TASKS-Macro

Yongxue

December 20, 2020

Contents

To be continued (*Basics<English Version>*)

1 Motivation

Originally TASKS has been an integral part of the ExSheets package [Nie19]. However, users told me that it indeed could be useful to have it as a stand-alone package not having to load the whole ExSheets beast just for having the TASKS environment available. Since I agree with this the environment has been extracted into a package if its own, tasks. Since then TASKS has been distributed as a package of its own but as part of the ExSheets bundle. With v0.10 I decided to make it a completely independent package. So the relation to ExSheets only is a historical one.
The reason for the tasks environment is an unwritten agreement in German maths textbooks (exspecially in (junior) high school textbooks) to organize exercises in columns counting horizontally rather than vertically. That is what tasks primarily is for. If you don’ t need this feature you’ re better of using traditional LaTeX lists and the enumitem package for customization.

Changes

- the option counter-format is deprecated. Labels can now be set quite similar to the way they are set in enumitem. This also made the enumerate option of the list template superfuous which has been removed accordingly.
- The commands \NewTasks and \RenewTasks have been renamed.
- The multiple choice lists have been removed.
- Custom defnitions can be put in a tasks.cfg fle which is automatically loaded if available.

2 License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lppl), version 1.3c or later (http://www.latex-project.org/ lppl.txt). The software has the status “maintained.”
TASKS requires the l3kernel [L3P] bundle, xparse1 and xtemplate.

3 How it works

3.1 Background

The TASKS environment is similar to a list like enumerate but not the same. Here are some of the diferences:

1. There is no pagebreak possible inside an item but only between items.
2. The enumeration default is a), b), c)
3. The body of the tasks environment is split at every occurrence of the item separator. For this reason the default separator is not \item but \task so it is unique to this environment only. This directly leads to ...
4. The fact that the tasks environment cannot be nested. You can, however, use an itemize environment or another "real" list in it.
5. A ffth diference: verbatim material cannot be used in it. You' ll have to use \string,
6. `\texttt` or \detokenize. If this won' t sufce then don' t use TASKS

The environments of TASKS are what I like to call "pseudo-environments". This means like environments defned by the package environ [Rob14] the body of the environment is read as argument before it is processed. This is why verbatim material cannot be used in TASKS' lists.

3.2 The Basics

`\begin{tasks}[<options>](<num of columns>)`
List like environment where the single items are introduced with \task.
Let' s see an example:
The environment takes the optional argument (<num of columns>) with which the number of columns used by the environment is specifed.

3.3 Items Spanning More Than One Column

Sometimes it may come in handy if an item is allowed to span more than one column. TASKS introduced in v0.10 supports items using the remaining space by adding an optional star to \task:

```
% \Sample is defined to contain some sample text:
% \def\sample{This is some sample text we will use to create a somewhat
% longer text spanning a few lines.}
% \def\Sample{\sample\ \sample\par\sample}
Some text before the list.
\begin{tasks}
\task \Sample
\task \Sample
\task \Sample
\end{tasks}
And also some text after it.
```

a) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

   This is some sample text we will use to create a somewhat longer text spanning a few lines.

b) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

   This is some sample text we will use to create a somewhat longer text spanning a few lines.

c) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

   This is some sample text we will use to create a somewhat longer text spanning a few lines.

And also some text after it.

```latex
\begin{tasks}(2)
\task \Sample
\task \sample\ \sample
\task \sample
\task \Sample
\task \sample\par\sample
\end{tasks}
```

a) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

b) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

c) This is some sample text we will use to create a somewhat longer text spanning a few lines.

d) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

e) This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

3.4 Items Spanning More Than One Column

Sometimes it may come in handy if an item is allowed to span more than one column. tasks introduced in v0.10 supports items using the remaining space by adding an optional star to \task.

```
\begin{tasks}(3)
\task \sample
\task * \sample
\task * \sample
\task \sample
\task \sample
\end{tasks}
```

a) This is some sample text we will use to create a somewhat longer text spanning a few lines.

b) This is some sample text we will use to create a somewhat longer text spanning a few lines.

c) This is some sample text we will use to create a somewhat longer text spanning a few lines.

d) This is some sample text we will use to create a somewhat longer text spanning a few lines.

e) This is some sample text we will use to create a somewhat longer text spanning a few lines.

