The ufrgscca, and associated, Packages Version 2.3 (extended documentation)

Alceu Frigeri[*]

February 2024

Abstract

This bundled is aimed at producing undergraduate students final work/report at UFRGS/EE (Engineering School at the Federal University of Rio Grande do Sul), closely following ABNT rules (Brazilian Association for Technical Norms). It is composed of a main class, *ufrgscca*, and a set of auxiliary packages, some of which can be used independently.

Contents

[*]https://github.com/alceu-frigeri/ufrgscca

1 Introduction

ABNT rules can be quite challenging at times (read: bibliography style/references) and sometimes just odd (line spacing, front matter, page layout), nevertheless it is a *Brazilian Standard* for typography whose students at UFRGS should grow cherished to follow.

In short, as of version 2.3 the bundle is composed of a class, *ufrgscca* (based on *report* class), which pre-loads, as needed: *ufrgscca-abnt*, *ufrgscca-core*, *ufrgscca-cover* and *ufrgscca-lists*. The packages *ufrgscca-forms*, *ufrgscca-curr*, *ufrgscca-coord* and *ufrgscca-ppc* need to be loaded explicitly, thought.

N.B.: This bundle requires a quite recent LaTeX2e kernel, at least as recent as June 2022, which allows to declare package options using the *key* =*value* system and declare commands with `\NewDocumentCommand`, out-of-the-box.

Version 2.0: Starting with Version 2.0, this bundle has been fully re-written with *Expl3* and *starray*.

Version 1.12a: In case you need the old version, for some odd *expl3* compatibility issue, you can find it at `https://github.com/alceu-frigeri/ufrgscca/releases/tag/v1.12a`

1.1 Current Version

For the sake of the “maintainer’s sanity”, since this is a bundle, all files are saved with the same version (bundle version: 2.3)

2 ufrgscca Class

The following packages are always pre-loaded: *etex*, *etoolbox*, *lmodern*, *fontenc (T1)*, *inputenc (utf8)*, *silence* and *ufrgscca-abnt*, *ufrgscca-cover*, *ufrgscca-core* and *hyperref* and (if it exists) a *local.tex* file.

Other, auxiliary, packages are also pre-loaded, depending on the class options used, and finally *report* class (the exception being in case of the `article` class option).

Being based on the report class, all report class options are valid, in addition to the ones below.

2.1 Class Options

`tocdepth` use: *`tocdepth=number`*, whereas ⟨`number`⟩ indicates the deepest sectioning to appears in the Table of Contents (0 being the top section, which is `\chapter` for report based classes, 1 being `\section`, and so on.) The default is 3 (`\subsubsection`).

`secdepth` use: *`secdepth=number`*, whereas ⟨`number`⟩ indicates the deepest sectioning to be numbered. (0 being the top section, which is `\chapter` for report based classes, 1 being `\section`, and so on.) The default is 4 (`\paragraph`).

`english` the default language being Portuguese, this option changes locale to English.

`brazilian` in some rare cases (to be further investigated) babel seems to get confused about which language is active, this "shouldn't be necessary" but one can explicitly tell babel to use THIS language (which should, otherwise, be the default one).

`relnum` by default, figures, tables, etc. are numbered as a continuous series. With this switch, those lists are reset at each chapter, e.g. Figure 5.1 instead of Figure 23.

`openright` in case of printed material, this assures that a `\chapter` always starts at an odd page, which is relevant in case of printing out (double sided) the document.

`oneside` in case the document will be printed in single side sheets, otherwise it's assumed a two-sided printing.

`strict-abnt` to assure asymmetric margins, as defined by ABNT: inner ones greater than outer ones, which matters if you are going to print the doc and make a book of it, but makes it odd to look at in a computer screen, reason by which the current default setting is for symmetric margins (same text width).

`pretextontoc` "pre-text" elements, like "list of..." will be inserted in the "table of contents".

`yearsonly` Approval page, at it's bottom, will display the years only (instead of the default mouth year construct).

`timesroman` will set the default font to Roman (using the obsolete mathptmx package, based on a free replacement of the proprietary Times New Roman (by Microsoft) and Times Roman (By Adobe)) instead of the default Latin Modern Roman font. As a side effect, the package *`microtype`* isn't loaded (can't be used), resulting in a sub-optimal overall layout. NB. The alternative (newer and maintained) packages *`newtxtext`*/*`newtxmath`* can't be used due to some packages incompatibilities.

`repeatfields` in case of authors with multiple publications, their names will be repeated for each entry. In the default setting the author's name is written only in the first entry, and replaced by underscores in the other entries.

`xlists` this will load the *`ufrgscca-lists`* package, for the definition of new floats/lists.

`xpacks`	this will load a series of packages, which can be handy when writing Engineering reports: *relsize*, *keyval*, *graphicx*, *mathtools*, *mathrsfs*, *amsfonts*, *amssymb*, *empheq*, *amsthm*, *extarrows*, *mathfixs*, *bigdelim*, *circuitikz* and *steimenz* and tikz libraries: *fit*, *math*, *calc*, *shapes.geometry*, *shapes.misc*, *shapes.multipart*, *graphs*, *3d*, *positioning*, *shadows* and *babel*. One is advised to look after each package documentation (ctan.org) for further information.
`chapternopagenum`	to suppress the page numbers at chapters begin.
`nomicrotype`	in some rare cases, *microtype* might hurt page layout, this allows the suppression of *microtype*.
`showframes`	for layout proof only, it will draw frames around each page main parts.
`showlabels`	it will put a reference mark in each label created, and print out it's name.
`nofontwarning`	in case of *ufrgscca-ppc* is loaded, it will suppress some font related warnings.
`nolocal`	this will suppress the loading of any local.tex file, which would, otherwise, be loaded.
`article`	this will load the class *article* instead of *report*, it's meant to document the class itself.
`nogeometry`	the package *geometry* won't be loaded. In case one wants to fully customize the page geometry
`oldrenews`	Some, deprecated, renews will be in effect: `\maketitle`, `\author` `\begin{abstract}`. For backwards compatibility only.
`texlive`	this is a reserved key, in case some workaround for texlive is needed.
`overleaf`	this is a reserved key, in case some workaround for overleaf is needed.
`miktex`	this is a reserved key, in case some workaround for miktex is needed.

2.2 Class Declared Commands

`\autonameref`
`\annexref`
`\autoannexref`

`\autonameref [⟨sep⟩] {⟨label⟩} [⟨spc⟩]`
`\annexref {⟨label⟩}`
`\autoannexref [⟨sep⟩] {⟨label⟩} [⟨spc⟩]`

The *hyperref* package, sometimes, gets the `\autoref` name wrong (when referencing an annex), the `\annexref {⟨label⟩}` will assure the correct annex name is used.
`\autonameref {⟨label⟩}` produces an entry of the form '`\autoref {⟨label⟩}` ⟨sep⟩ `\nameref {⟨label⟩}` ⟨spc⟩'
`\autoannexref {⟨label⟩}` produces an entry of the form '`\annexref {⟨label⟩}` ⟨sep⟩ `\nameref {⟨label⟩}` ⟨spc⟩'
The default ⟨sep⟩ being a comma, and the default ⟨spc⟩ being empty space.

