The thalie package[*]
A package to typeset drama plays

Louis Paternault
spalax(at)gresille(dot)org

June 26, 2014

Abstract

This package is meant to typeset drama plays using LaTeX. It provides commands to introduce characters' lines, to render stage direction, to divide a play into acts and scenes, to automatically build the dramatis personæ, etc.

Contents

[*] This document corresponds to thalie v0.6, dated 2014/06/26.

1 Introduction

This document introduces the thalie package, used to typeset drama plays.

1.1 Other classes and packages

This package is far from being the only one that can be used to render drama plays. If you do not like my work, you can use one of the following ones (and I guess there exists others): `drama` [5], `dramatist` [6], `play` [7], `screenplay` [8], `sides` [9].

I began to write this package in 2010, and now, at the end of 2012, I must admit that I wonder why I started this... There are already several such packages in CTAN[1], and the `dramatist` package seems really nice (I borrowed some ideas and copied some code from it). There are actually a few improvements in my package compared to `dramatist`: in my package, plays, acts and scenes appear in the table of contents; it is possible to include several plays in a single document; there are more options when building the dramatis personæ; headers and footers are taken into account, etc. But these improvements are small; it might have been smarter to contribute to `dramatist` instead of starting my own package. The good part is that I have a package that exactly[2] fits my needs, and I learned how to write a LaTeX package.

Oh, yes! I know why I started this: I am a geek...

1.2 License

This work may be distributed and/or modified under the conditions of the LaTeXProject Public License, either version 1.3 of this license or (at your option) any later version.

Further information can be found in the `.dtx` file used to build this document.

In short (but this paragraph has no legal value), you can use this package freely to render your drama plays, and modify it almost freely. Nevertheless, if you like my work, you can send me smiles, cakes or poetry, or, better, the text of the plays you write using my package[3].

[1] `http://www.ctan.org/topic/drama-script`

[2] Well, not exactly: see the wish list page 25. But I do not think `dramatist` can do this either.

[3] Seriously, it would make me *really* happy!

Command	Result (in English)	Result (in French)
`\playname`	Play	Pièce
`\actname`	Act	Acte
`\scenename`	Scene	Scène
`\interludename`	Interlude	Intermède
`\pausename`	Pause	Pause
`\curtainname`	Curtain	Rideau

Table 1: Localization commands

1.3 Acknowledgements

I borrowed ideas and lines of code from the LaTeX packages `drama`[5] and `dramatist`[6].

I used the following guides to package my package: *How to Package Your LaTeX Package* [3], and *LaTeX 2ε for class and package writers* [4].

1.4 Examples

All the examples are taken from Edmond Rostand's *Cyrano de Bergerac* [2]. The snippets are from the original (French) version of this play, roughly translated by myself (which means that they must be of poor quality).

A part of this play (now in public domain), is attached to this file, as a working example of this package.

1.5 Overview

Documentation about how to use this package is given in section 2. In particular, section 2.3 explains how to use acts and scenes, section 2.4 explains how to define characters, and use these definitions to introduce characters' lines, and section 2.5 describes commands used to render stage directions.

Implementation is given in section 3.

2 Usage

2.1 Localization

`\playname`
`\actname`
`\scenename`
`\interludename`
`\pausename`
`\curtainname`

Language cannot be defined directly in this package. Instead, the language currently used by the `babel` package (i.e. the language returned by `babel`'s `\languagename` command) is used to display names.

Package `babel`, and its languages you want to use, must be loaded *before* this package, otherwise localization will not be available.

The effect of choosing a language or another is to translate some words. The corresponding commands, and the words they correspond to, are listed in table 1.

If the option for your language does not exist[4], or if you want to change the

[4] You can also send me the translation for your language, to improve this package.

Name	Available options	Default
characterstyle	bold margin center simple arden imprimerie-verse imprimerie-prose	simple
playstyle	center bigcenter box custom	box
actstyle		bigcenter
scenestyle		center
playlevel	part chapter section ... subparagraph	chapter
actlevel		section
scenelevel		subsection
interludelevel	play act scene	act

Table 2: Package options summary

default words used here, you can redefine the commands listed in table 1. For instance use \renewcommand{\curtainname}{Tel\`on} for Spanish (if I am not wrong).

2.2 Package options

2.2.1 Summary

Package options are summed up in table 2.

2.2.2 Style

The way characters' lines are displayed, as well as play, act and scene titles, can be set when loading the options. To set character style, use option characterstyle=⟨*style*⟩. Available styles, and indication to use a custom one, are described in section 2.4.4.

Several play, act and scene title styles are defined. Choose it using options playstyle=⟨*style*⟩, actstyle=⟨*style*⟩ and scenestyle=⟨*style*⟩. Description of available styles, and how to define your own one, are described in section 2.3.2.

2.2.3 Sectioning levels

If you use a table of contents, or if you also use "usual" sectioning commands (\chapter, \section and so on), the relative importance of plays, acts and scenes is important. You can set this using options playlevel=⟨*level*⟩, actlevel=⟨*level*⟩ and scenelevel=⟨*level*⟩. The argument is the name of the corresponding sectioning level, i.e. one of part, chapter, section, subsection, subsubsection, paragraph and subparagraph.

Setting the interlude level is slightly different. While setting it using interludelevel=⟨*level*⟩, instead of choosing one of LaTeX vanilla sectioning levels as the level, you may choose play, act or scene. It defines if an interlude is at the same level as a play, an act or a scene.

2.3 Sectioning

Here begin the parts explicitly relating to drama.

