The nnext package

Nicholas LaCara
nick.lacara@gmail.com

October 6, 2020

Abstract

This package is an add-on for the gb4e example package used in linguistics. It implements the `\Next`, `\NNext`, `\Last`, and `\LLast` commands from the linguex package or the `\nextx`, `\anextx`, `\lastx`, `\blastx`, and `\bblastx` commands from the expex package. The package takes its name from the distinctively named `\NNext` command found in linguex.

1 Introduction

The popular linguistics example package linguex and the less popular (though more powerful) package expex allow users to refer to previously or to-be defined examples through mnemonic commands such as `\Next` or `\nextx`. The popular package gb4e lacks any such functionality. The goal of this package is to provide this functionality for gb4e and related packages so that users that need (or want) to use gb4e but are more comfortable with linguex or expex can still have access to these macros while using gb4e.

This package is currently compatible with the original gb4e package as well as the modified version langsci-gb4e used by Language Science Press and with gb4e-emulate by Alan Munn (which reimplements gb4e's functionality with the package enumitem).[1]

2 Usage

2.1 Loading the package

The package should be loaded in your preamble with `\usepackage{nnext}`. It should be loaded *after* gb4e (or whichever variant you are using).

2.2 Package options

The package has a few options that may be specified when loaded:

[1] Unfortunately, gb4e-emulate is not on CTAN. It is, however, available at its GitHub repository: `https://github.com/amunn/gb4e-emulate`

`linguex` This option defines the macros `\Next`, `\NNext`, `\Last`, and `\LLast`, as found in the linguex package. It is the default package option and does not need to be specified.

`expex` This option defines the macros `\nextx`, `\anextx`, `\lastx`, `\blastx`, and `\bblastx`, as found in the expex package.

`noparens` This option disables the use of parentheses around example numbers that are on by default when linguex emulation is used. This is the typical behavior for gb4e and expex.

2.3 Macros

2.3.1 linguex emulation mode (default)

If the user loads the package with the linguex option or, indeed, with no options specified, the package will provide the following macros.

`\Next`
- The macro `\Next` will print the next example number. So, if the previous example was (15), this will print (16).

`\NNext`
- The macro `\NNext` will print the example number after next. So, if the previous example was (15), this will print (17).

`\Last`
- The macro `\Last` will print the previous example number. So, if the previous example was (15), this will print (15).

`\LLast`
- The macro `\LLast` will print the example number before last. So, if the previous example was (15), this will print (14).

Usage of these macros is straightforward. Simply deploy them in running text, as in the following code:

```
If you want to refer to the next example, type \Next.
If you want to refer to the example after that, type \NNext.

  \begin{exe}
    \ex[]{This is an example.}
    \ex[*]{This are another example.}
  \end{exe}

\noindent If you want to refer to the previous example,
type \Last. If you want to refer to the example before
that, type \LLast.
```

This code produces the following output:

> If you want to refer to the next example, type (1). If you want to refer to the example after that, type (2).
>
> (1) This is an example.
>
> (2) * This are another example.
>
> If you want to refer to the previous example, type (2). If you want to refer to the example before that, type (1).

2.3.2 expex emulation mode

If the user loads the package with the expex option the package will provide the following macros.

\nextx
- The macro \nextx will print the next example number. So, if the previous example was (15), this will print (16).

\anextx
- The macro \anextx will print the example number after next. So, if the previous example was (15), this will print (17).

\lastx
- The macro \lastx will print the previous example number. So, if the previous example was (15), this will print (15).

\blastx
- The macro \blastx will print the example number before last. So, if the previous example was (15), this will print (14).

\bblastx
- The macro \bblastx will print the example number before last. So, if the previous example was (15), this will print (13).

Usage of these macros is straightforward. Simply deploy them in running text, as in the following code:

```
If you want to refer to the next example, type (\nextx).
If you want to refer to the example after that, type (\anextx).

  \begin{exe}
    \ex[]{This is an example.}
    \ex[*]{This are another example.}
  \end{exe}

\noindent If you want to refer to the previous example,
type (\lastx). If you want to refer to the example before
that, type (\blastx). If you want to refer to the example
before that, type (\bblastx).
```

This code produces the following output:

If you want to refer to the next example, type (3). If you want to refer to the example after that, type (4).