2.3 Class Known Hooks

`\miktexHack`
`\overleafHack`
`\livetexHack`

`\miktexHack`
`\overleafHack`
`\livetexHack`
Case some workaround is needed due an unexpected error (when upgrading packages/TEXsystem) the class "knows" about those three hooks. They will be executed if, and only if, they are user defined and the corresponding package option is used, i.e., for example, for the hook `\miktexHack` to be used/called by the class *ufrgscca*, one has to: a) define it and b) use the class option `miktex`.

3 ufrgscca-abnt Package

This package is the one that sets the page layout (using *geometry*, *titlesec* and *titletoc*) and adjusts the main float environments (figure, tables, captions). It can be used as a stand alone package, regardless of the underlying class.

The following packages are always pre-loaded: *babel*, *csquotes*, *geometry*, *appendix*, *titlesec*, *titletoc*, *enumitem*, *chngctr*, *caption*, *biblatex*, *microtype*, *array*, *nicematrix*, *contour* and *soul*.

Take note that *biblatex* is loaded with the **biber** option, to correctly handle ABNT biography style.

3.1 Package Options

`strict-abnt`	to assure asymmetric margins, as defined by ABNT: inner ones greater than outer ones, which matters if you are going to print the doc and make a book of it, but makes it odd to look at in a computer screen, reason by which the current default setting is for symmetric margins (same text width).
`chapternopagenum`	to suppress the page numbers at chapters begin.
`relnum`	by default, figures, tables, etc. are numbered as a continuous series. With this switch, those lists are reset at each chapter, e.g. Figure 5.1 instead of Figure 23.
`repeatfields`	in case of authors with multiple publications, their names will be repeated for each entry. In the default setting the author's name is written only in the first entry, and replaced by underscores in the other entries.
`yearsonly`	In some cover pages (like the ones for TCC) the bottom of the approval's page, will only displays the year (instead of the default mouth year construction).

`nomicrotype`	in some rare cases, *`microtype`* might hurt page layout, this allows the suppression of *`microtype`*.
`showframes`	for layout proof only, it will draw frames around each page main parts.
`showlabels`	it will put a reference mark in each label created, and print out it's name.
`tocdepth`	use: *`tocdepth`* = ⟨number⟩, whereas ⟨number⟩ indicates the deepest sectioning to appears in the Table of Contents (0 being the top section, which is `\chapter` for report based classes, 1 being `\section`, and so on.) The default value being 3 (`\subsubsection`).
`secdepth`	use: *`secdepth`* = ⟨number⟩, whereas ⟨number⟩ indicates the deepest sectioning to be numbered. (0 being the top section, which is `\chapter` for report based classes, 1 being `\section`, and so on.) The default value being 4 (`\paragraph`).

3.2 Commands

`\mainkeyword`
`\otherkeyword`

new: 2023/11/18

```
\mainkeyword {⟨keyword⟩}
\otherkeyword {⟨keyword⟩}
```

These command can be invoked many times, it will construct a list of keywords to be used when printing out the (main/other)abstract environment.

Note: the old `\keyword {⟨keyword⟩}` gets defined, as an allias to `\mainkeyword`, in case one use the *`oldrenews`* class option is used.

`\sourcecitation`
`\note`

```
\sourcecitation {⟨source⟩}
\note {⟨text⟩}
```

When describing floating elements (like figure, tables, circuits) one always has to cite the source of it, and in some cases it might be necessary to add a special note. Those assure uniformity when doing that.

`\nonum`
`\notoc`

```
\nonum\chapter {⟨chap.title⟩}
\nonum\section {⟨sec.title⟩}
\notoc\chapter {⟨chap.title⟩}
\notoc\section {⟨sec.title⟩}
```

In some cases, it might be necessary to create a numberless chapters or sections. Those two commands can be used as a *prefix* to any sectioning command. Whilst `\nonum` will just suppress the sectioning number, the `\notoc` will also suppress it from the table of contents.

LaTeX Code:

```
\nonum\chapter{some title} %this one will appear in the toc
\notoc\section{some other title} %this won't even appear in the toc
```

\tightul

\tightul {⟨text⟩}

This will *underline* a short text, take note that ⟨text⟩ 'can't be broken' (think paragraph justification), which can lead to *text overflows* and bad justification.

LaTeX Code:

```
\tightul{Some text example}%
```

LaTeX Result:

Some text example

\NewChapListEnv

\NewChapListEnv {⟨envname⟩}{⟨displayname⟩}

Creates those *chapter like* lists, like 'List of Symbols' or 'List of acronyms'. With it, a new environment is created, ⟨envname⟩, with an associated 'numberless' chapter name ⟨displayname⟩. The newly created environment will implement a *description* like environment (thanks to ***enumitem***) with an optional and a mandatory argument (see below).

LaTeX Code:

```
\def\listabbrvname{Lista de Abreviaturas}
\NewChapListEnv{listofabbrv}{\listabbrvname} % this is the actual code
    used in ufrgscca-abnt.sty
```

\pubdate
\today
\monthname

update: 2023/11/30

\pubdatedate [⟨day⟩] {⟨month⟩}{⟨year⟩}
\today
\monthname

\pubdate sets the publication date. If not called by the user it *defaults* to current month / year. \today returns the current *locale* date, whilst \monthname returns the *locale* name of the current month.

N.B.: If the package option oldrenews is used, the command \date will be redefined as an allias to \pubdate.

3.3 Environments

mainabstract

new: 2023/11/18

\begin{mainabstract} [⟨lang⟩] {⟨keywords⟩}...\end{mainabstract}

mainabstract is defined as a numberless chapter based on the current locale (default: Portuguese), at the end of it the keywords list created with \mainkeyword will be added.

LaTeX Code:

```
\mainkeyword{a keyword}
\mainkeyword{another keyword}
\begin{mainabstract} some short summary of things\ldots
\end{mainabstract}
```

otherabstract `\begin{otherabstract} [⟨lang⟩] {⟨...⟩}\end{otherabstract}`

update: 2023/11/18

This is the environment to create an abstract in a language other than the default one. The default value for ⟨lang⟩ is english, and it can be any value that *babel* understands. A keywords list created with `\otherkeyword` will be added at the end of it.

LaTeX Code:

```
\otherkeyword{a keyword}
\otherkeyword{another keyword}
\begin{otherabstract}[english] some short summary of things\ldots
\end{otherabstract}
```

listofabbrv `\begin{listofabbrv} [⟨enum-opt⟩] {⟨ABBRV⟩}...\end{listofabbrv}`
listofsymbols `\begin{listofsymbols} [⟨enum-opt⟩] {⟨SYMB⟩}...\end{listofsymbols}`

Both environments create a description like list preceded by a numberless (`\nonum`) chapter. ⟨enum-opt⟩ is any *enumitem* list valid key. Whereas ⟨ABBRV⟩ / ⟨SYMB⟩ are just the 'biggest' abbreviation/symbol to be used as a tab reference.

appendix `\begin{appendix} .... \end{appendix}`
annex `\begin{annex} .... \end{annex}`

Those two environments start the appendices and annex chapters (using locale). Chapters are alphabetic *numbered* (starting at A).