2.3.1 Levels

`\play`
`\play*`
`\act`
`\act*`
`\scene`
`\scene*`

To introduce a new play, act or scene, use commands `\play`, `\act` and `\scene`. Their behaviour is as close as the "usual" sectioning commands (`\chapter`, `\section` and so on) as possible, i.e.:

- Their signature is `\play[`⟨*short title*⟩`]{`⟨*longtitle*⟩`}` (the optional short title is the one used in the table of content, and in headers and footers).
- A starred version (`\play*`, `\act*` and `\scene*`) is provided, which inserts a play (or act, or scene) which is not numbered, and does not insert any line in the table of content.
- Headers and footers are changed (more information in section 2.3.4).

Both commands `\act` and `\scene` (and their starred versions) are designed to deal with empty titles. Indeed, it is common not to give any name to acts and scenes.

By default, a play is as deep (regarding to the table of contents) as a chapter, an act as a section, and a scene as a subsection. But this can be set using package options `playlevel`, `actlevel` and `scenelevel` (see packages options, page 4). That way, you can use in your document plays, acts and scenes as well as chapters, sections and so on. It can be useful if you want a foreword, and appendix, etc.

It is not compulsory to use all three commands `\play`, `\act` and `\scene`. The rule of thumb is: if only one element exists, skip the corresponding command: if your document has a single play, you should ignore `\play`; if your document has several single act plays, set `playlevel=section,scenelevel=subsection` and ignore `\act`; etc.

`\interlude`
`\interlude*`

You may want to use interludes, which are acts or scenes which are not numbered, but which should appear in the table of content. Command `\interlude[`⟨*short title*⟩`]{`⟨*long title*⟩`}` has this purpose.

You may choose the sectioning level an interlude is equivalent to in the package options. If your interludes are acts, use `interludelevel=act`; if they are scenes, use `interludelevel=scene`.

`\curtain`

At last, to mark the end of an act or of the play, you can use command `\curtain`, which prints the word `\curtainname` in the middle of its own line.

2.3.2 Title styles

Several styles are available to render play, act and scene titles. Choose them using package options `playstyle`, `actstyle` and `scenestyle`. Default is `playstyle=box`, `actstyle=bigcenter`, `scenestyle=center`.

Custom titles Unfortunately, as play, act and scene titles are not considered (by LaTeX) as usual sections, package `titlesec`[5] cannot be used to use alternative section titles. Here is the way to set your own one.

`\customplay`
`\customact`
`\customscene`

When loading the package, use `custom` as the style of the title you want to customize (e.g. `actstyle=custom`). Then, commands `\customact{`⟨*counter*⟩`}{`⟨*title*⟩`}` and `\customact*{`⟨*title*⟩`}` will be called by this package to render titles. You *must* define them. Figure 1 gives the example of the definition of the `center` style.

The first argument of `\customact` is the label of the act being printed (that is, `\theact` for an act, `\theplay` for a play, etc.), its second argument is its title. Command `\customact*` only has one argument, which is the act title.

2.3.3 Labels and counters

`\theplay`
`\theact`
`\thescene`

Using the same tools as `\chapter`, `\section` and so on, it is possible to define the way counters of plays, acts and scenes are displayed. You can do this by redefining `\theplay`, `\theact` and `\thescene`. For example, to have acts numbered using letters, use `\renewcommand{\theact}{\Alph{act}}`.

2.3.4 Headers and footers

`\playmark`
`\actmark`
`\scenemark`

Once again, similar tools as those used by `\section` are provided to deal with headers and footers. When introducing, a new play, act or scene, respectively, commands `\playmark{`⟨*label*⟩`}`, `\actmark{`⟨*label*⟩`}` and `\scenemark{`⟨*label*⟩`}` are called, so that titles can be used in headers and footers. If the default behaviour does not suit you (which should be the case if you did not choose the default option for `playlevel`, `actlevel` or `scenelevel`), you can redefine them.

2.4 Characters

This part explains how to define characters, introduce character's lines, and build and display the dramatis personæ.

2.4.1 Dramatis personæ

Definition of characters is done in document body. As it is possible to have several plays in a single documents (for a collection of plays or sketches), it is possible to define several dramatis personæ. A new one disables the character commands defined by the previous one.

`dramatis`

Definition of characters is done inside the `dramatis` environment. If the `hidden` option is given, the dramatis personæ is not printed (its only purpose is then to define the character commands).

```
\begin{dramatis}[⟨hidden⟩]
```

Then, several commands are available to define characters, and organize character definitions.

		empty ⟨*name*⟩		⟨*name*⟩	
		no ⟨*cmd*⟩	⟨*cmd*⟩	no ⟨*cmd*⟩	⟨*cmd*⟩
no ⟨*drama*⟩	no ⟨*desc*⟩			silent[iv]	default[i] hidden[iii]
	⟨*desc*⟩	description only[ii]		silent[iv]	default[i]
⟨*drama*⟩	no ⟨*desc*⟩	silent[iv]			default[i]
	⟨*desc*⟩	silent[iv]			default[i]

An empty cell means that the corresponding combination is forbidden. The superscript number refers to the list of special character definitions (page 7).

Table 4: (Not) defining arguments in character definition

`\characterspace` Command `\characterspace` put some vertical space into the dramatis personæ.

2.4.2 Character definition

`\character` **Basic definition** To define a character, use command `\character`.

`\character[`⟨cmd=*command*,drama=*dramatis*,desc=*description*⟩`]{`⟨*name*⟩`}`

The mandatory argument is the name of the character, as it will appear in each of this character's line. It is later possible to redefine it using command `\setcharactername` (see part 2.4.3). Optional arguments are:

desc is a description of your character, appearing in the dramatis personæ;

cmd is the name of the command that will be used to introduce this character's lines in the remaining part of your document;

drama is the name of your character, as it will appear in the dramatis personæ. The name of the character (mandatory argument) is used as a default value.

