

The SKB package - Create and maintain a repository for long-living documents

Sven van der Meer

2011-06-03 v0.52

Abstract

This package provides macros that help to build a repository for long living documents. It focuses on structure and re-use of text, code, figures etc. The basic concept is to first separate structure from content (i.e. text about a topic from the structure it is presented by) and then separating the content from the actual published document, thus enabling easy re-use of text blocks in different publications (i.e. text about a protocol in a short article about this protocol as well as in a book about many protocols); all without constantly copying or changing text. As a side effect, using the document classes provided, it hides a lot of L^AT_EX from someone who just wants to write articles and books.

Contents

1	The Intent	4
2	The Story	4
2.1	The Short Story	4
2.2	The Long Story	5
3	The Concept: Separate Things	7
3.1	Separate Content from Structure: the Repository Folder	7
3.2	Separating different Parts of a Document	8
3.2.1	Bibliography, Acronyms and Figures	8
3.2.2	Publications and Content	10
3.2.3	The Final Directory Structure	11
4	User Manual	12
4.1	Getting Started	12
4.1.1	The SKB Distribution	12
4.1.2	Installation	13
4.1.3	Rebuild the SKB from Source	14
4.1.4	Configuration: skbconfig	15

4.1.5	Configuration: View Options Used	16
4.1.6	Creating a Directory Structure	17
4.2	Files, Figures and Slides	18
4.2.1	Files and Headings	18
4.2.2	Figures	19
4.2.3	Slides	21
4.3	Filenames, Acronyms and References	22
4.3.1	Path and File Names	22
4.3.2	Loading Acronyms and Bibliographic Information	23
4.4	Other useful Macros	23
4.4.1	Emphasising Text	23
4.4.2	Environments for lists and enumerates	24
4.4.3	Macros for PDF Info	24
4.4.4	Listings Styles and Support	25
4.4.5	Optional Text – Versions and Optional	25
5	Examples	26
6	Implementation: Kernel	27
6.1	Required Packages	28
6.2	Conditiona/Optional Text Support	28
6.3	Provide Command	30
6.4	Macro Redefinitions	30
6.5	At End of Document	30
6.6	Package Configuration	31
6.7	Generic Input Macro	33
6.8	Kernel support for skbinput	33
7	Implementation: Configuring the SKB	35
7.1	Changing Configuration: skbconfig	35
7.1.1	The Macro Options	35
7.1.2	The Macro	35
7.2	Changing Configuration: skb.cfg and skblockal.cfg	36
7.3	Viewing Configuration: skboptionsused	36
8	Implementation: Files, Figures and Slides	37
8.1	Declaring Headings: skbheading	37
8.2	Loading T _E X files: skbinput	37
8.2.1	Macro Options	37
8.2.2	The Macro	39
8.3	Loading Figures: skbfigure	40
8.3.1	Macro Options	40
8.3.2	The Macro	41
8.4	Loading Slides: skbslide	43
8.4.1	Some Extentions	43
8.4.2	Macro Options	43

8.4.3	The Macro	44
9	Implementation: Filenames, Acronyms and References	45
9.1	Path and File Names	45
9.2	Loading Acronyms	46
9.3	Loading Reference Database	46
10	Implementation: Other useful Macros	46
10.1	Emphasising Text: skbem	46
10.1.1	Macro Options	46
10.1.2	The Macro	47
10.2	Emphasising Text: skbcode	47
10.3	List Environments: skbnolist and skbnoteenum	47
10.4	Acronyms in Footnotes: skbacft	48
10.5	PDF Meta Information: skbpdfinfo and more	48
10.6	Listing Styles and Support	49
11	Experimental Macros	51
11.1	Defining new relative Headings: skbheadingude	51
11.1.1	Macro Options	51
11.1.2	The Macro	52
12	The Configuration File skb.cfg	52
13	The SKB Classes	53
13.1	The Class skbarticle	53
13.1.1	Loaded Packages	54
13.1.2	Memoir Options	55
13.1.3	Misc Settings	56
13.2	The Class skbbook	56
13.2.1	Loaded Packages	56
13.2.2	Memoir Options	57
13.2.3	Misc Settings	58
13.3	The Class skbbeamer	59
13.3.1	Loaded Packages	59
13.3.2	Misc Settings	60
13.4	The Class skblncsbeamer	60
13.4.1	Loaded Packages	61
13.4.2	Memoir Options	62
13.4.3	Misc Settings	63
13.5	The Class skblncsppt	63
13.5.1	Loaded Packages	64
13.5.2	Memoir Options	65
13.5.3	Misc Settings	66
13.6	The Class skbmoderncv	66
13.6.1	Loaded Packages	67

13.6.2 Misc Settings	67
13.7 Macros	68
14 History and Change Log	70
14.1 v0.10 from 06-Jul-2010	70
14.2 v0.20 from 08-Jul-2010	70
14.3 v0.30 from 14-Jul-2010	70
14.4 v0.31 from 20-Jul-2010	71
14.5 v0.32 from 20-Jul-2010	72
14.6 v0.4 from 21-Jul-2010	72
14.7 v0.5 from 04-Aug-2010	72
14.8 v0.51 from 12-May-2011	74
14.9 v0.52 from 03-Jun-2011	74

1 The Intent

Provide a \LaTeX package that helps to create and maintain a repository for long-living documents. It's probably not usefull for some short-term articles, however, I learned that most of my short term articles eventually become part of my long-term documents. Here you go. The repository should allow for easy access to 'stuff': text blocks, senteces, paragraphs, sections, complete chapters. But also to figures, code snippets, examples, etc. And I do want to limit the amount of repetition of information. For example, if I use a certain example in an article I might want to use the same (identical) example in a book or a presentation or lecture notes. If I only copy the example I have to maintain several sources, and over time I will not remember which of them is normative. As a side effect, I also want to optimise document creation and limit the \LaTeX or document class specific code in my documents.

2 The Story

2.1 The Short Story

I have written papers, done a lot of presentations, provided some book chapters, still working on a book, participated in many research proposals and projects, and created tons of notes and figures. As of early 2009, most of that information was distributed over the repositories of different projects and organisations I worked for, in some document management systems, on several websites, databases, my preferred email client (which changed twice), different computers and later even different external hard drives and **USB**¹ sticks. Looking for specific text or a particular figure could easily end in a days work. Tools like desktop search engines can help to find 'stuff'. I used them, but if they found anything it was hard to maintain the context it was written in and some formats or sources were out of

¹Universal Serial Bus (**USB**)

reach for them. Even worse with figures and the many versions some of them evolved in over time. After multiple jobs and several years, all I had is kind of a very messy base of knowledge, well-hidden somewhere, thus very difficult to locate and impossible to maintain.

So I started early 2009 to re-organise my 'stuff'. At the same time, I did realise that moving away from \LaTeX was part of the problem (and I thought using the other text processor would help, it actually didn't, long-term). So \LaTeX became, again, the text processor of choice, and with it the ability for a complete different approach to organise my 'stuff'. This was the moment the **SKB**² was created. **SKB** stands for Sven's Knowledge Base. The \LaTeX package **skb**, described in this article, forms part of a larger software system that uses **SQL**³ite databases, a small **PHP**⁴ framework, Apache for **HTML**⁵ access and recently also a Java port.

My document repository uses the **skb** package, so most of my documents are eventually \LaTeX documents. I am saying eventually because I still use other tools (like Microsoft's Powerpoint), but integrate their output in my repository. I do all my figures these days using Inkscape, so the source is **SVG**⁶ and the output for \LaTeX documents **PDF**⁷. For editing the text files I do flip between UE Studio and LeD. Parts of the content (such as acronyms and bibliographic information) are maintained in **SQL**ite databases and exported to \LaTeX . This package now shows how I build my documents.

2.2 The Long Story

Over several years of writing documents (articles, books, reports, standards, research proposals) ideas and concepts became distributed (actually a euphemism for 'hidden') within many many documents (in all sorts of formats) located at many many locations (such as local file system, document management system, subversion/perforce systems, web servers, email clients). The problems associated to this situation are manifold:

- Ideas/concepts are hidden, often un-accessible and, as I experienced, search tools are of limited help.
- The documents are written in all sorts of formats or available only in (usually proprietary) binary formats. Ever tried to open a document written in **MS**⁸ WinWord 6.0 with customised document template in a newer version of the same programme? You know then what I am talking about.
- Reusing the ideas/concepts, once found in a document and managed to open that very document, usually involves huge amount of re-formatting. This will produce mistakes. Ever tried to use a **BIB \TeX** generated reference list,

²Sven's Knowledge Base (**SKB**)

³Structured Query Language (**SQL**)

⁴PHP Hypertext Preprocessor (**PHP**)

⁵Hyper Text Markup Language (**HTML**)

⁶Scalable Vector Graphics (**SVG**)

⁷Portable Document Format (**PDF**)

⁸Microsoft (**MS**)

found in a PDF file in a new article? I found better ways to spend my nights and weekends (yes, I am married and I have a garden).

- Over time, it can become very difficult to distinguish between different versions of a document, concept and/or idea. As it happens in real life, things move on even in computing and the related sciences. Documents are written for a specific historic context, which might but often will not appear in their abstract (or the name of the folder they are stored in).
- The above issues do apply to figures and presentations as much as to the text part of documents. Reorganising my documents/figures/presentations I did find way too many duplicates. I have used too many graphic software packages in the past 10 years which don't exist anymore, or which do not run on the latest version of my preferred operating system. Some of the figures are only available in some sort of low-resolution bitmap, rendering them useless even for a non-peer-reviewed article today (the original source got 'lost', in most cases because someone removed the project folder after the project was terminated).

A solution is to create a unified document repository, then use this repository as 'normative source' to create documents for specific purposes while leaving the text blocks, headings, figures, presentations, references, acronyms and all other reusable 'stuff' in the repository for the next document which might (hopefully will) benefit from them. This can (did it for me already) save a lot of time, demands archiving (of published documents, thus creating a traceable history), helps to keep important information updated (without jeopardising any other work) and prevents losing any 'stuff'.

The repository needs a few rules, a (customisable) structure but beside that only a bit of effort to be maintained. To give an example: while writing the first version of this article (May 11, 2009), I have moved 4 lecture notes, 2 presentations, 1 book chapter, 1 book (in writing), 1 textbook (for students, with 4 chapters) and 4 articles from my 'mess' into my repository. This involved some re-formatting (plus the occasional re-drawing) to bring the original sources into the target formats. At the same time I did develop the rules of my repository, the structure and the (mostly L^AT_EX) code (and re-wrote/structured/ruled most of them a few times). I ended up with 1,314 files in 87 folders, which create 9 articles, 2 books, 1 textbook, 3 lecture notes and this document (note: the number of articles increased, because I could re-assemble 'stuff' for new uses, spending some five minutes per one new article). I did remove roughly 100 pages of text (take the classic Spring LNCS⁹ format and you get the point of the number of characters) and some 40 figures (all duplicates). I did find way too many errors in the original sources (most of which have been created by 're-using' them earlier from even more-original-sources).

⁹Lecture Notes in Computer Science (LNCS)

3 The Concept: Separate Things

You already got the idea that separation is important, reading about published documents and a normative repository. The basic idea is: think separation – separate as much as you can, but not more. I know that this sounds like a strange idea when the goal is a unified repository, but it is essential. So we separate several concerns (taking a concept of distributed system design). So if we want to facilitate re-usability, we have to:

1. separate content of a document from its structure and
2. separate the different parts of a document.

For the impatient:

1. Separating content from structure means to identify small, coherent blocks of information, i.e. text describing a certain aspect or an example, and put them separated into the repository folder.
2. Separating parts of a document means to separate the part that is important for publishing from the part that is important for the content and put them into different places (one in the publish folder and the other one in the repository older). It also means to build a separate repository for figures, since they could be used in many different small blocks of information.

3.1 Separate Content from Structure: the Repository Folder

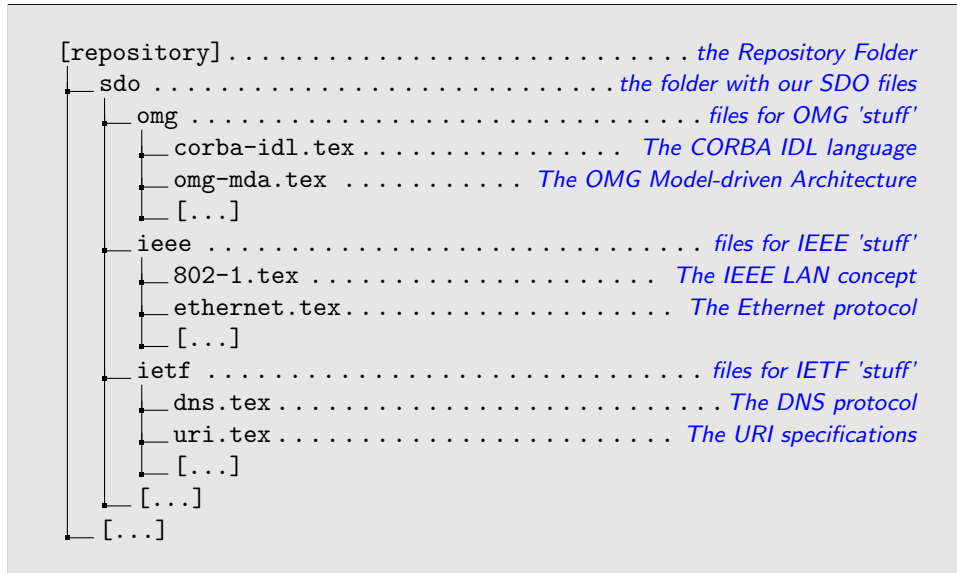
Now, separating the structure from the content first. The structure of a document (we stay with classic text documents like articles, books, etc. for a while) is words in sentences in paragraphs in (sub-) sections in chapters (if its a book, of not only sections)¹⁰. We collect sentences and paragraphs but separate them from headings. \LaTeX is doing that already with the macros for chapters and sections. We go one step further and provide a generic way to identify a heading with the **SKB** macro `\setheading`. This allows to select the appropriate \LaTeX heading level at a later stage having the context of that later stage in mind (i.e. it might be a section in an article but a chapter in a book). Now we create a structure for the resulting files in our repository, adding meaningful names to the directories and files. Obviously the names of these folders should provide some idea about the general characterisation of the files they contain. Example? Well, if you collect information from **SDO** the top folder could be named `sdo` and the sub-folders using the respective Standard Defining Organisation (**SDO**) acronyms, such as `omg` for the **OMG**¹¹ and `ieee` for the **IEEE**¹² and `ietf` for the **IETF**¹³. We put all this in a folder named repository, making it explicit that here is were we find all our normative content. The following example shows how that looks like.

¹⁰One very meticulous person might add 'characters' and mention that there are more ways to think about a document's structure. But that person is not me. The structure I used fits the purpose (as of now), if it doesn't anymore I will look further.

¹¹Object Management Group (**OMG**)

¹²Institute of Electrical and Electronic Engineers (**IEEE**)

¹³Internet Engineering Task Force (**IETF**)



The result: we have a structure of files, containing our 'stuff', integrated in a structure of folders that allows us to find it (the better this structure the more helpful it is, and remember this is a 'personal' repository, so your personal flavour is king).

3.2 Separating different Parts of a Document

The next step is to separate the remaining parts of a document based on their semantics. You are probably doing that already if you maintain a database for bibliographic information and have many of your articles using it. But we can and should do much more than that.

3.2.1 Bibliography, Acronyms and Figures

So the combination of \LaTeX and \BibTeX already helps us for this separation. Using the acronym package, we can extend this to acronyms. Looking into the highly common re-use of figures, we should look into this as well. Let's take the organisation of bibliographic information as a template. I store them using \BibTeX and process them with the biblatex package (but that is not critical for now). My \BibTeX database is in a special folder (we can call it **references** for the moment) and it uses a file structure that helps me to find information. This structure is based on the \BibTeX and biblatex classification, i.e. inproceedings, article, book, thesis, standard, etc.

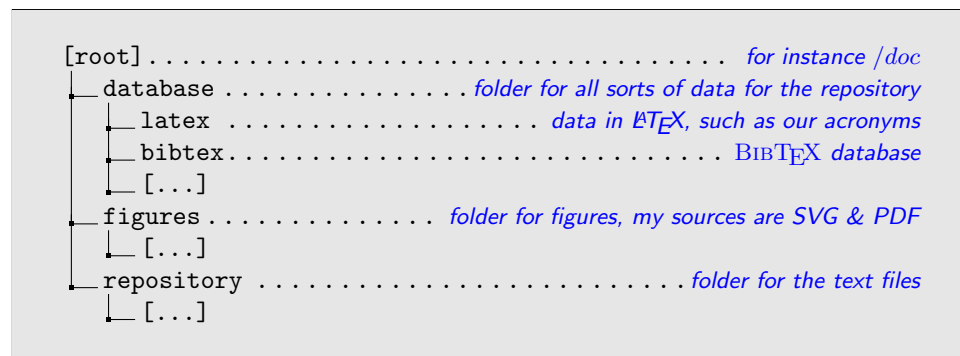
Now, I can do the same for figures: put them into a special folder (we can call it **figures** for the moment), which contains the source of the figures and the

generated formats I need for my documents (usually **PDF**, some **PNG**¹⁴). Now I can reference these figures from the repository as well as for other use cases, such as web publishing or presentations¹⁵.

Last not least, the **acronym** package allows for an automatic handling of acronyms, including list of acronyms. It is very similar to **BIBTEX** in that it will only show the acronyms used in a document out of a (potentially large) database.

One might also want to create other specific structures, such as for programming code. Don't stop yourself, it's easier to combine things later (if the separation is not effective) than to separate things that are hugely integrated into each other. For one of my internal projects, a parser generation environment based on **ANTLR**¹⁶, I created a special folder for the EBNF specifications along with railroad diagrams. Now I can use them in my book and my papers.

Now we name the folders for the separated content. This is straight forward, although you might want to use different names (don't worry, the **skb** supports that). We add to the already created repository folder the things we need for figures (**figures**) and combine acronyms and **BIBTEX** in a folder called **database**, separating the data from all other content¹⁷. Now the directory structure looks like this:



What did we do so far? We did separate the different parts of our documents. The more clinical you are, the better the result will be. But please note: separate as much as you should, not as you could. If you don't find a reason for separating 'stuff', then don't do it!