3.4 Tabular New Columns

Thanks to *array* some new columns types are defined:

P `P{⟨width⟩}` Normal text, ragged left.

B `B{⟨width⟩}` Bold text, ragged left.
C `C{⟨width⟩}` Normal text, centered.
R `R{⟨width⟩}` Normal text, ragged left.
L `L{⟨width⟩}` Normal text, ragged right.
J `J{⟨width⟩}` Normal text, justified.

3.5 enumitem Extra Keys

Besides the *default* keys defined by the *enumitem* package a few others are defined for author's convenience:

ppc, tcc *ppc* and *tcc* are alias of each other, and just assure that lists indentation will be the same as paragraphs default.

parindent with *parindent*, the list number/mark is aligned with paragraph indentation.

noindent *noindent* removes the label indentation.

LaTeX Code: LaTeX Result:

```
\begin{enumerate}[tcc]
  \item some A
  \item some B
\end{enumerate}
\begin{enumerate}[tcc,parindent]
  \item some A
  \item some B
\end{enumerate}
\begin{enumerate}[parindent]
  \item some A
  \item some B
\end{enumerate}
\begin{enumerate}[noindent]
  \item some A
  \item some B
\end{enumerate}

New paragraph, for reference.
```

1. some A
2. some B

1. some A
2. some B

1. some A
2. some B

1. some A
2. some B

New paragraph, for reference.

tight allows for very tight lists (no indentation) to be used, for instance, inside quotes. N.B. don't use it in normal paragraph mode, otherwise the labels will spill outside the default text window.

miditemsep *miditemsep* halves items separation, as an alternative to *noitemsep* from *enumitem*

LaTeX Code: LaTeX Result:

```
\begin{enumerate}[tcc]
  \item some A
  \item some B
\end{enumerate}
\begin{enumerate}[tcc,miditemsep]
  \item some A
  \item some B
\end{enumerate}
\begin{enumerate}[tcc,noitemsep]
  \item some A
  \item some B
\end{enumerate}
```

1. some A

2. some B

1. some A
2. some B

1. some A
2. some B

bullet	for simple itemized lists, it will replace the default black dot by an 'open bullet'

LaTeX Code: LaTeX Result:

```
\begin{itemize}[tcc,miditemsep]
  \item some A
  \item some B
  \item some C
\end{itemize}
\begin{itemize}[tcc,bullet,
    miditemsep]
  \item some A
  \item some B
  \item some C
\end{itemize}
```

- some A
- some B
- some C

◦ some A
◦ some B
◦ some C

arabic	That's the *default* enumerate style. Arabic numbers, starting at 1, followed by a dot.
arabic)	Label will be constructed as number followed by a parenthesis.
(arabic)	Label will be enclosed by parenthesis.
*arabic**	(for secondary lists) Label will be constructed by the label of the outer list, this item number and a final dot.
arabic)*	(for secondary lists) Label will be constructed by the label of the outer list, this item number and a final parenthesis.
roman	This and below keys are the same as the arabic ones, but using lower case roman numbers.
roman)	lower case roman number, followed by a parenthesis.
(roman)	enclosed by parenthesis.
*roman**	preceding one followed by roman number and a final dot.
roman)*	same, followed by a final parenthesis.
Roman	This and below keys are the same as the arabic ones, but using upper case roman numbers.
Roman)	upper case roman number, followed by a parenthesis.
(Roman)	enclosed by parenthesis.
*Roman**	preceding one followed by roman number and a final dot.
Roman)*	same, followed by a final parenthesis.
alpha	This and below keys are the same as the arabic ones, but using lower case alpha numbers.

alpha)	lower case alpha number, followed by a parenthesis.
(alpha)	enclosed by parenthesis.
*alpha**	preceding one followed by alpha number and a final dot.
alpha)*	same, followed by a final parenthesis.
Alpha	This and below keys are the same as the arabic ones, but using upper case alpha numbers.
Alpha)	upper case roman number, followed by a parenthesis.
(Alpha)	enclosed by parenthesis.
*Alpha**	preceding one followed by roman number and a final dot.
Alpha)*	same, followed by a final parenthesis.

LaTeX Code: LaTeX Result:

```latex
\begin{enumerate}[tcc,roman]
  \item some A
  \item some B
  \item some C
\end{enumerate}
\begin{enumerate}[tcc,Roman]
  \item some A
  \item some B
  \begin{enumerate}[tcc,alpha*]
    \item some A
    \item some B
    \item some C
  \end{enumerate}
  \item some C
\end{enumerate}
\begin{enumerate}[tcc,arabic]
  \item some A
  \item some B
  \begin{enumerate}[tcc,roman*)]
    \item some A
    \item some B
    \item some C
  \end{enumerate}
  \item some C
\end{enumerate}
```

```
    i.  some A

   ii.  some B

  iii.  some C

    I.  some A

   II.  some B

          II.a.  some A
          II.b.  some B
          II.c.  some C

  III.  some C

    1.  some A

    2.  some B

           2.i)  some A
          2.ii)  some B
         2.iii)  some C

    3.  some C
```

4 ufrgscca-core Package

The *ufrgscca-core* package defines a set of commands for student's and activity's related info. It is needed by most/all of the bundled packages.

All data is stored in two main *starray* defined as follow:

Activity's Structure Definition:

```
{
  name , acronym ,
  coord . struct
    {
      name , title ,
      article , Article , narticle , Narticle , carticle , Carticle ,
    } ,
  calendar . struct
    {
      date ,  week , event ,
    } ,
    chkmarked , chkunmarked , chkref
}
```

Student's Structure Definition:

```
{
 first , last , name , Nproc , ID , email , worktitle ,
       article , Article , narticle , Narticle , carticle , Carticle ,
 remarks , checklist , brief , reason ,
 board-local , board-date , board-time , gradeavrg , grade ,
 flag-graded , %%% IF gradeavrg AND finalgrade already calculated (or defined)
 flag-exam ,
 flag-ff ,
 flag-dismiss , %%% IF it was the 1st semester.
 flag-newpage , %% if it should go in a new page (board)
 flag-distinctboard , %% if advisor isn't in the board
 flag-approved ,
 flag-coadvisor ,
 advisor . struct {
   first , last , name , institution , title , email , phone ,
       article , Article , narticle , Narticle , carticle , Carticle ,
   assessment
 } ,
 coadvisor . struct {
   first , last , name , institution , title , email , phone ,
       article , Article , narticle , Narticle , carticle , Carticle ,
   reason
 } ,
 reviewer . struct {
   first , last , name , institution , title , email , phone ,
       article , Article , narticle , Narticle , carticle , Carticle ,
   pointA , pointB , pointC , pointD , grade , gradetype ,
 } ,
 altreviewer . struct {
   first , last , name , institution , title , email , phone ,
       article , Article , narticle , Narticle , carticle , Carticle ,
 } ,
 internship . struct {
   company , field , start , end , length ,
 } ,
 tutor . struct {
   first , last , name , title , email , phone ,
       article , Article , narticle , Narticle , carticle , Carticle ,
 } ,
 supervisor . struct {
   first , last , name , register , title , office , email , phone ,
       article , Article , narticle , Narticle , carticle , Carticle ,
 } ,
}
```

4.1 Core Commands

`\NewActivity`

new: 2023/11/18

`\NewActivity` {⟨act-hash⟩}

This will create a new 'activity'. Predefined ones being: `course`, `tccI`, `tccII`, `internship` and `internship-opt`.