If *cmd* is defined, this command creates two new commands: `\`⟨*cmd*⟩ and `\`⟨*cmd*⟩`name`. The first one is used to introduce a character's line. The second one prints the character's name. An error is raised if a command with any of these two names already exists.

An example of the use of this command is given in figure 2.

Special character definition Although optional arguments are not mandatory, not defining them, or leaving the mandatory argument blank, have special meaning. The combination are summed up in table 4.

(i) **Default definition (⟨*name*⟩ and ⟨*cmd*⟩ are given; ⟨*desc*⟩ and ⟨*drama*⟩ may be omitted):** The character is defined as described in 2.4.2. If

[5] `http://www.ctan.org/pkg/titlesec`

```
\newcommand\customact[2]{
  \begin{center}
    \textsc{#1}

    #2
  \end{center}
}
\WithSuffix\newcommand\customact*[1]{\customact{}{#1}}
%
```

Figure 1: Example of custom act definition

```
\begin{dramatis}
  \character[cmd={cyrano}, drama={Cyrano de Bergerac}]{Cyrano}
  \character[cmd={lebret}]{Le Bret}
  \character[cmd={bellerose}]{Bellerose}
\end{dramatis}

\bigskip

\lebret[to \cyranoname, holding his arm]
Let's talk !

\cyrano
Wait for the crowd to leave.  \did{To \bellerosename} Can I stay?
```

Cyrano de Bergerac
Le Bret
Bellerose

Le Bret, *to Cyrano, holding his arm* : Let’s talk !
Cyrano : Wait for the crowd to leave. *(To Bellerose)* Can I stay?

Figure 2: Example of character definition

⟨*description*⟩ is omitted, no description appear in the dramatis personæ; if no ⟨*drama*⟩ is given, character in the dramatis personæ has the same name as it will have in the document. Example:

```
\character[drama={A ghost}, desc={the king's ghost},
           cmd={ghost}]{The ghost}
```

(ii) **Description only (everything omitted but ⟨*desc*⟩):** A description is inserted in the dramatis personæ. Useful to add characters such as *The kings' armies*. Example:

```
\character[desc={The kings' armies}]{}
```

(iii) **Hidden character (⟨*drama*⟩ is empty (defined, but empty), ⟨*cmd*⟩ and ⟨*name*⟩ are defined, ⟨*desc*⟩ is omitted)** Definition of a character that does not appear in the dramatis personæ. Example:

```
\character[drama={}, cmd={postman}]{The postman}
```

(iv) **Silent character (one of ⟨*name*⟩ or ⟨*drama*⟩ is defined; ⟨*desc*⟩ may be omitted; ⟨*cmd*⟩ is omitted):** The character only appear in the dramatis personæ. It will not be used elsewhere in the document. An optional description may also appear in the dramatis personæ. Example:

```
\character{A priest}
```

`charactergroup`

Group of characters It is possible to group several characters' definition if they have the same description. This is done with environment `charactergroup{`⟨*description*⟩`}`. For example, one can use code of figure 3 to define the three sons of another character.

The effect of this code will be, in the dramatis personæ, to have a nice brace mapping the three characters to their common description.

`\disposablecharacter`

Disposable character One can need to define characters that are used only once (or a few times). Command `\disposablecharacter` is here to help.

`\disposablecharacter[`⟨*directions*⟩`]{`⟨*name*⟩`}`

This command, used in the body, introduces a line for character `name`, with optional stage directions. It is rendered the same way regular characters are rendered, but nothing is added to the dramatis personæ. It is used in figure 4.

2.4.3 Changing character's name

`\setcharactername`

It is possible to change the name that appears to introduce character's lines within the text. This can be used, for example, when a character is first refered

[6] I admit I am cheating here. As it was difficult to find a single play that illustrates every feature of this package, this snippet, in the original *Cyrano de Bergerac*, is used to give the names of the actors playing the two musicians, which is a bit different from the purpose of the `charactergroup` environment. Still, you can see how it is rendered, which is the important part here.

Gaston Henry Damon	} Two musicians

```
\begin{dramatis}
\begin{charactergroup}{Two musicians}
  \character[desc={Gaston Henry}]{}
  \character[desc={Damon}]{}
\end{charactergroup}
\end{dramatis}
```

Figure 3: Defining a group of characters[6]

On the first scenes of Cyrano de Bergerac, the main character (Cyrano) is somewhere in a crowd, and only appears as *a voice*. He appears as Cyrano in the following verses.

```
\begin{dramatis}[hidden]
  \character[desc={Cyrano de Bergerac}, cmd=cyrano]{The voice}
  \character[cmd=montfleury]{Montfleury}
\end{dramatis}

\cyrano Leave!

\disposablecharacter{The crowd} Oh!

\montfleury[losing his voice] ``Happy he who far away from court\ldots''

\setcharactername{cyrano}{Cyrano}

\cyrano[emerging from the crowd, standing up on a chair {[}\ldots{]} with
    a terrible nose] Ah! I am getting angry!\ldots
```

The voice : Leave!
The crowd : Oh!
Montfleury, *losing his voice* : “Happy he who far away from court. . . ”
Cyrano, *emerging from the crowd, standing up on a chair [. . .] with a terrible nose* : Ah! I am getting angry!. . .

Figure 4: Example of changing the name of a character

to as *A sailor*, until we learn his real identity, which is *Liz's father*. An example is given in figure 4.

```
\setcharactername{⟨command⟩}{⟨name⟩}
```

This command takes two mandatory arguments. The first one, `command`, is the command used to introduce this character's line. The second one is the name to display for this character.

2.4.4 Lines

To introduce characters' lines, use the commands defined in the character definition (see the `\character` command, or the example in figure 2).