¹⁴Portable Network Graphics (**PNG**)

¹⁵My figures are exclusively in **SVG** using inkscape (www.inkscape.org). This has the advantage of a standardised, text-based format with many export options. All my figures are in a single root folder, structured very similar to the document folders created above, but separated from it. This makes re-use of figures very easy.

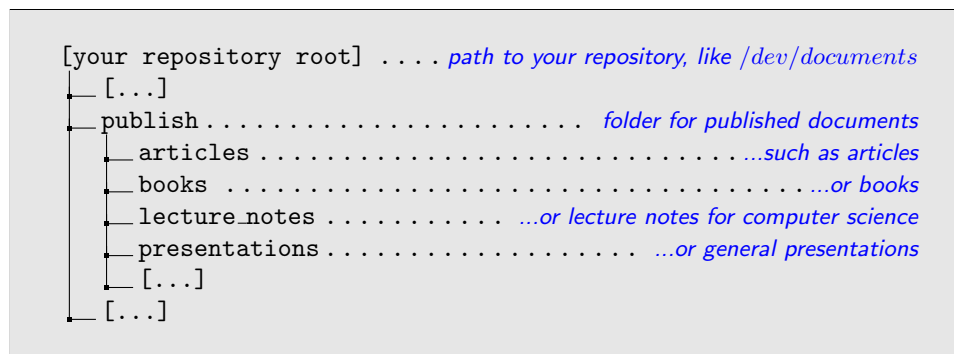
¹⁶Another Tool for Language Recognition (**ANTLR**)

¹⁷Now, the reason for the database folder and its structure is that the whole **SKB** contains more databases, all of which reside here. If you want to simply apply this to **L^AT_EX** documents you might want to use a different structural approach.

3.2.2 Publications and Content

Here is where it might get slightly more complicated than in the first few steps. And you might see already that the reason for that is separation! We didn't finish the separation, we have to go one step further. And that means to separate now the contents (with the references and acronyms and figures) from the reason to publish a document. This last step of separation is more conceptual, being focused on the *why?* and *where?* and *how?* we publish, rather than being focused on the *what?* we publish.

So we do publish for many reasons: articles for research, project proposals, reports, lecture notes, standard documents, annotated presentations, sometimes even books. We publish for a specific purpose, in a specific (soon historic) context, using the requested format (and style sheets) and a particular structure of our document that fits the purpose. That means we organise and structure our content every time according to these constraints. Thus we need a new directory structure for that, since we will not reuse that as often as our 'stuff' itself. Remember, we use the `skb` macro `\skbheading` for headings, not the classical \LaTeX macros like `\section`, so our files effectively do not contain much information about their place in the structure, only that they claim one ¹⁸. This comes in handy now, since all we have actually to do is to assign a document heading level to every repository file we load. Let's create a folder for the published documents and call it `published` with a set of sub-folders that help us to understand the general context of the publication. My directory structure could look like this:



We could, and it usually makes sense to do so, go one step further in that separation. This time within the documents in the `published` folder. The reason is the structure of \LaTeX documents: they do need some commands specific to \LaTeX , which we don't necessarily want to mix with the commands that input our content (the files from repository). So it would make sense to have another pair of documents here, one containing all \LaTeX commands needed to create a document, and one containing all the commands that include our content. Say we have a few

¹⁸Currently experimental, but soon to be ready, there will be an extension to the `\skbheading` macro that allows a little bit more information to be put in the repository files. For the moment this is captured in the `\skbheadingduc` macro.

articles for state of the art discussions on *naming*, *object-models* and *protocols*, we could create the following structure for the `article` folder :

```

articles ..... our articles
├─ naming.tex ..... the file creating an article on naming
├─ object-models.tex ..... the file creating an article on object-models
├─ protocols.tex ..... the file creating an article on protocols
├─ tex ..... a folder containing the tex files that include our content
│   └─ naming.tex ..... the file including all content for naming
│   └─ object-models.tex ... the file including all content for object-models
│   └─ protocols.tex ..... the file including all content for protocols

```

Now everything is structured, thus simple again. If we are looking for content, we go to the `repository` directory and the directory names help us to find what we are looking for. If we need a figure, we do the same at the `figures` directory. `acronyms` and `bibtex` help with the respective other content. If we want a specific published document, we simply check the `published` directory and will have a look into a `tex` sub-directory to see what content is include how.

Good news, the separation is finished! What have we done? We have separated the contents from the structure by creating, created a separate directory structure for figures, another one for bibliographic data, one for acronyms and finally a complete directory structure for published documents. Content and title form a pair, include figure, use acronyms and references and are combined in the published documents. At this point we can start calling it document repository.

3.2.3 The Final Directory Structure

As this is the final and complete root directory of our repository:

```

[your repository root] .... path to your repository, like /dev/documents
├─ database ..... folder for all sorts of data for the repository
│   └─ latex ... this is were  $\LaTeX$ data will be collected, such as our acronyms
│   └─ bibtex ..... folder for all  $\text{BIB}\text{T}\text{E}\text{X}$  reference files
├─ figures ..... folder for figures, my sources are SVG & PDF
├─ publish ..... folder for published documents
│   └─ articles .....such as articles
│   └─ books .....or books
│   └─ lecture_notes .....or lecture notes for computer science
│   └─ presentations .....or general presentations
└─ repository ..... folder for the text content

```

4 User Manual

The **SKB** provides macros that simplify file handling and hide some \LaTeX code (i.e. for figures) from the user, thus helping everyone to focus on the actual document one wants to write. There are a few macros, and they can be categorised as follows.

- Installation, rebuilt and configuration: this part deals with the installation of the package with your local \LaTeX distribution, the rebuilt of the styles, classes and documentation (all of them are provided, but you never know, it might become useful) and the configuration of the **SKB** using configuration files and the macro `\skbconfig`.
- Files, figures and slides: the combination of `\skbheading` and `\skbinput` will process files and the document level of headings. The macro `\skbfigure` makes it easy to include figures in your document and the macro `\skbslide` helps with **PDF** slides and annotations (if you are not using a classic \LaTeX solution such as the beamer package).
- Filenames, acronyms and references: here we deal with macros that provide access to the path and filenames the **SKB** maintains, plus loading acronym and reference databases.
- Other useful macros: there are some more macros that complete the **SKB**. There are little helpers for emphasising text, limiting the space between items in some list environments, putting acronyms into footnotes, filling meta information for **PDF** files and last not least macros that help dealing with optional and conditional text.

For the impatient, we start with a few examples. The first one shows how to use the **SKB** to produce a simple article. The second one explains how the documentation for the **SKB** is created using most of the **SKB** macros. Then we detail the usage of all the macros, following the above introduced categorisation.

4.1 Getting Started

4.1.1 The **SKB** Distribution

The **SKB** distribution that one can download from SourceForge or **CTAN**¹⁹ contains the source files for the **SKB**, the generated classes and styles, the generated documentation and the source files for the user guide. The following example shows the structure and content of the **SKB** distribution.

¹⁹the Comprehensive TeX Archive Network (**CTAN**)

[start folder]	
doc	<i>The generated PDFs and User Guide Sources</i>
[user-guide]	<i>Sources for the User Guide</i>
skb.pdf	<i>The generated Documentation</i>
skb-guide.pdf	<i>The User Guide only</i>
run	<i>The generated Class and Style Files</i>
skb.cfg	<i>The global Configuration File</i>
skb.sty	<i>The Style File</i>
*.cls	<i>The Class Files</i>
source	<i>The Source files</i>
skb.dtx	<i>Documented L^AT_EX Source File</i>
skb.ins	<i>The L^AT_EX Installer File</i>
*.txt	<i>Manifest, Licence, Todo List and History as plain Text</i>

4.1.2 Installation

There are two ways to install the **SKB**. The first option is have it automatically installed by your L^AT_EX distribution using T_EXLife or the CTAN archive²⁰. The second option is a manual installation doing the following steps:

1. Go to your L^AT_EX distribution to the folder `tex/latex`.
2. Create a folder `skb`.
3. Copy all files from the directory `run` of this package to the newly created folder `tex/latex/skb`.
4. Update the filename database of your L^AT_EX distribution. Please see the manual or help files of your L^AT_EX distribution for details.

If you want copy the source and documentation files as well, then do the following steps. We start with the documentation:

1. Go to your L^AT_EX distribution to the folder `doc/latex`.
2. Create a folder `skb`.
3. Copy all files from the directory `doc` of this package to the newly created folder `doc/latex/skb`.

And now the source files of the **SKB**:

1. Go to your L^AT_EX distribution to the folder `source/latex`.
2. Create a folder `skb`.
3. Copy all files and directories from the directory `source` of this package to the newly created folder `source/latex/skb`.

Now you have installed the **SKB** and you are ready to use it.

²⁰Note: Version 0.5 of the **SKB** has been submitted to CTAN and is available at <http://www.ctan.org/tex-archive/macros/latex/contrib/skb/>. But it will take a while to reach all mirrors and even longer for TeX Live and automatic installation with your preferred T_EX distribution

4.1.3 Rebuild the SKB from Source

The **SKB** class and style files as well as the documentation can be rebuilt from its sources very easily. The class and style files are part of the documented \LaTeX sources in the file `source/skb.dtx` and the \LaTeX installer (`source/skb.ins`) provides all necessary instructions. Simply follow the steps shown in the first part of the following example. All you have to do then is to copy the files created to your \LaTeX distribution.

```
## Build style and class files
$cd run
latex ../source/skb.ins
-> creates: skb.cfg, skb.sty and all cls files
```

The **SKB** documentation comes in several different ways. The file `source/skb.dtx` contains the documented source while the files in `doc/user-guide` can be used to generate the User Guide without source documentation, the **SKB** presentation (animated and not animated) and the lecture notes for the presentation.

When processing the file `source/skb.dtx`, the User Guide will automatically be included in the generated **PDF** if it is found in either of the directories `source/./doc/user-guide` (when using the **SKB** original distribution) or `source/./doc/latex/skb/user-guide` (when the **SKB** is already installed with your \LaTeX distribution).

The following example shows how to generate the complete **SKB** documentation. Please note that the sequence is partially important, for instance the file `ug-slides-noanim` must be processed before the file `ug-slides-notes`.

```
$cd doc
$pdflatex ../source/skb.dtx
$bibtex skb
$pdflatex ../source/skb.dtx
$pdflatex ../source/skb.dtx

## Build User Guide
$cd doc
$pdflatex user-guide/user-guide
$bibtex user-guide
$pdflatex user-guide/user-guide
$pdflatex user-guide/user-guide

## Build Presentation with Animations
$cd doc
$pdflatex user-guide/ug-slides-anim
$bibtex user-guide
$pdflatex user-guide/ug-slides-anim
$pdflatex user-guide/ug-slides-anim

## Build Presentation without Animations
$pdflatex user-guide/ug-slides-noanim
$bibtex user-guide
$pdflatex user-guide/ug-slides-noanim
$pdflatex user-guide/ug-slides-noanim

## Build Notes for Presentation
$pdflatex user-guide/ug-slides-noanim
$bibtex user-guide
$pdflatex user-guide/ug-slides-noanim
$pdflatex user-guide/ug-slides-noanim
```

Please note that the **SKB** documentation is heavily using the **SKB** macros, so you will need to have the style and class files installed before you can rebuild the documentation.

4.1.4 Configuration: `skbconfig`

`\skbconfig` There are multiple options to configure the **SKB**. The following list contains all possible options starting with the least significant. That means that the higher priority settings, which overwrite other settings, will be listed at the bottom.

- Change the file `skb.sty` in your \LaTeX distribution. This might require administrator (root) privileges. This option, while possible, is not recommended.
- Change the file `skb.cfg` in your \LaTeX distribution. This might require administrator (root) privileges. This option is suitable for a system wide configuration, say on your own computer or laptop.
- Create a file `skblocal.cfg` in your personal \LaTeX style/template directory. This will be the most convenient way to configure the **SKB**. Note: you might need to refresh the file database of your \LaTeX distribution.
- Use `\skbconfig` in your documents.

If you chose option 1 we assume you know what you are doing. In case you chose options 2-3, you can use the macro `\skbconfig` to do the configuration for you. The macro comes with options for all possible settings of the **SKB**. [Table 1](#) describes all options and shows their default value. Please note that the **SKB** can currently not handle inputs from directories outside the root hierarchy. However, one can call `\skbconfig` anytime to change the root directory, but be careful with potential side effects!

The macro `\skbconfig` requires one argument, which describes where the configuration has been changed. This is helpful in combination with the macro `\skboptionsused` to trace configuration settings. For instance, in the files `skb.cfg` and `skblocal.cfg` we should use the respective filename. When changing configuration settings elsewhere, some other descriptive text should be useful.

The following code shows the example for `\kbconfig`. The first one is the default content of the file `skb.cfg`. It basically sets all possible configuration options to their default value.

```
%default content of skb.cfg
\skbconfig[
  root=/doc,fig=figures,sli=slides
  acr=database/latex,acrfile=acronym,
  bib=database/bibtex,bibfile=bibliograhpy,
  rep=repository,pub=publish
]{skb.cfg}

%using relative path for root and no directory structure
\skbconfig[
  root=.,rep=,pub=,fig=,sli=,
  acr=,acrfile=acronym,
```

Table 1: Options for skbconfig

Option	Description	Default
<code>root</code>	Sets the root path of the SKB. Everything that the SKB processes should be located below the root.	<code>/doc</code>
<code>pub</code>	Sets the path for the published documents.	<code>publish</code>
<code>rep</code>	Sets the path for the repository documents.	<code>/repository</code>
<code>fig</code>	Sets the path for figures.	<code>/figures</code>
<code>sli</code>	Sets the path for the slides.	<code>/transparencies</code>
<code>acr</code> , <code>acrfile</code>	The SKB uses the <code>acronym</code> package and these two macros detail the directory (<code>acr</code>) and the file (<code>acrfile</code>) where the acronyms can be found.	<code>acr</code> : <code>database/latex</code> <code>acrfile</code> : <code>acronym</code>
<code>bib</code> , <code>bibfile</code>	These two macros detail the directory (<code>bib</code>) and the main file (<code>bibfile</code>) where bibliographic information (BIB _T _E _X database) can be found.	<code>bib</code> : <code>database/bibtex</code> <code>bibfile</code> : <code>bibliography</code>

```
bib,bibfile=bibliograhpy
]{myfile.tex}
```

If you want to change the configuration settings for a single document without any directory structure, overwriting all default settings (from `skb.sty`, `skb.cfg` and `skbllocal.cfg` and using the current relative path, you can use the second example shown above.

To trace the configuration settings, you can use `\skboptionsused`. Please see `###` for details on this macro.

4.1.5 Configuration: View Options Used

`\skboptionsused` This macro will print out a warning including the currently used configuration information and the change list for each of them. For example, if the configuration for `root` has not been changed the output for `root` will be

```
- root [skb.sty]: /doc
```

but if the configuration for `fig` has been changed using `\skbconfig` to `graphics` the output for `root` will be

```
- fib [skb.sty, skbconfig]: graphics
```

This macro is automatically called at the end of processing.

When creating the documentation for the **SKB** by running `pdflatex skb.dtx`, the following output will be created:

```
Package skb Warning: Options last changed by: skb-presentation
(skb) Change log:
(skb) - root = skb.sty, ug-slides-noanim.tex
(skb) - acr = skb.sty
```

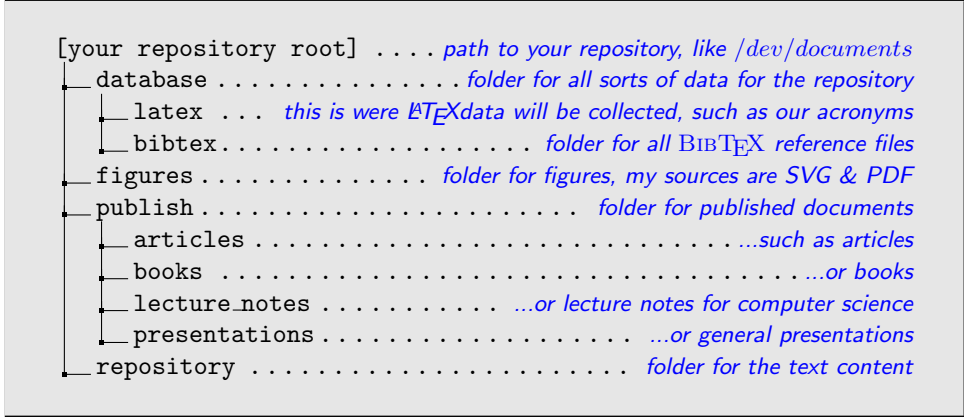


```
(skb) - acrfile = skb.sty
(skb) - bib = skb.sty
(skb) - bibfile = skb.sty
(skb) - rep = skb.sty
(skb) - pub = skb.sty, ug-slides-noanim.tex
(skb) - fig = skb.sty
(skb) - sli = skb.sty, skb-presentation
(skb) Last set Path/File Options:
(skb) - file root = user-guide/
(skb) - path root = user-guide
(skb) - file acr = user-guide/database/latex/acronym
(skb) - file bib = user-guide/database/bibtex/bibliography
(skb) - path bib = user-guide/database/bibtex
(skb) - path rep = user-guide/repository/
(skb) - path pub = user-guide//
(skb) - path fig = user-guide/figures/
(skb) - path sli = user-guide/slides/ .
```

The change log shows that all configuration options have been set by `skb.sty` and later by `skb.cfg`. Furthermore, the configuration option `root` has been changed by `skb.dtx`.

4.1.6 Creating a Directory Structure

The real power (and possibly madness) of the `SKB` comes with the separation of different parts of a document into different directory structures. For the user guide, we assume the following general directory structure .



To create this structure, go to the directory were you want to put all your documents, say `/doc`. Now create the folders `database`, `figures`, `publish` and `repository` and the respective sub-folders as shown above. Finally, configure the `SKB` by either editing one of the configuration files or adding the following line to all of your published documents (and of course change the text `myfile.tex` to something that tells you about the location of the configuration change):

```
\skbconfig[root=/doc,
acr=database/latex,acrfile=acronym,
bib=database/bibtex,bibfile=bibliograhpy,
rep=repository,pub=publish,
fig=figures,sli=slides
```

```
]{myfile.tex}
```

The directory structures for the publish folder and the repository folder reflect different views to your document base. The publish folder contains documents that are published for a reason (i.e. articles, books, presentations, white papers, work in progress) while the repository folder contains content, most likely structured following a content-specific categorisation. The choice of how the directory structure looks like is yours, and of course you could also have multiple document directories with completely different structures, for instance one for computer science publications and one for a gardening book. The **SKB** does not set any limit, since it can be configured very flexibly to your needs (please see [subsection 4.1.4](#) for more details) .

