}{
  \revy@thumbindex{}
}
```

`\rectothumbtabwithtitlefor`

`\rectothumbtabwithtitlefor{`⟨*title*⟩`}`

Sets only the thumb tab for the entr(y/ies) with the title given in the parameter. It's unclear why you'd want to, but the option is included, for completeness.

Implementation

```
\newcommand{\rectothumbtabwithtitlefor}[1]{
  \revy@thumbindex{
    \filtername{#1}
    \writetitlestrue
  }
}
```

`\versothumbtabfor`
`\versothumbindexwithtitles`
`\versothumbindex`
`\versothumbtabwithtitlefor`

These macros also come in left oriented versions, suitable for the margin of verso pages.

`\versothumbtabfor{`⟨*titel*⟩`}`

`\versothumbindexwithtitle`

`\versothumbindex`

`\versothumbtabwithtitlefor{`⟨*titel*⟩`}`

Implementation

```
\newcommand{\versothumbindex}{
  \revy@thumbindex{
    \movetoleftside
  }
}
\newcommand{\versothumbindexwithtitles}{
  \revy@thumbindex{
    \movetoleftside
    \writetitlestrue
  }
}
\newcommand{\versothumbtabfor}[1]{
  \revy@thumbindex{
    \movetoleftside
    \filtername{#1}
  }
}
\newcommand{\versothumbtabwithtitlefor}[1]{
  \revy@thumbindex{
    \movetoleftside
    \filtername{#1}
    \writetitlestrue
  }
}
```

Implementation

And thus concludes the conditional group for `\if@thumbindex`.

```
}
```

The thumbindex hasn't been enabled in the example file in appendix A, since there's nothing to index. However, if you were to place the example plan file above in a file along side `Example.tex` named `aktoversigt.plan`, and replace the class invocation at the top of `Example.tex` with

```
\documentclass[thumbindex]{ucph-revy}
```

then the typeset result should have a thumb marking like the one in figure 1. Nonexistent files are simply ignored when constructing the thumb index.

You might also like to have a page with an index of the thumb markings, something similar to figure 2, but scaled to a full page. Appendix B hold a complete example of one way to make a page like that. If you have access to the package's source files, you can also extract that code as the file `Thumbindex.tex`, by running the file `ucph-revy-ex-thumb.ins` through LaTeX.

2.3 The infoblock

ucph–revy's modified title block contains some additinal information that is useful in organizing a show. Figure 3 shows an example of how the title block looks with every possilbe piece of informaiton included.

Figure 3

`\maketitle`

The Meaning of Liff 3001

An example to illustrate the use of ucph–revy

written by an examplesmith

Melody: Monty Python: "Always look on the bright side of life"
(https://youtu.be/SJUhlRoBL8M)

Status: example	Version 1.0
TEX–responsible Probably you	April 4, 2025
(1 minute, 47 seconds)	2 pages

An example of how ucph–revy's tilte block looks with every possible piece of information filled in. It is set with the command `\maketitle`.

We give ucph–revy this information with a series of preamble commands. For the sake of the example, the example file includes all of these commands, but they are not required for a ucph–revy document to compile. If they don't make sense to include in a document, just leave them out. The macros were:

`\version` `\version{1.0}`

Implementation

```
\def\version#1{\def\@version{#1}}
```

`\version` is the only one of these commands that may not be omitted, because version conrol is important[3].

Implementation

The version number is made a requirement by these commands:

```
\def\@version{\@ifundefined{the@version}
{\typein[\versionsnr]{Indtast revyteksten versionsnummer: }%
\global\let\the@version=\versionsnr}{}\the@version}
```

---

[3] You might be of the opinion that there are better ways of doing version control in this far future year 2023. In which case, you may take this as a reminder to actually use one of them.

`\revyname` `\revyyear` `\title`

```
\revyname{The Meaning of Liff}
\revyyear{3001}
\title[Example]{An example to illustrate the use of ucph--revy}
```

Implementation

```
\def\revyname#1{\def\@revyname{#1}}
\def\revyyear#1{\def\@revyyear{#1}}
```

This demonstrates the use of the optional parameter to `\title`, which is a short version of the title. The short title is put in the page headers, and can be used for content lists. In the example in appendix A, the optional parameter is skipped, since its title is of a reasonable length.

Implementation

```
\let\revy@oldtitle=\title
\renewcommand{\title}[2][\relax]{%
  \ifx\relax#1 \shorttitle{\@title}\else\shorttitle{#1}\fi
  \revy@oldtitle{#2}%
}
```

`\shorttitle` The short title may also be set with the macro `\shorttitle{`⟨*short title*⟩`}`.

Implementation

```
\newcommand{\shorttitle}[1]{\def\theshorttitle{#1}}
\shorttitle{\@title}
```

If ucph–revy isn't given a revue name, year or a title, it will use the defaults of "DIKUrevy", "1973" and "En sketch".

Implementation

```
\def\@revyyear{1973}
\def\@revyname{DIKUrevy}
\def\@title{En sketch}
```

`\author` `\status` `\eta` `\responsible` `\melody` The rest aren't typeset if they are not defined.