(3) This is an example.

(4) * This are another example.

If you want to refer to the previous example, type (4). If you want to refer to the example before that, type (3). If you want to refer to the example before that, type (2).

2.4 Class compatibility

I have tested the package with a variety of common LaTeX document classes (including the ams-LaTeX classes, scr* group of classes, tufte-handout, and the exam class, among several others) and believe it should be broadly compatible with most people's set-ups (so far I have only found it to be incompatible with

the standalone class). If you have an issue, drop me an email and I'll see if I can't make it work for you.

3 Rationale

This package is really just a quality-of-life add-on for people using the gb4e example package.

The initial motivation for this package came from a colleague who needed to typeset a document for submission to a publisher but was unable to use linguex due to constraints imposed by the publisher's class file. She asked specifically if there was any way to get gb4e to replicate the functionality of the linguex commands `\Next`, `\NNext`, and `\Last`, among others.

I figured it wouldn't be too difficult to code up a solution and send it to her (which I did!), but given that there are likely other people who, due to various externally imposed requirements, might need to use gb4e when they are used to another example pacakge, I thought a more robust solution might be appropriate. This package attempts to provide that.

I also hope that this package can help people migrate to gb4e from linguex. There are several benefits to using gb4e over linguex, including a more robust syntax that is more aligned with standard LaTeX syntax and less sensitive to code formatting such as line-breaks. That said, gb4e is a little less user-friendly than linguex, so I think that providing one of the features that linguex users expect can help people to migrate if they so choose.

4 Some notes for linguex users

The gb4e package behaves slightly differently from linguex in a handful of ways. A few of those differences may impact how you experience using this package.

- gb4e handles example numbering in footnotes poorly, using numbering from the main body of the text (linguex uses roman numerals in footnotes, one of the ways in which it is better). This is something to keep in mind if you put an example in a footnote.
- Relatedly, if you use gb4e's commands for customizing example label numbers (*e.g.*, `\exi` and `\exr`), the code here will not use those special labels.

5 Implementation

In the following section, I present the fully commented code for the package.

5.1 Dependencies

I use the xspace package to make the spacing after example numbers right when using the `\Next`, `\Last`, *etc*., commands. This mimics the behavior of linguex.

```
\RequirePackage{xspace}
```

I also use ifthen for various booleans set by the package options.

```
\RequirePackage{ifthen}
```

Because this package is designed to work with a range of gb4e variants, there is no line in the code for loading gb4e. Documents loading nnext without loading a gb4e variant will compile will compile, but using any of the included macros will cause an error.

5.2 Booleans

Here I define what the booleans will be. The first will set the package to emulate linguex's `\Next`, `\NNext`, `\Last`, and `\LLast` commands. The boolean is set to 'false' initially.

```
\newboolean{emulatelinguex}
\setboolean{emulatelinguex}{false}
```

The next boolean will set the package to emulate expex's `\nextx`, `\lastx`, `\blastx`, `\anextx`, and `\bblastx` commands. The boolean is set to 'false' initially.

```
\newboolean{emulateexpex}
\setboolean{emulateexpex}{false}
```

The final boolean will set whether the example numbers in the text should be set with surrounding parentheses or not. The boolean is set to 'true' initially.

```
\newboolean{parentheses}
\setboolean{parentheses}{true}
```

5.3 Package options

Here is where various package options are defined. All the package options do is manipulate the values of the booleans declared in the previous subsection.

By using the package option `linguex`, the user sets the `emulatelinguex` boolean to 'true'.

```
\DeclareOption{linguex}{
    \setboolean{emulatelinguex}{true}
}
```

By using the package option `expex`, the user sets the `emulateexpex` boolean to 'true' and the `emulatelinguex` and `parentheses` booleans to 'false'.

```
\DeclareOption{expex}{
    \setboolean{emulateexpex}{true}
    \setboolean{emulatelinguex}{false}
    \setboolean{parentheses}{false}
}
```

By using the package option `noparens`, the user sets the `parentheses` boolean to 'false'.

```
\DeclareOption{noparens}{
```

```
18          \setboolean{parentheses}{false}
19      }
```