TSAKS also supports items that span all columns in any case by adding an optional bang to \task.

```
\begin{tasks}(3)
\task \sample \task! \sample \task! \sample
\task \sample \task \sample
\end{tasks}
```

a) This is some sample text we will use to create a somewhat longer text spanning a few lines.

b) This is some sample text we will use to create a somewhat longer text spanning a few lines.

c) This is some sample text we will use to create a somewhat longer text spanning a few lines.

d) This is some sample text we will use to create a somewhat longer text spanning a few lines.

e) This is some sample text we will use to create a somewhat longer text spanning a few lines.

The optional star has itself an optional argument with parentheses where you can specify the number of columns the item is supposed to span:

```
\settasks{debug}
\begin{tasks}(4)
\task the first
\task the second
\task the third
\task the fourth
\task * (3) the fifth item is way too long for this and needs three columns
\task the sixth
\task the seventh
\task * (2) the eighth item is way too long for this and needs two columns
\task the nineth
\task the tenth
\end{tasks}
```

```
    a) the first        b) the second       c) the third
d) the fourth

    e) (3) the fifth item is way too long for this and needs three columns

    f) the sixth        g) the seventh      h) (2) the eighth item is way too long for
                                               this and needs two columns

    i) the nineth       j) the tenth
```

If there are not enough columns left (say two columns but you said \task * (3)) the argument is ignored and the maximum number of remaining columns is used (two in case of our example). Both optional star and optional bang can be combined with the optional argument for a custom label:

```
 \begin{tasks}(3)
 \task \sample
 \task * \sample
\task * [(x)] \sample
\task \sample
\task \sample
\end{tasks}
```

a) This is some sample text we will use to create a somewhat longer text spanning a few lines.

b) This is some sample text we will use to create a somewhat longer text spanning a few lines.

c) [(x)] This is some sample text we will use to create a somewhat longer text spanning a few lines.

d) This is some sample text we will use to create a somewhat longer text spanning a few lines.

e) This is some sample text we will use to create a somewhat longer text spanning a few lines.

Forcing a new item line manually is also possible introduced in v0.9 (2013/04/07) using the following command:
\startnewitemline
Introduce a new line in a tasks environment.
While this works it also needs a bit of care since the width of the items doesn’t change which means in order to use the full width you’d have to use trickery like \rlap which then means the danger of the item text sticking into the margin.

```
\begin{tasks}(4)
\task the first
\task the second
\task the third
\task the fourth
\task \rlap{the fifth item is way too long for this so we start a new row
    }
\startnewitemline
\task the sixth
\task the seventh
\task \rlap{the eighth item also is too long} \startnewitemline
\task the nineth
\task the tenth
\end{tasks}
```

a) the first b) the second c) the third
d) the fourth

e) the fifth item is way too long for this so we start a new row

f) the sixth g) the seventh h) the eighth item also is too long

i) the nineth j) the tenth

4 Available Options

The tasks package does not have any package options.
The environment tasks has a number of options, though, namely the following ones that can be set using a setup command:
\settasks<options>
Setup command for tasks.
style = <instance> (initially empty)
Choose the instance to be used. Read more on this in section 8.1.
label-format = <code> (initially empty)
Can be used to apply a formatting like, e.g., \bfseries to the labels. This may be code accepting the item as mandatory argument.

label = <code> Default: `\alph *`)

Sets a custom label. The * is replaced by task . This is heavily inspired by enumitem’ s [Bez19] label option.

ref = <code> (initially empty)

New Works like label but sets the output of the reference by setting \the<counter>(\thetask in the default setting). label-width = <dim> Default: 1em

Sets the width of the item labels. label-offset = <dim> Default: .3333em

Sets the ofset, i.e., the distance between label and item. item-format = <code>(initially empty)

Can be used to apply a formatting like, e.g., `\bfseries` to the items. This may be code accepting the item as mandatory argument.