```
\author{an examplesmith}
\status{example}
\eta{$1$ minute, $47$ seconds}
\responsible{Probably you}
\melody{Monty Python: ``Always look on the bright side of life''
  (\url{https://youtu.be/SJUhlRoBL8M})}
```

Implementation

These new `\if` macros keep track of which of these things have been defined.

```
\newif\if@author\@authorfalse
\newif\if@status\@statusfalse
\newif\if@eta\@etafalse
\newif\if@responsible\@responsiblefalse
\newif\if@melody\@melodyfalse
```

And these macros define them:

```
\def\author#1{\def\@author{#1}\@authortrue}
\def\status#1{\def\@status{#1}\@statustrue}
\def\eta#1{\def\@eta{#1}\@etatrue}
\def\responsible#1{\def\@responsible{#1}\@responsibletrue}
\def\melody#1{\def\@melody{#1}\@melodytrue}
```

Ensuring that the internal macros aren't undefined:

```
\def\@author{}
\def\@status{}
\def\@eta{}
\def\@responsible{}
\def\@melody{}
```

`\auteurs` And as a bonus, we've retained `\auteurs` as an alias of `\author`, for fancy folk.

Implementation

```
\let\auteurs\author
```

`\writtenbyname`
`\melodyname`
`\responsibletext`
`\statustext`

```
\writtenbyname{written by}
\melodyname{Melody:}
\responsibletext{\TeX--responsible}
```

The labels on some of the bits of info are in danish by default: "skrevet af", "Melodi:" and "TEX–ansvarlig:", respectively. These commands overwrite those defaults. There is also a command for changing the "Status" label, which happens not to be necessary in english:

`\statustext {⟨text⟩}`

Implementation

```
\def\writtenbyname#1{\def\@writtenbyname{#1}}
\def\melodyname#1{\def\@melodyname{#1}}
\def\responsibletext#1{\def\@responsibletext{#1}}
\def\statustext#1{\def\@statustext{#1}}
\def\@writtenbyname{skrevet af}
\def\@melodyname{Melodi:}
\def\@responsibletext{\TeX--ansvarlig:}
\def\@statustext{Status:}
```

`\pagessum` The page count is a touch more complicated, and requires a macro that takes 1 argument (a ⟨*number*⟩), and evaluates to the grammatically correct phrase for that page count. Thus `\pagessum{0}` becomes “0 sider” (the plural), while `\pagessum{1}` becomes “1 side” (the singular).

The default macro, for danish, is

```
\newcommand{\pagessum}[1]{#1 side\ifnum1=#1\else r\fi}
```

In the example, such a macro is given for english as

```
\renewcommand{\pagessum}[1]{#1 page\ifnum#1=1\else s\fi}
```

Implementation

`\maketitle` is rewritten to create a title segment like what was shown in figure 3.

```
\def\maketitle{{\thispagestyle{empty}
  \vspace*{-\headheight}\vspace*{-\headsep}
  \centering{\Large\@revyname{} \@revyyear}\vspace{5pt}\\
  {\LARGE \bf \@title}\vspace{5pt}\\
  \if@author{\large\@writtenbyname{} \@author}\vspace{5pt}\\\fi
  \if@melody{\sl\@melodyname{} \@melody}\vspace{5pt}\\\fi

  \if@leftfields
    \begin{varwidth}[t]{\bigheaderwidth}\raggedleft
      \if@status{\@statustext{} \@status}\vspace{2pt}\\\fi
      \if@responsible{\@responsibletext{} \@responsible}\vspace{2pt}\\\fi
      \if@eta{(\@eta)}\\\fi
    \end{varwidth}%
```

The explicit space here keeps the foratting regular.