Note: This will create a *starray*, the ⟨act-hash⟩ being it's ⟨hash⟩.

```
\ActivitySet              \ActivitySet [⟨act-hash⟩] {⟨acronym⟩} {⟨name⟩}
\ActivitySetCoordTitle    \ActivitySetCoordTitle [⟨act-hash⟩] {⟨title⟩}
\ActivitySetCoord         \ActivitySetCoord [⟨act-hash⟩] {⟨name⟩} [⟨gender⟩]
new: 2023/11/18
```

These will set an Activity many fields. ⟨acronym⟩ and ⟨name⟩ being the short (acronym) and long name of an activity. ⟨title⟩ is the coordinator formal title, and so on.

```
\ActivitySelect   \ActivitySelect {⟨act-hash⟩}
\Activity         \Activity [⟨act-hash⟩] {⟨act-field⟩}
\ActivityCoord    \ActivityCoord [⟨act-hash⟩] {⟨coord-field⟩}
new: 2023/11/18
```

`\ActivitySelect` just sets ⟨act-hash⟩ as the current activity (set's the *starray* iter). `\Activity` and `\ActivityCoord` gets the corresponding field. Possible values for ⟨act-field⟩ are: `name` and `acronym`. Possible values for ⟨coord-field⟩ are: `name`, `title`, `article`, `Article`, `narticle`, `Narticle`, `carticle` and `Carticle`, as defined by *Activity's structure* (see 4).

4.2 Core Auxiliary Commands

```
\studentselect    \studentselect {⟨student-hash⟩}
new: 2023/11/18
```

Select a student based on it's hash.

```
\DataFields    \DataFields {⟨starray-ref⟩} {⟨field⟩}
\eDataSet      \eDataSet [⟨act-hash⟩] {⟨starray-ref⟩}
\eDataFields   \eDataFields {⟨field⟩}
new: 2023/11/18
```

These are, respectively, `\starray_get_prop:nn`, `\starray_term_syntax:n` and `\starray_parsed_get_prop:n` from *starray*. One can reference/get any field from the main *starray* defined structures: *student* and *activity* as defined at chapter 4.

```
\studentiterate            \studentiterate {⟨code⟩}
\studentadvisoriterate     \studentadvisoriterate {⟨code⟩}
new: 2023/11/18
update: 2023/12/02
```

These are `\starray_iterate_over:nn` from *starray*. ⟨code⟩ will be executed for every defined student, `\studentiterate` or student's advisor, `\studentadvisoriterate`.

```
\ActivityCalendarIterate    \ActivityCalendarIterate {⟨code⟩}
new: 2023/11/29
```

This is `\starray_iterate_over:nn` from *starray*. ⟨code⟩ will be executed for every defined calendar item.

4.3 Core Specific Commands

The following commands are more or less self-explanatory, ⟨ID⟩ is the student's university ID. ⟨Nproc⟩ is the process/request number. ⟨gender⟩ can be either 'm' or 'f'.

`\student`
`\studentinfo`

update: 2023/11/18

```
\student [⟨student-hash⟩] {⟨last⟩} {⟨first⟩} [⟨gender⟩]
\studentinfo [⟨Nproc⟩] {⟨ID⟩} {⟨email⟩}
```

N.B.: If the package option `oldrenews` is used, the command `\author` and `\authorinfo` will be redefined as an allias to `\student` and `\studentinfo`.

`\workbrief`
`\advisorreview`
`\coadvisorreason`
`\workchange`

new: 2023/11/18

```
\workbrief {⟨work-summary⟩}
\advisorreview {⟨advisor's-review⟩}
\coadvisorreason {⟨reason-for-a-coadvisor⟩}
\workchange {⟨reason-for-the-change⟩}
```

Those commands are only of use when using *ufrgscca-forms*. `\workbrief` sets the work initial summary, `\coadvisorreason` sets the justification for having a co-advisor, `\advisorreview` sets the advisor's review, `\workchange` sets the reason for the work's theme change.

`\advisor`
`\advisorinfo`

```
\advisor [⟨title⟩] {⟨last⟩} {⟨first⟩} [⟨gender⟩]
\advisorinfo {⟨Institut⟩} {⟨title-info⟩} {⟨email⟩} {⟨phone⟩}
```

`\coadvisor`
`\coadvisorinfo`

```
\coadvisor [⟨title⟩] {⟨last⟩} {⟨first⟩} [⟨gender⟩]
\coadvisorinfo {⟨Institut⟩} {⟨title-info⟩} {⟨email⟩} {⟨phone⟩}
```

`\distinctboard`

new: 2023/11/18

```
\distinctboard
```

For the rare case in which the advisor won't take part in the examiner's board.

`\examiner`
`\examinerinfo`

```
\examiner [⟨title⟩] {⟨last⟩} {⟨first⟩} [⟨gender⟩]
\examinerinfo {⟨Institut⟩} {⟨title-info⟩} {⟨email⟩} {⟨phone⟩}
```

`\altexaminer`
`\altexaminerinfo`

```
\altexaminer [⟨title⟩] {⟨last⟩} {⟨first⟩} [⟨gender⟩]
\altexaminerinfo {⟨Institut⟩} {⟨title-info⟩} {⟨email⟩} {⟨phone⟩}
```

\internship

new: 2023/11/18

`\internship {⟨company⟩}{⟨field⟩}{⟨start⟩}{⟨end⟩}{⟨length⟩}`

\tutor
\tutorinfo

`\tutor [⟨title⟩] {⟨last⟩}{⟨first⟩} [⟨gender⟩]`
`\tutorinfo {⟨Institut⟩}{⟨title-info⟩}{⟨email⟩}{⟨phone⟩}`

\supervisor
\supervisorinfo

`\supervisor [⟨title⟩] {⟨last⟩}{⟨first⟩} [⟨gender⟩]`
`\supervisorinfo {⟨register⟩}{⟨office⟩}{⟨email⟩}{⟨phone⟩}`

N.B.: The commands `\advisor`, `\coadvisor`, `\examiner` and `\altexaminer` are meant to be used in a 'final work' doc. The Macros `\internship`, `\tutor` and `\supervisor` in case of an internship report.

5 ufrgscca-cover Package

This package is the one that sets the front pages, depending on the kind of 'report' being generated.