4.2 Files, Figures and Slides

4.2.1 Files and Headings

`\skbinput`
`\skbheading`

Just to remember: we have two different types of files in two different directory structures. The repository folder stores the content and the publish folder stores everything needed to produce complete documents. For the content in the repository, we mark headings with the macro `\skbheading`. This macro has no options and no arguments and is simply called with the text for the heading, as shown in the following example.

```
\skbheading{My Heading}
```

Leaving the argument empty will have the same effect as calling the original \LaTeX macros directly with an empty argument.

The association of a \LaTeX document level with the heading is then defined for the published documents (in the publish folder) using the macro `\skbinput`. This macro provides a number of options and requires one argument. The following examples shows a few use cases for `\skbinput`. For all possible options, please see [Table 3](#)

```
1 \skbinput{myfile}
2 \skbinput[from=rep]{myfile}
3 \skbinput[from=pub]{myfile}
4 \skbinput[level=chapter]{myfile}
5 \skbinput[from=pub,level=chapter]{myfile}
6 \skbinput[from=pub]{test/myfile}
```

Let's start with the simplest form, one argument only shown in line 1. The macro will look for a file called `myfile.tex` in the root directory of the **SKB**. The file extension `.tex` is automatically added, and since we did not specify any special directory the root directory is used instead. If the file is not found, the macro will throw an error providing the full path and filename it did try to load.

Line 2 shows how we can load the file `myfile.tex` from the repository folder. As you can see, the option `from` is supplied with the argument `rep`, which in fact directs the macro to look for `myfile.tex` in the repository folder. Should the file be located in the folder for published documents, we simply change the `from` option to `pub` as shown in line 3.

Table 3: Options for skbinput

Option	Description	Values
<code>from</code>	Set the directory from where the file should be loaded.	<code>pub</code> , <code>rep</code> , <code>fig</code> , <code>sli</code>
<code>level</code>	Set the document level to be used for the next occurrence of <code>\skbheading</code>	<code>book</code> , <code>part</code> , <code>title</code> , <code>chapter</code> , <code>section</code> , <code>subsection</code> , <code>subsubsection</code>

If `from` is used and neither `pub` nor `rep` is given, the macro will throw an error.

To associate a document level for the heading, we can use the option `level` to define which level we want. This option understands all standard document levels from the memoir package: `book`, `part`, `title`, `chapter`, `section`, `subsection` and `subsubsection`. So, if we want to load `myfile.tex` as a chapter we simply invoke `\skbinput` as shown in line 4 of the example.

Since `myfile.tex` is part of the repository, we combine the two options `from` and `level` (see line 5). This call to `\input` will load `myfile.tex` from the repository and use `\chapter` for the heading found in that file. If `myfile.tex` is in a sub folder, we simply add that sub folder to the filename. An example is shown in line 6 assuming the the file is located in the repository sub-folder `examples`.

4.2.2 Figures

`\skbfigure` The classic way to add figures to your document is to have a **PDF** or **PNG** or **JPG**²¹ file ready, include it using `\includegraphics` while putting it into a box to resize it (i.e. to the width of the text in your document), putting this very box into a figure environment so that L^AT_EX can process list of figures etc. and of course adding `\label` and `\caption` to it. Here is some L^AT_EX example, which also uses the `center` environment:

```
\begin{figure}\begin{center}
  \resizebox{\textwidth}{!}{
    \includegraphics[width=\textwidth]{../figures/myfig}
  }
  \caption{My Figure}\label{myfig}
\end{center}\end{figure}
```

With the **SKB** macro `\skbfigure` things become a little bit simpler. `\skbfigure` takes a number of options and one argument. The following code shows a number of examples for using this macro.

```
1 \skbfigure{myfig}
2 \skbfigure[figure,center]{myfig}
3 \skbfigure[figure,center,width=\textwidth]{myfig}
4 \skbfigure[figure,center,
5   caption=My Figure,label=myfig]{myfig}
```

Let's start with the easy usage, simply using the one argument to load a figure, as shown in line 1. This call will simply use `\includegraphics` and `\resizebox`

²¹Joint Photographic Experts Group (JPG)

to load the figure `myfig` from the figure directory, so we do not need to say `../figures` anymore. To use the figure and the center environment, we simply add two options requesting exactly that, as shown in line 2. In other words, using the option `figure` will put the `myfig` in a figure environment and using the option `center` will center the figure.

Similar for width and height information. Say the figure should be rescaled to the width of the text in your document you simply add `width` to the options, as shown in line 3 Use `height` for height or both options if required. Note that the width and the height are automatically applied to the `\resizebox` and `\includegraphics`. You can also add caption and label information using the respective options (lines 4 and 5). Now we will have the same result as the classic L^AT_EX example. You can also add the required position for your figure, if using the figure environment applying the option `position` with the usual parameters, including `H` from the float environment.

Table 4: Options for `skbfigure`

Option	Description
<code>width</code>	Set the width to be used with <code>\resizebox</code> and <code>\includegraphics</code> .
<code>height</code>	Set the height to be used with <code>\resizebox</code> and <code>\includegraphics</code> .
<code>center</code>	Use center environment.
<code>figure</code>	Use figure environment.
<code>position</code>	The position to be used within figure environment. This option will be ignored if not combined with <code>figure</code> .
<code>caption</code>	The caption to be used. Ignored if the option <code>figure</code> is not used.
<code>label</code>	The label to be used. Ignored if the option <code>figure</code> is not used.
<code>multiinclude</code>	The label to be used. Ignored if the option <code>figure</code> is not used.

The last option for the macro `\skbfigure` is called `multiinclude`. It can be used with the beamer package to realise animations by loading a series of images and showing them in sequence with or without overlaying. If used, this option will overwrite all other options resulting in a simple call to `\multiinclude` within a resised box. One can use all standard multiinclude parameters with `\skbfigure`, just omit the enclosing brackets. For instance, if you want to use multiinclude on the `myfig` with the options `<+->` call

```
\skbfigure [ multiinclude=+-]{myfig}
```

The figure size will be automatically set to `\textwidth` and the height to `!`. The start of the multiinclude is fixed to be 0, the format is PDF. For more information on how to use multiinclude please refer to `mpmulti` and `beamer` packages.

4.2.3 Slides

`\skbslide` This macro helps to create lecture notes (handouts) using PDF slides and L^AT_EX notes without using the beamer package. The reason for adding this to the SKB was to integrate slides from sources outside the L^AT_EX universe (i.e. Microsoft Powerpoint). Some of my presentations are done using Powerpoint, but for handouts I do prefer using L^AT_EX thus benefiting from many of the automated features it provides (references, acronyms). As a nice side effect, the output generated is scalable (assuming that the PDF sources of the slides contain scalable images rather than bitmaps, which can be easily realised using for instance Inkscape's EMF export within Microsoft Powerpoint slides).

The macro `\skbslide` provides all means to include PDF slides with or without annotations, annotations only and it can load the annotations using different mechanisms. The macro offers two options to set the input path for the slides and the annotations: `slidefrom` and `notefrom`. If `slidefrom` is used, then the slide (PDF) file will be loaded from the requested path (`sli`, `rep` or `pub`). If `notefrom` is used, then the annotation (T_EX) file will be loaded from the requested path (`sli`, `rep` or `pub`). The default path for slides and annotations is the path for slides.

The third option `annotate` requests to load annotations. If not used, no annotations will be loaded. It can be used in combination with the two arguments to automated loading annotations.

The two arguments of this macro define the files for the slide and the annotation. They can be used as follows:

- Argument 1 is the slide to be loaded. If a name is given, we load the PDF using `\inputgraphics` with width being `\textwidth`. If no name is given, no slide will be loaded.
- Argument 2 is the file with the annotations in combination with the option `annotate`. If this option is not used then no annotations will be loaded. If the option is used and no name is given, then the annotation is loaded from a file with the same name as the slide plus the extension `.tex`. If this option is used and a name is given then this file will be loaded.

This provides the following combinations for `\skbslide`

- Slide only: argument 1 has the name for the PDF, argument 2 is empty
- Annotation only: argument 1 is empty, argument 2 has the name for the T_EX file, option `annotate` used
- Slide with Annotation 1: argument 1 has the name for the PDF, argument 2 has the name for the T_EX file, option `annotate` used
- Slide with Annotation 2: argument 1 has the name for the PDF, argument is empty, option `annotate` used
- do nothing: leave both arguments empty

Some examples on how to use `\skbslide`:

```
1 \skbslide{myslides/slide1}{}  
2 \skbslide{myslides/slide2}{}\clearpage
```

```

3 \skbslide[annotate]{myslides/slide3}{}
4 \skbslide[annotate,notefrom=rep]
5   {myslides/theme1}{text/theme1}
6 \skbslide[annotate,notefrom=rep,slidefrom=rep]
7   {text/theme2}{text/theme2}

```

In line 1 and 2 we load `myslides/slide1.pdf` and `myslides/slide2.pdf` from the default directory without any annotations and clear the page after that. In line 3 we load `myslides/slide2.pdf` and request this slide to be annotated without giving a specific file name, thus loading `myslides/slide3.tex`, both files from the default slides directory. In lines 4&5 we change the directory for the notes and request a particular file to be loaded, resulting in the slide loaded as `myslides/theme1.pdf` from the slides directory and the annotations loaded as `text/theme1.tex` from the repository. Finally, in lines 6&7 we change both folders to the repository, this loading `text/theme2.pdf` and `text/theme2.tex` from the repository.

`\skbslidecite` The macro `\skbslidecite` provides some simple means to add citations to annotated slides. It takes two arguments, the first one for the type of citation and the second one for the actual citation. Here a simple example:

```

1 \skbslidecite{Slide}{\cite{tanenbaum-andrew:book:2003}}
2 \skbslidecite{Notes}{\cite{standard:IETF:RFC:1155}}

```

The first line states that the slide contains material from a book of Tannenbaum and the second line states that the annotation contains material from an [IETF RFC](#)²² standard documents ([1]). Since this macro is very simple, any content can be given for the two arguments.

4.3 Filenames, Acronyms and References

4.3.1 Path and File Names

`\skbfileroot` The **SKB** provides a number of macros to directly create path and file names. Most of these macros are actually used within the **SKB**, but they might also be useful for users to access files without using the provided specialised macros (such as `\skbinput`). The following macros are provided:

- `\skbpathroot` – returns the set root path of the **SKB**.
- `\skbfileroot` – returns the set root path and adds `/#1`, i.e. the directory separator and the argument provided.
- `\skbfileacr` – returns the path (including root) and file name for the acronym database.
- `\skbfilebib` – returns the path (including root) and file name for the file that loads the reference database (`BIBTEX`).
- `\skbpathbib` – returns the path (including root) to the reference database.
- `\skbfilerep` – returns the path to the repository and adds `/#1`, i.e. the directory separator and the argument provided.

²²Request for Comment ([RFC](#))

- `\skbfilepub` – returns the path to the folder with the published documents and adds `/#1`, i.e. the directory separator and the argument provided.
- `\skbfilefig` – returns the path to the figure folder and adds `/#1`, i.e. the directory separator and the argument provided.
- `\skbfilesli` – returns the path to the slide folder and adds `/#1`, i.e. the directory separator and the argument provided.

4.3.2 Loading Acronyms and Bibliographic Information

`\skbacronyms` These two macros can be used to load the acronym database (`\skbacronyms`)
`\skbbibtex` and the references (`\skbbibtex`). Both macros behave identical: they use `\InputIfFileExists` to check whether the acronym or bibtex file exists. If so, they simply input the file. If not, they use `\PackageError` to throw an error with a help message, showing the requested database file to input. One should use `\skbacronyms` at the place in the document where the list of acronyms should be printed and `\skbbibtex` at the beginning of the document to load the bibliographic information.

4.4 Other useful Macros

4.4.1 Emphasising Text

`\skbem` Highlighting or emphasising text is an important aspect of many technical documents. One can use \LaTeX macros directly to set text in italic or bold. This has the disadvantage that there is no meaningful information given as on why that text is treated in a special way. Furthermore, when the editor requires to change certain highlights, it will be very difficult to go through a large document and figure out which text is to be changed.

To prevent that from happening, one can use \LaTeX macros to actually distinguish between different highlighted text. A simple start is provided by the `SKB`. It is simply because, at the moment, it only supports three different ways and no further meaningful information. But it is a start.

The macro `\skbem` comes with three different options. The option `bold` will set the text given in the argument in bold face. Similar, the option `italic` will set it italic. Last not least, the option `code` will use another `SKB` macro (`\skbcode`) for typesetting the argument text. The following code shows some examples for the macro:

```
Use \cmd{\skbem} to produce \skbem[bold]{bold},
\skbem[italic]{italic} or \skbem[code]{type writer} text.

The example above shows the macro \skbem[code]{skbem} with
the option \skbem[italic]{bold} and \skbem[bold]{italic}.
```

And here the final type setting of that example:

Use `\skbem` to produce **bold**, *italic* or `type writer` text.

The example above shows the macro `skbem` with the option *bold* and **italic**.

`\skbcode` This macro `\skbcode` is a facade for calling the macro `\stinline` from the `listing` package with a basic style that uses type writer font (`ttfamily`).

4.4.2 Environments for lists and enumerates

`\skbnotelist` These two environments mimic the macro `\tightlists` from the memoir package.
`\skbnoteenum` It might be useful when not using memoir to minimise the margin between items in lists (`itemize`) and enumerations (`enumerate`).

Both environments do the following:

- Store current value of `\parskip` and `\itemsep`.
- Set `\parskip` and `\itemsep` to 0cm.
- Use the original environments (`itemize` for `skbnotelist` and `enumerate` for `skbnoteenum`)
- Set `\parskip` and `\itemsep` back to their original value.

Here is an example using first the classic list environment (`itemize`) and then the **SKB** macro `\kbnotelist` ²³ ²⁴ :

- Lorem ipsum dolor sit amet, consectetur adipiscing 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.
- Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
- Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Now list with `\skbnotelist`:

- Lorem ipsum dolor sit amet, consectetur adipiscing 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.
- Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
- Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Note: both macros will only change the margins of the memoir package if not loaded!

4.4.3 Macros for PDF Info

`\skbtitle` The macro `\skbtitle` will set the title to be used for **PDF** info. The default for

²³For those who are interested, the ‘Lorem Ipsum’ is the standard phrase commonly used since the 1500s.

²⁴The following examples might have no effect on annotated slides...

the title is an empty string.

- `\skbauthor` The macro `\skbauthor` will set the author information to be used for PDF info. The default for the author is an empty string.
- `\skbsubject` The macro `\skbsubject` will set the subject information to be used for PDF info. The default for the subject is an empty string.
- `\skbkeywords` The macro `\skbkeywords` will set the keywords to be used for PDF info. The default for the keywords is an empty string.
- `\skbpdfinfo` The macro `\skbpdfinfo` will call the macro `\pdfinfo` to set the meta information in the created PDF output file. The `SKB` automatically calls this macro just before finishing the process of the main document, using the information provided by the above described macros. Furthermore, the date of the PDF file will be set.

4.4.4 Listings Styles and Support

The `SKB` comes with a few predefined styles for the listing package. Most of them use type writer font in scriptsize, arrange a grey box around the listing and set the keywords to Blue4.

- generic – for any generic listing without specifying a language and no line numbers.
- genericLN – same as generic, just with line number in the left side, which means allowing extra space left to the listing box.
- gentab – almost the same as generic, but without definitions for frame and numbers, which seem to collide with some table environments.
- genericLNspecial – same as genericLN, just with a lighter grey for the box.
- beamer-example – style designed for examples in beamer frames.
- beamer-exampleLN – same as beamer-example, just with line numbers on the left, which means allowing extra space left to the listing box.
- javaCode – generic style plus language Java.

- `\lstdefinestyle` There is also one macro supported, which sets the listing style back to normal, i.e. after changing it in the text. Some macros in the `SKB` make use of this. All that `\lstdefinestyle` does is setting the basic style back to type writer font.

4.4.5 Optional Text – Versions and Optional

The `SKB` provides two means to include text and other L^AT_EX commands on an optional basis. They are pre-configured and will be automatically set/unset according to the three main document types the `SKB` supports:

- text – is equivalent to any classic text document, for instance an article or a book.

- `slide` – is used to identify slides, for instance beamer frames.
- `note` – is used to identify lecture notes or handouts, in essence annotated slides (frames).
- `anim` – for beamer frames, used for text with animation activated.
- `noanim` – for beamer frames, used for text with animation deactivated.
- `memoir` – used for documents that include the memoir package.

We use the packages `versions` and `optional` and support both. The main difference is that with `versions` one has to use `\begin` and `\end` while with `optional` one can use more than one of the above introduced types. The macros for provided for optional text are:

- `\skbmodetext` and options using `text` – will be valid if neither beamer nor beamerarticle is loaded (normal text).
- `\skbmodeslide` and options using `slide` – will be valid if the beamer package is loaded (slides).
- `\skbmodenote` and options using `note` – will be valid if the beamerarticle package is loaded (annotated slides).
- `\skbmodeanim` and options using `anim` – will be valid if the beamer package is loaded and the `SKB` is loaded with the argument `beameranim`
- `\skbmodenoanim` and options using `noanim` – will be valid if the beamer package is loaded and the `SKB` is loaded with the argument `beamernoanim`
- `\skbmodememoir` and options using `memoir` – will be valid if the memoir package is loaded

The following code shows a few examples on how to use the optional text.