```
    \ \hspace{.5em}\vrule{}\hspace{.5em}
    \begin{varwidth}[t]{\smallheaderwidth}\raggedright
      Version \@version\vspace{2pt}\\
      \today\vspace{2pt}\\
      \pagessum{\getpagerefnumber{lastpage}}
    \end{varwidth}

  \else
    Version \@version
    \ \hspace{.5em}\vrule\hspace{.5em}
    \today
    \ \hspace{.5em}\vrule\hspace{.5em}
    \pagessum{\getpagerefnumber{lastpage}}
  \fi
  \vskip 5pt }}
```

`varwidth` (*env.*) The environment `varwidth` controls the placement of the two–column part of the title block. It comes from

```
\RequirePackage{varwidth}
```

`\getpagerefnumber` is an expandable version of `\pageref` from

```
\RequirePackage{refcount}
```

`\if@leftfields` This macro is defined to effectively be[4] `\if@responsible` ∨ `\if@status` ∨ `\if@eta`. In the absence of a real ∨–operator for TeX's `\if`, it uses 1's and 0's as standins for boolean values.

```
\def\if@leftfields{
  \def\@responsiblenum{\ifx\iftrue\if@responsible 1 \else 0 \fi}
  \def\@statusnum{\ifx\iftrue\if@status 1 \else 0 \fi}
  \def\@etanum{\ifx\iftrue\if@status 1 \else 0 \fi}

  \ifnum1=\ifnum1=\@responsiblenum 1
    \else \ifnum1=\@statusnum 1
    \else \ifnum1=\@etanum 1
    \else 0
  \fi\fi\fi
}
```

`\smallheaderwidth` `\bigheaderwidth` And these lengths were used.

```
\newlength{\smallheaderwidth}
\setlength{\smallheaderwidth}{22ex}
\newlength{\bigheaderwidth}
\setlength{\bigheaderwidth}{\textwidth}
\addtolength{\bigheaderwidth}{-\smallheaderwidth}
\addtolength{\bigheaderwidth}{-1.5em}
```

2.3.1 Page headers

The class also defines its own header style, which is illustrated in figure 4, and contains some of the same information.

Figure 4

The Meaning of Liff 3001
Version 1.0 April 4, 2025 *Example* Side 2/2

An example of the headers that ucph–revy define.

These headers belong ot a new `pagestyle` named `revyheadings`

Implementation

`\ps@revyheadings`

```
\gdef\ps@revyheadings{
  \def\@oddhead{\vbox to 0pt{\vss
      \hbox to \textwidth{\hfil\rectoheaderthumbtab}
      \hbox{\rm\strut\@revyname{} \@revyyear}
      \hbox to \textwidth{Version \@version\quad\today
        \hfil {\large\sl\theshorttitle}\hfil
        Side \rm\thepage/\pageref{lastpage}}
      \hrule}}
```

---

[4] ∨ being logical OR.

```
  \def\@evenhead{\vbox to 0pt{\vss
      \hbox{\versoheaderthumbtab}
      \hbox to \textwidth{\hfil\rm\strut\@revyname{} \@revyyear}
      \hbox to \textwidth{Side \rm\thepage/\pageref{lastpage}
        \hfil {\large\sl\theshorttitle}\hfil
        Version \@version\quad\today}
      \hrule}}
  \def\@oddfoot{}\def\@evenfoot{}}
```

`\rectoheaderthumbtab` `\versoheaderthumbtab` Here, the `\rectoheaderthumbtab` and `\versoheaderthumbtab` are macros that place the thumb markings correctly in relation to the page header, and call the drawing macro, so long as `ucph-revy` has been given the option `thumbindex`, see section 2.2. If not, they do nothing.

```
\newcommand{\rectoheaderthumbtab}{%
  \if@thumbindex
    \rectothumbtabfor{\@title}%
    \vspace*{ -\paperheight }\vspace*{ -\voffset }\vspace*{ 0.32in }%
    \hspace{-\paperwidth }\hspace{ 1in }\hspace{ \oddsidemargin }%
    \hspace{ \hoffset }\hspace{ \textwidth }\hspace{-.6cm}%
  \fi
}
\newcommand{\versoheaderthumbtab}{%
  \if@thumbindex
    \hspace{ -2.5in }\hspace{ -\evensidemargin }%
    \hspace{ -\hoffset }\hspace{ -1in }%
    \versothumbtabfor{\@title}%
    \vspace*{ -\paperheight }\vspace*{ -\voffset }\vspace*{ 0.32in }%
  \fi
}
```

The new `pagestyle` is activated by default.

Implementation