5.1 Defined Commands

\MakeCoverPages

new: 2023/11/18

`\MakeCoverPages {⟨type⟩}`

This is the main command, which will typeset the front matter, from the information already given. ⟨type⟩ sets the 'kind' of cover pages to be generated. Currently, it can be one of:

tccI	Generate 3 pages, a first cover one, a second with work's description and third last one with work's approval for TCC-I
tccII	Generate 3 pages, a first cover one, a second with work's description and third last one with work's approval for TCC-II
internship	Generate 2 pages, a first cover one, a second with work's approval for internship report
internship-opt	Generate 2 pages, a first cover one, a second with work's approval for optional internship report
class-report	Generate 1 cover page

N.B.: If the package option `oldrenews` is used, the command `\maketitle` will be redefined as an allias to `\MakeCoverPages`.

\location

`\location {⟨city⟩}{⟨state⟩}`

To redefine the default values of ⟨city⟩ and ⟨state⟩ (Porto Alegre and RS).

`\class` — new: 2023/11/18

`\class {⟨code⟩}{⟨name⟩}`

To set the class code and name, for the cover page, in case of a class report.

`\SetCoverFields` — new: 2023/11/18

`\SetCoverFields {⟨type⟩}{⟨field⟩}{⟨value⟩}`

This allows to redefine the aforementioned ⟨types⟩ and create new types of cover pages. ⟨field⟩ is one of:

`clist`	this defines which kind, and order, of pages will be generated. Possible values are: `cover`, `desc` and `approval`.
`top`	This will be the common top matter used.
`students`	How students names, authors, will be presented
`title`	The title to be used
`bottom`	The bottom of the cover page.
`text-descpage`	The text presented in the desc page.
`advisor-descpage`	Advisor's matter.
`bottom-descpage`	The bottom of the desc page.
`text-approvalpage`	The text presented in the approval page.
`advisor-approvalpage`	Advisor´s matter in the approval page.
`bottom-approvalpage`	The bottom of the approval page.

6 ufrgscca-forms Package

This package defines just two user commands to generate specific forms needed at UFRGS/EE.

6.1 *Forms Defined Commands*

`\tcforms` `\tcemptyforms` — update: 2023/05/29

`\tcforms {⟨formslist⟩}`
`\tcemptyforms {⟨formslist⟩}`

The command `\tcforms` will generate the many forms (⟨formslist⟩) using the information from *local.tex*, whilst `\tcemptyforms` will generate said forms with 'blanks' (to be fulfilled by hand, for instance).

⟨formslist⟩ is a csv list of:

`reqform-I`	
`reqform-II`	Registration requirement form.
`coadvisor-I`	
`coadvisor-II`	Coadvisor justification form.
`boardapproval-I`	
`boardapproval-II`	Boards approval form.
`advisorsapproval-I`	
`advisorsapproval-II`	Advisors approval form.
`receipts-II`	Receipts forms (one per board member).
`examinersforms-I`	

`examinersforms-II`	Grades and correction forms (per board member).
`rectifyapproval-I`	
`rectifyapproval-II`	Corrections approval form.
`internreqform`	Internship Registration requirement form.
`internsupervisorform`	Internship Supervisor evaluation form.
`interntutorform`	Internship tutor evaluation form.

Please note that those '-I' regards TCC-I, while '-II' regards TCC-II.

`\SetForm`
`\MakeForm`
new: 2023/11/18

`\SetForm {⟨form-hash⟩}{⟨field⟩}{⟨code⟩}`
`\MakeForm {⟨form-hash⟩}`

`\SetForm` can be used to set new forms (or redefine existent ones). ⟨form-hash⟩ being a free identifier. Possible ⟨field⟩ values are *heading*, *title*, *opening*, *body*, *closing* and *footnone*. `\MakeForm` typesets the selected form.

7 ufrgscca-lists Package

The following packages are always pre-loaded: *newfloat*, *listings* and *xcolor*. It defines a new *floating environment codelist*. Combined with *listings* one can typeset exempts of a *code listing*.

7.1 Environment

codelist

`\begin{codelist} ... \end{codelist}`

`\caption` will be named 'Listing' (Listagem).

LaTeX Code:

```
\begin{codelist}[htbp]
  \caption{sample C code}
  \label{code01}
  \begin{lstlisting}[language=C]
    struct i2c_msg
    {
      __u16 addr;     /* endereco do escravo */
      __u16 flags;
    }
  \end{lstlisting}
  {\sourcecitation{\textcite{Garg:SMA-2000}}}
\end{codelist}
```

7.2 Declared Commands

`listofcodelist`

`\listofcodelist`

This will create the 'List of ...' associated with the *codelist* environment.

\DeclareNewFloat

\DeclareNewFloat {⟨env-name⟩} {⟨file-ext⟩} {⟨listname⟩} {⟨listofname⟩}

A new float environment, named ***env-name***, will be created. Captions will be associated (numbered) as ⟨**listname**⟩ **num:**. Finally, an associated command \listof... will be defined, using ⟨listofname⟩ as a numberless \chapter title.

LaTeX Code:

```
\def\listingname{Listing}%
\def\listlistingname{List of Listings}%
\DeclareNewFloat{codelist}{lox}{\listingname}{\listlistingname}%%
%% after that, one can do as in the previous example
%%
%% the list of, will be created as
\listofcodelist
```

8 ufrgscca-coord Package (extended documentation)

This package defines a set of auxiliary commands meant to support the Professor coordinating students work. it will always pre-load the ***longtable*** and ***ufrgscca-forms*** packages.

N.B. It might be also useful to use the commands defined at subsection 6.1, Forms Defined Commands.

A *report document* is composed of 2 main parts:

1. A global preamble, where one sets
   1.a. the current semester, Course/TCC/internship coordinator's names, etc. ,
   1.b. auxiliary data, like students *check list* items and
   1.c. students data.
2. A 'final part' whereas one set which reports are to be generated.

One can (should) use the commands listed at subsection 4.1, Core Commands, and these below:

Check List

\checkdef
\checklist

update: 2023/11/18

\checkdef {⟨LxCy⟩}{⟨check-item⟩}{⟨check-text⟩}
\checklist {⟨check-items-list⟩}

Whereas one has a '5x5 matrix' (⟨checkLC⟩ being one of L1C1...L1C5, ... , L5C1...L5C5). ⟨chek-item⟩ is a free identifier (to be used with the \checklist), and ⟨check-text⟩ the text to appear in the 'check list report'. Note this is a list **per activity** (the current one being set).
\checklist set's those items for the current student. ⟨check-items-list⟩ is a comma separated list of ⟨check-item⟩.

