The ufrgscca, and associated, Packages Version 1.0.1 (extended documentation)

Alceu Frigeri

September 2022

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

1 Introduction

ABNT rules can be quite challenging some 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 1.0.1 the bundle is composed of a class, `ufrgscca` (based on the standard LaTeX2e `report` class), which pre-loads all other, as needed, packages: `ufrgscca-abnt`, `ufrgscca-core`, `ufrgscca-cover`, `ufrgscca-forms`, `ufrgscca-gen`, `ufrgscca-lists`, `ufrgscca-curr`, `ufrgscca-coord`, `ufrgscca-ppc`.
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 new `key = value` system and declare commands with `\NewDocumentCommand`, out-of-the-box.

1.1 Current Version

For the sake of the 'maintainers' sanity, since this is a bundle, all files are saved with the same version (bundle version), with two exceptions: `ufrgscca-curr.sty` `ufrgscca-ppc.sty` which are less tested than the others, and somewhat in what one would call 'beta' state. Better said, all files are version 1.0.1, except `ufrgscca-curr` and `ufrgscca-ppc` whose versions are 1.0.1beta.

2 ufrgscca Class

The following packages are always pre-loaded: *etex, etoolbox, lmodern, fontenc (T1), inputenc (utf8), silence, ufrgscca-abnt, ufrgscca-gen, ufrgscca-cover, ufrgscca-core, hyperref* and (if it exists) a *local.tex* file.
Other set of auxiliary packages are also pre-loaded, depending on the class options used, and finally it loads (normally) the `report` class (the exception being if one uses the `dctools` option).
Being based on the report class, one can use all class options one would with a report, plus the ones listed below.

`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`).

`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 will assure 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).

`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, steimenz* and tikz libraries: *fit, math, calc, shapes.geometry, shapes.misc, shapes.multipart, graphs, 3d, positioning, shadows, babel*. One is advised to look after each package documentation (ctan.org) for further information.

`report` in case the doc is just a class assignment with, possibly, many co-authors. It changes mainly the front matter, which is simplified (no referral page, for instance).

`internship` in case the doc is an internship report.

`forms` in the process of submitting a student final work/report, there is a series of forms to be submitted, this allows the customization of said forms in a simple way.

`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.

`dctools` this will change page layout and base class to article, it is meant to document the class itself.

`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), therefore 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.

3 ufrgscca-abnt Package

This package is the one that sets the page layout (using *geometry*, *titlesec*, *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.

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`).

dctools this will change page layout and base class to article, it is meant to document the class itself.

`\keyword` `\keyword{⟨keyword⟩}`
This command can be invoked many times, it will construct a list of keywords to be used when printing out the abstract environment.

`\sourcecitation` `\sourcecitation{⟨source⟩}`
`\note` `\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` `\nonum\chapter{⟨chap.title⟩}`
`\notoc` `\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:	LaTeX Result:
`\tightul{Some text example}%`	Some text example

`\NewChapListEnv` `\NewChapListEnv{⟨envname⟩}{⟨displayname⟩}`
This is the command used to created 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
```

`\date` `\date[⟨day⟩]{⟨month⟩}{⟨year⟩}`
`\today` `\today`
`\monthname` `\monthname`
The command `\date` is redefined, to allow a separation between the many arguments ⟨day⟩, ⟨month⟩ and ⟨year⟩. 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.

3.2 Environments

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

The standard environment *abstract* is redefined as a numberless chapter based on the current locale (default: Portuguese), at the end of it the keywords list created with \keyword will be added.
LaTeX Code:

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

otherabstract

\begin{otherabstract} [⟨lang⟩] {⟨keywords⟩}...\end{otherabstract}
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. The ⟨keywords⟩ are just a list of keywords which will be added at the end of the *otherabstract*.
LaTeX Code:

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

listofabbrv
listofsymbols

\begin{listofabbrv} [⟨enum-opt⟩] {⟨ABBRV⟩}...\end{listofabbrv}
\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
annex

\begin{appendix}.... \end{appendix}
\begin{annex}.... \end{annex}
Those two environments start the appendices and annex chapters (using locale). Chapters are alphabetic *numbered* (starting at A).

3.3 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.

3.4 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

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:

```
\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

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

4 ufrgscca-core Package

The *ufrgscca-core* package defines a set of commands for authors, students, advisors and examiners names and related info. It is needed by most/all of the tc bundled packages.