```
\pagestyle{revyheadings}
```

`\ps@empty` But we also redefine the `pagestyle` “enpty” to be

```
\gdef\ps@empty{
  \def\@oddhead{\vbox to 0pt{\vss
      \hbox to \textwidth {\hfil\rectoheaderthumbtab }
      \vbox to 2em{}}}
  \def\@evenhead{\vbox to 0pt{\vss\hbox{\versoheaderthumbtab}}}
  \def\@oddfoot{}\def\@evenfoot{}}
```

2.4 Assignement lists

We provide a number of environments for setting information about the material in a file in lists. In the example file, we produce something like the lists shown in figure 5, with the passage

```
\rolename{Roles:}
\begin{roles}
```

```
  \role{KA}[Graham] Arthur, king of all the Britons
  \role{P}[Michael] Peasant
  \role{RC}[John] Roman Centurion
  % [...]
\end{roles}
\propname{Props:}
\begin{props}
  \prop{Large sign}
  \prop{Field of crosses}
\end{props}
```

Figure 5

Roles:

KA (Graham)	Arthur, king of all the Britons
P (Michael)	Peasant
RC (John)	Roman Centurion

Props:

Large sign
Field of crosses

Examples of the assignment lists that can be typeset with ucph–revy, for noting cast lists, among other things.

Implementation

`revy@list` (*env.*) Both those environments are derived from this environment.

```
\newenvironment{revy@list}[1]%
{{\smallskip\noindent\Large\bf#1}
  \begin{list}{}{
      \labelwidth 8em
      \leftmargin 10em
      \rightmargin 0em
      \labelsep 1em
      \listparindent 0em
      \topsep 1ex
      \partopsep 0ex
      \parsep 1ex
      \itemsep -1ex\relax
    }%
  }%
  {\end{list}\smallskip\hfil\rule{6cm}{0.1mm}\medskip\par}
```

The argument that this environment takes is the title for the list.

`\revy@itemwithout`
`\revy@itemwith`
`\revy@itemname`

As the `revy@list` environment is derived from the `list` environment, it expects entries in its list to ultimately be called with `\item`. However, for the environments presented to the end user, we like to provide item commands that more closely match the intended use. `\revy@itemname` bundles the process of creating those commands, while `\revy@itemwith` and `\revy@itemwithout` hold the translation to an `\item` command.

```
\def\revy@itemwithout#1{\item [{\bf #1} \hfill] }
\def\revy@itemwith#1[#2]{\item [{\bf #1} (#2) \hfill] }
\def\revy@itemname#1{%
  \expandafter\def\csname #1\endcsname##1{%
    \@ifnextchar [ %]
    {\revy@itemwith{##1}}{\revy@itemwithout{##1}}%
  }
}
```

The argument is the name of the macro that will be created for setting items on our lists.

Some examples of how these are used will follow.

`roles` (*env.*)

In the role list, each role is set with the macro

`\role{`⟨*abr.*⟩`}[`⟨*actor*⟩`]`⟨*description*⟩

Implementation

The roles list environment is here derived from `revy@list`.

```
\newenvironment{roles}{
  \begin{revy@list}{\@rolename}
    \revy@itemname{role}
 }{
  \end{revy@list}
}
```

`props` (*env.*)
`costumes` (*env.*)
`mics` (*env.*)
`instructors` (*env.*)

Beyond the roles environment, **ucph-revy** defines these four additional environments, for typesetting lists that may be useful to a production. Of these, the examples in figures 1 and 5 used `props`.

Within these environments, `props`, `costumes`, `mics` and `instructors`, the macros for setting a list item are

`\prop{`⟨*prop*⟩`}[`⟨*responsible*⟩`]`⟨*description*⟩,

`\costume{`⟨*costume*⟩`}[`⟨*responsible*⟩`]`⟨*description*⟩,

`\mic{`⟨*abr.*⟩`}[`⟨*actor*⟩`]`⟨*microphon*⟩

and

`\instructor[`⟨*title*⟩`]`⟨*name*⟩,

respectively.

Their definitions run along the same lines as the `roles` environment.

```
\newenvironment{props}{
  \begin{revy@list}{\@propname}
    \revy@itemname{prop}
  }{
  \end{revy@list}
}
\newenvironment{costumes}{
  \begin{revy@list}{\@costumename}
    \revy@itemname{costume}
  }{
  \end{revy@list}
}
\newenvironment{mics}{
  \begin{revy@list}{\@micname}
    \revy@itemname{mic}
  }{
  \end{revy@list}
}
```

However, the structure of the item macro in `instructors` doesn't fit the pattern, and so needs its own particular implementation.