Choose style Several styles are available, to typeset character's name and lines in different ways. Change the style by loading package using option `characterstyle={⟨style⟩}` (available styles are `bold`, `center`, `margin`, `simple`, `imprimerie-verse`, `imprimerie-prose`). Note that `imprimerie-verse` and `imprimerie-prose` styles are French *de facto* standards for typesetting drama plays, respectively in verse and in prose, as defined by the *Imprimerie nationale* [1], and `arden` tries to mimick typesetting of the Arden Shakespeare series. Default style is `simple`.

`\speakswithoutdirection`
`\speakswithdirection`

Customize style If available styles does not fit your need, you can also define your own one. To do so, you can redefine the following commands.

```
\speakswithoutdirection{⟨name⟩}
\speakswithdirection{⟨name⟩}{⟨direction⟩}
```

The first one (`\speakswithoutdirection`) is invoked to display a character's name to introduce its line. It takes one argument, which is the character's line. The second one (`\speakswithdirection`) is also invoked to display a character's name to introduce its line, but it takes a second argument, which is stage directions to be printed together with character's name.

See also section 2.5 to see other ways to print stage directions.

2.5 Stage directions

Let us begin with a warning: since we could not figure out a nice command name made from "stage directions", we used the French word *(didascalie)* to build up command names.

`\did`
`dida`

We define two ways to render stage directions: a short one, that is printed in-line, and a long one, printed in its own paragraph. They act a bit like `$\formula$` and `\[\formula\]` for formulas. The first way is the command `\did{⟨directions⟩}`. The second way is the environment `dida`.

`\onstage`

Some stage directions are displayed centered under the scene or act definition (I often see it in classical drama, where the list of characters appearing in each scene is displayed that way). Command `\onstage{`⟨*directions*⟩`}` can be used to display such information.

`\pause` As we often need to mark pauses in theater, we created the command `\pause`, which is a shortcut for `\did{\pausename}`.

3 Implementation

3.1 Required packages

Loading some packages.

```
\RequirePackage{ifthen}
% Chapters, sections, etc
\RequirePackage{etoolbox}
\RequirePackage{suffix}
% Dramatis person\ae{} is written using a table. Package "longtable" is used to be
% able to write it on several pages.
\RequirePackage{longtable}
% Handling spaces after having introduces characters' lines.
\RequirePackage{xspace}
```

3.2 Package options

Manage package options.

```
\RequirePackage{pgfopts}

\pgfkeys{
  % Character style
  /THALIE/.cd,
  characterstyle/.value required,
  characterstyle/.default=simple,
  characterstyle/.is choice,
  characterstyle/bold/.code=\def\@characterstyle{bold},
  characterstyle/center/.code=\def\@characterstyle{center},
  characterstyle/margin/.code=\def\@characterstyle{margin},
  characterstyle/simple/.code=\def\@characterstyle{simple},
  characterstyle/arden/.code=\def\@characterstyle{arden},
  characterstyle/imprimerie-verse/.code=\def\@characterstyle{imprimerie-verse},
  characterstyle/imprimerie-prose/.code=\def\@characterstyle{imprimerie-prose},
  characterstyle,
}

\pgfkeys{
  % play style
  /THALIE/.cd,
  playstyle/.value required,
  playstyle/.default=box,
  playstyle/.is choice,
  playstyle/center/.code=\def\@playstyle{center},
  playstyle/bigcenter/.code=\def\@playstyle{bigcenter},
  playstyle/box/.code=\def\@playstyle{box},
  playstyle/custom/.code=\def\@playstyle{custom},
  playstyle,
}
\pgfkeys{
  % act style
  /THALIE/.cd,
  actstyle/.value required,
  actstyle/.default=bigcenter,
  actstyle/.is choice,
  actstyle/center/.code=\def\@actstyle{center},
  actstyle/bigcenter/.code=\def\@actstyle{bigcenter},
  actstyle/box/.code=\def\@actstyle{box},
  actstyle/custom/.code=\def\@actstyle{custom},
  actstyle,
}
\pgfkeys{
  % scene style
  /THALIE/.cd,
  scenestyle/.value required,
  scenestyle/.default=center,
  scenestyle/.is choice,
  scenestyle/center/.code=\def\@scenestyle{center},
  scenestyle/bigcenter/.code=\def\@scenestyle{bigcenter},
  scenestyle/box/.code=\def\@scenestyle{box},
  scenestyle/custom/.code=\def\@scenestyle{custom},
  scenestyle,
}

\pgfkeys{
  % play level
  /THALIE/.cd,
  playlevel/.value required,
  playlevel/.default=chapter,
  playlevel/.store in=\@playlevel,
  playlevel,
}
\pgfkeys{
  % act level
  /THALIE/.cd,
  actlevel/.value required,
  actlevel/.default=section,
  actlevel/.store in=\@actlevel,
  actlevel,
}
\pgfkeys{
  % scene level
  /THALIE/.cd,
  scenelevel/.value required,
  scenelevel/.default=subsection,
  scenelevel/.store in=\@scenelevel,
  scenelevel,
}
\pgfkeys{
  % interlude level
  /THALIE/.cd,
  interludelevel/.value required,
  interludelevel/.default=act,
  interludelevel/.is choice,
  interludelevel/play/.code=\def\@interludelevel{play},
  interludelevel/act/.code=\def\@interludelevel{act},
  interludelevel/scene/.code=\def\@interludelevel{scene},
  interludelevel,
}

\ProcessPgfPackageOptions{/THALIE}
```