4.1 Core Forms Commands

\tccbrief
\tcccoadvisorbrief
\tccadvisorsreview

\tccbrief {⟨brief⟩}
\tcccoadvisorbrief {⟨brief⟩}
\tccadvisorsreview {⟨brief⟩}

Those commands are only of use when using *ufrgscca-forms*. \tccbrief sets the work initial summary, \tcccoadvisorbrief sets the justification for having a co-advisor, \tccadvisorsreview sets the advisor's review.

4.2 Core Global Commands

\location

\location {⟨city⟩} {⟨state⟩}

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

\TCCcoord
\TCCcoordtitle

\TCCcoord {⟨(title) full name⟩} [⟨gender⟩]
\TCCcoordtitle {⟨coordinator denomination⟩}

\coursecoord
\coursecoordtitle

\coursecoord [⟨(title) full name⟩] [⟨gender⟩]
\coursecoordtitle {⟨course coordinator denomination⟩}

⟨coordinator denomination⟩ and ⟨course coordinator denomination⟩ are the full 'job title' of their position. ⟨gender⟩ can be either 'm' or 'f'.

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'.

\author
\authorinfo
\student
\studentinfo

\author {⟨last⟩} {⟨first⟩} [⟨gender⟩]
\authorinfo [⟨Nproc⟩] {⟨ID⟩} {⟨email⟩}
\student {⟨last⟩} {⟨first⟩} [⟨gender⟩]
\studentinfo [⟨Nproc⟩] {⟨ID⟩} {⟨email⟩}

\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⟩}`

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

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

5 ufrgscca-cover Package

This package is the one that sets the front pages, depending on the kind of 'report' being generated. The default being to generate 3 cover pages: an identification on, followed by presentation one, then an referral/approval one.

5.1 Package Options

report in case the doc is just a class assignment with, possibly, many co-authors. It changes mainly the front matter, which is simplified (no referral page, for instance).

internship in case the report is a internship one.

5.2 Defined Commands

`\maketitle` `\maketitle`
This is the only main command, which will typeset the front matter. It requires that all *specific info* be already set up (like work title, author's name, affiliation, etc.)

`\course` `\course {⟨arg⟩}`
`\courseacronym` `\courseacronym {⟨arg⟩}`
`\graduationtitle` `\graduationtitle {⟨arg⟩}`
`\university` `\university {⟨arg⟩}`
`\universityacronym` `\universityacronym {⟨arg⟩}`
`\universitydivision` `\universitydivision {⟨arg⟩}`
`\divisiongradcouncil` `\divisiongradcouncil {⟨arg⟩}`
`\department` `\department {⟨arg⟩}`
`\classcode` `\classcode {⟨arg⟩}`
`\classname` `\classname {⟨arg⟩}`
`\subject` `\subject {⟨arg⟩}`
In case some customization is needed, one can change them as needed. The default values are set for the *control and automation* course at UFRGS/EE.

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` `\tcforms {⟨formslist⟩}`
`\tcemptyforms` `\tcemptyforms {⟨formslist⟩}`
`\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 any of:

reqform Registration requirement form.
coadvisor Coadvisor justification form.
boardsapproval Boards approval form.
advisorsapproval Advisors approval form.
receipts Receipts forms (one per board member).
examinersforms Grades and correction forms (per board member).
rectifyapprovalform Corrections approval form.

7 ufrgscca-lists Package

The following packages are always pre-loaded: *newfloat*, *listings* and *xcolor*. It defines a new *floating environment*. Combined with *listings* one can typeset exempts of *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 previous 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-gen Package (extended documentation)

Just two set of commands are defined, one is kind of a 'command factory' aimed at creating macros in a standard way, while the other helps create 'case like' commands.

8.1 Package Options

family sets the family name, defaults to *tcdef*.
group sets the group name, defaults to *gen*.

8.2 Defined Commands

\cmdfactory `\cmdfactory [⟨fam⟩] <⟨grp⟩> {⟨cmd-list⟩}`
\factory `\factory [⟨fam⟩] <⟨grp⟩> {⟨cmd⟩}`
\tcgen@cdef `\tcgen@cdef [⟨fam⟩] <⟨grp⟩> {⟨cmd⟩} {⟨new-val⟩}`

`\cmdfactory` is the actual command meant to be used (the other two are just auxiliary ones). ⟨cmd-lst⟩ is a csv list of commands. ⟨fam⟩ is the command *family* (defaults to *tcdef*) and ⟨grp⟩ is the family group (defaults to *gen*).
The newly created commands will be based on `\tcgen@cdef` (the actual assigment command) having the form `\cmd {⟨new-val⟩}`, accepting a single mandatory value. Internally ⟨new-val⟩ will be stored in a macro likely named `\fam@grp@cmd`.
`\factory` is basically the same as `\cmdfactory`, whilst to create just one new command (it is the command called by `\cmdfactory` via `\forcsvlist`.)