5 Examples

A Simple Article

Take the article that describes the state of the art in protocols. Remember, we have all the contents for that in our `repository` directory. We go the directory that has the published articles `published/articles` and create a new file say `protocols.tex` .

```
\documentclass{skbarticle}

\begin{document}
  \author{Sven van der Meer}
  \title{Protocols, Formats and Communication Services}
  \maketitle
  \tableofcontents*
  \bigskip

  \skbinput[from=rep]{sota/protocols}

  \section{Introduction}
  \skbinput[from=rep,level=subsection]
    {sota/protocols/data_encoding}
```

```

\skbinput [from=rep, level=subsection]
  {sota/protocols/message-formates}
\skbinput [from=rep, level=subsection]
  {sota/protocols/protocols}
\skbinput [from=rep, level=subsection]
  {sota/protocols/protocol-services}

\skbinput [from=rep, level=section]{sdo/omg/corba-giop}
\skbinput [from=rep, level=section]{sdo/ietf/snmp-protocol}
\skbinput [from=rep, level=section]{sdo/itu/x700-cmip}
\skbinput [from=rep, level=section]{sdo/w3c/http}

\end{document}
\endinput

```

The article uses the class `skbarticle`. That class will load the `SKB` package and the memoir class and do all settings we need. It prepares the title page and prints the table of contents like any other \LaTeX article. It uses `\skbinput` to load files from the repository. The first one is loaded without requesting a level. In other words, there is some text right at the beginning of our article, without any special heading, like an abstract.

Then we do start the section 'Introduction' and collect a few files with their heading categorised as sub-sections. Reading the directory and file names, we can already guess what the introduction will be doing: it introduces general protocol concepts with regard to data encoding, protocol message formats, protocols themselves and protocol services. The last block loads four files with headings categorised as sections. Using the directory names, we see that the remaining article describes the protocols The General Inter-ORB Protocol (`GIOP`) defined by the `OMG`, Simple Network Management Protocol (`SNMP`) by the `IETF`, Common Management Information Protocol (`CMIP`) by the International Telecommunication Unit (`ITU`) and finally Hyper Text Transfer Protocol (`HTTP`) by the World Wide Web Consortium (`W3C`).

Finally, we load acronyms and bibliography and finishing the article. This example will create a table of contents similar to this:

```

1 Introduction . . . . . 1
  1.1 Data Encoding . . . . . 2
  1.2 Message Formats . . . . . 5
  1.3 Protocols . . . . . 7
  1.4 Protocol Services . . . . . 9
2 General Inter-ORB Protocol . . . . . 10
3 Simple Network Management Protocol . . . . . 13
4 Common Management Information Protocol . . . . . 15
5 Hypertext Transport Protocol . . . . . 18

```

Job done. Now we can use \LaTeX or PDF- \LaTeX to compile our article.

6 Implementation: Kernel

First we do announce the package.

```
1 \*skbpackage)
```

```

2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{skb}[2011/06/03 Sven's Knowledge Base - SKB for LaTeX v0.52]

```

Next we process the package's options. To do that, we define a new if that indicates if we process slides with or without animation, and then we set that new if accordingly.

```

4 \newif\if@skbBeamerAnim
5 \@skbBeamerAnimfalse
6 \DeclareOption{beameranim}{\@skbBeamerAnimtrue}
7 \DeclareOption{beamernoanim}{\@skbBeamerAnimfalse}
8 \ProcessOptions\relax

```

6.1 Required Packages

Now we load a few packages that we need within the **SKB**. We use `keyval` to allow for options in macros, the `listings` package for all listings, `dirtree` to show tree structures similar to a directory tree, `ifpdf` to establish whether we use **PDF** or not, `datetime` to get the current date and the `versions` package to allow for optional text. Note: some packages, such as the package `optional`, are loaded at a later stage.

```

9 \RequirePackage{keyval}
10 \RequirePackage{listings}
11 \RequirePackage{dirtree}
12 \RequirePackage{ifpdf}
13 \RequirePackage{datetime}
14 \RequirePackage{versions}

```

6.2 Conditiona/Optional Text Support

Now we set everything that we need to provide optional text. Basically, we want to distinguish between the following modes: `text` (normal text), `slide` (for slides), `note` (for slite annotations), `anim` (for animated slides), `noanim` (for non-animated slides) and `memoir` (if we use the `memoir` package).

We start with the `memoir` package. First we define a configuration value (used when loading the package `optional`) and a new if (telling us later if `memoir` is loaded or not).

```

15 \def\skb@cfg@memoir{}
16 \newif\ifSkbMemoirLoaded

```

Now we test for the `memoir` package. Note, if this package is loaded after the **SKB**, this test and all following actions will fail. If the package is loaded, then we set the if to true, activate (include) the environment `skbmodememoir` and set our configuration value to the string `" , memoir"`. If the `memoir` package is not loaded, then we set the if to false, deactivate (exclude) the environment `skbmode-`

memoir and load the package booktabs (to provide the commands `\toprule` and `\bottomrule`).

```

17 \@ifclassloaded{memoir}
18   {\SkbMemoirLoadedtrue
19     \includeversion{skbmodememoir}
20     \def\skb@cfg@memoir{,memoir}}
21   {\SkbMemoirLoadedfalse
22     \excludeversion{skbmodememoir}
23     \RequirePackage{booktabs}}

```

Now we check for the style beamerarticle. We define an if, set its default value to false and test for of the package is loaded (if so, we change the if to true).

```

24 \newif\ifSkbBeamerArticleLoaded
25 \SkbBeamerArticleLoadedfalse
26 \@ifpackageloaded{beamerarticle}{\SkbBeamerArticleLoadedtrue}{}

```

Now we check for the beamer package. e define an if, set its default value to false and test for of the package is loaded (if so, we change the if to true).

```

27 \newif\ifSkbBeamerLoaded
28 \SkbBeamerLoadedfalse
29 \@ifclassloaded{beamer}{\SkbBeamerLoadedtrue}{}

```

Now we process the first optional text support. First, we define a configuration value for beamer animations. If animations are requested (skb package option, see above), we set that value to the string ",anim" and activate (include) the environment skbmodeanim and deactivate (exclude) the environment skbmodenoanim. If no-animation is requested (skb package option, see above) or as default we set the value to the string ",noanim" and deactivate (exclude) the environment skbmodeanim and activate (include) the environment skbmodenoanim.

```

30 \def\skb@cfg@beameranim{}
31 \if@skbBeamerAnim
32   \def\skb@cfg@beameranim{,anim}
33   \excludeversion{skbmodenoanim}
34   \includeversion{skbmodeanim}
35 \else
36   \def\skb@cfg@beameranim{,noanim}
37   \excludeversion{skbmodeanim}
38   \includeversion{skbmodenoanim}
39 \fi

```

Now we are ready to provide for all other optional text support. The code configures the environments skbmodetext, skbmodenote and skbmodeslide and loads the optional package depending if we have the beamer package loaded or have the package beamerarticle loaded or have none of the two packages loaded. The environments (package versions) are excluded or included accordingly. The package optional is loaded with the respective option activated (text, note or slide) and using the two configuration values we have defined above (these values are either

empty having no effect or contain the option to be included).

```

40 \ifSkbBeamerLoaded
41   \excludeversion{skbmodetext}
42   \excludeversion{skbmodenote}
43   \includeversion{skbmodeslide}
44   \RequirePackage[slide\skb@cfg@memoir\skb@cfg@beameranim]{optional}
45 \else\ifSkbBeamerArticleLoaded
46   \excludeversion{skbmodetext}
47   \includeversion{skbmodenote}
48   \excludeversion{skbmodeslide}
49   \RequirePackage[note\skb@cfg@memoir\skb@cfg@beameranim]{optional}
50 \else
51   \includeversion{skbmodetext}
52   \excludeversion{skbmodenote}
53   \excludeversion{skbmodeslide}
54   \RequirePackage[text\skb@cfg@memoir\skb@cfg@beameranim]{optional}
55 \fi\fi

```

6.3 Provide Command

The **SKB** provides for a few commands that the documentation (and maybe your documents as well) expect to be available. The first two are for typesetting SKB and BibTeX, the rest are simply useful.

```

\BibTeX
\DescribeMacro
\cmdprint
\cmd
56 \providecommand{\BibTeX}{\scshape Bib}\TeX}
57 \providecommand{\DescribeMacro}[1]{\relax}
58 \providecommand{\cmdprint}[1]{\texttt{\string#1}}
59 \providecommand{\cmd}[1]{\cmdprint{#1}}%

```

6.4 Macro Redefinitions

The **SKB** documentation uses the package `dirtree` and we want to have some of its default settings changed. For the comments, the default configuration we want is an small, italic serif font in blue; and for the style part we want a type writer font in black.

```

60 \renewcommand*{\DTstylecomment}{\itshape\sffamily\color{blue}\small}
61 \renewcommand*{\DTstyle}{\ttfamily\textcolor{black}}

```

6.5 At End of Document

Last not least, we define what should happen at the end of the processing of the input document. At them moment, we call `\skbpdfinfo` to set **PDF** meta information and `\skboptionsused` to print out the change log and current set of **SKB** configuration options.

```

62 \AtEndDocument{
63   \skbpdfinfo
64   \skboptionsused
65 }

```

6.6 Package Configuration

The basic idea of the **SKB** is that different parts of a document (figures, slides, repository, published documents) reside in different folders. So the main configuration of the **SKB** is to provide macros to set and get these folders and to load files from them.

To simplify coding, we introduce some macros that handle configuration information. These macros will be used by the **SKB** package to define, set and get configuration information. The macros also store the origin of changes to the configuration information.

`\skb@tmp` This variable is used to temporarily store macros and strings. The value can change anytime a new **SKB** macro is called.

```
66 \newcommand{\skb@tmp}{}

```

`\skb@cfg@origlast` Is used to store the last location (second argument of `\skbconfig`) where any configuration information has been changed. The currently possible locations are `skb.sty` for default values, `skb.cfg` for the general configuration file, `skblocal.cfg` for the local configuration file and `skbconfig` when the macro `\skbconfig` was called.

```
67 \newcommand{\skb@cfg@origlast}{skb.sty}

```

`\skb@defCfgVars` This macro is used to define new configuration information. It defines two new macros, one for the name of the configuration information and one for storing a change log. The first argument is the name to be used and the second argument the default initialisation. For instance, to add the configuration information for the root path with the default value `‘/doc’` call

```
\skb@defCfgVars{root}{/doc}
```

```

68 \newcommand{\skb@defCfgVars}[2]{
69   \@namedef{skb@cfg@var@#1}{#2}
70   \@namedef{skb@cfg@orig@#1}{skb.sty}
71 }

```

`\skb@setCfgVars` Alter configuration information and append the location from where its called (second argument of `\skbconfig` taken from `\kb@cfg@origlast`) to the change log.

```

72 \newcommand{\skb@setCfgVars}[2]{
73   \@namedef{skb@cfg@var@#1}{#2}
74   \xandafter\protected@edef\csname skb@cfg@orig@#1\endcsname%

```

```
75   {\csname skb@cfg@orig@#1\endcsname,\space \skb@cfg@origlast}%
76 }
```

`\skb@getCfgVars` This macro provides access to configuration values. It is used everywhere in the **SKB** to retrieve configuration values.

```
77 \newcommand{\skb@getCfgVars}[1]{%
78   \csname skb@cfg@var@#1\endcsname%
79 }
```

Now we use `\skb@defCfgVars` to initialise all configuration values the **SKB** uses.

`\skb@cfg@var@root` The first one is the root directory. Everything that the **SKB** processes should be located below the root. The **SKB** can currently not handle inputs from directories outside the root hierarchy (Note: one can call `\skbconfig` anytime to change the root directory, but be careful with potential side effects!). The default value for the root directory is `/doc`.

```
80 \skb@defCfgVars{root}{/doc}
```

`\skb@cfg@var@acr` These two values define the directory and the file name for the acronym database.
`\skb@cfg@var@acrfile` The **SKB** uses the `acronym` package and the two macros detail the directory (`acr`) and the file (`acrfile`) where the acronyms can be found. The default for the directory is `database/latex` and the default for the file is `acronym`.

```
81 \skb@defCfgVars{acr}{database/latex}
82 \skb@defCfgVars{acrfile}{acronyms}
```

`\skb@cfg@var@bib` These two values define the directory and the file name for the **BIB_TE_X** database.
`\skb@cfg@var@bibfile` The two macros detail the directory (`bib`) and the main file (`bibfile`) where bibliographic information can be found. The default for the directory is `database/bibtex` and the default for the file is `bibliography.tex`.

```
83 \skb@defCfgVars{bib}{database/bibtex}
84 \skb@defCfgVars{bibfile}{bibliography.tex}
```

`\skb@cfg@var@rep` This value points to the `repository` directory. The default value is `repository`.

```
85 \skb@defCfgVars{rep}{repository}
```

`\skb@cfg@var@pub` This value points to the folder with the published documents. The default value is `publish`.

```
86 \skb@defCfgVars{pub}{publish}
```

`\skb@cfg@var@fig` This value points to the directory for figures. The default value is `figures`.

```
87 \skb@defCfgVars{fig}{figures}
```

`\skb@cfg@var@sli` This value points to the directory for slides. The default value is `transparencies`.

```
88 \skb@defCfgVars{sli}{transparencies}
```


6.7 Generic Input Macro

`\skb@input@doife` `\skb@input@doife` is the generic input macro. It expects four arguments. The first argument is the **SKB** macro that should be used to input a file. The second argument is the actual file to be loaded, without file extension. The third argument is the file extension to be used. The fourth argument is plain text that should be added to the help message in case an error occurred while loading the file. If the second and third argument are empty, we assume that the first argument already contains directory and file and file extension information.

```
89 \newcommand{\skb@input@doife}[4]{%
90   \def\filearg{#2}
91   \ifx\filearg\empty%
92     \edef\intfile{\csname #1\endcsname}%
93   \else%
94     \edef\intfile{\csname #1\endcsname{#2}#3}%
95   \fi%
96   \InputIfFileExists{\intfile}{}%
97   {\PackageError{skb}%
98     {file not found: \intfile}%
99     {I did not find the requested file #4,%
100      \MessageBreak please check: \intfile%
101      \MessageBreak <return> to continue, no file loaded}%
102   }%
103 }
```

6.8 Kernel support for skbinput

This is the actual core functionality of the **SKB** package: flexibly load files from various pre-defined locations (folders). We start with a few macros that we can use later to test options using the package keyval.

`\skb@input@var@rep` This macro represents the string "rep", which will be later used to test for macro options, for instance in `\skbinput`.

```
104 \def\skb@input@var@rep{rep}
```

`\skb@input@var@pub` This macro represents the string "pub", which will be later used to test for macro options, for instance in `\skbinput`.

```
105 \def\skb@input@var@pub{pub}
```

`\skb@input@var@fig` This macro represents the string "fig", which will be later used to test for macro options, for instance in `\skbinput`.

```
106 \def\skb@input@var@fig{fig}
```

`\skb@input@var@sli` This macro represents the string "sli", which will be later used to test for macro options, for instance in `\skbinput`.

```
107 \def\skb@input@var@sli{sli}
```

The next set of macros will load files from various supported folders. All of them behave identical: they expect argument 1 being the requested file and use `\InputIfFileExists` to check whether this file exists. If so, they simply input the file using `\input`. If not, they use `\PackageError` to throw an error with a help message, showing the requested directory and file. The extension `.tex` is automatically added to the argument, which in turn should only contain the path and the basename of the file.

```

\skb@input@doroot Load a given .tex file from the root directory.

108 \newcommand{\skb@input@doroot}[1]{%
109   \def\intarg{#1}
110   \skb@input@doife{skbfileroot}{\intarg}{.tex}{in given location}
111 }

\skb@input@dorep Load a given .tex file from the repository.

112 \newcommand{\skb@input@dorep}[1]{%
113   \def\intarg{#1}
114   \skb@input@doife{skbfilerrep}{\intarg}{.tex}{in the repository}
115 }

\skb@input@dopub Load a given .tex file from the directory with the published documents.

116 \newcommand{\skb@input@dopub}[1]{%
117   \def\intarg{#1}
118   \skb@input@doife{skbfilepub}{\intarg}{.tex}{in the published document folder}
119 }

\skb@input@dofig Load a given .tex file from the figure directory.

120 \newcommand{\skb@input@dofig}[1]{%
121   \def\intarg{#1}
122   \skb@input@doife{skbfilefig}{\intarg}{.tex}{in the figure folder}
123 }

\skb@input@dosli Load a given .tex file from the slide directory.

124 \newcommand{\skb@input@dosli}[1]{%
125   \def\intarg{#1}
126   \skb@input@doife{skbfilesli}{\intarg}{.tex}{in the slide folder}
127 }

\skb@input@call These two macros are used to load files. \skb@input@call will point to the
\skb@input@set currently requested load macro (see above). \skb@input@set sets the default load
option in \skb@input@call to \skb@input@doroot. That means if no option is
given for an input directory, then the SKB root directory will be used.

128 \def\skb@input@call{}
129 \newcommand\skb@input@set{%
130   \gdef\skb@input@call{\skb@input@doroot}
131 }

```

7 Implementation: Configuring the SKB

7.1 Changing Configuration: skbconfig

7.1.1 The Macro Options

The macro provides one option per **SKB** configuration value. Each option expects one parameter; the new value. The options are `root` (for the root directory), `acr` (for the acronym directory), `acrfile` (for the acronym file), `bib` (for the bibtex directory), `bibfile` (for the bibtex file), `rep` (for the repository directory), `pub` (for the directory with the published documents) and `sli` (for the directory with slides).

```
132 \define@key{skbconfig}{root}[]{\skb@setCfgVars{root}{#1}}
133 \define@key{skbconfig}{acr}[]{\skb@setCfgVars{acr}{#1}}
134 \define@key{skbconfig}{acrfile}[]{\skb@setCfgVars{acrfile}{#1}}
135 \define@key{skbconfig}{bib}[]{\skb@setCfgVars{bib}{#1}}
136 \define@key{skbconfig}{bibfile}[]{\skb@setCfgVars{bibfile}{#1}}
137 \define@key{skbconfig}{rep}[]{\skb@setCfgVars{rep}{#1}}
138 \define@key{skbconfig}{pub}[]{\skb@setCfgVars{pub}{#1}}
139 \define@key{skbconfig}{fig}[]{\skb@setCfgVars{fig}{#1}}
140 \define@key{skbconfig}{sli}[]{\skb@setCfgVars{sli}{#1}}
```

7.1.2 The Macro

`\skbconfig` This macro allows to change the main directory and path information for the **SKB**. It reads the provided options and changes the requested values in the **SKB**. The macro takes one argument which will set the origin of the configuration change. If this argument is empty, the origin will be set to `skbconfig`.

```
141 \newcommand{\skbconfig}[2][]{
142   \def\intarg{#2}
```

If no second argument is given, then set `\skb@cfg@origlast` to the string "skbconfig" (this macro's name) otherwise use the second argument to set `\skb@cfg@origlast`. In both cases, print out a general warning about the change of configuration values for later trace or debugging.

```
143   \ifx\intarg\empty
144     \renewcommand{\skb@cfg@origlast}{skbconfig}
145     \PackageWarning{skb}{load options overwritten by skbconfig}
146   \else
147     \renewcommand{\skb@cfg@origlast}{#2}
148     \PackageWarning{skb}{load options overwritten by #2}
149   \fi
```

Now use the `keyval` package to process the options. They will set the respective configuration values, so there is nothing else to do here.

```

150 \setkeys{skbconfig}{#1}
151 }

```

7.2 Changing Configuration: `skb.cfg` and `skbllocal.cfg`

The **SKB** can also be configured using external configuration files. Two files will be loaded if they exist:

- `skb.cfg` – Should be used with the installed package in your $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ distribution. If it exists, it will overwrite the default options for directories and paths.
- `skbllocal.cfg` – Should be used in your local styles/template directory. If it exists, it will overwrite the default options as well as the options loaded with `skb.cfg`.