3.3 Localization

\playname \actname \scenename \interludename \curtainname \pausename

Definition of the commands used for localization. The *only* way to choose the language to use is by loading the `babel` package before loading this one.

```
\newcommand{\playname}{Play}
\newcommand{\actname}{Act}
\newcommand{\scenename}{Scene}
\newcommand{\interludename}{Interlude}
\newcommand{\curtainname}{Curtain}
\newcommand{\pausename}{Pause}
\@ifpackageloaded{babel}{
    \addto\captionsfrench{%
      \renewcommand{\playname}{Pi\`ece}
      \renewcommand{\actname}{Acte}
      \renewcommand{\scenename}{Sc\`ene}
      \renewcommand{\interludename}{Interm\`ede}
      \renewcommand{\curtainname}{Rideau}
      \renewcommand{\pausename}{Pause}
    }
    \addto\captionsenglish{%
      % Useless: these are the default...
    }
}{}
```

3.4 Sectioning

3.4.1 Headears, footers, counters, etc.

playmark actmark scenemark

Define commands `\playmark`, `\actmark` and `\scenemark`, which are involved in headers and footers definition.

```
\newcommand{\playmark}[1]{%
```

```
  \markboth{\MakeUppercase{#1}}{}%
}
\newcommand{\actmark}[1]{%
  \markright{\MakeUppercase{#1}}%
}
\newcommand{\scenemark}[1]{%
}
```

`theplay`
`theact`
`thescene`

Defines counters for plays, acts and scenes, and the associated labels (`\theplay`, `\theact`, `\thescene`).

```
\newcounter{play}
\renewcommand{\theplay}{\arabic{play}}
\newcounter{act}[play]
\renewcommand{\theact}{\Roman{act}}
\newcounter{scene}[act]
\renewcommand{\thescene}{\arabic{scene}}
```

3.4.2 Styles

Style difinition. Command `\@displaytitle` is later used by commands `\play`, `\act` and `\scene` (and their starred version) to typeset the title.

```
\newcommand{\@displaytitle}[3]{
  % Arguments:
  % - Style
  % - Label (none = not in toc)
  % - Title
  \ifthenelse{\equal{#1}{center}}{
    \begin{center}
      \textsc{#2}

      #3
    \end{center}
  }{\ifthenelse{\equal{#1}{bigcenter}}{
    \begin{center}
      \Large
      \textsc{#2}

      #3
    \end{center}
  }{\ifthenelse{\equal{#1}{box}}{
    \begin{center}
      \framebox{\begin{minipage}{0.7\textwidth}
      \begin{center}
        \Large \bfseries
      \vspace{0.5em}

      #2
      \ifthenelse{\equal{#3}{} \OR \equal{#2}{}}{}{---}
      #3

      \vspace{0.5em}
    \end{center}
    \end{minipage}}
    \end{center}
    \vspace{1em}
  }{}}}
}
```

3.4.3 Sectioning commands

Some general commands to handle clearing pages, and table of contents.

```
\newcommand\@clearpage[1]{%
  % Clear page if necessary
  \ifthenelse{\equal{#1}{part} \OR \equal{#1}{chapter}}{
    \cleardoublepage
    \thispagestyle{empty}
  }{}
}

```

\play
\play*

Define sectioning commands to introduce plays. As for \section (and other) sectioning command, the starred version does the same, excepted that nothing is written in the table of content.

```
\newcommand{\play}[2][]{%
  \refstepcounter{play}
  \ifthenelse{\equal{#1}{}}{
    \def\@short{#2}
  }{
    \def\@short{#1}
  }
  \@clearpage{\@playlevel}
  \playmark{\@short}
  \addcontentsline{toc}{\@playlevel}{\@short}
  \ifthenelse{\equal{\@playstyle}{custom}}{
    \customplay{\theplay}{#2}
  }{
    \@displaytitle{\@playstyle}{}{#2}
  }
}
\WithSuffix\newcommand\play*[1]{%
  \@clearpage{\@playlevel}
  \ifthenelse{\equal{\@playstyle}{custom}}{
    \customplay*{#1}
  }{
    \@displaytitle{\@playstyle}{}{#1}
  }
}
```

\act
\act*

Define commands to introduce acts.

```
\newcommand{\act}[2][]{%
  \refstepcounter{act}
  \ifthenelse{\equal{#1}{}}{
    \def\@short{#2}
  }{
    \def\@short{#1}
  }
  \ifthenelse{\equal{\@short}{}}{
    \def\@label{\actname{} \theact{}}
  }{
    \def\@label{\actname{} \theact{}\xspace: }
  }
  \@clearpage{\@actlevel}
  \actmark{\@label\@short}
  \addcontentsline{toc}{\@actlevel}{\@label\@short}
  \ifthenelse{\equal{\@actstyle}{custom}}{
    \customact{\theact}{#2}
  }{
    \@displaytitle{\@actstyle}{\actname{} \theact}{#2}
  }
}
\WithSuffix\newcommand\act*[1]{%
  \@clearpage{\@actlevel}
  \ifthenelse{\equal{\@actstyle}{custom}}{
    \customact*{#1}
  }{
    \@displaytitle{\@actstyle}{}{#1}
  }
}
```

\scene
\scene*

Define commands to introduce scenes.

```
\newcommand{\scene}[2][]{%
  \refstepcounter{scene}
  \ifthenelse{\equal{#1}{}}{
    \def\@short{#2}
  }{
    \def\@short{#1}
  }
  \ifthenelse{\equal{\@short}{}}{
    \def\@label{\scenename{} \thescene{}}
  }{
    \def\@label{\scenename{} \thescene{}\xspace: }
  }
  \@clearpage{\@scenelevel}
  \scenemark{\@label\@short}
  \addcontentsline{toc}{\@scenelevel}{\@label\@short}
  \ifthenelse{\equal{\@scenestyle}{custom}}{
    \customscene{\thescene}{#2}
  }{
    \@displaytitle{\@scenestyle}{\scenename{} \thescene}{#2}
  }
}
\WithSuffix\newcommand\scene*[1]{%
  \@clearpage{\@scenelevel}
  \ifthenelse{\equal{\@scenestyle}{custom}}{
    \customscene*{#1}
  }{
    \@displaytitle{\@scenestyle}{}{#1}
  }
}
```

