

L^AT_EX for linguists

Sylvia Blaho & Patrik Bye

CASTL

University of Tromsø

Spring 2004

Class 3: commands, counters, lengths, lists

- Defining and redefining commands
- Using counters
- Using lengths
- Using and customising lists
- Typesetting linguistic examples with `gb4e` and `linguex`.

1 Defining your own commands

- (1) Generalised command definition:

```
\newcommand{\commandname}[number of arguments]{definition}
```

- (2) Commands with no arguments:

`\newcommand{\ipa}{\textipa}` saves you a lot of typing. Now `\ipa{A}` gives you the exact same result as `\textipa{A}`, that is α .

`\newcommand\eaaccu[1]{\tipaUpperaccent[.3ex]{\scriptsize{e}}{u}}` saves you even more typing.

Instead of `\tipaUpperaccent[.3ex]{\scriptsize{e}}{u}`, you can type `\eaaccu{}` and get $\overset{e}{u}$.

Now `\ipa{A}` gives you the exact same result as `\textipa{A}`, that is α .

- (3) Commands with one argument

```
\newcommand{\gloss}[1]{‘#1’}
```

Then saying `\gloss{fluffy bunny}` will output ‘fluffy bunny’.

Why is this useful? Recall the L^AT_EX principle of separating form and content: you only specify the *function* of an expression in the text, and change the form when you want:

```
\renewcommand{\gloss}[1]{\textsc{#1}}
```

Now, `\gloss{fluffy bunny}` will yield FLUFFY BUNNY.

- (4) Commands with two arguments

```
\newcommand{\eg}[2]{\textit{#1} ‘{#2}’}
```

Now `\eg{bolyhos nyuszika}{fluffy bunny}` gives *bolyhos nyuszika* ‘fluffy bunny’.

- (5) Commands with an optional argument

```
\newcommand{\commandname}[number of arguments][opt]{definition}
```

`opt` is the default value for the optional argument.

There can only be one optional argument in a user-defined command. This is always the first one in the definition (`#1`).

```
\newcommand{\eg}[3][\]{\textit{#2} \textipa{#1} ‘{#3}’}
```

Now `\eg[/bojhoS *nusikA/]{bolyhos nyuszika}{fluffy bunny}` gives

bolyhos nyuszika /bojhoʃ ɲusika/ ‘fluffy bunny’,

whereas `\eg{bolyhos nyuszika}{fluffy bunny}` gives *bolyhos nyuszika* ‘fluffy bunny’

- (6) If you try to define a command that has already been defined, you get an error message. Similarly, if you try to redefine a command that has not been defined yet, $\text{\TeX}$ will complain.

If you’re not sure whether a command has already been defined or not, for example because it is defined in one of the packages that you’re using but you want it to work for all your documents, you can use `\providecommand`. This command has the same syntax as `\newcommand` and `\renewcommand`. If the command you’re trying to define with `\providecommand` already exists, the new definition will be ignored, and you can redefine the existing command with `\providecommand`. If the command didn’t exist, `\providecommand` will define it.

Since $\text{\TeX}$ won’t actually tell you if the command you defined with `\providecommand` already existed or not, it’s a good idea to always use `\renewcommand` after `\providecommand`.

2 Counters

- (7) Some predefined counters in $\text{\LaTeX}$:

chapter	paragraph	figure
section	subparagraph	table
subsection	page	footnote
subsubsection	equation	

The commands that manipulate these counters usually have the same name as the corresponding command. For instance, the command `\subsection` increments the value of the counter `subsection` by one and prints the current value of `section` and `subsection`, separated by a period.

(8) Defining your own counter

```
\newcounter{countername}[incounter]
```

`countername` cannot be the same as the name of an already existing counter. The optional argument is the name of an already existing counter, and has the effect that the newly defined counter is set to zero whenever the value of `incounter` is changed.

(9) Printing counter values

<code>\arabic{counter}</code>	Arabic number
<code>\Roman{counter}</code>	capital Roman number
<code>\roman{counter}</code>	lowercase Roman number
<code>\Alph{counter}</code>	uppercase letter
<code>\alph{counter}</code>	lowercase letter
<code>\fnsymbol{counter}</code>	footnote symbol

Each counter has a corresponding `\thecounter` command, which consists of `\the` immediately followed by the name of the counter. Its default setting is `\arabic{counter}`, but it can be redefined.