```
\newenvironment{instructors}{
  \begin{revy@list}{\@instructorname}
    \def\@instructorwith[##1]{\item [{\bf ##1} \hfill] }
    \def\@instructorwithout{\item}
    \def\instructor{%
      \@ifnextchar [{\@instructorwith}{\@instructorwithout}%]
    }
  }{
  \end{revy@list}
}
```

`\rolename`
`\costumename`
`\propname`
`\micname`
`\instructorname`

The default headings for these environments—“Roller:”, “Kostumer:”, “Rekvisitter:”, “Mikrofoner” and “Instruktører:”—may be changed with the macros

`\rolename{`⟨*text*⟩`}`,

`\costumename{`⟨*text*⟩`}`,

`\propname{`⟨*text*⟩`}`,

`\micname{`⟨*text*⟩`}`

og

`\instructorname{`⟨*text*⟩`}`

Implementation

```
\def\rolename#1{\def\@rolename{#1}}
\def\costumename#1{\def\@costumename{#1}}
\def\propname#1{\def\@propname{#1}}
\def\micname#1{\def\@micname{#1}}
\def\instructorname#1{\def\@instructorname{#1}}
```

The macros that store these names are

```
\def\@rolename{Roller:}
\def\@costumename{Kostumer:}
\def\@propname{Rekvisitter:}
\def\@micname{Mikrofoner:}
\def\@instructorname{Instruktører:}
```

2.5 Body text

2.5.1 `sketch`

`sketch` (*env.*) To typeset spoken lines, as in the example shown in figure 6, use the `sketch` envrionment.

Figure 6

⋮

RC *(To KA)*: Did you write th... wait a minute. *(Off stage)* It's him! *(Back at KA)* How did you get here?

KA *attempts to sidle away.*

RC: Get him!

Several Roman soldiers appear, and drag ***KA*** *off stage, while* ***KA*** *attempts to protest.*

⋮

An example of how spoken lines are typeset, using the `sketch` environment.

Implementation

`sketch`, like the assignment list environments, is derived from the `list` environment. The environment `revy@bodycommon` introduces macros that are shared between the `sketch` and `song` environments. It is defined on page 29.

```
\newenvironment{sketch}{%
  \begin{revy@bodycommon}
  \rm\begin{list}{}{
      \labelwidth 2em
      \leftmargin 3em
```

```
    \rightmargin 0em
    \labelsep 0.5em
    \listparindent 2em
    \topsep 1ex
    \partopsep 1ex
    \parsep 0ex
    \itemsep 1ex\relax}%
  \item \rule{0.2em}{0em}\vspace{-1em}\par %}
```

`\says` Inside the `sketch` environment, the macro `\says` sets each line of dialouge. Its complete syntax is:

`\says{`⟨*abr.*⟩`}[`⟨*direction*⟩`]` ⟨*line*⟩

Implementation

`\says` and `\scene` (see page 30) effectively shift the environment back and forth between two modes of typesetting. We mess around with `\leftskip` to accomplish changes in indentation.

```
  \newdimen\old@leftskip
  \old@leftskip \leftskip
  \newdimen\short@leftskip
  \short@leftskip \leftskip
  \advance\short@leftskip -1.5em
```

`\says`

```
\def\says##1{\@ifnextchar [{\@saysas{##1}}{\@says{##1}}} %]
\def\@saysas##1[##2]{\rm\item [{\bf ##1 }{\it (##2)\,}\hfill:]
  \leftskip \old@leftskip}
\def\@says##1{\rm\item [{\bf ##1\,}\hfill:]\leftskip \old@leftskip}
```

`\scene`

```
\def\scene{\medskip\par\noindent\it\leftskip \short@leftskip }
```

Thus, for example, part of the first line of figure 6 gets created with the following code:

```
\says{RC}[To KA] Did you write th... wait a minute. \act{Off stage} (...)
```

Implementation

And here, the `sketch` environment is concluded.

```
  }{%
  \end{list}
  \end{revy@bodycommon}
}
```

Figure 7

⋮

CC: ...be silly chumps,
Just purse you're lips and whistle,
That's the thing.

And, always look on the bright side of life,

⋮

An example of how song lyrics are typeset, using the `song` environment.

2.5.2 `song`

`song` (*env.*) The `song` environment is for song lyrics, which are set as demonstrated in figure 7.

Implementation

The `song` environment is also implemented as a derivative of `list` and uses `revy@bodycommon`.