\interlude \interlude*

Define commands to introduce interludes.

```
\newcommand{\interlude}[2][]{%
  \ifthenelse{\equal{#1}{}}{
    \def\@short{#2}
  }{
    \def\@short{#1}
  }
  \ifthenelse{\equal{\@short}{}}{
    \def\@label{\interludename{}}
  }{
    \def\@label{\interludename{}\xspace: }
  }
  \ifthenelse{\equal{\@interludelevel}{play}}{
    \@clearpage{\@playlevel}
    \playmark{\@label\@short}
    \addcontentsline{toc}{\@playlevel}{\@label\@short}
    \@displaytitle{\@playstyle}{\interludename}{#2}
  }{\ifthenelse{\equal{\@interludelevel}{act}}{
    \@clearpage{\@actlevel}
    \actmark{\@label\@short}
    \addcontentsline{toc}{\@actlevel}{\@label\@short}
    \@displaytitle{\@actstyle}{\interludename}{#2}
  }{% \@interludelevel is scene
    \@clearpage{\@scenelevel}
    \scenemark{\@label\@short}
    \addcontentsline{toc}{\@scenelevel}{\@label\@short}
    \@displaytitle{\@scenestyle}{\interludename}{#2}
  }}
}
\WithSuffix\newcommand\interlude*[1]{%
  \ifthenelse{\equal{\@interludelevel}{play}}{
    \@clearpage{\@playlevel}
    \@displaytitle{\@playstyle}{\interludename}{#1}
  }{\ifthenelse{\equal{\@interludelevel}{act}}{
    \@clearpage{\@actlevel}
    \@displaytitle{\@actstyle}{\interludename}{#1}
  }{% \@interludelevel is scene
    \@clearpage{\@scenelevel}
    \@displaytitle{\@scenestyle}{\interludename}{#1}
  }}
}
```

3.4.4 Curtain

\curtain

Used to mark the end of an act. Prints “curtain” centered in its own line.

```
% Curtain
\newcommand\curtain{
  \begin{center}
    \sffamily\LARGE\bfseries \textsc{\curtainname}
  \end{center}
}
```

3.5 Characters

3.5.1 Characters’ line

Define the command `\@speaks`, wich introduce a character’s line. It is not meant to be used by user, but will be used by further commands. It takes two arguments: the character’s name, and optionally, a stage direction.

```
\newcommand{\@speaks}[2][]{%
  \ifthenelse{\equal{#1}{}}{%
    \speakswithoutdirection{#2}%
  }{%
    \speakswithdirection{#2}{#1}%
  }\xspace%
}
```

\speakswithdirection
\speakswithoutdirection

Definition of styles for introducing characters’ lines. These commands can be overloaded by user.

```
\ifthenelse{\equal{\@characterstyle}{bold}}{%
  % Bold style
  \newcommand\speakswithdirection[2]{%
    \noindent%
    {\bfseries\sffamily #1} \emph{(#2)}\xspace:%
  }
  \newcommand\speakswithoutdirection[1]{%
    \noindent%
    {\bfseries\sffamily #1\xspace:}%
  }%
}{}%
\ifthenelse{\equal{\@characterstyle}{center}}{%
  % Center style
  \newcommand\speakswithdirection[2]{%
    \begin{center}%
    \textsc{#1},\\\emph{#2}%
    \end{center}%
  }%
  \newcommand\speakswithoutdirection[1]{%
    \begin{center}%
    \textsc{#1}%
    \end{center}%
  }%
}{}%
\ifthenelse{\equal{\@characterstyle}{imprimerie-verse}}{%
  % Style for verse plays defined by the French Imprimerie nationale
  \newcommand\speakswithdirection[2]{%
    \begin{center}%
    \textsc{#1}, \emph{#2}%
    \end{center}%
  }%
  \newcommand\speakswithoutdirection[1]{%
    \begin{center}%
    \textsc{#1}%
    \end{center}%
  }%
}{}%
\ifthenelse{\equal{\@characterstyle}{imprimerie-prose}}{%
  % Style for prose plays defined by the French Imprimerie nationale
  \newcommand\speakswithdirection[2]{%
    \noindent\hspace*{-\parindent}\textsc{#1}, \emph{#2}\xspace:%
  }%
  \newcommand\speakswithoutdirection[1]{%
    \noindent\hspace*{-\parindent}\textsc{#1}\xspace:%
  }%
}{}%
\ifthenelse{\equal{\@characterstyle}{arden}}{%
  \newcommand\speakswithdirection[2]{%
    \noindent\hspace*{-\parindent}\textsc{\MakeLowercase{#1}} [\emph{#2}]\quad%
  }%
  \newcommand\speakswithoutdirection[1]{%
    \noindent\hspace*{-\parindent}\textsc{\MakeLowercase{#1}}\quad%
  }%
}{}%
\ifthenelse{\equal{\@characterstyle}{simple}}{%
  % Simple style
  \newcommand\speakswithdirection[2]{%
    \indent\textsc{#1}, \emph{#2}\xspace:%
  }%
  \newcommand\speakswithoutdirection[1]{%
    \indent\textsc{#1}\xspace:%
  }%
}{}%
\ifthenelse{\equal{\@characterstyle}{margin}}{%
  % Margin style
  \setlength{\leftskip}{3cm}
  \newcommand\speakswithdirection[2]{%
    \hspace{-3cm} #1 #2
  }
  \newcommand\speakswithoutdirection[1]{%
    \hspace{-3cm} #1
  }%
}{}%
```