\mkswitch `\mkswitch [⟨default⟩] {⟨sw-name⟩}`
\addcase `\addcase {⟨sw-name⟩} {⟨str-case⟩} {⟨code⟩}`

`mkswitch` will create a command, `\sw-name {⟨case⟩}`, which will behave like a switch/case in other programming languages. ⟨default⟩ is the code to be executed in case a *switching value* isn't defined. `\addcase` adds *cases*, one by one, to the switch. ⟨str-case⟩ can be any `\csname` valid name. ⟨code⟩ is the code to be executed.
LaTeX Code:

```
\mkswitch[\gr@depcut]\gr@case@angle
\addcase\gr@case@angle{}{\def\gr@ANG{0}}
\addcase\gr@case@angle{A}{\def\gr@ANG{\gr@A}}
%%
%% actual use of the switch
\gr@case@angle{A}  % this will result in \def\gr@ANG{\gr@A}
```

9 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. One can select the reports/forms to be generated using the package options or the command `\setreports {⟨keys⟩}`
N.B. It might be also useful to use the commands defined at subsection 6.1, Forms Defined Commands .

9.1 Package/Report Options

calendar Calendar for the period.
checklist a students check list.
report a student control report.
boards exam board dates.
boarddates exam board dates with highlighted dates.
studentlist a simple student list.
revforms per student reviews forms.

referral per student referral letters.
cocertificate per student coadvisor certificate letter (if any).

9.2 Defined Commands

The *report document* to be created is composed of 2 main parts:

1. A global preamble, where one sets
   1.a. the current semester, Course/TCC coordinator 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.

9.2.1 Global Commands I

One can (should) use the commands listed at subsection 4.2, Core Global Commands , and these below:

\tcccalendareventdate
\boardstitleB
\boardsOBS
\TCCperiod

\tcccalendareventdate{⟨date⟩}
\boardstitleB{⟨titleB⟩}
\boardsOBS{⟨obs⟩}
\TCCperiod{⟨semester⟩}

Use \tcccalendareventdate to set the date of a given 'event' (the list of 'calendar events' are (might have been) set in the *ufrgscca-ptBR-coord.def* or *ufrgscca-en-coord.def* file). boardstitleB sets a 2nd title line for the 'boards schedule report'. \boardsOBS allows to add an observation (⟨obs⟩) for the 'boards schedule report', finally, \TCCperiod{⟨s⟩}ets the current semester value.

\checkdef

\checkdef{⟨checkLC⟩}{⟨check-item⟩}{⟨check-text⟩}

Whereas one has a '4x5 alphabetic matrix', lines A to D, columns A to E. ⟨checkLC⟩ being one element of that matrix (from checkAA up to checkDE), ⟨chek-item⟩ is a free identifier (to be used with the \checklist), and ⟨check-text⟩ the text to appear in the 'check list report'. So, for instance:
LaTeX Code:

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

\checkdef{checkAB}{board}{Banca def.}             %
\checkdef{checkBB}{board-date}{Data defesa}       % 'board-date' is
    associated with the BB position

\checkdef{checkAE}{tcc-final}{TCC final}          %
\checkdef{checkBE}{approval}{Aprovação Correções} %
\checkdef{checkDE}{exam}{Em Exame}                % 'exam' (display 'Em
    Exame') is associated with the DE 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.
```

Be aware that, \checkdef can and should be only used at the preamble, whereas \checklist can only be used at the 'student data definition' context (meaning, inside the \NewStudent command).

9.2.2 Global Commands II

\NewStudent

`\NewStudent {⟨studentname⟩} {⟨code⟩}`

This is the main command describing each ⟨student⟩ associated work, advisor and exam board. In ⟨code⟩ one should use the commands defined in subsection 4.3, Core Specific Commands , and subsubsection 9.2.3, Student Specific Commands (although one can use any valid LaTeX 2_ε preamble code, keep in mind those will be executed BEFORE `\begin{document}`), to describe a student work. So, for instance:
LaTeX Code:

```
\NewStudent{Artur}{
  \student{last}{first}[m]

  \studentinfo[]{243716}{email@somewhere}
  \TCCtitle{work title}
  \advisor{de Amorin}{Heraldo José}[m]
  \coadvisor{Camargo Nardelli}{Vítor}[m]
  \examinergrades{9.2}{8.5}{9.2}
  \examiner{Götz}{Marcelo}[m]
  \examinergrades{10}{9.5}{9.5}
  \examiner{Comparsi Laranja}{Rafael Antônio}
  \examinergrades{8.5}{8.5}{8}
  \altexaminer{Ventura Bayan Henriques}{Renato}
  %%
  %%
  \timeslot[Teams]{12/11}{15:30}

  \studentFate[Dismiss]  %% FF or Dismiss ??
}
```

N.B. Internally, `\NewStudent` will create a command named `\studentname`, with a *hook* named `\studentname.hook` (the dot is part of the hook's name).

9.2.3 Student Specific Commands

\studentFate

`\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 C or D (in case a student got in exam), 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)

\studenttimeslot
\timeslot

`\studenttimeslot [⟨local⟩] {⟨date⟩} {⟨time⟩}`
`\timeslot [⟨local⟩] {⟨date⟩} {⟨time⟩}`

`\timeslot` is just an alias of `\studenttimeslot`. They set, for the Boards Report, the ⟨local⟩, ⟨date⟩ and ⟨time⟩ in which a student will have its work presented. Those commands are meant to be used 'inside' a `\NewStudent` command.

\studentTCCtitle
\TCCtitle
\studentremark

`\studentTCCtitle {⟨title⟩}`
`\TCCtitle {⟨title⟩}`
`\studentremark {⟨remark⟩}`

`\TCCtitle` is also just an alias to `\studentTCCtitle` which just 'defines' the current student "work's title". `\studentremark` just inserts a ⟨remark⟩, which will appear in the *report*'s report (. . . report option).

\DistinctBoard
\DefaultBoard

`\DistinctBoard`
`\DefaultBoard`

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 informing a student's name (via `\student`) and before informing its advisor name (via `\advisor`). For instance:
LaTeX Code:

```
\NewStudent{Artur}{
  \student{last}{first}[m]

  \studentinfo[]{243716}{email@somewhere}
  \TCCtitle{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 `\examinersgrades {⟨N1⟩} {⟨N2⟩} {⟨N3⟩}`
Quite obvious, this set the grades given by an examiner (the one defined by the 'last' `\examiner` before this.).

\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.

\addtostudent `\addtostudent {⟨student⟩} {⟨code⟩}`
⟨code⟩ will be appended to the command created with `\NewStudent`. ⟨student⟩ must be an already defined one, whilst ⟨code⟩ can be anything valid in the context of a `\NewStudent` as described in subsubsection 9.2.2, Global Commands II .

\setreports `\setreports {⟨rep-list⟩}`
⟨rep-list⟩ is a csv list of keys as defined at subsection 9.1, Package/Report Options .

\setstudentlist `\setstudentlist {⟨listID⟩} {⟨list⟩}`
This command will define/create a list named ⟨listID⟩ composed of a csv ⟨list⟩ of student names (as defined by `\Newstudent`).

\tcreports `\tcreports [⟨rep-list⟩] {⟨listID⟩}`
This is the main command, to be used only once, at the end of the file. It will typeset the reports, as set by `\setreports`, using the student list identified by ⟨listID⟩. ⟨rep-list⟩ is a csv list of keys as defined at subsection 9.1, Package/Report Options .

10 ufrgscca-ppc Package (beta) (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.

10.1 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.

10.2 Defined Commands

`\maketitle`

`\maketitle`

`\maketitle` is redefined for the specifics of a *PPC document*.

The next few commands use a finite set of ⟨status⟩ which are a pre-defined 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` `\textmark [⟨status⟩] {⟨text⟩}`
`\comment` `\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. `\comment` is suppressed unless the option `showind` is used.

`\indref` `\indref* [⟨status⟩] {⟨fam⟩} {⟨ID⟩} {⟨comment⟩}`
`\indreflst` `\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
\acro
\acrol
\acrols
\acrosl
\acrofoot
\printacrolist

\acrodef {⟨acroID⟩}acronymlong
\acro {⟨acroID⟩}
\acrol {⟨acroID⟩}
\acrols {⟨acroID⟩}
\acrosl {⟨acroID⟩}
\acrofoot {⟨acroID⟩}
\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.