We use `\InputIfFileExists` to test if the configuration file exist. If true, we load the configuration file and print out a general warning for later trace or debugging. If not, we simply do nothing.

```

152 \InputIfFileExists{skb.cfg}{%
153 \PackageWarning{skb}{load options from skb.cfg}
154 }{}
155 \InputIfFileExists{skbllocal.cfg}{%
156 \PackageWarning{skb}{load options from skbllocal.cfg}
157 }{}

```

7.3 Viewing Configuration: `skboptionsused`

`\skboptionsused` This macro can be used to print out a message (as package warning), which contains the change log and the currently used value for all **SKB** configuration values.

```

158 \newcommand{\skboptionsused}{
159 \PackageWarningNoLine{skb}{%
160 Options last changed by: \skb@cfg@origlast \MessageBreak
161 Change log: \MessageBreak
162 - root = \skb@cfg@orig@root \MessageBreak
163 - acr = \skb@cfg@orig@acr \MessageBreak
164 - acrfile = \skb@cfg@orig@acrfile \MessageBreak
165 - bib = \skb@cfg@orig@bib \MessageBreak
166 - bibfile = \skb@cfg@orig@bibfile \MessageBreak
167 - rep = \skb@cfg@orig@rep \MessageBreak
168 - pub = \skb@cfg@orig@pub \MessageBreak
169 - fig = \skb@cfg@orig@fig \MessageBreak
170 - sli = \skb@cfg@orig@sli \MessageBreak
171 Last set Path/File Options: \MessageBreak
172 - file root = \skbfileroot{} \MessageBreak
173 - path root = \skbpathroot \MessageBreak

```

```

174 - file acr = \skbfileacr \MessageBreak
175 - file bib = \skbfilebib \MessageBreak
176 - path bib = \skbpathbib \MessageBreak
177 - path rep = \skbfilerep{} \MessageBreak
178 - path pub = \skbfilepub{} \MessageBreak
179 - path fig = \skbfilefig{} \MessageBreak
180 - path sli = \skbfilesli{}
181 }
182 }

```

8 Implementation: Files, Figures and Slides

8.1 Declaring Headings: skbheading

`\skbheading` This macro can be used everywhere to declare a new heading and let the **SKB** decide which document level to use. The actual document level must be declared in the loading file using `\skbinput` with the option `level`, otherwise this command will have no effect.

```

183 \newcommand{\skbheading}[1]{
184   \ifx\empty\skb@inputLevel
185     #1
186   \else%
187     \skb@inputLevel{#1}%
188   \fi
189 }

```

8.2 Loading T_EX files: skbinput

8.2.1 Macro Options

`skbinput: opt from` The option `from` is used to point to one of the following **SKB** directories: the repository (`from=rep`), the folder with the published documents (`from=pub`), the figure folder (`from=fig`) or the slide folder (`from=sli`). The option is optional, but when used must give one of the those values. The **SKB** will throw an error otherwise. The implementation works as follows: if the option is used, its parameter is evaluated. Depending on which **SKB** directories is requested, the value `\skb@input@call` is set to point to the respective load macro. For instance, if the requested directory is the repository (`from=rep`) then `\skb@input@call` will be pointed to `\skb@input@dorep`.

```

190 \define@key{skbinput}{from}[]{}%
191   \def\intarg{#1}
192   \ifx\skb@input@var@rep\intarg
193     \gdef\skb@input@call{\skb@input@dorep}
194   \else\ifx\skb@input@var@pub\intarg

```

```

195 \gdef\skb@input@call{\skb@input@dopub}
196 \else\ifx\skb@input@var@fig\intarg
197 \gdef\skb@input@call{\skb@input@dofig}
198 \else\ifx\skb@input@var@sli\intarg
199 \gdef\skb@input@call{\skb@input@dosli}
200 \else
201 \PackageError{skb}%
202 {Value for option \@tempa\space not supported: \intarg}%
203 {I do not know the value \intarg\space for the option \@tempa.%
204 \MessageBreak Please use either "rep", "pub", "fig" or "sli".%
205 \MessageBreak <return> to continue, no file will be loaded}
206 \fi\fi\fi\fi
207 }

```

`skbinput: opt level` The option `level` is used to define the document level to be used for the next occurrence of `\skbheading`. Supported are all document levels known to \LaTeX and no check is done whether the currently used document class supports them or not (for instance, the article class does not support the document level chapter, however, memoir supports it even in article mode). The supported parameters for this option are: `book` (memoir package), `part` (memoir package), `title` (base \LaTeX classes), `chapter` (\LaTeX book class), `section` (base \LaTeX classes), `subsection` (base \LaTeX classes) and `subsubsection` (base \LaTeX classes).

The option is optional, but when used must give one of the above described values. The package will throw an error otherwise.

We start by defining the macros we use later for testing the option. This might be a slightly awkward way to do it, I am still looking into optimising this code. Anyway, we define everything we need for book, part, title, chapter, section, subsection and subsubsection.

```

208 \def\skb@inputLevelBook{book}
209 \def\skb@inputLevelPart{part}
210 \def\skb@inputLevelTitle{title}
211 \def\skb@inputLevelChapter{chapter}
212 \def\skb@inputLevelSection{section}
213 \def\skb@inputLevelSubSection{subsection}
214 \def\skb@inputLevelSubSubSection{subsubsection}

```

Now we define a macro that will be used to point to the selected input level (`\skb@inputLevel`) and a macro that will be used to set the default input level to be empty (i.e. do nothing, `\skb@SetInputLevel`).

```

215 \def\skb@inputLevel{}
216 \newcommand\skb@SetInputLevel{\gdef\skb@inputLevel{}}

```

And here is the actual definition of the option `level`. For each supported parameter (introduced and defined above) we test if it was provided calling the option (put into `\intarg` on start) and if so we point `\skb@inputLevel` to the \LaTeX macro realising that document level. For instance, if the requested level is subsection we point `\skb@inputLevel` to the \LaTeX macro `\subsection`. That means we

can later simply call `\skb@inputLevel` to instruct L^AT_EX to realise the requested document level. In case the parameter is not supported, the option will throw an error along with a help message.

```

217 \define@key{skbinput}{level}[]{}%
218 \def\intarg{#1}
219 \ifx\skb@inputLevelBook\intarg
220   \let\skb@inputLevel=\book
221 \else\ifx\skb@inputLevelPart\intarg
222   \let\skb@inputLevel=\part
223 \else\ifx\skb@inputLevelTitle\intarg
224   \let\skb@inputLevel=\title
225 \else\ifx\skb@inputLevelChapter\intarg
226   \let\skb@inputLevel=\chapter
227 \else\ifx\skb@inputLevelSection\intarg
228   \let\skb@inputLevel=\section
229 \else\ifx\skb@inputLevelSubSection\intarg
230   \let\skb@inputLevel=\subsection
231 \else\ifx\skb@inputLevelSubSubSection\intarg
232   \let\skb@inputLevel=\subsubsection
233 \else
234   \PackageError{skb}%
235     {Value for option \@tempa\space not supported: \intarg}%
236     {I do not know the value \intarg\space for the option \@tempa.%
237     \MessageBreak Please use only: book, part, title, chapter,%
238     \MessageBreak section, subsection or subsubsection.%
239     \MessageBreak <return> to continue, no level will be set and heading is ignored}
240 \fi\fi\fi\fi\fi\fi\fi
241 }

```

8.2.2 The Macro

`\skbinput` This macro will load a `.tex` file from the root directory or from an **SKB** known directory (if option `from` is applied). It will also configure the document level macro for the next use of `\skbheading`, if the option `level` is applied. If `level` is not used, then `\skbheading` will have no effect. The macro first sets the input level to be empty (`\skb@input@set`) and the input macro to the default value (`\skb@input@set`). The it processes the options (using the `keyval` package) and finally calls `\skb@input@call` to realise the load of the requested file.

```

242 \newcommand\skbinput[2]{}%
243 \skb@input@set
244 \skb@SetInputLevel
245 \setkeys{skbinput}{#1}
246 \skb@input@call{#2}
247 }

```

8.3 Loading Figures: `skbfigure`

8.3.1 Macro Options

This macro supports a number of options. To be able to test for the applied options, we first define a few macros that will be used by `\skbfigure` to realise the requested figure input. We define one macro per option supported.

```
248 \def\skb@FigureOptWidth{}
249 \def\skb@FigureOptHeight{}
250 \def\skb@FigureOptCenter{}
251 \def\skb@FigureOptFigure{}
252 \def\skb@FigureOptPosition{}
253 \def\skb@FigureOptCaption{}
254 \def\skb@FigureOptLabel{}
255 \def\skb@FigureOptMultiinclude{}
```

To be able to reset all of these macros before processing a figure, we define a reset macro.

```
256 \newcommand{\skb@figureOptReset}{
257   \gdef\skb@FigureOptWidth{}
258   \gdef\skb@FigureOptHeight{}
259   \gdef\skb@FigureOptCenter{}
260   \gdef\skb@FigureOptFigure{}
261   \gdef\skb@FigureOptPosition{}
262   \gdef\skb@FigureOptCaption{}
263   \gdef\skb@FigureOptLabel{}
264   \gdef\skb@FigureOptMultiinclude{}
265 }
```

Now we define all options for `\skbfigure`. All options work the same way: they either take the parameter given and put it into the corresponding macro we defined above or simply set the corresponding macro to true. This way we can test these corresponding macros for being empty (default) or not and then decide how to process the figure input.

`skbfigure opt width` The first one is called `width` used for the width of `\resizebox` and `\includegraphics`.

```
266 \define@key{skbfigures}{width}[]{}
267 \gdef\skb@FigureOptWidth{#1}
268 }
```

`skbfigure opt height` The option `height` is used for the height of `\resizebox` and `\includegraphics`.

```
269 \define@key{skbfigures}{height}[]{}
270 \gdef\skb@FigureOptHeight{#1}
271 }
```

`skbfigure opt center` The option `center` is used to trigger the center environment (so it only needs to set true).


```

272 \define@key{skbfigures}{center}[true]{%
273   \gdef\skb@FigureOptCenter{true}
274 }

```

`skbfigure opt figure` The option `figure` is used to trigger the figure environment (so it only needs to set `true`).

```

275 \define@key{skbfigures}{figure}[true]{%
276   \gdef\skb@FigureOptFigure{true}
277 }

```

`skbfigure opt position` The option `position` is used to fix the position when figure environment is used

```

278 \define@key{skbfigures}{position}[]{%
279   \gdef\skb@FigureOptPosition{\begin{figure}[\#1]}
280 }

```

`skbfigure opt caption` The option `caption` is used to define the caption of the figure used as `\caption`

```

281 \define@key{skbfigures}{caption}[]{%
282   \gdef\skb@FigureOptCaption{\caption{\#1}}
283 }

```

`skbfigure opt label` The option `label` is used to define the label of the figure used as `\label`

```

284 \define@key{skbfigures}{label}[]{%
285   \gdef\skb@FigureOptLabel{\label{fig:\#1}}
286 }

```

`skbfigure opt multiinclude` The option `multiinclude` is a special option to use `\multiinclude`, automatically deactivates all other options

```

287 \define@key{skbfigures}{multiinclude}[]{%
288   \gdef\skb@FigureOptMultiinclude{\#1}
289 }

```

8.3.2 The Macro

`\skbfigure` `\skbfigure` itself expects options (processed using `keyval`) and the actual file to be included. The file name should start at the figure root directory.

```

290 \newcommand{\skbfigure}[2][]{

```

First, we call our reset function and then use `keyval` to process the options.

```

291   \skb@figureOptReset
292   \setkeys{skbfigures}{\#1}%
293 }

```

Now we process the options `figure` and `position` to decide if and how to use the figure environment. If the `figure` option has been used, we test if the `position` option has been used as well. If `figure` and `position` have been used, we call

`\skb@FigureOptPosition`, which expands to `\beginfigure[option]`. If only the figure option was used, we directly invoke `\beginfigure`.

```

294 \ifx\skb@FigureOptFigure\empty\else
295   \ifx\skb@FigureOptPosition\empty
296     \begin{figure}
297   \else
298     \skb@FigureOptPosition
299   \fi
300 \fi

```

Next is the center option. If it was used, we call `\begincenter`.

```

301 \ifx\skb@FigureOptCenter\empty\else\begin{center}\fi
302

```

The core of the macro. If the option `multiinclude` was not used, we proceed load the figure as we would usually do with L^AT_EX. If `multiinclude` was used, then we simply call `\multiinclude` with the given overlay information, starting at number 0, using PDF format and scaling everything to `\textwidth`.

```

303 \ifx\skb@FigureOptMultiinclude\empty
304   \ifx\skb@FigureOptWidth\empty
305     \ifx\skb@FigureOptHeight\empty
306       \resizebox{!}{!}%
307       {\includegraphics[]%
308        {\skbfilefig{#2}}}
309     \else
310       \resizebox{!}{\skb@FigureOptHeight}%
311       {\includegraphics[height=\skb@FigureOptHeight]%
312        {\skbfilefig{#2}}}
313     \fi
314   \else
315     \ifx\skb@FigureOptHeight\empty
316       \resizebox{\skb@FigureOptWidth}{!}%
317       {\includegraphics[width=\skb@FigureOptWidth]%
318        {\skbfilefig{#2}}}
319     \else
320       \resizebox{\skb@FigureOptWidth}%
321       {\skb@FigureOptHeight}%
322       {\includegraphics[%
323        width=\skb@FigureOptWidth,%
324        height=\skb@FigureOptHeight]%
325        {\skbfilefig{#2}}}
326     \fi
327   \fi
328 \else
329   \resizebox{\textwidth}{!}%
330   {\multiinclude[<\skb@FigureOptMultiinclude>]%
331    [start=0,format=pdf,graphics={width=\textwidth}]%
332    {\skbfilefig{#2}}}

```

```

333 \fi
334

```

If we did use the figure environment, then we check for given caption and label.

```

335 \ifx\skb@FigureOptFigure\empty\else%
336   \skb@FigureOptCaption
337   \skb@FigureOptLabel
338 \fi%
339

```

And finally we close the figure and center environments if we did open them earlier.

```

340 \ifx\skb@FigureOptCenter\empty\else\end{center}\fi
341 \ifx\skb@FigureOptFigure\empty\else\end{figure}\fi
342 }

```

8.4 Loading Slides: skbslide

This macro allows to load a (configurable) combination of PDF slide and L^AT_EX annotation to be loaded in a single call.

8.4.1 Some Extentions

`\skb@slides@callpath` The first is a macro that will maintain the current path and file for loading slides.

```

343 \def\skb@slides@callpath{}

```

`\skb@slides@doslinote` The second is a macro to load annotations from the slide folder.

```

344 \newcommand{\skb@slides@doslinote}[1]{%
345   \def\intarg{#1}
346   \skb@input@doife{skbfilesli}{\intarg}{.tex}{in the slides folder}
347 }

```

8.4.2 Macro Options

`\skbslideopt slidefrom` The option `slidefrom` is used to point to one of the following **SKB** directories: `sli` (the folder for slides) or `pub` (the folder for published documents) or `rep` (the repository directory). The option is optional, but when used must give one of the above described values. The **SKB** will throw an error otherwise.

```

348 \define@key{skbslide}{slidefrom}[]{}%
349 \def\intarg{#1}
350 \ifx\skb@input@var@sli\intarg
351   \let\skb@slides@callpath=\skbfilesli
352 \else\ifx\skb@input@var@pub\intarg
353   \let\skb@slides@callpath=\skbfilepub
354 \else\ifx\skb@input@var@rep\intarg

```

```

355 \let\skb@slides@callpath=\skbfilerep
356 \else
357 \PackageError{skb}%
358 {Value for option \@tempa\space not supported: \intarg}%
359 {I do not know the value \intarg\space for the option \@tempa.%
360 \MessageBreak Please use either "pub", "rep" or "sli".%
361 \MessageBreak <return> to continue, no file will be loaded}
362 \fi\fi\fi
363 }

```

`\skbslideopt notefrom` The option `notefrom` is used to point to one of the following **SKB** directories: `sli` (the folder for slides) or `pub` (the folder for published documents) or `rep` (the repository directory). The option is optional, but when used must give one of the above described values. The **SKB** will throw an error otherwise.

```

364 \define@key{skb@slide}{notefrom}[]{}%
365 \def\intarg{#1}
366 \ifx\skb@input@var@sli\intarg
367 \gdef\skb@input@call{\skb@slides@doslinote}
368 \else\ifx\skb@input@var@pub\intarg
369 \gdef\skb@input@call{\skb@input@dopub}
370 \else\ifx\skb@input@var@rep\intarg
371 \gdef\skb@input@call{\skb@input@dorep}
372 \else
373 \PackageError{skb}%
374 {Value for option \@tempa\space not supported: \intarg}%
375 {I do not know the value \intarg\space for the option \@tempa.%
376 \MessageBreak Please use either "pub", "rep" or "sli".%
377 \MessageBreak <return> to continue, no file will be loaded}
378 \fi\fi\fi
379 }

```

`\skbslideopt annotate` The option `annotate` requests to load annotations for the slide. If not given, no annotations will be loaded.

```

380 \def\skb@slides@loadnote{}
381 \define@key{skb@slide}{annotate}[true]{}%
382 \gdef\skb@slides@loadnote{true}
383 }

```

8.4.3 The Macro

`\skbslide` This macro will load the slide and annotation, depending on the options provided.

```

384 \newcommand\skbslide[3]{}%
385 \gdef\skb@slides@loadnote{}
386 \gdef\skb@input@call{\skb@slides@doslinote}
387 \let\skb@slides@callpath=\skbfilesli
388 \setkeys{skb@slide}{#1}
389

```

```

390 \def\sl{#2}
391 \def\an{#3}
392
393 \ifx\sl\empty\else
394   \begin{figure}[!bh]
395     \resizebox{\textwidth}{!}{\includegraphics[width=\textwidth]{\skb@slides@callpath{#2}}}
396   \end{figure}
397 \fi
398
399 \ifx\skb@slides@loadnote\empty\else
400   \ifx\an\empty
401     \skb@input@call{#2}
402     \clearpage
403   \else
404     \skb@input@call{#3}
405     \clearpage
406   \fi
407 \fi
408 }

```

`\skbslide` This simple macro can help to provide standardised citations on annotation pages.

```

409 \newcommand{\skbslidecite}[2]{\small Source \textit{#2}: \textit{#1} \normalsize}

```

9 Implementation: Filenames, Acronyms and References

9.1 Path and File Names

These macros are used within the **SKB** to generate path and filenames for all known directories and files. They basically provide user-level access to kernel-level processed configuration data. All path names, except root, are fully qualified from root. All filenames are fully qualified from root. Macros that expect an argument use that very argument as the requested filename to provide path and filename.