3.5.2 Dramatis personæ

dramatis In this environment are defined characters.

```
\newcommand{\@dramatis@clear}{}
\provideboolean{@dramatis@hidden}
\newenvironment{dramatis}[1][]{
  \@dramatis@clear{}
  \undef{\@dramatis@clear}
  \undef{\@dramatis@hook}
  \newcommand{\@dramatis@hook}{}
  \ifthenelse{\equal{#1}{hidden}}{
    \setboolean{@dramatis@hidden}{true}
  }{\ifthenelse{\equal{#1}{}}{
    \setboolean{@dramatis@hidden}{false}
    \gappto{\@dramatis@hook}{\begin{longtable}{l}}
  }{
    \ClassError{thalie}{%
      Environment dramatis does not accept option #1.%
    }{
    }
  }
  }
}{%
  \ifthenelse{\boolean{@dramatis@hidden}}{%
  }{%
    \gappto{\@dramatis@hook}{\end{longtable}}
    \@dramatis@hook{}%
  }
}
```

Generic character output

```
\newcommand{\@character}[2]{%
  \ifthenelse{\( \equal{#1}{} \) \AND \( \equal{#2}{} \)}{%
    ~%
  }{}%
  \ifthenelse{\( \NOT \equal{#1}{} \) \AND \( \equal{#2}{} \)}{%
    #1%
  }{}%
  \ifthenelse{\( \equal{#1}{} \) \AND \( \NOT \equal{#2}{} \)}{%
    #2%
  }{}%
  \ifthenelse{\( \NOT \equal{#1}{} \) \AND \( \NOT \equal{#2}{} \)}{%
    #1, #2%
  }{}%
  \tabularnewline
}
```

\characterspace Add space in the dramatis personæ.

```
\newcommand{\characterspace}{%
  \gappto{\@dramatis@hook}{\@character{}{~}}
}
```

charactergroup Gathers definition of characters that share the same description.

```
\newenvironment{charactergroup}[1]{%
  \gdef\@groupname{#1}
  \gappto{\@dramatis@hook}{
    \hspace{-7.2pt}\begin{math}\left.
    \begin{tabular}{ll}
  }
}{%
  \gappto{\@dramatis@hook}{
    \end{tabular}
    \right\} \end{math}
  }
  \protected@xappto{\@dramatis@hook}{\@groupname}
  \gappto{\@dramatis@hook}{\tabularnewline}
}
```

3.5.3 Character definitions

\setcharactername Set (or change) the name used to introduce the lines of a character.

```
% The tough part of this code is to deal with optional argument.
\newcommand{\setcharactername}[2]{%
    \expandafter\gdef\csname#1name\endcsname{%
      #2\xspace%
    }%
    \expandafter\gdef\csname#1\endcsname{%
      \@ifnextchar[{%
        \defcharcommand@with{#2}%
      }{%
        \defcharcommand@without{#2}%
      }%
    }%
    \xappto{\@dramatis@clear}{%
      \global\noexpand\csundef{#1}%
      \global\noexpand\csundef{#1name}%
    }%
}
```

Define the command used to introduce a character's line. It takes two arguments: the first one is the name of the command to define, and the second one is the name of the character corresponding to this command.

```
\newcommand{\@definecharactercommand}[2]{%
  \@ifundefined{#1}{%
  }{%
    \ClassError{thalie}{%
      A command named \@backslashchar#1 already exists. We cannot define a new
      one.%
    }{%
      Choose another command name to introduce character #2's lines.%
    }%
  }%
  \@ifundefined{#1name}{%
  }{%
    \ClassError{thalie}{%
      A command named \@backslashchar#1name already exists. We cannot define a
      new one.%
    }{%
      Choose another command name to introduce character #2's lines, such that
      when a new command is defined by adding "name" to it, it does not
      conflict with an existing one.
    }%
  }%
  \setcharactername{#1}{#2}%
}
\def\defcharcommand@with#1[#2]{\@speaks[#2]{#1}}
\def\defcharcommand@without#1{\@speaks{#1}}
```

`\character` Define a character: put it in the dramatis personæ, and define corresponding commands.

```
\pgfkeys{
  % Character definition
  /CHARACTER/.is family, /CHARACTER,
  cmd/.value required,
  cmd/.store in=\@cmd,
  drama/.value required,
  drama/.store in=\@drama,
  desc/.value required,
  desc/.store in=\@desc,
}
\newcommand{\character}[2][]{
  \undef{\@drama}
  \undef{\@cmd}
  \undef{\@desc}
  \pgfkeys{/CHARACTER, #1}%
 % Forbidden combinations
  \ifthenelse{
    \( \( \NOT \isundefined{\@cmd} \) \AND \equal{#2}{} \) \OR
    \( \equal{#2}{} \AND \isundefined{\@cmd} \AND \isundefined{\@desc} \AND \isundefined{\@dram
    \( \isundefined{\@cmd} \AND \( \NOT \equal{#2}{} \) \AND \( \NOT \isundefined{\@drama} \) \
  }{
    \ClassError{thalie}{Invalid character definition.}{All combination of omitted arguments are
  }{
    % Defining character command
    \ifthenelse{\( \NOT \isundefined{\@cmd} \) \AND \( \NOT \equal{#2}{} \)}{
      \@definecharactercommand{\@cmd}{#2}
    }{
    }
    \ifthenelse{
      \( \NOT \equal{#2}{} \) \AND \( \NOT \isundefined{\@cmd} \) \AND \( \NOT \isundefined{\@d
    }{
      \ifthenelse{\equal{\@drama}{}}{
      % Hidden character. Nothing added to dramatis personae
      }{
      % Populating dramatis personae
        \protected@xappto{\@dramatis@hook}{\noexpand\@character{\@drama}{}}
      }
    }{
      % Populating dramatis personae
      \@ifundefined{@desc}{\def\@desc{}}{}
      \@ifundefined{@drama}{\def\@drama{#2}}{}
      \protected@xappto{\@dramatis@hook}{\noexpand\@character{\@drama}{\@desc}}
    }
  }
}
```