item-indent = <dim> Default: 2.5em

The indent of an item, i.e., the horizontal space available for both label and label-ofset. If

$$\text{indent} = \text{labelwidth} + \text{label-offset}$$

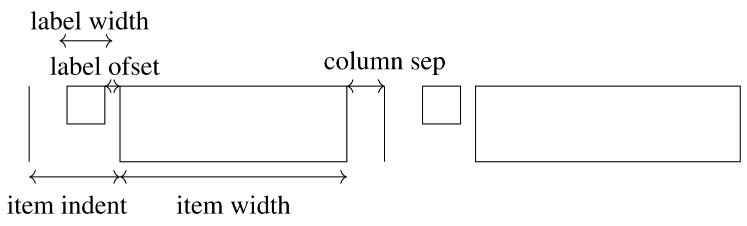

Figure 1: A visual representation of the used lengths.

the label will align with the textblock above (if label-align = {left} is set). Please see fgure 1 for a sketch of the available lengths and how they are set.

column-sep = <dim> Default: 0pt

A horizontal length that is inserted between columns ot items. label-align = left |*right*| center Default: left

Determine show the labels are aligned with in the label-box whose width is set with label-width . before-skip = {<skip>} Default: 0pt

Sets the skip before the list. after-skip = {<skip>} Default: 0pt Sets the skip after the list.

after-item-skip = {<skip>} Default: 1ex plus 1ex minus 1ex

This vertical skip is inserted between rows of items.

resume = true|false Default: false

The enumeration will resume from a previous tasks environment. In order to use this option properly you shouldn’ t mix diferent tasks environments that both count their items. start ={<integer>} Default: 1

Set the starting value with which the list starts counting.

counter = {<counter>} Default: task

The counter to be used to count the items. debug = true|false Default: false

If set to true \fboxsep is set to 0pt inside the tasks environment and \fbox is used to draw a frame around the label boxes and the item boxes.

Now the same list as above but with three columns and a diferent label:

```
\begin{tasks}[label=(\roman * ),label-width=4ex](2)
\task \Sample
\task \sample
\task \sample
\task \Sample
\task \sample
\end{tasks}
```

() This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

() This is some sample text we will use to create a somewhat longer text spanning a few lines.

() This is some sample text we will use to create a somewhat longer text spanning a few lines.

() This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

Let’s use it inside a question, i.e., inside xsim’ s exercise environment [Nie20]:

```latex
% since settings are local the following ones will be lost
% outside this example;
\settasks{
label = \theexercise.\arabic * ,
item-indent = 2em ,
label-width = 2em ,
label-offset = 0pt
}
\begin{exercise}
I have these two tasks for you. Shall we begin?
\begin{tasks}(2)
\task The first task: easy!
\task The second task: even more so!
\end{tasks}
\end{exercise}
\begin{solution}[print]
Now, let's see\ldots\ ah, yes:
\begin{tasks}
\task This is the first solution. Told you it was easy.
\task This is the second solution. And of course you knew that!
\end{tasks}
\end{solution}
```

I have these two tasks for you. Shall we begin?

1. The first task: easy! 2. The second task: even more so!

Now, let's see… ah, yes:

1. This is the first solution. Told you it was easy.

2. This is the second solution. And of course you knew that!

Finally let' s see what the debug option does (you could see it already on page 6):

```
\settasks{debug}
\begin{tasks}(2)
\task \Sample
\task \Sample
\end{tasks}
```

a) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

b) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

5 Available Instances

There are currently three additional instances for the tasks object available:

itemize uses \labelitemi as labels.

enumerate enumerates the items with 1., 2., ...

```
\begin{tasks}[style=itemize](2)
\task that's just how\ldots
\task \ldots we expected
\end{tasks}
\begin{tasks}[style=enumerate](2)
\task that's just how\ldots
\task \ldots we expected
\end{tasks}
```

● that's just how… ● …we expected

☐ that's just how… ☐ …we expected

6 Custom Labels

If you want to change a single label inside a list, you can use the optional argument of \task. This will temporarily overwrite the default label.

```
\begin{tasks}[style=itemize](2)
\task that's just how\ldots
\task[+] \ldots we expected
\task that's just how\ldots
\task \ldots we expected
\end{tasks}
```

• that's just how. . . + . . . we expected

• that's just how. . . • . . . we expected

You' ve already seen examples for the label option.
label = <code> Default: \alph *)
It can be used to set the label for a list. A * inside is always replaced by the current counter name inside braces. It can contain formatting instructions like \bfseries but it can be cleaner to use
label-format = <code> (initially empty)
instead. This is especially true since the label also sets \the<counter> where you usually don' t want to have formatting instructions. Another way to deal with this issue is the option
ref = <code> (initially empty)
which sets \the hcounteri (\thetask in the default setting).

```
\begin{tasks}[label=\arabic * .,ref=\arabic * ]
\task first item
\task second item \label{foo}
\end{tasks} See item~\ref{foo} without dot.
```

a) first item

b) second item

See item b) without dot.