(10) Changing counter values

Every counter is set to zero by default. The value of a counter can be changed by the following commands:

```
\setcounter{counter}{number}
```

Assigns the value `number` to `counter`.

```
\addtocounter{counter}{number}
```

Increases the value of `counter` by `number` (can be positive or negative).

```
\stepcounter{counter}
```

Increases the value of `counter` by one and resets all its subcounters.

```
\refstepcounter{counter}
```

Same as `\stepcounter`, except that it also sets `counter` to be the current counter for the cross-referencing command `\label` (see below).

(11) The value of a counter can be used as a number with `\value{counter}`.

```
\setcounter{mypage}{\value{page}}
```

(12) You can refer to the value of a counter at a certain point in the text by using the commands `\label` and `\ref`. So, for instance, saying `\label{labelname}` after `\section` will store the current value of `section`, which you can print by `\ref{labelname}`.

You need to $\LaTeX$ your document twice when using cross-references!

(13) Defining your own example numbering

```
\newcounter{mycounter}
```

This command defines a new counter for $\TeX$ to use. It is necessary to issue this command to make sure you chose a counter name that hasn't been used yet.

```
\newcommand{\myexample}{\refstepcounter{mycounter}(\arabic{mycounter})}
```

This command does two things.

1. it increases the counter by one (remember, all counters are set to zero by default)
2. it tells T_EX to output the current value of the counter in arabic numbers, enclosed in parentheses.

Useful if you want to save space:

- | | |
|-------------|--------------------|
| (1) example | (2) second example |
| third | fourth |

3 Lists

- | | | |
|----------------------|--|--|
| (14) Itemized lists: | <ul style="list-style-type: none">• First item.• Second item, and some more text to show indentation. | <pre>\begin{itemize} \item First item. \item Second item... \end{itemize}</pre> |
| (15) Numbered lists: | <ol style="list-style-type: none">1. First item.2. Second item, and some more text to show indentation. | <pre>\begin{enumerate} \item First item. \item Second item... \end{enumerate}</pre> |
| (16) Descriptions: | <p>first First item.</p> <p>second Second item, and some more text to show indentation.</p> | <pre>\begin{description} \item[first] First item. \item[second] Second item... \end{description}</pre> |

(17) Nested lists:

- First level. `\begin{itemize}`
- Second level. `\item First level.`
- * Third level. `\begin{itemize}`
 - Fourth level. `\item Second level.`
 - Fourth again. `\begin{itemize}`
 - * Third level. `\item Third level.`
 - Second level. `\begin{itemize}`
 - Fourth level. `\item Fourth level.`
 - Fourth again. `\item Fourth again.`
- First level. `\end{itemize}`

`\item Third level.`

`\end{itemize}`

`\item Second level.`

`\end{itemize}`

`\item First level.`

`\end{itemize}`

`\begin{enumerate}`

1. First level. `\item First level.`

(a) Second level. `\begin{enumerate}`

i. Third level. `\item Second level.`

A. Fourth level. `\begin{enumerate}`

B. Fourth again. `\item Third level.`

ii. Third level. `\begin{enumerate}`

(b) Second level. `\item Fourth level.`

`\item Fourth again.`

`\end{enumerate}`

`\item Third level.`

`\end{enumerate}`

`\item Second level.`

`\end{enumerate}`

`\item First level.`

`\end{enumerate}`

(18) Manual labels

`\begin{itemize}`

α TIPA. `\item[\textipa{A}] \tipa.`

♥ Heart. `\item[$\heartsuit$] Heart.`

`\end{itemize}`

(19) Redefining labels

```
\renewcommand{\labelitemiii}{+}
(labelitemi,labelitemii,labelitemiii,labelitemiv)
\renewcommand{\labelenumi}{\roman{enumi}.}
(labelenumi,labelenumii,labelenumiii,labelenumiv)
(\arabic,\roman,\Roman,\alph,\Alph)
```

(20) Generalised lists

```
\begin{list}  
  {label declarations}  
  {list declarations}  
\end{list}
```

3.1 Lengths

(21) Fixed lengths

`cm` centimetre
`mm` millimetre
`in` inch (1 in = 2.54 cm)
`pt` point (1 in = 72.27 pt)
`bp` big point (1 in = 72 bp)
`pc` pica (1 pc = 12 pt)

(22) Font-specific lengths

`em` the width of the capital M
`ex` the height of the letter x

(23) Commands for changing lengths

```
\setlength{\mylength}{length_spec}
```

Assigns the value of `length_spec` to `\mylength`. Note that `length_spec` can be a length specification or another length command.

```
\addtolength{\mylength}{length_spec}
```

Adds `length_spec` to the current value of `\mylength`. If `length_spec` has a negative value, `\mylength` decreases.