```
\newenvironment{song}{%
  \begin{revy@bodycommon}
  \rm\begin{list}{}{%
      \raggedright
      \labelwidth 4.5em
      \leftmargin 7em
      \rightmargin 0em
      \labelsep 2em
      \listparindent -2em
      \topsep 0ex
      \partopsep \bigskipamount
      \parsep .6\baselineskip
      \itemsep -.6\baselineskip
      \relax
    }
  \item[] \rule{0.2em}{0em}\vspace{-\baselineskip}\strut\par %}
```

`\sings` Within the `song` environment, the macro `\sings` sets song lyrics. Its full syntax is

`\sings{`⟨*abr.*⟩`}[`⟨*direction*⟩`]` ⟨*lyric*⟩

Line breaks are significant in song lyrics, and so the `song` chanees how LaTeX reacts to carriage returns in the source file. For example, the lyric shown in figure 7 is the product, in part, of this source code:

```
\sings{CC} ...be silly chumps,
           Just purse you're lips and whistle,
            That's the thing.
```

Implementation

`\sings` and `\scene` shift back and forth between to even more different states within the `song` environment than was the case in `sketch`. Much of the magic relates to the macros `\obeycr` and `\restorecr`.

```
  \newdimen\old@leftskip
  \old@leftskip \leftskip
  \newdimen\short@leftskip
  \short@leftskip \leftskip
  \advance\short@leftskip -5.5em
```

`\sings`

```
\def\sings##1{\obeycr
  \@ifnextchar [{\@singsas{##1}}{\@sings{##1}}} %}
\def\@singsas##1[##2]{\rm\item [\hskip\@leftsingpad{\bf ##1 }{\it
    (##2)\,}:\hskip\@rightsingpad]\hskip-2em\leftskip \old@leftskip}
\def\@sings##1{\rm\item [\hskip\@leftsingpad{\bf ##1\,}:\hskip
  \@rightsingpad]\hskip-2em\leftskip \old@leftskip}
```

`\scene`

```
\let\old@sings=\sings
\def\scene{%
  \def\sings{\let\sings=\old@sings \medskip\sings}
  \medskip\par\leftskip \short@leftskip \restorecr
  \it\noindent\relax
}
```

`\@leftsingpad` and `\@rightsingpad` allow us to set lyrics to be left or right justfied.

```
  \newskip\@leftsingpad
  \newskip\@rightsingpad
  \def\flushsingsright{\@leftsingpad 0pt plus 1fill\@rightsingpad 0pt\relax}
  \def\flushsingsleft{\@leftsingpad 0pt\@rightsingpad 0pt plus 1fill\relax}
```

We set left justification as the default.

```
  \flushsingsleft
```

Implementation

We call `\obeycr` here, along with some macros that get LaTeX to be less concerned about the proper length of lines.

```
  \tolerance10000\hfuzz21cm\obeycr
}{
  \end{list}
  \end{revy@bodycommon}
}
```

Implementation

`\obeycr` `\restorecr` The macros that make the line breaking magic possible, are

```
\newif\if@newpar
{
  \catcode`\^^M=13
```

At this point, we must be careful with carriage returns in the code itself...

```
  \gdef\@carriagereturn{%
    \@ifnextchar^^M{%
      \@newpartrue%
    }{%
      \@ifnextchar\end{}{%
        \@ifnextchar\sings{%
          \if@newpar\bigskip\par\fi\@newparfalse%
        }{%
          \if@newpar\par\else\\\fi\@newparfalse%
        }%
      }%
    }%
  }%
  \gdef\obeycr{%
    \catcode`\^^M=13 %
    \let^^M=\@carriagereturn%
    \@gobblecr%
  }%
  \gdef\restorecr{\catcode`\^^M=5 }%
}
```

2.5.3 Both `sketch` and `song`

The `sketch` and `song` macros both provide the following macros:

Implementation

`revy@bodycommon` (*env.*) They are packaged in the environment `revy@bodycommon`.

```
\newenvironment{revy@bodycommon}{
```

`\role` A macro for formatting role abbreviations within the text.

Implementation

This macro has the same name as the item command inside the `roles` environment. But it's OK, because that command is defined inside `roles`'s internal `group`.

```
  \let\role = \textbf
```

`\does` A macro for giving instructions for a particular role that aren't a spoken line, such as

```
\does{KA} attempts to sidle away.
```

Implementation

The 1em exdent is a hardcoded length that matches how `song` and `sketch` are defined right now. I'm sure it won't jump up and bite anyone in the future...