`\skbpathroot` This macro returns the currently set root path.

```

410 \newcommand{\skbpathroot}{\skb@getCfgVars{root}}

```

`\skbfileroot` This macro takes the given argument and prefixes the root path to it.

```

411 \newcommand{\skbfileroot}[1]{\skb@getCfgVars{root}/#1}

```

`\skbfileacr` This macro returns the file of the acronym database.

```

412 \newcommand{\skbfileacr}{\skb@getCfgVars{root}/\skb@getCfgVars{acr}/\skb@getCfgVars{acrfile}}

```

`\skbpathbib` This macro returns the path to the reference library.

```

413 \newcommand{\skbpathbib}{\skb@getCfgVars{root}/\skb@getCfgVars{bib}}
\skbfilebib This macro returns the file that is used to load the reference library.
414 \newcommand{\skbfilebib}{\skb@getCfgVars{root}/\skb@getCfgVars{bib}/\skb@getCfgVars{bibfile}}
\skbfilerep This macro takes the provided argument and prefixes the path to the repository
to it.
415 \newcommand{\skbfilerep}[1]{\skb@getCfgVars{root}/\skb@getCfgVars{rep}/#1}
\skbfilepub This macro takes the provided argument and prefixes the path to the published
documents to it.
416 \newcommand{\skbfilepub}[1]{\skb@getCfgVars{root}/\skb@getCfgVars{pub}/#1}
\skbfilefig This macro takes the provided argument and prefixes the path to the figures to it.
417 \newcommand{\skbfilefig}[1]{\skb@getCfgVars{root}/\skb@getCfgVars{fig}/#1}
\skbfilesli This macro takes the provided argument and prefixes the path to the slides to it.
418 \newcommand{\skbfilesli}[1]{\skb@getCfgVars{root}/\skb@getCfgVars{sli}/#1}

```

9.2 Loading Acronyms

`\skbacronyms` This macro will load the acronym database. It should be used at the place in your document where you want the list of acronyms to appear. If the file is not found, an error is thrown.

```

419 \newcommand{\skbacronyms}{%
420 \skb@input@doife{\skbfileacr}{-}{-}{for acronym database}
421 }

```

9.3 Loading Reference Database

`\skbbibtex` This macro will load the reference database. It should be used before you start the actual document. If the file is not found, an error is thrown.

```

422 \newcommand{\skbbibtex}{%
423 \skb@input@doife{\skbfilebib}{-}{-}{for bibtex database}
424 }

```

10 Implementation: Other useful Macros

10.1 Emphasising Text: `skbem`

10.1.1 Macro Options

`skbem opt italic` This option will typeset the given text for `\skbem` using italic font.

```

425 \def\skb@emCmd{}
426 \define@key{skbem}{italic}[true]{%
427   \gdef\skb@emCmd{\textit}%
428 }%

```

`skbem opt bold` This option will typeset the given text for `\skbem` using bold font.

```

429 \define@key{skbem}{bold}[true]{%
430   \gdef\skb@emCmd{\textbf}%
431 }%

```

`skbem opt code` This option will typeset the given text for `\skbem` using the command `\skbcode` (see below).

```

432 \define@key{skbem}{code}[true]{%
433   \gdef\skb@emCmd{\skbcode}%
434 }%

```

10.1.2 The Macro

`\skbem` This macro helps to emphasise text in an explicit way (as compared to use font commands within the actual text). Simply call with the one of the option to emphasise text.

```

435 \newcommand{\skbem}[2] [] {%
436   \gdef\skb@emCmd{}%
437   \setkeys{skbem}{#1}%
438   \skb@emCmd{#2}%
439 }%

```

10.2 Emphasising Text: `skbcode`

`\skbcode` This macro is a facade for calling `\lstinline` with `basicstyle` set to type writer font. It is used by `skbem` with the option `code` to call `\lstinline` but can also be called directly.

```

440 \newcommand{\skbcode}[1] {%
441   \lstinline[basicstyle=\ttfamily]{#1}%
442 }%

```

10.3 List Environments: `skbnotelist` and `skbnoteenum`

These environments simulate `\tightlist` from the memoir package. They work identical: call the environment `itemize` (for `skbnotelist`) or `enumerate` (for `skbnoteenum`) and set the two values to 0 (thus minimising the margin between items).

`\skbnotelist` New Environment `skbnotelist` to minimise the margin between list items.

```

443 \newenvironment{skbnotelist}

```

```

444 {\begin{itemize}
445   \ifSkbMemoirLoaded\else
446     \setlength{\parskip}{0cm}\setlength{\itemsep}{0cm}
447   \fi
448 }
449 {\end{itemize}}

```

`\skbnoteenum` New Environment skbnotelist to minimise the margin between list items.

```

450 \newenvironment{skbnoteenum}%
451 {\begin{enumerate}
452   \ifSkbMemoirLoaded\else
453     \setlength{\parskip}{0cm}\setlength{\itemsep}{0cm}
454   \fi
455 }
456 {\end{enumerate}}

```

10.4 Acronyms in Footnotes: `skbacft`

`\skbacft` This macro provides some functionality that the `acronym` package does not offer: introducing acronyms in a footnote (if they are used the first time) or simply use the short form. I found this is useful when writing books, where sometimes introducing acronym in the normal text flow somehow disturbs that very flow.

```

457 \newcommand{\skbacft}[1]{%
458   \ifAC@dua
459     \ifAC@starred\acl*{#1}\else\acl{#1}\fi%
460   \else
461     \expandafter\ifx\csname ac@#1\endcsname\AC@used%
462       \acs{#1}%
463     \else
464       \acs{#1}\footnote{\acf{#1}}%
465     \fi
466   \fi}

```

10.5 PDF Meta Information: `skbpdfinfo` and more

`\skbtitle` This macro allows to set text for the title of the generated PDF.

```

467 \def\skb@TitleText{}
468 \newcommand{\skbtitle}[1]{\gdef\skb@TitleText{#1}}

```

`\skbauthor` This macro allows to set text for the author of the generated PDF.

```

469 \def\skb@AuthorText{}
470 \newcommand{\skbauthor}[1]{\gdef\skb@AuthorText{#1}}

```

`\skbsubject` This macro allows to set text for the subject of the generated PDF.

```

471 \def\skb@SubjectText{}

```



```

472 \newcommand{\skbsubject}[1]{\gdef\skb@SubjectText{#1}}
\skbkeywords This macro allows to set text for the keywords of the generated PDF.
473 \def\skb@KeywordsText{}
474 \newcommand{\skbkeywords}[1]{\gdef\skb@KeywordsText{#1}}
\skbpdfinfo This macro will set the PDF information in the generated PDF. It first checks
if we are in PDF mode, and then uses the information from \skb@AuthorText,
\skb@TitleText plus subject and keywords from above.
475 \newcommand{\skbpdfinfo}{%
476   \ifpdf
477     \pdfinfo{
478       /Author (\skb@AuthorText)
479       /Title (\skb@TitleText)
480       /ModDate (D:\pdfdate)
481       /Subject (\skb@SubjectText)
482       /Keywords (\skb@KeywordsText)
483     }
484   \fi
485 }

```

10.6 Listing Styles and Support

The **SKB** comes with a few pre-defined styles for the `listing` package. Most of these predefined styles use type writer font in scriptsize, arrange a grey box around the listing and set the keywords to Blue4.

The first style is the for any generic listing without specifying a language and no line numbers.

```

486 \lstdefinestyle{generic}
487   {basicstyle=\scriptsize\ttfamily, backgroundcolor=\color[gray]{.9},
488     frame=single, framerule=.5pt, numbers=none,
489     linewidth=0.99\textwidth, xleftmargin=3pt,
490     keywordstyle=\bfseries\color{Blue4},
491     identifierstyle=\bfseries}

```

This style is designed for listings within tables. It is similar to the generic one above, except that the definitions for frame and numbers are not used, which seem to collide with some table environments.

```

492 \lstdefinestyle{gentab}
493   {basicstyle=\scriptsize\ttfamily, backgroundcolor=\color[gray]{.9},
494     framerule=0pt,
495     linewidth=.86\textwidth, xleftmargin=3pt,
496     keywordstyle=\bfseries\color{Blue4},
497     identifierstyle=\bfseries}

```

This style is the same as the generic one above, except that it switches on line numbers and allows extra space for them within the grey box.

```
498 \lstdefinestyle{genericLN}
499     {basicstyle=\scriptsize\ttfamily, backgroundcolor=\color[gray]{.9},
500     frame=single, framerule=.5pt, numbers=left,
501     linewidth=0.99\textwidth, xleftmargin=20pt,
502     keywordstyle=\bfseries\color{Blue4},
503     identifierstyle=\bfseries}
```

This style is based on the style `genricLN`, basically using a slightly brighter grey for the box.

```
504 \lstdefinestyle{genericLNspecial}
505     {basicstyle=\small\ttfamily, backgroundcolor=\color[gray]{.97},
506     frame=single, framerule=.5pt, numbers=left,
507     linewidth=0.99\textwidth, xleftmargin=20pt,
508     keywordstyle=\bfseries\color{Blue4},
509     identifierstyle=\bfseries}
```

This style is designed for examples within slides (frames) using the `beamer` package.

```
510 \lstdefinestyle{beamer-example}
511     {basicstyle=\scriptsize\ttfamily,
512     frame=single, framerule=0pt, numbers=none,
513     linewidth=0.99\textwidth, xleftmargin=3pt,
514     keywordstyle=\bfseries\color{Blue4},
515     identifierstyle=\bfseries}
```

This style is designed for examples within slides (frames) using the `beamer` with added line numbers.

```
516 \lstdefinestyle{beamer-exampleLN}
517     {basicstyle=\scriptsize\ttfamily,
518     frame=single, framerule=0pt, numbers=left,
519     linewidth=0.99\textwidth, xleftmargin=20pt,
520     keywordstyle=\bfseries\color{Blue4},
521     identifierstyle=\bfseries}
```

This style uses the definitions from the generic style above and set the language to Java.

```
522 \lstdefinestyle{javaCode}
523     {basicstyle=\scriptsize\ttfamily, backgroundcolor=\color[gray]{.9},
524     frame=single, framerule=0pt, language=JAVA,
525     numbers=none,
526     keywordstyle=\bfseries\color{Blue4},
527     identifierstyle=,
528     linewidth=0.99\columnwidth}
```

This style can be used to set ‘normal’ style after changing it.

```
529 \lstdefinestyle{inText}
```

530 {basicstyle=\ttfamily}

11 Experimental Macros

This part of the **SKB** is experimental. Please do not use it for production code or important documents. The macros in this section will be moved as soon as they are stable, or simply removed. They can, as long as they stay in this section, be changed at any time in future releases.

11.1 Defining new relative Headings: `skbheadingdc`

When we set the document level with `\skbheading`, it might be useful to actually have a macro that allows to relatively change headings. This is useful if we have more than one heading in a repository file, where the first one defines the heading and will get an associative document level from the calling document while any subsequent heading might need to go one level up or down. The macro here works as long as we don't need to recursively store document levels. So it is not stable right now and makes only sense if used for single headings.

First, a macro that we use to point to the new heading (rather than the one used by `\skbinput`).

```
531 \def\skb@newHeading{}
```

11.1.1 Macro Options

Now the option `down`, which indicates that this heading should be one level down from the previous one.

```
532 \define@key{skbheadings}{down}[true]{%
533   \ifx\skb@inputLevel\part
534     \let\skb@newHeading=\chapter
535     \let\skb@inputLevel=\chapter
536   \else\ifx\skb@inputLevel\chapter
537     \let\skb@newHeading=\section
538     \let\skb@inputLevel=\section
539   \else\ifx\skb@inputLevel\section
540     \let\skb@newHeading=\subsection
541     \let\skb@inputLevel=\subsection
542   \else\ifx\skb@inputLevel\subsection
543     \let\skb@newHeading=\subsubsection
544     \let\skb@inputLevel=\subsubsection
545   \else
546     \KV@err{Invalid current level for SkbNewHeading(down),
547           please use: part, chapter, section or subsection}
548   \fi\fi\fi\fi
```

```
549 }
```

Now the option `up`, which indicates that this heading should be one level up from the previous one.

```
550 \define@key{skbheadings}{up}[true]{%
551   \ifx\skb@inputLevel\chapter
552     \let\skb@newHeading=\part
553     \let\skb@inputLevel=\part
554   \else\ifx\skb@inputLevel\section
555     \let\skb@newHeading=\chapter
556     \let\skb@inputLevel=\chapter
557   \else\ifx\skb@inputLevel\subsection
558     \let\skb@newHeading=\section
559     \let\skb@inputLevel=\section
560   \else\ifx\skb@inputLevel\subsubsection
561     \let\skb@newHeading=\subsection
562     \let\skb@inputLevel=\subsection
563   \else
564     \KV@err{Invalid current level for SkbNewHeading(up),
565           please use: chapter, section, subsection or subsubsection}
566   \fi\fi\fi\fi
567 }
```

Now the option `last`, which indicates that this heading should be on the same level as the previous one.

```
568 \define@key{skbheadings}{last}[true]{%
569   \let\skb@newHeading=\skb@inputLevel%
570 }
```

11.1.2 The Macro

`\skbheadingudc`

```
571 \newcommand{\skbheadingudc}[2][ ]{%
572   \gdef\skb@newHeading{#1}%
573   \setkeys{skbheadings}{#1}%
574   \ifx\empty\skb@newHeading\else%
575     \skb@newHeading{#2}%
576   \fi
577 }
578 </skbpackage>
```

12 The Configuration File `skb.cfg`

This file is used to overwrite the default values for the **SKB** configuration options. It calls the macro `\skbconfig` using all possible options of that very macro and

providing usefull text as origin of the configuration change `skb.cfg`. Use this as template for the local configuration file `skblocal.cfg` if you need one.

```
579 <*skbcfg>
580 \skbconfig[root=/doc,
581         acr=database/latex,
582         acrfile=acronyms,
583         bib=database/bibtex,
584         bibfile=bibliography.tex,
585         rep=repository,
586         pub=publish,
587         fig=figures,
588         sli=slides
589     ]{skb.cfg}
590 </skbcfg>
```

13 The **SKB** Classes

13.1 The Class `skbarticle`

This class is an example on how to use the **SKB** with memoir. I use `skbarticle` for my articles. Using this class as a template, one can easily write other classes or change/overwrite the settings done here.

First, we announce the package and the font definition file.

```
591 <*skbarticle>
592 \NeedsTeXFormat{LaTeX2e}
593 \ProvidesClass{skbarticle}[2011/06/03 The SKB Article class v0.52]
```

Now we load the memoir class with the following options:

- `10pt` - for 10 point font size
- `a4paper` - I am European, so A4 paper makes sense here
- `extrafontsizes` - tbd
- `twoside` - I want my articles to be set with different even/odd pages
- `onecolumn` - I don't necessarily like 2-columns for my articles
- `openright` - tbd
- `article` - use memoir as if it is an article

```
594 \LoadClass[10pt,a4paper,extrafontsizes,twoside,onecolumn,openright,article]{memoir}
```

Load the **SKB**.

```
595 \RequirePackage{skb}
```

13.1.1 Loaded Packages

I prefer BibLaTeX over plain BibTeX, and other parts of the SKB (such as the LAMP²⁵ server) produce BibLaTeX. The options are:

- style - is set to alphanumeric, much better to find/remember references. If writing for IEEE or LNCS, numeric would be the preferred option.
- sorting - is set to none, not needed here.
- hyperref - I want to have hyperref with my citations

```
596 \RequirePackage[style=alphanumeric,sorting=none,hyperref]{biblatex}
```

Load the acronym package and print only the acronyms actually used in the document. This might move into the SKB package later.

```
597 \RequirePackage[printonlyused]{acronym}
```

Load a view packages that I tend to use quite often:

- etoolbox - etoolbox
- comment - Add comments to your L^AT_EX files
- graphicx - Enhanced graphic support, with key/value interface for include graphics
- longtable - Helps with tables that span multiple pages
- colortbl - Allows coloured cells in tables

```
598 \RequirePackage{etoolbox,comment,graphicx,longtable,colortbl}
```

And some more packages needed quite often:

- textcomp - Special characters, such as ® and ©
- gensymb - Generic characters (math and text mode), such as °, °C, ‰, µ and Ω
- wasysym - Adds characters from *wasy* font, such as ☺, ☒ and ☹
- units - Typeset units correctly (and produce 'nice' fractions), such as 10 m/s and 1/2
- float - Improves interface for floating environments (such as figures, tables)
- xmpmulti - tbd

```
599 \RequirePackage{textcomp,gensymb,wasysym,units,xmpmulti,float}
```

The xcolor package provides driver independent access to all sorts of colour tins, shades, tones and mixes. I like x11names, as you can tell.

```
600 \RequirePackage[x11names]{xcolor}
```

²⁵Linux, Apache, MySQL and PHP (LAMP)

The hyperref package provides layout for hyper references, such as URLs and references within a document, such as acronyms, citations and the table of contents. We use the option colorlinks and then provide the colors we prefer for links (linkcolor), citations (citecolor) and URLs (urlcolor).

```
601 \RequirePackage[colorlinks,%  
602             linkcolor=Brown4,%  
603             citecolor=SeaGreen4,%  
604             urlcolor=RoyalBlue3%  
605             ]{hyperref}  
606 %\RequirePackage[colorlinks,linkcolor=blue]{hyperref}
```

13.1.2 Memoir Options

Not sure, but I don't think semi-iso-pages are good. So not used right now.

```
607 %\semiisopage
```

Change the margins for even and odd pages. Odd to 1cm and even to 1cm.

```
608 \setlength{\oddsidemargin}{1cm}  
609 \setlength{\evensidemargin}{0cm}
```

Set width and height for the text. At the moment only the width, to 15cm

```
610 \setlength{\textwidth}{15cm}  
611 %\setlength{\textheight}{24cm}
```

Don't use chapter numbers in sections, thus making them looking like sections in a classic article (1 instead of the default 0.1)

```
612 \def\thesection{\arabic{section}}
```

Allow table of contents to go up to sub-sections

```
613 \settocdepth{subsection}
```

And numbering up to subsubsections

```
614 \setsecnumdepth{subsubsection}
```

For lists, memoir provides different layouts. We use tightlists here, but can switch that to firmlists if needed

```
615 \tightlists  
616 %\firmlists
```

What are these for? I forgot...

```
617 \midsloppy  
618 \raggedbottom
```

13.1.3 Misc Settings

Finally, we do set the sort option for the bibliography to anyt (biblatex)

```
619 \ExecuteBibliographyOptions{sorting=anyt}
```

There is no code for `\AtBeginDocument` and `\AtEndDocument`, so we are done now.

```
620 \</skbarticle>
```

13.2 The Class `skbbook`

This class is an example on how to use the `SKB` with memoir. I use `skbbook` for my books. Using this class as a template, one can easily write other classes or change/overwrite the settings done here.