10.3 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.

11 ufrgscca-curr Package (beta) (extended documentation)

This package is mostly in beta state, some parts of it should be identified as alpha state. Those are mostly rushed out adaptations of other "solutions at hand". Literally, try to use it at your own peril.
The background of it: To have the ability to "describe" (store the information in a "structured way") an University Course curricula and later on have the possibility to presented that same information in many different ways (including a dependence graph). To an extended, most of it is done (and working), but hopeless lacking more testing and debugging.
Why is it included in the bundle? Well, it is needed, in part for completeness, by *ufrgscca-ppc* which is "locally important".

11.1 Commands Creating the many lists

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).

\semdef

\semdef [⟨pos⟩] <⟨cod⟩> {⟨semID⟩}
This "defines" a semester, ⟨semID⟩, and associates with it a ⟨cod⟩ (for reference) and a ⟨pos⟩ (to be used by, for instance, *ufrgscca-curr-graph*.)

\classdef
\setclass
\classremark

\classdef [⟨topicID⟩] <⟨pos⟩> {⟨classID⟩} {⟨cred⟩} {⟨typ⟩} {⟨name⟩} {⟨desc⟩}
\setclass {⟨classID⟩}
\classremark {⟨remark⟩}

\classdef defines a class, associating with a ⟨topicID⟩, ⟨pos⟩ (for *ufrgscca-curr-graph*), ⟨classID⟩, number and type, ⟨typ⟩, of credits, ⟨cred⟩, a long name, ⟨name⟩ and description, ⟨desc⟩. \classremark adds an extra remark to it.
The following commands always refer to the "last defined" \classdef unless \setclass is used, which changes the "current class" for the following commands.

\depdef
\altdef

\depdef [⟨topicID⟩] <⟨pos⟩> {⟨classID⟩}
\altdef

\depdef inserts/creates a "class dependency" list. The highlight color (if used) is usually defined by the current class topic (informing ⟨topicID⟩ changes the highlight color). ⟨pos⟩ is used by *ufrgscca-curr-graph* to determine the incident line angle.
\altdef defines/start and alternate dependency list.

\bibdef

\bibdef [⟨type⟩] {⟨text⟩}

This is used to set a list of bibliographies, one per issued command. The default ⟨type⟩ value is just *bib*, possible values (as understood by *ufrgscca-curr-tab*) are *bib*, *basic* and *comp*.

11.2 List Processing Commands

\LstClass
\LstDep
\LstTopic

Those are the main loop commands that go through the lists.
\LstClass [⟨cmd⟩] {⟨semID⟩}
\LstDep [⟨cmd⟩] <⟨ang⟩> {⟨classID⟩}
\LstTopic [⟨cmd⟩] {⟨topicID⟩}
⟨cmd⟩ can be any command accepting a single argument. It will, in fact, be the one defining the way the data will be, effectively, be presented.
\LstClass will process ⟨cmd⟩ over all classes associated with ⟨semID⟩.
\LstDep will process ⟨cmd⟩ over all dependency classes associated with ⟨classID⟩.
\LstTopic will process ⟨cmd⟩ over all classes associated with ⟨topicID⟩.

12 ufrgscca-curr-tab Package (alpha) (extended documentation)

This is truly a work in progress (based on some old ideas), not really tested. It shall be revised and, mostly sure, it will be changed (no compatibility guaranties). It always pre-load *ufrgscca-curr* and *longtable*.

12.1 Tabular Presentation Commands

\TabEtp
\TabTopic

\TabEtp <⟨type⟩> [⟨sectioning⟩]c {⟨semID⟩}
\TabTopic [⟨type⟩] {⟨topicID⟩}
\TabEtp will construct a longtable with all classes associated with ⟨semID⟩ (including it's dependencies and bibliography).
\TabTopic will construct a longtable with all classes associated with ⟨topicID⟩.

13 ufrgscca-curr-graph Package (alpha) (extended documentation)

13.1 Graph Presentation Command

Ironically, this is the "oldest" of the *-curr-* packages, but it is the less tested one, and the one whose code is more prone to fail in unexpected ways, be advised: do not try to use it, unless you know the internal code well. It always pre-load *ufrgscca-curr* (N.B. it also depends on *tikz*).

\GraphSem

`\GraphSem [⟨type⟩] {⟨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.