Note that `length_spec` can be a length specification or another length command.

```
\settowidth{\mylength}{text}
```

Sets `\mylength` to the natural length of the specified text.

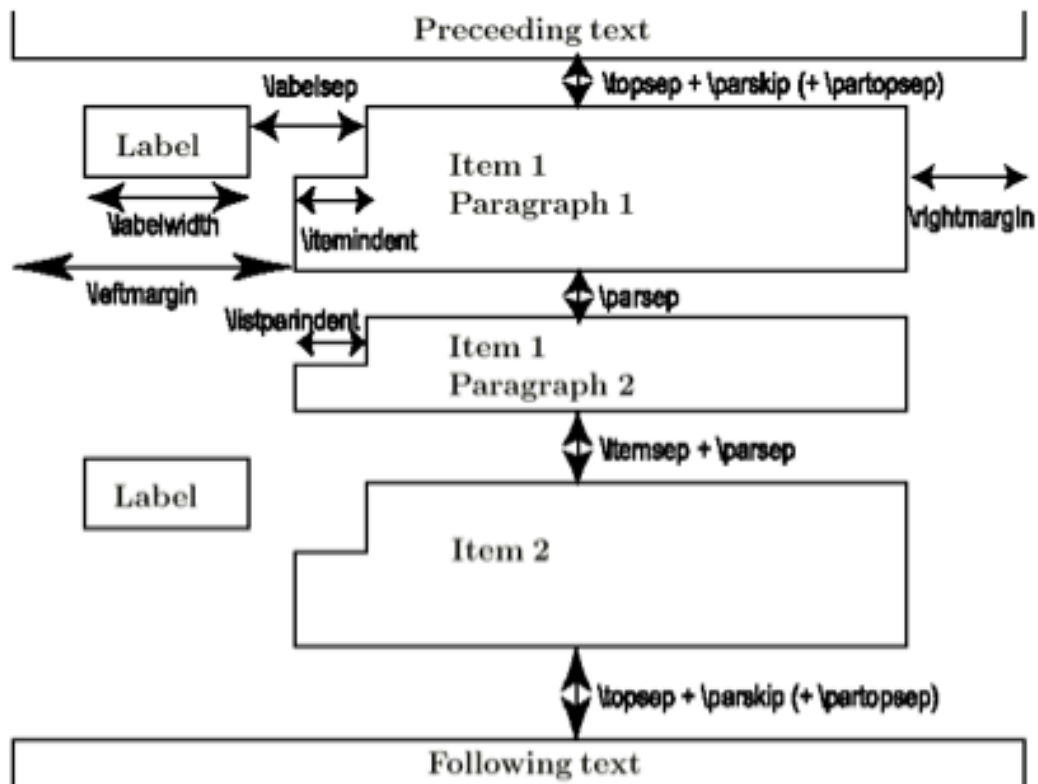
```
\settoheight{\mylength}{text}
```

Sets `\mylength` to the natural height of the specified text.

```
\settodepth{\mylength}{text}
```

Sets `\mylength` to the natural depth of the specified text.

(24) You can define a new length by `\newlength{lengthname}`. The default value of `\newlength` is 0 pt.



This is text preceding the list. ♡ Some text. ♡ More text, and yet some more text to see indentation. And this is text following the list.	This is text preceding the list. <pre>\begin{list}{\\$\\heartsuit\$} {\setlength{\rightmargin}{\leftmargin} \setlength{\itemsep}{-0.3cm}} \item Some text. \item More text... \end{list}</pre> And this is text following the list.
---	---

FIGURE I: <i>First figure.</i> FIGURE II: <i>Second figure.</i> FIGURE III: <i>Third figure. And now some more text to show sizes and in- dentation.</i> <i>And here is another ¶ in Figure III.</i>	<pre>\newcounter{myfigure} \begin{list}{\textsc{Figure} ~\upshape{\Roman{myfigure}}:\hfill} {\usecounter{myfigure} \itshape \setlength{\labelwidth}{2cm} \setlength{\labelsep}{0.5cm}} \item First figure. \item Second figure. \item Third figure. And now... \end{list}</pre>
--	---

The 1950's: Sex. The 1960's: Drugs. The 1970's: Rock'n'roll.	<pre>\newcounter{decade} \newcounter{year} \begin{list} {\setcounter{year}{\value{decade}*10}% The \arabic{year}'s:\hfill\stepcounter{decade}}% {\usecounter{year}} \setcounter{decade}{195} \item Sex. \item Drugs. \item Rock'n'roll. \end{list}</pre>
---	--

You need the package `calc` for this (part of the standard distribution).