First, we announce the package and the font definition file.

```
621 \<*skbbook>
622 \NeedsTeXFormat{LaTeX2e}
623 \ProvidesClass{skbbook}[2011/06/03 The SKB Book class v0.52]
```

Now we load the memoir class with the following options:

- 11pt - for 11 point font size
- a4paper - I am European, so A4 paper makes sense here
- extrafontsizes - tbd
- twoside - I want my articles to be set with different even/odd pages
- onecolumn - I don't necessarily like 2-columns for my articles
- openright - tbd

```
624 \LoadClass[11pt,a4paper,extrafontsizes,twoside,onecolumn,openright]{memoir}
```

Load the `SKB`.

```
625 \RequirePackage{skb}
```

13.2.1 Loaded Packages

I prefer BibLaTeX over plain BibTeX, and other parts of the `SKB` (such as the `LAMP` server) produce BibLaTeX. The options are:

- style - is set to alphanumeric, much better to find/remember references. If writing for IEEE or LNCS, numeric would be the preferred option.
- sorting - is set to none, not needed here.
- hyperref - I want to have hyperref with my citations

```
626 \RequirePackage[style=alphanumeric,sorting=none,hyperref]{biblatex}
```


Load the acronym package and print only the acronyms actually used in the document. This might move into the **SKB** package later

```
627 \RequirePackage[printonlyused]{acronym}
```

Load a view packages that I tend to use quite often:

- etoolbox - etoolbox
- comment - Add comments to your L^AT_EX files
- graphicx - Enhanced graphic support, with key/value interface for include graphics
- longtable - Helps with tables that span multiple pages
- colortbl - Allows coloured cells in tables

```
628 \RequirePackage{etoolbox,comment,graphicx,longtable,colortbl}
```

And some more packages needed quite often:

- textcomp - Special characters, such as \mathbb{R} and \mathbb{C}
- gensymb - Generic characters (math and text mode), such as $^\circ$, $^\circ\text{C}$, $\%$, μ and Ω
- wasysym - Adds characters from *wasy* font, such as \odot , \boxtimes and \circlearrowleft
- units - Typeset units correctly (and produce 'nice' fractions), such as $10^m/\text{s}$ and $1/2$
- float - Improves interface for floating environments (such as figures, tables)
- xmpmulti - tbd

```
629 \RequirePackage{textcomp,gensymb,wasysym,units,xmpmulti,float}
```

The xcolor package provides driver independent access to all sorts of colour tins, shades, tones and mixes. I like x11names, as you can tell.

```
630 \RequirePackage[x11names]{xcolor}
```

The hyperref package provides layout for hyper references, such as URLs and references within a document, such as acronyms, citations and the table of contents. We use the option colorlinks and then provide the colors we prefer for links (linkcolor), citations (citecolor) and URLs (urlcolor).

```
631 \RequirePackage[colorlinks,%  
632             linkcolor=Brown4,%  
633             citecolor=SeaGreen4,%  
634             urlcolor=RoyalBlue3%  
635             ]{hyperref}  
636 %\RequirePackage[colorlinks,linkcolor=blue]{hyperref}
```

13.2.2 Memoir Options

Not sure, but I don't think semi-iso-pages are good. So not used right now.

```
637 %\semiisopage
```

Set the head styles to komalike (the other nice style is memman).

```
638 \headstyles{komalike}
```

Change the margins for even and odd pages. Odd to .5cm and even to 0cm.

```
639 \setlength{\oddsidemargin}{.5cm}
```

```
640 \setlength{\evensidemargin}{0cm}
```

Set width and height for the text. Width to 15cm and length to 22cm.

```
641 \setlength{\textwidth}{15cm}
```

```
642 \setlength{\textheight}{22cm}
```

Get half a centimeter back from the topmargin.

```
643 \setlength{\topmargin}{-.5cm}
```

Allow table of contents to go up to subsub-sections

```
644 \settocdepth{subsubsection}
```

And numbering up to subsubsections

```
645 \setsecnumdepth{subsubsection}
```

For lists, memoir provides different layouts. We use tightlists here, but can switch that to firmlists if needed

```
646 \tightlists
```

```
647 %\firmlists
```

What are these for? I forgot...

```
648 \midsloppy
```

```
649 \raggedbottom
```

Chapters should look like the memoir veelo style.

```
650 \chapterstyle{veelo}
```

13.2.3 Misc Settings

Finally, we do set the sort option for the bibliography to anyt (biblatex)

```
651 \ExecuteBibliographyOptions{sorting=anyt}
```

There is no code for `\AtBeginDocument` and `\AtEndDocument`, so we are done now.

```
652 </skbbook>
```

13.3 The Class `skbbeamer`

This class is an example on how to use the `SKB` with memoir. I use `skbbeamer` for my beamer presentations. Using this class as a template, one can easily write other classes or change/overwrite the settings done here.

First, we announce the package and the font definition file and process the options.

```
653 \*skbbeamer
654 \NeedsTeXFormat{LaTeX2e}
655 \ProvidesClass{skbbeamer}[2011/06/03 The SKB Beamer class v0.52]
656 \DeclareOption{beameranim}{\PassOptionsToPackage{\CurrentOption}{skb}}
657 \DeclareOption{beamernoanim}{\PassOptionsToPackage{\CurrentOption}{skb}}
658 \ProcessOptions\relax
```

Now we load the `xcolor` package and then the beamer class. That should load the `x11names` some of the `SKB` listing styles use while not creating any clash between the packages `beamer` and `xcolor`.

```
659 \RequirePackage[x11names]{xcolor}
660 \LoadClass[x11names]{beamer}
```

Load the `SKB`.

```
661 \RequirePackage{skb}
```

13.3.1 Loaded Packages

I prefer BibLaTeX over plain BibTeX, and other parts of the `SKB` (such as the `LAMP` server) produce BibLaTeX. The options are:

- `style` - is set to `alphanumeric`, much better to find/remember references. If writing for IEEE or LNCS, `numeric` would be the preferred option.
- `sorting` - is set to `none`, not needed here.
- `hyperref` - I want to have `hyperref` with my citations

```
662 \RequirePackage[style=alphanumeric,sorting=none,hyperref]{biblatex}
```

Load the `acronym` package and print only the acronyms actually used in the document. This might move into the `SKB` package later

```
663 \RequirePackage[printonlyused]{acronym}
```

Load a view packages that I tend to use quite often:

- `etoolbox` - `etoolbox`
- `comment` - Add comments to your `LATEX` files
- `graphicx` - Enhanced graphic support, with key/value interface for include graphics
- `longtable` - Helps with tables that span multiple pages

- colortbl - Allows coloured cells in tables

```
664 \RequirePackage{etoolbox,comment,graphicx,longtable,colortbl}
```

And some more packages needed quite often:

- textcomp - Special characters, such as ® and ©
- gensymb - Generic characters (math and text mode), such as °, °C, ‰, µ and Ω
- wasysym - Adds characters from *wasy* font, such as ☺, ☒ and ☺
- units - Typeset units correctly (and produce 'nice' fractions), such as 10 m/s and 1/2
- float - Improves interface for floating environments (such as figures, tables)
- xmpmulti - tbd

```
665 \RequirePackage{textcomp,gensymb,wasysym,units,xmpmulti,float}
```

13.3.2 Misc Settings

And some default settings for the dirtree package.

```
666 \renewcommand*{DTstylecomment}{\itshape\sffamily\color{blue}\scriptsize}
667 \setlength{DTbaselineskip}{10pt}
668 \DTsetlength{0.2em}{1em}{0.2em}{0.4pt}{1.6pt}
669 \renewcommand*{DTstyle}{\scriptsize\ttfamily\textcolor{black}}
```

There is no code for `\AtBeginDocument` and `\AtEndDocument`, so we are done now.

```
670 </skbbeamer>
```

13.4 The Class skblncsbeamer

This class is an example on how to use the **SKB** with memoir. I use skblncsbeamer for my beamer based handouts. Using this class as a template, one can easily write other classes or change/overwrite the settings done here.

First, we announce the package and the font definition file.

```
671 <*skblncsbeamer>
672 \NeedsTeXFormat{LaTeX2e}
673 \ProvidesClass{skblncsbeamer}[2011/06/03 The SKB LNCS Beamer class v0.52]
```

Just in case there is no `\titlepage` declared, the beamerarticle wants that.

```
674 \providecommand{\titlepage}{}</pre>

```

Now we load the memoir class with the following options:

- 9pt - for 9 point font size

- a4paper - I am European, so A4 paper makes sense here
- extrafontsizes - tbd
- twoside - I want my articles to be set with different even/odd pages
- onecolumn - I don't necessarily like 2-columns for my articles
- openright - tbd
- article - use memoir as if it is an article
- x11names - this option will be forwarded to the xcolor/graphics packages

```
675 \LoadClass[9pt,a4paper,extrafontsizes,twoside,onecolumn,openright,article,x11names]{memoir}
```

For Beamer handouts, we need the beamerarticle package to load the frame thumbnails.

```
676 \RequirePackage{beamerarticle,pgf}
```

Load the **SKB**.

```
677 \RequirePackage{skb}
```

13.4.1 Loaded Packages

I prefer BibLaTeX over plain BiB_TE_X, and other parts of the **SKB** (such as the **LAMP** server) produce BibLaTeX. The options are:

- style - is set to alphanumeric, much better to find/remember references. If writing for IEEE or LNCS, numeric would be the preferred option.
- sorting - is set to none, not needed here.
- hyperref - I want to have hyperref with my citations

```
678 \RequirePackage[style=alphanumeric,sorting=none,hyperref]{biblatex}
```

Load the acronym package and print only the acronyms actually used in the document. This might move into the **SKB** package later.

```
679 \RequirePackage[printonlyused]{acronym}
```

Load a view packages that I tend to use quite often:

- etoolbox - etoolbox
- comment - Add comments to your L^AT_EX files
- graphicx - Enhanced graphic support, with key/value interface for include graphics
- longtable - Helps with tables that span multiple pages
- colortbl - Allows coloured cells in tables

```
680 \RequirePackage{etoolbox,comment,graphicx,longtable,colortbl}
```

And some more packages needed quite often:

- textcomp - Special characters, such as ® and ©

- gensymb - Generic characters (math and text mode), such as °, °C, %, μ and Ω
- wasysym - Adds characters from *wasy* font, such as ☺, ☒ and ↻
- units - Typeset units correctly (and produce 'nice' fractions), such as 10 m/s and 1/2
- float - Improves interface for floating environments (such as figures, tables)
- xmpmulti - tbd

```
681 \RequirePackage{textcomp,gensymb,wasysym,units,xmpmulti}
```

13.4.2 Memoir Options

Not sure, but I don't think semi-iso-pages are good. So not used right now.

```
682 %\semiisopage
```

We do want to list files.

```
683 \listfiles
```

Change the margins for even and odd pages. Odd to 0cm and even to 1cm.

```
684 \setlength{\oddsidemargin}{0cm}
685 \setlength{\evensidemargin}{0cm}
```

Set width and height for the text. Width to 15cm and height to 24.5cm.

```
686 \setlength{\textwidth}{15cm}
687 \setlength{\textheight}{24.5cm}
```

Get half a centimeter back from the topmargin.

```
688 \setlength{\topmargin}{-1.5cm}
```

Don't use chapter numbers in sections, thus making them looking like sections in a classic article (1 instead of the default 0.1)

```
689 \def\thesection{\arabic{section}}
```

Allow table of contents to go up to sub-sections

```
690 \settocdepth{subsection}
```

And numbering up to subsubsections

```
691 \setsecnumdepth{subsubsection}
```

Set the head styles to komalike (the other nice style is memman).

```
692 \headstyles{komalike}
```

For lists, memoir provides different layouts. We use tightlists here, but can switch that to firmlists if needed

```
693 \tightlists
694 %\firmlists
```

What are these for? I forgot...

```
695 \midsloppy
696 \raggedbottom
```

Set parindent to 0pt and parskip to 0.2pt.

```
697 \parindent0pt
698 \setlength{\parskip}{0.2cm}
```

13.4.3 Misc Settings

Do an index.

```
699 \makeindex
```

Before we start with the actual document, we want the title slide and the table of contents on the first page.

```
700 \AtBeginDocument{
701   \resizebox{\textwidth}{!}{\includeslide{title}}
702   \bigskip
703   \tableofcontents*
704   \bigskip
705   \newpage
706 }
```

There is no code for `\AtEndDocument`, so we are done now.

```
707 </skblncsbeamer>
```

13.5 The Class `skblncsppt`

This class is an example on how to use the **SKB** with memoir. I use `skblncsppt` for handouts (anotated slides) based on Microsoft's PPT. Reason for that is that the **PDF** export and print routines in Microsoft Office 2010 no longer support vector images for the slide thumbnails, which renders handouts almost useless. So I do print the PPT slides into **PDF** (screen resolution, that way one avoids frames around the slides), and then \LaTeX to generate handouts. Using this class as a template, one can easily write other classes or change/overwrite the settings done here.

First, we announce the package and the font definition file.

```
708 <*skblncsppt>
709 \NeedsTeXFormat{LaTeX2e}
710 \ProvidesClass{skblncsppt}[2011/06/03 The SKB LNCS PPT class v0.52]
```

Now we load the memoir class with the following options:

- 9pt - for 9 point font size
- a4paper - I am European, so A4 paper makes sense here
- extrafontsizes - tbd
- twoside - I want my articles to be set with different even/odd pages
- onecolumn - I don't necessarily like 2-columns for my articles
- openright - tbd
- article - use memoir as if it is an article

```
711 \LoadClass[9pt,a4paper,extrafontsizes,twoside,onecolumn,openright,article]{memoir}
```

Load the **SKB**.

```
712 \RequirePackage{skb}
```

13.5.1 Loaded Packages

I prefer BibLaTeX over plain BibTeX, and other parts of the **SKB** (such as the **LAMP** server) produce BibLaTeX. The options are:

- style - is set to alphanumeric, much better to find/remember references. If writing for IEEE or LNCS, numeric would be the preferred option.
- sorting - is set to none, not needed here.
- hyperref - I want to have hyperref with my citations

```
713 \RequirePackage[style=alphanumeric,sorting=none,hyperref]{biblatex}
```

Load the acronym package and print only the acronyms actually used in the document. This might move into the **SKB** package later.

```
714 \RequirePackage[printonlyused]{acronym}
```

Load a view packages that I tend to use quite often:

- etoolbox - etoolbox
- comment - Add comments to your L^AT_EX files
- graphicx - Enhanced graphic support, with key/value interface for include graphics
- longtable - Helps with tables that span multiple pages
- colortbl - Allows coloured cells in tables

```
715 \RequirePackage{etoolbox,comment,graphicx,longtable,colortbl}
```

And some more packages needed quite often:

- textcomp - Special characters, such as ® and ©
- gensymb - Generic characters (math and text mode), such as °, °C, ‰, µ and Ω

- wasysym - Adds characters from *wasy* font, such as ☺, ☒ and ☺
- units - Typeset units correctly (and produce 'nice' fractions), such as 10 m/s and 1/2
- float - Improves interface for floating environments (such as figures, tables)
- xmpmulti - tbd

```
716 \RequirePackage{textcomp, gensymb, wasysym, units, xmpmulti, float}
```

The xcolor package provides driver independent access to all sorts of colour tins, shades, tones and mixes. I like x11names, as you can tell.

```
717 \RequirePackage[x11names]{xcolor}
```

The hyperref package provides layout for hyper references, such as URLs and references within a document, such as acronyms, citations and the table of contents. We use the option colorlinks and then provide the colors we prefer for links (linkcolor), citations (citecolor) and URLs (urlcolor).

```
718 \RequirePackage[colorlinks,%
719                 linkcolor=Brown4,%
720                 citecolor=SeaGreen4,%
721                 urlcolor=RoyalBlue3%
722                 ]{hyperref}
723 %\RequirePackage[colorlinks, linkcolor=blue]{hyperref}
```

13.5.2 Memoir Options

Not sure, but I don't think semi-iso-pages are good. So not used right now.

```
724 %\semiisopage
```

We do want to list files.

```
725 \listfiles
```

Change the margins for even and odd pages. Odd to 0cm and even to 1cm.

```
726 \setlength{\oddsidemargin}{0cm}
727 \setlength{\evensidemargin}{0cm}
```

Set width and height for the text. Width to 15cm and height to 24.5cm.

```
728 \setlength{\textwidth}{15cm}
729 \setlength{\textheight}{24.5cm}
```

Get half a centimeter back from the topmargin.

```
730 \setlength{\topmargin}{-1.5cm}
```

Don't use chapter numbers in sections, thus making them looking like sections in a classic article (1 instead of the default 0.1)

```
731 \def\thesection{\arabic{section}}
```

Allow table of contents to go up to sub-sections

```
732 \settocdepth{subsection}
```

And numbering up to subsubsections

```
733 \setsecnumdepth{subsubsection}
```

Set the head styles to komalike (the other nice style is memman).

```
734 \headstyles{komalike}
```

For lists, memoir provides different layouts. We use tightlists here, but can switch that to firmlists if needed

```
735 \tightlists
```

```
736 %\firmlists
```

What are these for? I forgot...

```
737 \midsloppy
```

```
738 \raggedbottom
```

We want ruled pages and arabic page numbering.

```
739 \pagestyle{ruled}
```

```
740 \pagenumbering{arabic}
```

13.5.3 Misc Settings

Do an index.

```
741 \makeindex
```

There is no code for `\AtBeginDocument` and `\AtEndDocument`, so we are done now.

```
742 </skblncspt>
```

13.6 The Class `skbmoderncv`

This class integrates the `moderncv` package into the **SKB**. I use `moderncv` for my own CV and with the **SKB** I am able to maintain all information while producing different documents (i.e. one for a research proposal, for my website, for journals, etc.). This class provides some macros using `biblatex` to create several lists of publications, i.e. a different list for articles, proceedings, conference papers per year, etc.