LaTeX Code:

```
\checkdef{L1C1}{tcc-part}{Rel. Parcial}        % this creates the '
  check item' tcc-part and associates it with the L1C1 position (first
  line, first column), display text 'Rel. Parcial'
\checkdef{L2C1}{partOK}{Aprov. Rel. Parcial}   % this creates 'partOK'
  and associates it with L2C1 position

\checkdef{L1C2}{board}{Banca def.}             %
\checkdef{L2C2}{board-date}{Data defesa}       % 'board-date' is
  associated with the L2C2 position

\checkdef{L1C5}{tcc-final}{TCC final}          %
\checkdef{L2C5}{approval}{Aprovação Correções} %
\checkdef{L4C5}{exam}{Em Exame}                % 'exam' (display 'Em
  Exame') is associated with the L4C5 position
%%
%%
%% later on, one can use (inside a \NewStudent command)
\checklist{tcc-part,partOK,exam}                  % this will, for a
  given student, 'mark' the 'tcc-part', 'partOK' and 'exam' items.
```

Auxiliary / Report Specific

\ActivitySetNewEvent
\ActivitySetEventDay

new: 2023/11/18

\ActivitySetNewEvent [⟨act-hash⟩] {⟨event-hash⟩}{⟨event-desc⟩}
\ActivitySetEventDay [⟨act-hash⟩] {⟨event-hash⟩}{⟨day⟩}{⟨week⟩}

An activity may have a calendar/set of associated events. ⟨event-hash⟩ is just a hash to reference it (*starray* hash). ⟨event-desc⟩ is the text associated with it. ⟨day⟩ and ⟨week⟩ the associated date.

\studentremark
\studentnewpage
\distinctboard

new: 2023/11/18

\studentremark {⟨remarks⟩}
\studentnewpage [⟨student-hash⟩]
\distinctboard

Those commands are only of use when using *ufrgscca-coord*. \studentremark sets a free remark text (notes about). Whilst, \distinctboard and \studentnewpage set the *flag-distinctboard* and *flag-newpage* flags..

```
\studentCase
\studentAdvCase
\studentCoadvCase
\studentDismissCase
\studentNewPageCase
\studentDistinctBoardCase
\studentReviewerCase
```

new: 2023/11/18
update: 2024/01/15

```
\studentCase {⟨if-A-B-C⟩}{⟨if-D⟩}{⟨if-Exam-C⟩}{⟨if-Exam-D⟩}
    {⟨if-FF⟩}
\studentAdvCase {⟨if-more-than-one⟩}{⟨if-not⟩}
\studentCoadvCase {⟨if-defined⟩}{⟨if-not⟩}
\studentDismissCase {⟨if-dismiss⟩}{⟨if-not⟩}
\studentNewPageCase {⟨if-newpage⟩}{⟨if-not⟩}
\studentDistinctBoardCase {⟨if-distinct⟩}{⟨if-not⟩}
\studentReviewerCase {⟨if-marked⟩}{⟨if-not⟩}
```

These are a set of auxiliary conditionals, for instance, `\StudentCase` will execute *only one* of the ⟨`if-`⟩ accordingly.

```
\professor
\advisorprof
\coadvisorprof
\examinerprof
\altexaminerprof
```

new: 2023/11/30

```
\professor [⟨prof-ref⟩] {⟨last⟩}{⟨first⟩}{⟨email⟩}{⟨phone⟩} [⟨gender⟩]
\advisorprof {⟨prof-ref⟩}
\coadvisorprof {⟨prof-ref⟩}
\examinerprof {⟨prof-ref⟩}
\altexaminerprof {⟨prof-ref⟩}
```

`\professor` creates a ⟨`prof-ref`⟩ alias to a person's full name. `\advisorprof` expands to a call for `\advisor` and `\advisorinfo` using the properties stored by `\professor`. Similarly, `\coadvisorprof`, `\examinerprof` and `\altexaminerprof` results in call to `\coadvisor`, `\coadvisorinfo`, `\examiner`, `\examinerinfo`, `\altexaminer` and `\altexaminerinfo` respectively.

Note: `\professor` creates/uses an auxiliary *`starray`*, just mapping names for users convenience.

```
\boardtitle
\boardobs
\semester
```

new: 2023/11/18

```
\boardtitle {⟨title⟩}
\boardobs {⟨obs⟩}
\semester {⟨semester⟩}
```

`\boardobs` allows to add an observation (⟨`obs`⟩) for the 'boards schedule report', and `\semester` sets the current semester value.

Student Specific Commands

```
\studentfate
```

update: 2024/02/15

```
\studentfate [⟨fate⟩]
```

This assigns the ⟨`fate`⟩ of a student, for those cases that one cannot rely on the 'calculated one' (from examiners individual grades). ⟨`fate`⟩ can be either *exam* (if the student is in exam, but didn't got a grade yet) *C* or *D* (in case a student in exam got graded), *FF* for those that haven't finished the work or *dismiss* for those that, for whatever reason, got dismissed. The default is 'do nothing' (no ⟨`fate`⟩ assigned)

Note: (updated in 2024/02/15) In case some other, odd, value is assigned, this command will record as if the student is in 'exam', with the given grade ⟨`fate`⟩ marked in bold red.

\studentaddtolist
new: 2023/12/04

\studentaddtolist {⟨listID⟩}

Adds the student to a given list (defined by ⟨listID⟩), to be later used by \sortstudentlist and \tcreport.

\checklist

\checklist {⟨csv-checkitems⟩}

⟨csv-checkitems⟩ is a csv list of valid 'items' (the ones defined by \checkdef) and it will 'mark' (check) the corresponding items for a given student.

\timeslot
update: 2023/11/29

\timeslot [⟨local⟩] {⟨date⟩} {⟨time⟩}

To set the ⟨local⟩, ⟨date⟩ and ⟨time⟩ of a student's presentation work.

> ***Note:*** ⟨date⟩ must be given in numerical form, either ⟨day⟩/⟨month⟩ or ⟨day⟩/⟨month⟩/⟨year⟩. The day-of-the-week will be obtained using *pgfcalendar*.
> Likewise, ⟨time⟩ must be given in a (24h) ⟨hour⟩:⟨min⟩ format.

\worktitle
\studentremark
update: 2023/11/18

\worktitle {⟨title⟩}
\studentremark {⟨remark⟩}

\worktitle sets the current student "work's title". \studentremark just inserts a ⟨remark⟩, which will appear in the *report*'s report.

\distinctboard
update: 2023/11/18

\distinctboard

Normally, the default, it's assumed that the student's advisor will also be a member of the student's exam board. For the ones in which this doesn't holds true, one should use the \distinctboard after setting a student's name (via \student) and before setting its advisor name (via \advisor).

For example:

LaTeX Code:

```
\student[Arthur]{last}{first}[m]

\studentinfo[]{243716}{email@somewhere}
\worktitle{work title}
\distinctboard
\advisor{de Amorin}{Heraldo José}[m]
\examiner{Götz}{Marcelo}[m]                 % He will be the 1st
  examiner
\examiner{Comparsi Laranja}{Rafael Antônio} % the 2nd
\examiner{Ventura Bayan Henriques}{Renato}  % the 3rd
```

`\examinergrades`
update: 2024/02/15

`\examinersgrades {⟨N1⟩}{⟨N2⟩}{⟨N3⟩}[⟨N4⟩]*`

Quite obvious, this set the grades given by an examiner (the one defined by the 'last' `\examiner` before this.). In case ⟨N4⟩ is given it's assumed the TCC-I case, otherwise TCC-II. The 'star' at the end will mark said reviewer, in red, when generating a report.