`\disposablecharacter` Disposable character (character used only a few times, defined on-the-fly, that does not appear in the dramatis personæ).

```
\newcommand{\disposablecharacter}[2][]{%
  \@speaks[#1]{#2}%
}
```

3.6 Stage directions

`\onstage` Centered stage direction.

```
\newcommand{\onstage}[1]{{\centering \emph{#1}\par\medskip}}
```

`\did` Inline stage directions.

```
\newcommand{\did}[1]{\emph{(#1)} }
```

`dida` Bigger stage directions, in its own paragraph.

```
\newenvironment{dida}{%
  \begin{quote}
  \begin{em}
  }{%
  \end{em}
  \end{quote}
}
```

`\pause` Shortcut for `\did{\pausename}`.

```
\newcommand\pause{\did{\pausename}}
```

Change History

v0.5

General: First published version. 24

v0.6

General: Added character styles `imprimerie-verse`, `imprimerie-prose` and `arden`. 11

`charactergroup`: Groups of characters are now aligned with other caracters. 22

Wish list

Verse

I would like to be able to render drama plays in verse, where a verse can be continued on the next line if relevant (as shown on figure 5). But I could not find a way to do this, which would not be too complicated for the user, and which I am able to implement.

References

[1] Imprimerie nationale, *Lexique des règles typographiques en usage à l'Imprimerie nationale*, 2002, ISBN : 978-2-7433-0482-9

[2] Edmond Rostand, *Cyrano de Bergerac*, 1897

[3] Scott Pakin, *How to Package Your LaTeX Package — Tutorial on writing .dtx and .ins files*, 2004, `http://www.ctan.org/pkg/dtxtut/`

[4] The LaTeX Team, *LaTeX2ε for class and package writers*, 2006, `http://www.ctan.org/pkg/clsguide`

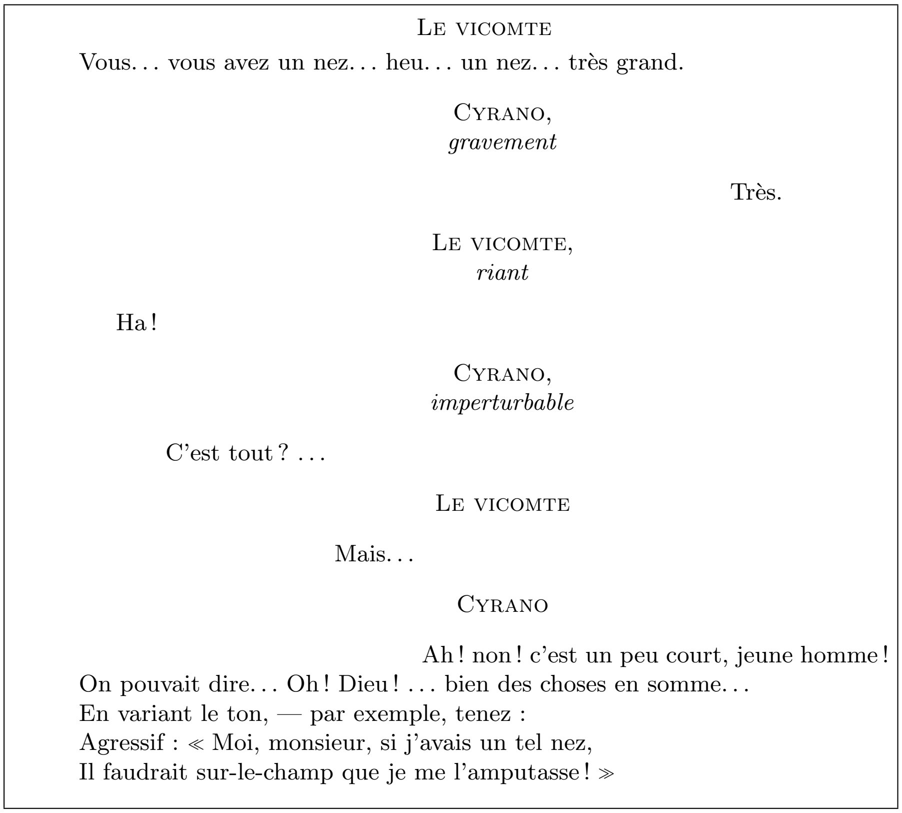
[5] Matt Swift, *drama — Production-style stage script in LaTeX*, 2001, `http://www.ctan.org/pkg/drama`

[6] Massimiliano Dominici, *dramatist — Typeset dramas, both in verse and in prose*, 2005, `http://www.ctan.org/pkg/dramatist`

[7] James Kilfiger, *play — Typeset drama using LaTeX*, 2001, `http://www.ctan.org/pkg/play`

[8] John Pate, *screenplay — A class file to typeset screenplays*, 2012, `http://www.ctan.org/pkg/screenplay`

[9] Wing L Mui, *sides — A LaTeX class for typesetting stage plays*, 2005, `http://www.ctan.org/pkg/sides`



Le vicomte

Vous… vous avez un nez… heu… un nez… très grand.

Cyrano,
gravement

Très.

Le vicomte,
riant

Ha !

Cyrano,
imperturbable

C'est tout ? …

Le vicomte

Mais…

Cyrano

Ah ! non ! c'est un peu court, jeune homme !
On pouvait dire… Oh ! Dieu ! … bien des choses en somme…
En variant le ton, — par exemple, tenez :
Agressif : ≪ Moi, monsieur, si j'avais un tel nez,
Il faudrait sur-le-champ que je me l'amputasse ! ≫

From *Cyrano de Bergerac*, by Edmond Rostand (act I, scene 4).

Figure 5: Wished verse rendering

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.