```
  \def\does##1{\scene \hspace{-1em}\role{##1}}
```

`\scene` `\scene` is used for stage directions, such as

```
 \scene P \& KA are startled.
```

Implementation

`\scene` was implemented inside the `sketch` and `song` environments.

`\act` `\act` is for directions, such as in

```
 (...) It's him! \act{Back at KA} How did you get here?
```

Implementation

```
  \def\act##1{\textit{(##1)}}
```

Both were used in the example in figure 6.

Implementation

And here, we end the the environment `revy@bodycommon`.

```
}{}
```

2.6 Miscellenia

2.6.1 SATyR

`\SaTyR` SATyRRevy, which is produced by the students at the Faculty of Science at the University of Copenhagen, is properly written by saying

```
\SaTyR{}Revy
```

Implementation

```
\newcommand{\SaTyR}{S\hspace*{-.2ex}\raisebox{-.15em}{A}\hspace*{-.5ex}TyR}
```

Implementation

2.6.2 The .aux file

For futue use, we will also ensure that our metadata is written to the `.aux` file at the end of the document.

This is also where we `label` the last page.

```
\let\revyinfo=\relax
\newtoks\@titletoks
\newtoks\@authortoks
\newtoks\@melodytoks
\newtoks\@revyrevynametoks
\newtoks\@revyrevyyeartoks
\let\end@document=\enddocument
\def\enddocument{\label{lastpage}\write\@auxout
  {\string\@ifundefined{revyinfo}{\string\def\string
  \revyinfo\string##1\string##2\string##3\string
  ##4\string##5\string##6\string##7\string##8{}}{}}\@titletoks=
  \expandafter{\@title}\@titletoks=
  \expandafter{\@title}\@authortoks=
  \expandafter{\@author}\@melodytoks=
  \expandafter{\@melody}\@revyrevynametoks=
  \expandafter{\@revyname}\@revyrevyyeartoks=
  \expandafter{\@revyyear}\write\@auxout
  {\string\revyinfo{\the\@titletoks}{\@version}
    {\today}%
    {\the\@authortoks}%
    {\the\@melodytoks}%
    {}%
    {\the\@revyrevynametoks}%
    {\the\@revyrevyyeartoks}%
  }%
  \end@document
}
```

References

[1] Victor Eijkhout. *TeX by Topic*, chapter 14.5.1. https://github.com/VictorEijkhout/tex-by-topic/raw/main/TeXbyTopic.pdf, 2019.

[2] Victor Eijkhout. *TeX by Topic*, chapter 13.8.6. https://github.com/VictorEijkhout/tex-by-topic/raw/main/TeXbyTopic.pdf, 2019.

[3] Peter Müller, Kristoffer Stensbo-Smidt, Troels Henriksen, Sebastian Paaske Tørholm, Sarah Brofeldt, Anders Komár Ravn, Kristoffer Levin Hansen, Asbjørn Preuss, and Mikkel Noer. FysikRevyTEX. https://github.com/FysikRevy/FysikRevyTeX, 2025.

[4] Overleaf. Understanding packages and class files. https://www.overleaf.com/learn/latex/Understanding_packages_and_class_files.

[5] Till Tantau. *The TikZ and PGF Packages*. https://mirrors.dotsrc.org/ctan/graphics/pgf/base/doc/pgfmanual.pdf, 2023.

Index

Numbers written in italic refer to the page 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.

A Complete example

```
%%
%% This is file `Example.tex',
%% generated with the docstrip utility.
%%
%% The original source files were:
%%
%% ucph-revy.dtx  (with options: `ex-en')
%%
%% See the generating file for its conditions on distribution and reuse.
%%
%% Also, for this file by itself, to the extent possible under law,
%% Kristoffer Levin Hansen has waived all copyright and related or
%% neighboring rights to Example.tex. This work is published from:
%% Denmark.
%%
%% http://creativecommons.org/publicdomain/zero/1.0/
%%
%% Note that this file probably contains text, characters and situations
%% covered by copyright beloninging to other entities.
%%
\documentclass{ucph-revy}

\usepackage[utf8]{inputenc}
\usepackage[OT1]{fontenc}
\usepackage{hyperref}
\urlstyle{sf}