Reports Command

`\setstudentlist`
update: 2023/11/29

`\setstudentlist {⟨listID⟩}{⟨list⟩}`

This command will define/create a list named ⟨listID⟩ composed of a csv ⟨list⟩ of student hashes (as defined by `\student [⟨student-hash⟩]{⟨...⟩}`).

`\sortstudentlist`
new: 2023/12/04

`\sortstudentlist {⟨listID⟩}`
`\sortstudentlist* {⟨listID⟩}`
`\sortstudentlist+ {⟨listID⟩}`

These will sort (and classify) a given student list defined by ⟨listID⟩. The star version sorts the list by student's full name, the *plus* version sorts the list by student's presentation date. By default, the list remains unchanged (no sort).

`\tcreports`

`\tcreports [⟨report-list⟩]{⟨listID⟩}`

This will typeset the many reports, using the student list defined by ⟨listID⟩. Where ⟨report-list⟩ is a csv list of keys as follow:

`calendar-I`	Calendar for the period, TCC-I.
`report-I`	a student control report, for TCC-I.
`checklist-I`	a student check list, for TCC-I.
`revforms-I`	per student reviews forms, TCC-I.
`referral-I`	per student referral letters, TCC-I.
`calendar-II`	Calendar for the period, TCC-II.
`report-II`	a student control report, for TCC-II.
`checklist-II`	a student check list, for TCC-II.
`revforms-II`	per student reviews forms, TCC-II.
`referral-II`	per student referral letters, TCC-II.

`boards`	exam board dates, TCC-II.
`attendance`	a simple student's attendance list.
`cocertificate`	per student coadvisor certificate letter (if any).

9 ufrgscca-ppc Package (extended documentation)

This contains a set of auxiliary commands to keep track of many *indicators* whilst writing a *PPC document* (which is going to be evaluated based on said *indicators*, though the track of those *indicators* themselves shall not appear in the final version of it). Keep in mind, when considering the use of it: "it works as is" but it hasn't being properly debugged, and it might change "as needed locally".

The packages *`longtable`*, *`pdfcomment`*, *`mdframed`* and *`ufrgscca-curr`* will always be pre-loaded.

9.1 Package Options

`showind`	(for drafts) it will display the report indicators, of those indicators whose family wasn't set to hide.
`indlong`	(for drafts) when displaying an indicator, the long version of them will be used.
`nocomments`	(for drafts) comments (created with the command `\comment {⟨⟩}` will be suppressed.

9.2 Defined Commands

The next few commands use a finite set of ⟨`status`⟩ which are a predefined list of:

`tbd`	"To Be Done"
`done`	"Done"
`review`	"to be reviewed"
`attention`	Needs Attention
`NSA`	NSA (portuguese for "do not apply")
`noref`	no references
`EAD`	EAD (portuguese for "distance learning")
`MDi`	course ware (portuguese for "didactic material")
`DOCs`	other DOCs
`default`	everything else

\declareindicator
\indicatorDesc
\indicatorText

`\declareindicator*+ [⟨status⟩] {⟨fam⟩} {⟨ID⟩} {⟨text⟩}`
`\indicatorDesc {⟨longdesc⟩} {⟨extra⟩}`
`\indicatorText {⟨text⟩}`

`\declareindicator` is the command to create/define a given "indicator". ⟨fam⟩ set's its *family* group, ⟨ID⟩ is the particular ID/term used to reference it (in a family of indicators), ⟨text⟩ is a short text describing it (it is the text displayed when using the `\indref` below.). `\indicatorDesc` adds a ⟨longdesc⟩ (long description) and ⟨extra⟩ (extra long description) to a defined `\declareindicator` (it will add those text fields to the "last declared one"). ⟨longdesc⟩ will also be displayed when using the `\indref` commands, but only if the `indlong` option was used. The ⟨extra⟩ will only be used/displayed with the `\PrintIndicators` command. Finally, `\indicatorText` adds a remark ⟨text⟩, which will be also printed out when using `\lstind` (akin of an explanation/remark field.)

\indsetstatus
\indsetview
\indsethide

`\indsetstatus [⟨status⟩] {⟨fam⟩} {⟨ID⟩}`
`\indsetview {⟨fam⟩}`
`\indsethide {⟨fam⟩}`

`indsetstatus` sets the ⟨status⟩ of a given indicator defined by ⟨fam⟩ and ⟨ID⟩. `\indsetview` and `indsethide` {⟨s⟩}et the visibility (or not) of a given "family" of indicators, meaning, if those indicators are going to be visible or not (command `\indref`, for instance) if the option `showind` is in use.

\lstind

`\lstind [⟨seclvl1⟩] [⟨seclvl2⟩] {⟨fam⟩}`

`\lstind` will produce a sectioning like list, ⟨seclvl1⟩ defaults to `\section` and ⟨seclvl2⟩ defaults to `\subsection`, those indicators marked with an * (when creating them) will be issued with ⟨seclvl1⟩, those marked with an + will be issued with ⟨seclvl2⟩. The indicator's short text will be the sectioning title, whilst the indicator's 'text' (the one assigned with `indicatorText` will be the sectioning body.)

\PrintIndicators

`\PrintIndicators [⟨fam⟩]`

`\PrintIndicators` will produce a "list of contents" like list (with cross reference to all used `\indref` pages). It will either issue a list of all `\declareindicator` or just the ones belonging to ⟨fam⟩. ⟨fam⟩ can be a csv list of families. Each entry will be composed by indicator's "family", "ID", "short text", "long text" and "extra description" but not the text issued with `\indicatorText`.

\helpindicators

`\helpindicators`

This will just prints, middle text, a quick "help text" listing the few main "indicators related command" (to help out those less LaTeX 2_ε savvy writers.)

\ifshowind

`\ifshowind {⟨code-ifshow⟩} {⟨code-ifnot⟩}`

Just a helping command, based on the package options. If the option `showind` was used, ⟨code-ifshow⟩ is executed, otherwise ⟨code-ifnot⟩.

\textmark
\comment

`\textmark [⟨status⟩] {⟨text⟩}`
`\comment [⟨status⟩] {⟨title⟩} {⟨text⟩}`

Those are annotation, remark commands. The difference being that `\textmark` will just highlight the ⟨text⟩ (using ⟨status⟩ "format"), whilst `comment` will create a "remark box" (the same used when inserting an indicator's reference, commands below).

N.B.: The command `\comment` is suppressed unless the option `showind` is used.