4 Linguistic examples

- (25) Probably the most frequently used packages are `linguex` and `gb4e`. We strongly recommend `linguex`, because it takes considerably less typing and is much easier to customise than `gb4e`. Warning: if you try to load both `gb4e` and `linguex` in the same document, you're most likely to get into trouble!

Note, however, that `linguex` uses parts of `gb4e`, so it will only work if you obtain `cgloss4e` and `xspace.sty` along with `linguex.sty`.

- (1) Here is an example.
- a. This is the second level of examples.
 - b. Still the second level.
 - (i) And the third level.
 - (ii) But don't go deeper than that.

Because you run out of labels.

```

\ex. Here is an example.
\ a. This is the second level of examples.
\ b. Still the second level.
\ a. And the third level.
\ b. But don't go deeper than that.
    \ a. Because you run out of labels.

```

The `\ex.` environment must be closed by a single blank line – this closes all embedded `\a.` environments as well.

(26) To close only one level of embedded lists:

```

(2)   Here is an example.
      a. This is the second level of examples.
      b. Still the second level.
         (i) And the third level.
         (ii) But don't go deeper than that.
      c. And back to level 2.

```

```

\ex. Here is an example.
\ a. This is the second level of examples.
\ b. Still the second level.
    \ a. And the third level.
    \ b. But don't go deeper than that.
    \ z.
\ b. And back to level 2.

```

(27) Glosses are automatically aligned with the example sentence

```

(3)   Here is an example.
      a. This is the second level of examples.
      b. Az első mondat.
         the first sentence
         'The first sentence.'

```

```

\ex. Here is an example.
\ a. This is the second level of examples.
\ bg. Az els\H{o} mondat.\
the first sentence\
'The first sentence.'

```

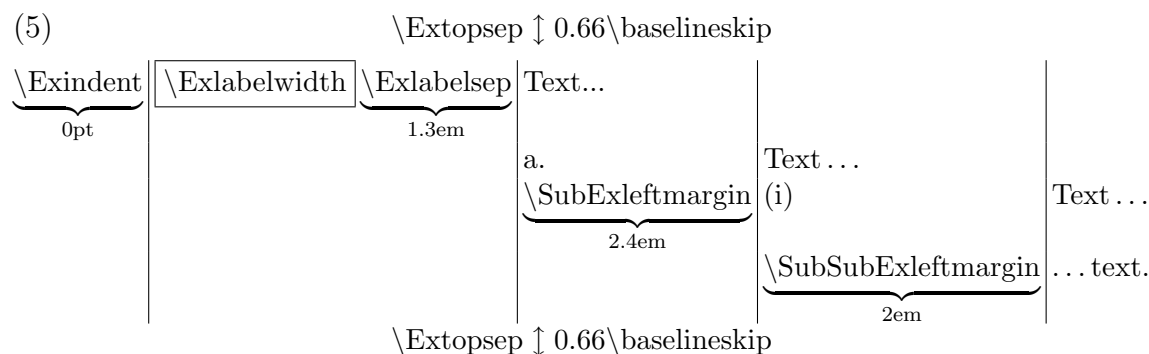
(28) Symbols for grammaticality judgements

```

(4)   *?Az mondat első.
      the sentence first
      \exg. *? Az mondat els\H{o}.\
            the sentence first\

```

(29) Customising lengths and margins



Note that the commands changing the defaults for `linguex` must *follow* `\begin{document}`.

(30) Customising labels

The counters used are `ExNo`, `SubExNo` and `SubSubExNo`, respectively, and treated like any other counter.

`\a.` Can be used without being embedded in `\ex.`, with the expected result:

a.	Example one.	<code>\a. Example one.</code>
b.	Example two.	<code>\b. Example two.</code>

The list commands also take optional arguments:

one	Example one.	<code>\a.[one] Example one.</code>
two	Example two.	<code>\b.[two] Example two.</code>

There should be no space between the command-final period and the optional argument!

For other things these packages can do, see their documentation.

Homework

1. Change the labels of the `enumerate` environment to “[I], [II], ...” for the first level, and to “(i-A), (i-B), ..., (ii-A), (ii-B)” for the second level.
2. Using the list environment, define a `definition` environment like the one below:

Definition 1:

This is a definition. Let’s put some more text here, so that you can see that the right side of the text extends to the right margin.

Definition 2:

This is another definition.

The indentation of the definitions is the same as the indentation of paragraphs in normal text. The label of the definitions is not indented, and it is typeset in bold italics.

3. Using `linguex`, typeset the following:

Principle C (Chomsky[81]): An R-expression is free only

- α) with respect to potential binders in A-positions,
- β) within the domain of its chain.