Two additional commands are defned which in some circumstances might prove useful:
\tasksifmeasuringTF{<true>}{<false>}
This command used inside a label checks if the label is typeset for measuring its width or if it is typeset "for real". There are also the variants \tasksifmeasuringT and \tasksifmeasuringF.
\tasklabel
New Holds the current label text.

7 New tasks-like Environments

It is possible to add custom environments that work like the tasks environment.
\NewTasksEnvironment[<options>]{<name>}[<separator>](<cols>)
\Defneenvironment <name> thatuses <eparator> tointroduceanewitem.
\Defaultfor <separator>
is \task, default for <cols> is 1 . The <options> are the ones described in section 4.
\RenewTasksEnvironment[<options>]{<name>}[<separator>](<cols>)
Renew environment previously defned with \NewTasksEnvironment.
The tasks environment is defned as follows:

```
    \NewTasksEnvironment{tasks}
```

The separator does not have to be a control sequence:

```
% preamble:
% \usepackage{fontawesome}
\NewTasksEnvironment[label=\faThumbsOUp,label-width=15pt]{done}[ * ]
\begin{done}
* First task
* Second task
\end{done}
```

♣ First task ♠ Second task

√ First task ♡ Second task

Although this might seem handy or even nice I strongly advice against using something diferent than a command sequence. Remember that the items will be split at every occurrence of the separator. So in order to use the separator (here for example for a starred variant of a command)

within an item it has to be hidden in braces. This is avoided of you use a command sequence which even doesn’ t have to be defned.
Please also keep in mind that the separator still has an optional star argument (see 4), an optional bang argument and the standard optional argument. Using * will prevent the optional star argument.

```
% preamble:
% \usepackage{fontawesome}
\NewTasksEnvironment[label=\faThumbsOUp,label-width=15pt]{done}[ * ]
\begin{done}(3)
* First task
* Second task
* ! Third task spanning the full width available
* Fourth task
\end{done}
```

∇ First task   »» Second task

8 Styling tasks

TASK uses xtemplate to declare additional instances for the lists.

8.1 The tasks Object

The object that’ s defned by tasks is the ‘tasks’ object. This time there are four instances available for the one template (again ‘default’) that was defned.

8.1.1 Available Options

This section only lists the options that can be used when defning an instance of the ‘defaul’ template. The following subsections will give some examples of their usage.

```
\DeclareTemplateInterface{tasks}{default}{3}{
% option : type = default
label : tokenlist = \alph * ) ,
indent : length = 2.5em ,
label-format : tokenlist ,
label-width : length = 1em ,
label-offset : length = .3333em ,
after-item-skip : skip = 1ex plus 1ex minus 1ex
}
```

8.1.2 Predefned Instances

This is rather brief this time:

```
% alphabetize: a) b) c)
\DeclareInstance{tasks}{alphabetize}{default}{}
% itemize
\DeclareInstance {tasks} {itemize} {default}{
label-width = 1.125em ,
label = \labelitemi} % enumerate:
\DeclareInstance {tasks} {enumerate} {default}
{label = \arabic * . }
```

References

[1] Javier Bezos. enumitem. Version 3.9. June 20, 2019. url: https://ctan.org/pkg/enumitem.

[2] The LaTeX3 Project Team. l3kernel. Sept. 19, 2019. url: https://www.ctan.org/pkg/l3kernel/.

[3] Clemens Niederberger. exsheets. Version 0.21k. Sept. 30, 2019. url: https://ctan.org/pkg/exsheets.

[4] Clemens Niederberger. xsim. Version 0.19a. Mar. 19, 2020. url: https://ctan.org/pkg/xsim.

[5] Will Robertson. environ. Version 0.3. May 4, 2014. url: https://ctan.org/pkg/environ .