First, we announce the package and the font definition file and process the options.

```
743 <*skbmoderncv>
```

```
744 \NeedsTeXFormat{LaTeX2e}
```

```
745 \ProvidesClass{skbmoderncv}[2011/06/03 The SKB Modern CV class v0.52]
```

Now we load the moderncv package. By default we use an 11 point font and A4 paper. Once can change these settings later in a CV document.

```
746 \LoadClass[11pt,a4paper]{moderncv}
```

Load the **SKB**.

```
747 \RequirePackage{skb}
```

13.6.1 Loaded Packages

I prefer BibLaTeX over plain BIB_TE_X, and other parts of the **SKB** (such as the **LAMP** server) produce BibLaTeX. The options are:

- style - is set to alphabetic, much better to find/remember references. If writing for IEEE or LNCS, numeric would be the preferred option.
- sorting - is set to ynt, year first, then author name then title.
- bibstyle - is set to standard, which means there are no labels (we set labels as enumerate list later).
- hyperref - I want to have hyperref with my citations

```
748 \RequirePackage[style=alphabetic,sorting=ynt,bibstyle=standard,hyperref]{biblatex}
```

Load the eurofont package for the Euro sign.

```
749 \RequirePackage{eurofont}
```

Load the enumitem package, which is used later to change indents on the enumerate list for publication references.

```
750 \RequirePackage{enumitem}
```

Load the xcolor and hyperref packages to allow painting URLs in publication references.

```
751 \RequirePackage[x11names]{xcolor}
752 \RequirePackage[colorlinks,%
753                 linkcolor=Brown4,%
754                 citecolor=SeaGreen4,%
755                 urlcolor=RoyalBlue3,%
756                 pdftex
757                 ]{hyperref}
```

13.6.2 Misc Settings

Configure moderncv to use the classic layout.

```
758 \moderncvtheme{classic}
```

Some definitions for list symbols.

```
759 \newcommand{\up}[1]{\ensuremath{\text{\texttrm{\scriptsize#1}}}}
760 \renewcommand{\listitemsymbol}{\textendash}
```

Define a new heading for BibLaTeX doing nothing, we want to set headings manually in the CV.

```
761 \defbibheading{None}{}
```

13.7 Macros

`\skbcvrefplain` This macro prints a list of references without any labels. It expects a list of citation references, and empty list is valid and will result in an empty reference list. The macro defines a new environment for the BibTeX list adding an indent to the list using moderncv counters. `\hintscolumnwidth` is the left column of an entry and `\separatorcolumnwidth` is the middle column providing a spacer between left and right.

With the new environment defined, the macro then opens a new refsegment (BibLaTeX) to group all references without impacting the overall reference list. It uses `\nocite` to mark all references as being used without printing them, and then `\printbibliography` for printing the list. The options here are:

- heading - refers to the new heading we have defined earlier, called None.
- segment - points to the current refsegment, so only references used here will be part of the list.
- maxnames - the maximum names shown per entry, we want to see as many as possible.
- minnames - the minimum names shown per entry, we want to see as many as possible.

```
762 \newcommand{\skbcvrefplain}[1]{%
763   \defbibenvironment{bibliography}
764     {\list}{%
765       \setlength{\parindent}{\hintscolumnwidth}
766       \addtolength{\parindent}{\separatorcolumnwidth}
767       \leftmargin\parindent
768       \setlength{\parindent}{0pt}
769       \itemindent\parindent
770       \itemsep\bibitemsep
771       \parsep\bibparsep
772     }}
773   {\endlist}
774   {\item}
775
776   \begin{refsegment}
777     \nocite{#1}
778     \printbibliography[heading=None,segment=\therefsegment,maxnames=20,minnames=20]
779   \end{refsegment}
780 }
```

`\skbcvrefenum` This macro prints a list of references as an enumerated list, which helps to see how many entries a list has and also makes it easier to read the list in the final CV. It expects a list of citation references, and empty list is valid and will result in an empty reference list. The macro defines a new environment for the BibTeX list using an enumerate environment, which was of course already changed loading the enumitem package. First, we calculate the length for `\parindent` by setting it to `\intscolumnwidth`, addint `\separatorcolumnwidth` and finally adding another 1pt to it. This will make the items of the enumeration list appear within the left column of the CV. We then apply a few options to the enumerate environment:

- `leftmargin` - set to the newly calculate `\parindent`.
- `labelsep` - set to the spacer defined in moderncv `\separatorcolumnwidth`.
- `label` - set to use arabic numbers without any trailing full stop.
- `noitemsep` - no vertical separation between items, we want a small list.
- `topsep` - finally add a top separator for the items of 1pt.

To reset `\parindent`, we set it back to 0pt once the enumerate environment is closed.

With the new environment defined, the macro then opens a new refsegment (BibLaTeX) to group all references without impacting the overall reference list. It uses `\nocite` to mark all references as being used without printing them, and then `\printbibliography` for printing the list. The options here are:

- `heading` - refers to the new heading we have defined earlier, called None.
- `segment` - points to the current refsegment, so only references used here will be part of the list.
- `maxnames` - the maximum names shown per entry, we want to see as many as possible.
- `minnames` - the minimum names shown per entry, we want to see as many as possible.

```

781 \newcommand{\skbcvrefenum}[1]{%
782   \defbibenvironment{bibliography}
783     {\setlength{\parindent}{\hintscolumnwidth}
784      \addtolength{\parindent}{\separatorcolumnwidth}
785      \addtolength{\parindent}{1pt}
786      \begin{enumerate}[leftmargin=\parindent,labelsep=\separatorcolumnwidth,label*=\arabic*,noi
787     ]}
788     {\end{enumerate}%
789     \setlength{\parindent}{0pt}
790     }
791     {\item}
792
793   \begin{refsegment}
794     \nocite{#1}
795     \printbibliography[heading=None,segment=\therefsegment,maxnames=20,minnames=20]
796   \end{refsegment}
797 }
```

There is no code for `\AtBeginDocument` and `\AtEndDocument`, so we are done now.

798 `</skbmoderncv>`

14 History and Change Log

14.1 v0.10 from 06-Jul-2010

- first source forge release of the skb
- at this stage a collection of .sty and .tex files
- documentation in a separate pdf file
- included acronym list

14.2 v0.20 from 08-Jul-2010

- first L^AT_EX package version of the skb
- no changes in the documentation and no change in commands
- removed acronym list

14.3 v0.30 from 14-Jul-2010

- First dtx release of the skb, including the package and all classes introduced in v0.2
- Integrated parts of the v0.1 pdf as documentation and added documentation for many commands (not finished though)
- Re-write of all load commands (publish, repository, figures, acronyms, bib) and rename of all old load commands, new command names use only lower-cases in their names
- In rewrite, many commands could be removed w/o losing their functionality
- New Commands:
 - `\skbfigure` – load figures with some options
 - `\skbinput` – load files with some options
 - `\skbheading` – set heading text in a file loaded
 - `\skbheadingudc` – set heading relatively to the last heading level (up, down, current) (experimental)
 - `\skbem` – emphasise code using options
 - `\skbacft` – rename of `\SkbAcFT`
 - `\skbacronyms` – rename of `\SkbLoadAcronyms`
 - `\skbbibtex` – rename of `\SkbLoadBibtex`
 - environment `skbnotelist` – itemize list with `\parskip 0` and `\itemskip 0`
 - environment `skbnoteenum` – enumerate list with `\parskip 0` and `\itemskip 0`
- Replaced Commands:

- `\SkbSetTitle` \mapsto replaced by `\skbheading`
- `\SkbFigure` \mapsto removed, closest is `\skbfigure` (but changed behaviour!)
- `\listingInline` \mapsto replaced by `\skbem[code]`
- `\SkbEmIT` \mapsto replaced by `\skbem[italic]`
- `\SkbEmBF` \mapsto replaced by `\skbem[bold]`
- `\SkbAcFT` \mapsto replaced by `\skbacft`
- `\SkbLoadAcronyms` \mapsto replaced by `\skbacronyms`
- `\SkbLoadBibtex` \mapsto replaced by `\skbbibtex`
- `\SkbLoadRepository` \mapsto replaced by `\skbinput[from=rep]`
- `\SkbLoadPublish` \mapsto replaced by `\skbinput[from=pub]`
- `\SkbItemizeBegin` \mapsto replaced by `\begin{skbnotelist}`
- `\SkbItemizeEnd` \mapsto replaced by `\end{skbnotelist}`
- `\SkbEnumerateBegin` \mapsto replaced by `\begin{skbnoteenum}`
- `\SkbEnumerateEnd` \mapsto replaced by `\end{skbnoteenum}`
- `\SkbFigureBeamerTextWidth` \mapsto replaced by `\skbfigure[width=##]`
- `\SkbFigureBeamerTextHeight` \mapsto replaced by `\skbfigure[height=##]`
- `\SkbFigureBeamerNoResize` \mapsto replaced by `\skbfigure[]`
- `\SkbFigureBeamerTextWidthPDFMulti` \mapsto replaced by `\skbfigure[multiinclude=##]`

14.4 v0.31 from 20-Jul-2010

- fixed space problem in `\skbem`
- added error handling to the options `skbconfig` and `skbheading`
- added error handling for `skbinput` related macros
- separated documentation, `skb.dtx` is now using itself to create the documentation
- removed old code: `DeclareOptions` (none declared)
- changed a lot in the documentation
- prepare for CTAN submission, i.e. adding README and other things
- New Commands:
 - `\skbconfig` – change the path/file options
 - `\skbsubject` – add subject information for PDF
 - `\subkeywords` – add keyword information for PDF
 - `\skbpdfinfo` – generate PDF information
- Changed Commands:
 - `\skbfigure` – added option for position, moved caption/label from argument to option
 - `\title` – re-newed to store PDF info information (experimental)
 - `\author` – re-newed to store PDF info information (experimental)
- Replaced Commands:
 - `\SkbCodeInline` \mapsto replaced by `\skbcode`

14.5 v0.32 from 20-Jul-2010

- fastest re-release, I had built-in some problems and excluded important code in v0.31, fixed now

14.6 v0.4 from 21-Jul-2010

- major re-write of the kernel subsequently the documentation. Most internal macros will have been changed or removed, some are added. Also re-arranged the macros in the dtx file to (hopefully) optimise the documentation
- added input for `skb.cfg` and `skblocal.cfg` to overwrite package options with configuration files
- added `skb.cfg` to the distribution
- New Commands:
 - `\skbpathroot` – returns current root path
 - `\skbfileroot` – returns `root/#1`
 - `\skbfileacr` – returns current acronym path and file
 - `\skbfilebib` – returns current bibtex path and file
 - `\skbpathbib` – returns current bibtex path
 - `\skbfilerep` – returns `rep/#1`
 - `\skbfilepub` – returns `pub/#1`
 - `\skbfilefig` – returns `fig/#1`
 - `\skbfilesli` – returns `sli/#1`
 - `\skboptionsused` – prints a warning with change log of options and current values
- Changed Commands:
 - `\skbconfig` – added parameter to identify origin of the configuration change
- Replaced Commands:
 - `\SkbPathBib` \mapsto replaced by `\skbpathbib`
 - `\SkbPathFig` \mapsto replaced by `\skbfilefig`

14.7 v0.5 from 04-Aug-2010

- this is the first version on CTAN: <http://www.ctan.org/tex-archive/macros/latex/contrib/skb/>
- added example describing how the SKB uses itself to create parts of its documentation
- removed the redefinition of `\title` and `\author`, since they intererred with the beamer package definitions of these macros. added `\skbtitle` and `\skbauthor` instead.
- added `RequiredPackage` in the class `skbbeamer` before loading beamer to load `xcolors` with `x11names`
- added test for `nemoir` class: if loaded, then `skbnotelist` and `skbnoteenum` will have no effect; if not loaded, then the package `booktabs` will be loaded

(for top/mid/bottomrule

- added test for beamer package: depending if beamer or beamerarticle are loaded, the SKB will initialise a few new ifs
- added required package dirtree, and redefinition of some dirtree styles
- added two options to the SKB package: beameranim and beamernoanim
- added the package versions with the environments: skbmodetext, skbmode-note and skbmodeslide
- added the package optional with the options: text, note, slide, anim and noanim
- internally, the package optional also provides memoir
- changed the documentation, moved manual description to user guide in folder /doc, moved history.tex into the dtx file and changed most of the actual documentation (still not finished though)
- skbbeamer: corrected load of beamer package
- skblnclsbeamer: moved load of skb after beamerarticle to allow skb to create proper options
- added `\providecommand` for `\escribeMacro` and `\cmd`, so that we can use the user-guide in the dtx and stand-alone
- added conditional load of skb.dtx in the driver
- changed the sequence of definitions in the dtx file, again, hopefully the last time
- Bug Fixes (SF=sourceforge):
 - SF#3032749 (skboptionsused doesn't work) – fixed, changed `\skb@setCfgVars`
 - SF#3032752 (history section for v0.4 has wrong date) – fixed, changed the heading
 - SF#3032754 (skb.cfg missing/empty) – fixed, changed the installer (skb.ins) to generate it and my local scripts to put it into /run
 - SF#3033124 (renewcommand title/author doesn't work) – fixed, no renewcommand anymore, two new commands to set author/title for pdfinfo
 - SF#3038935 (skbinput not working w/o from) – fixed, can load from root directory now
- New Commands:
 - `\skbtitle` – title for PDF info
 - `\skbauthor` – author for PDF info
 - `\skbslide` – load slides and annotations
 - `\skbslidecite` – for citations on slide annotation pages
- Changed Commands:
 - `\skbinput` – added option to load tex files from figures directory (option fig)
- Replaced Commands:
 - `\SkbLoadSlideNotes` \mapsto replaced by `\skbslide` with option annotate and first argument only

- `\SkbLoadSlideNotesDifferent` \mapsto replaced by `\skbslide` with option `annotate` and both arguments
- `\SkbLoadSlideNotesExtern` \mapsto replaced by `\skbslide` with option `annotate` and both arguments and option `notefrom set`
- `\SkbLoadSlideNotes` \mapsto replaced by `\skbslide` without `annotate` and first argument only
- `\SkbLoadSlideOnlyNotes` \mapsto replaced by `\skbslide` with option `annotate` and second argument only
- `\SkbSlideSource` \mapsto replaced by `\skbslidecite`
- `\SkbBeamerAnimtrue` \mapsto replaced by options `beameranim` and `beamer-noanim` for `skbbeamer`
- `\SkbBeamerAnimtrue` \mapsto usage of this if replaced by `\opt` with `anim` and `noanim`

14.8 v0.51 from 12-May-2011

- worked on the documentation, lots of changes
- fixed a typo in `skb.cfg`, which made the bibliography file unloadable
- changed `linkcolor` from `AntiqueWhite4` to `Brown4`
- added acronym database (short version of the automatically generated) and acronym handling in the documentation
- removed `\SKB`, appropriately it's now an acronym rather than a special type setting
- added `bibtex` load to the documentation
- removed call to `\skbbibtex` from class files, users need to call that now manually. reason is that otherwise configuration changes for `bib/bibfile` have no effect
- changed load mechanism for the user guide, due to `bibtex` load problems
- changed the two `skbnote` environments (`list/enum`), removed unnecessary temp storage
- changed default acronym file name to `acronyms`, note the added 's'
- added `skbmoderncv` class using the `moderncv` package and adding some macros for reference lists using `biblatex`

14.9 v0.52 from 03-Jun-2011

- changed directory and package structure to be CTAN conform
- re-worked `README` to reflect new package structure, and added information on generating files from the source

Acronyms

ANTLR	A N o T her Tool for Language Recognition
CMIP	Common Management Information Protocol

CTAN	the Comprehensive TeX Archive Network
GIOP	The General Inter-ORB Protocol
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
ITU	International Telecommunication Unit
JPG	Joint Photographic Experts Group
LAMP	Linux, Apache, MySQL and PHP
LNCS	Lecture Notes in Computer Science
MS	Microsoft
OMG	Object Management Group
PDF	Portable Document Format
PHP	PHP Hypertext Preprocessor
PNG	Portable Network Graphics
RFC	Request for Comment
SDO	Standard Defining Organisation
SKB	Sven's Knowledge Base
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SVG	Scalable Vector Graphics
USB	Universal Serial Bus
W3C	World Wide Web Consortium

References

- [1] Internet Engineering Task Force. *Structure and identification of management information for TCP/IP-based internets*. Request for Comments 1155. (Standard, STD0016). 1990. URL: <http://www.ietf.org/rfc/rfc1155.txt>.