\indref
\indreflst

`\indref {⟨<⟩}*>[status]fam,ID,comment`
`\indreflst {⟨<⟩}*>[status]fam,IDlist,comment`

`\indref` creates a box (*TikZ* based *mdframed*) of the indicator denoted by ⟨fam⟩ and ⟨ID⟩. The family and IDs will be issued as the "frame title", the current indicator's ⟨status⟩ will be printed out (the whole box will be highlighted accordly), the short version of the indicator will be used (the long version will "appear" as a *pdfcomment*), finally any ⟨comment⟩ will be added to the text box. Each `\indref` box will have a link to the indicator's list (issued with `\PrintIndicators`). If the optional argument ⟨status⟩ is used, the indicator's status will be updated accordly. The star version also prints the indicator's long text.
`\indreflst` behaves similarly, with the difference that ⟨IDlist⟩ is a csv list of IDs (same family), moreover, each item of said list can have the form either just ⟨ID⟩ or ⟨status:ID⟩, in the last form, that ID will have its status updated, as well.

\fancyquote

`\fancyquote [⟨vspc⟩] {⟨text⟩} {⟨author⟩} {⟨dateref⟩}`

As quick "quote" hack, `\fancyquote` will typesets a ⟨text⟩ (small size, italic text, in a minipage environment) followed by ⟨author⟩ and ⟨dateref⟩. This is meant to be used after a `\chapter` or `\section` commands. ⟨vspc⟩ is to be used in case one has to adjust the vertical space between the sectioning command, and the quote one.

\labelhack

`\labelhack {⟨text⟩}`

As the name implies, it is a hack. In some cases (which we haven't manage to found why/what), *hyperref* would fail miserably when using the `\nameref` (in some cases getting the sectioning correct, but not the name!). This just assures that `\nameref` will use the correct sectioning name in those cases.

For Example:

```
    \section{this section}\labelhack{this section}\label{somelabel}
```

\acrodef	\acrodef {⟨acroID⟩}acronymlong
\acro	\acro {⟨acroID⟩}
\acrol	\acrol {⟨acroID⟩}
\acrols	\acrols {⟨acroID⟩}
\acrosl	\acrosl {⟨acroID⟩}
\acrofoot	\acrofoot {⟨acroID⟩}
\printacrolist	\printacrolist [⟨enumkeys⟩] {⟨widest⟩}

Those are yet another acronym hack. `\acrodef` “creates” an acronym, identified by ⟨acroID⟩, whose short (acronym) version is ⟨acronym⟩ and the long version in ⟨long⟩. `\acro` just typesets the ⟨acronym⟩, `\acrol` the ⟨long⟩ version. `\acrols` typesets the the long version followed by the short (using a comma as separator). `\acrosl` prints the short version first. Finally, `\acrofoot` typesets the short version in text and the long as a footnote. `\printacrolist` creates a description list based on the *listofabbrv* environment.

9.3 Environments

ppc.quote	\begin{ppc.quote} ... \end{ppc.quote}

This is just a tailored “quote” environment, using almost all page width, just in a smaller font size.

10 ufrgscca-curr Package (extended documentation)

The background of it: To have the ability to “describe” (store the information in a “structured way”) an University Course Curricula and have the possibility, later, to presented that same information in many different ways (including a dependence graph). All data is captured/stored in a set of 3 *starrays*:

topics’ Structure Definition:

```
{
  self  = , name = , color = , class lst = , %list (sequence) of classes
}
```

Classes’ Structure Definition:

```
{
 cred = , self = , name = , summary = , topic = , remark = ,
 bib seq = ,  bib basic seq = ,  bib compl seq = ,
 ref . struct = {
   curr = , sem = , kind = ,
 } ,
}
```

Curricula's Structure Definition:

```
{
  self = , name = , text = ,
  sem.struct = {
    pos = , self = , name = ,
    class . struct = {
      name = , kind = , obs = , pos = , color = ,
      prereqset . struct = {
        prereq . struct = {
          starred = , name = , ang = ,
        }
      } ,
    } ,
  } ,
}
```

The following commands "describe" a curricula, whereas one is a sequence of semesters ⟨semID⟩, each semester is composed by a list of classes, ⟨classID⟩, and each class has a list of dependencies, ⟨classID⟩ as `\depdef`. All those lists are stored as csv lists, so "processing them" can be systematized.

`\topicdef`
`\defaulttopic`

`\topicdef [⟨color⟩] {⟨topicID⟩} {⟨text⟩}`
`\defaulttopic {⟨topicID⟩}`

`\topicdef` defines ⟨topicID⟩ (to be used when describing a class) and associates a ⟨text⟩ description and a ⟨color⟩ (for topic highlight). `\defaulttopic` sets the default one (if not explicitly given when describing a class).

`\classdef`
`\classset`
update: 2023/11/18

`\classdef [⟨topicID⟩] {⟨classID⟩} {⟨cred⟩} {⟨name⟩}`
`\classset {⟨classID⟩}`

`\classdef` defines a class ⟨classID⟩ (with a given ⟨name⟩ and number of ⟨name⟩) associated with a given ⟨topicID⟩. `\classset` sets the current class.
The following commands always refer to the "last defined" `\classdef` unless `\setclass` is used, which changes the "current class" for the following commands.

`\csummary`
`\classremark`
`\bibdef`
new: 2023/11/18

`\csummary {⟨desc⟩}`
`\classremark {⟨remark⟩}`
`\bibdef [⟨type⟩] {⟨text⟩}`

`\csummary` sets a class summary, whilst `\classremark` annotates a 'note/remark'. `\bibdef` is used to set a list of bibliographies, one per issued command. The default ⟨type⟩ value is *`main`*, other possible values *`basic`* and *`compl`*.

`\currdef`
`\semdef`
update: 2023/11/18

`\currdef {⟨curr-ID⟩} {⟨short name⟩} {⟨long name⟩}`
`\semdef {⟨semID⟩} {⟨name⟩} {⟨pos⟩}`

`\currdef` creates a curricula (with a set of semesters defined as following). `\semdef` creates a semester, ⟨semID⟩, ⟨pos⟩ being a position 'hint' when creating a dependency graph (see below).

\depdef
\altdep

update: 2023/11/18

\depdef*<⟨pos⟩>{⟨classID⟩}
\altdep

\depdef inserts/creates a “class dependency” list. ⟨pos⟩ is used as a ’hint’ for the incident (dependency) line angle. The starred version is meant to be used when the ’dependency’ isn’t another class but rather, for instance, a number of credits.
\altdep defines/start and alternate dependency list.

\TabEtp
\TabTopic

update: 2023/11/18

\TabEtp* [⟨sectioning⟩] {⟨semID⟩} [⟨font-fmt⟩]
\TabTopic {⟨topicID⟩}

\TabEtp will construct a longtable with all classes associated with ⟨semID⟩. The default ’font size’, ⟨font-fmt⟩, is \footnotesize. The default ⟨sectioning⟩ is \notoc\section. The non-star version includes the bibliography lists as well.
\TabTopic will construct a longtable with all classes associated with a ⟨topicID⟩.

\GraphEtp

update: 2023/11/18

\GraphEtp {⟨semID⟩}

It will create a dependency graph for a given ⟨semId⟩. N.B. to start with, it is highly dependent on the semester sequence, one shall start with first semester and go from there.