\version{1.0}
%% The settings below may be omitted if their inclusion is undesired
\revyname{The Meaning of Liff}
\revyyear{3001}
\title{Example}
\author{an examplesmith}
\status{example}
\eta{$1$ minute, $47$ seconds}
\responsible{Probably you}
\melody{Monty Python: ``Always look on the bright side of life''
  (\url{https://youtu.be/SJUhlRoBL8M})}

\writtenbyname{written by}
\melodyname{Melody:}
\responsibletext{\TeX--responsible}

\renewcommand{\pagessum}[1]{#1 page\ifnum#1=1\else s\fi}

\begin{document}
\maketitle
```

```
\rolename{Roles:}
\begin{roles}
  \role{KA}[Graham] Arthur, king of all the Britons
  \role{P}[Michael] Peasant
  \role{RC}[John] Roman Centurion
  \role{R1...?} Several Roman Soldiers
  \role{CC}[Eric] Crucified Criminal
  \role{B}[Graham] Brian
\end{roles}
%% The props environment may be omitted if found to be superfluous
\propname{Props:}
\begin{props}
  \prop{Large sign}
  \prop{Field of crosses}
\end{props}

\begin{sketch}
  \scene Lights up on a muddy field with a castle in the background.

  \says{KA}[As he enters] Hello!

  \says{KA} Now, by rights, you should all be kneeling, given that I
  am Arthur, king of all the Britons. However, it has been brought to
  my attention that the audence seating is not laid out to allow for
  that. We had some considerable trouble during dress rehearsals. So I
  will excuse you from that requirement.

  Now, has anyone seen a particularly fancy cup? Perhaps out
  in the bar? You see, I am on a quest given to me by God, to find the
  holy... \act{Interrupted, as he steps in a hole.} Oh dear...

  \says{P}[Shrieking, as he rears up from the ground] Oi! What do you
  think you're doing!

  \says{P} You broke my sign!

  \says{KA} Sign? What sign?

  \says{P} Well, look.

  \does{P} raises up a sign that reads

  \begin{center}\sc Romanes Eunt Dom[ ]\end{center}

  The last bit of the sign has a foot--shaped hole in it.

  \says{P} You can't read it anymore!

  \says{KA} You certainly can! It says ``Romans go home.''
  \says{RC}[Appears from offstage] No it doesn't.

  \does{P \& KA} are startled.

  \says{RC}[To KA] Did you write th... wait a minute. \act{Off stage}
  It's him! \act{Back at KA} How did you get here?

  \does{KA} attempts to sidle away.

  \says{RC} Get him!

  \scene Several Roman soldiers appear, and drag \role{KA} off stage,
  while \role{KA} attempts to protest.

  \says{KA}[As he is dragged off] I am not him! I am Arthur, King of
  the Britons!

  \scene The scene changes to a field of crosses, with several
  crucified criminals and B.

  \does{CC} is in the middle of singing to \role{B}.
\end{sketch}
\begin{song}%
  \sings{CC} ...be silly chumps,
             Just purse you're lips and whistle,
             That's the thing.

             And, always look on the bright side of life,
             Always look on the right side of life,
             For life is quite absurd,
             An. death's the final word...
\end{song}
\begin{sketch}
  \says{RC}[Shouting as he walks on stage] Right, you lanky reprobate!
  Don't think that the mighty Roman bureaucracy doesn't keep track of
  its inmates

    \scene KA is dragged on stage by two Roman soldiers behind RA.

    \says{RA} Look, we have your designated cross ready for you
    right... \act{points to B'a cross, then falters when he notices
      B.}

    \scene Everyone looks back and forth between B and KA.

    \scene End scene
\end{sketch}
\end{document}
}
\endinput
```

```
%%
%% End of file `Example.tex'.
```

B Example of a thumb index page

```
\documentclass[a4paper,11pt,thumbindex]{article}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[danish]{babel}
\usepackage{calc}
\usepackage{tikz}

\version{1.0}
\title{Registerindeks}

\begin{document}
\thispagestyle{empty}
\newlength{\torightside}
\setlength{\torightside}{ \paperwidth - 1in - \hoffset %
  - \oddsidemargin + .6cm}
\newlength{\totop}
\setlength{\totop}{ -\headsep - \headheight - \topmargin %
  - \voffset - 1in - .5cm}

\vbox to 0pt{\vskip \totop \hbox to \torightside{\hss%
    \rectothumbindexwithtitles%
  }}
\vbox to 0pt{\tikz \path (0,0) -- (0, -.5\textheight)
  node[rotate=90]{\parbox{\textheight}{\maketitle}}; }
\end{document}
```