After defining our package options, we tell LaTeX to use linguex emulation as the default mode.

```
20      \ExecuteOptions{linguex}
21      \ProcessOptions\relax
```

5.4 Detecting gb4e variants

To make this compatible with different variants of gb4e, we need to detect which variant of gb4e the user is using, since there is some variation in what each variant uses for its example counter. Right now, the code is compatible with the following gb4e-like packages:

- gb4e
- langsci-gb4e (for Language Science Press books)
- gb4e-emulate (by Alan Munn, which reimplements gb4e using the package enumitem)

The code below checks to see if langsci-gb4e is loaded; if it isn't, it checks to see if gb4e-emulate is loaded; if it isn't, it checks to see if gb4e is loaded. If this fails, the package defaults to using the `exx` counter from gb4e on the assumption that the most likely package to be used instead of gb4e is a modded version. It also prints a warning to the terminal.

The reason this needs to be done is because each of these packages uses a different counter for example numbering under the hood:

- gb4e uses the counter `exx`
- langsci-gb4e uses the counter `equation`
- gb4e-emulate uses the counter `exei`

`\@countername` Once the right package is identified, the value of `\@countername` is defined to match the counter used by that package. This allows the macros defined below to get the current value of the example counter at any point in the document regardless of what example package is being used.

```
22      \@ifpackageloaded{langsci-gb4e}%
23        {\newcommand{\@countername}{equation}}
24        {\@ifpackageloaded{gb4e-emulate}%
25          {\newcommand{\@countername}{exei}}
26          {\@ifpackageloaded{gb4e}%
27            {\newcommand{\@countername}{exx}}
28            {\PackageWarningNoLine{nnext}{No known compatible %
29              example package loaded! Examples may not be correctly %
30              referenced, or there may be fatal errors}%
31              \newcommand{\@countername}{exx}}}}%
```

5.5 Manipulating parentheses

\@lparens
\@rparens
\@afterspace

linguex and expex behave differently with regard to whether the output of reference commands include parentheses: linguex includes them; expex (like gb4e) does not. The code here defines whether the macros include parentheses based on the booleans defined above. It does this by setting the commands \@lparens and \@rparens to either be parentheses or by defining them to be empty. It also defines whether there should be space included after the number (as in linguex) or not (as in expex). It does so, again, by defining the macro afterspace to either to expand to \xspace or to exapand to nothing. These macros are then used in the \printtmpcounter macro below.

```
\ifthenelse{\boolean{emulatelinguex}%
            \AND \boolean{parentheses}}{%
    \newcommand{\@lparens}{(}
    \newcommand{\@rparens}{)}
    \newcommand{\@afterspace}{\xspace}}{%
    \newcommand{\@lparens}{}
    \newcommand{\@rparens}{}
    \newcommand{\@afterspace}{}}%
```

5.6 Generic macros

linguex implements its commands by getting the current value of the example counter, storing that value in a temporary counter, and then adding or subtracting from that temporary counter. This is the approach that I take below. To replicate this functionality with gb4e, we only have to do this with the counter that it (or its variants) uses

5.6.1 Setting up the temporary counter

First, we create a temporary counter, which will be used to store the value of the next, last, or whichever example number the macro is looking for.

```
\newcounter{tmpcounter}
```

\settmpcounter

Now we define the command \settmpcounter, which gets the value of the gb4e example counter (the name of which is stored in \@countername) and sets the value of the temporary counter to this value plus some integer:

```
\newcommand{\settmpcounter}[1]{%
\setcounter{tmpcounter}{\value{\@countername}}%
    \addtocounter{tmpcounter}{#1}}
```

\printtmpcounter

Now we define a command for printing the value of temporary counter with or without parentheses and with or without the trailing space, depending on whether linguex or expex is being emulated. This relies on how \@lparens, \@rparens, and \@afterspace were defined above.

```
\newcommand{\printtmpcounter}[1]{\settmpcounter{#1}%
\@lparens\thetmpcounter\@rparens\@afterspace}
```

5.6.2 Generic definitions

Now we define some generic, under-the-hood macros for commands that add to or subtract from the temporary counters and print those numbers as though they were references. These under-the-hood commands will be assigned user-facing names depending on whether linguex or expex is being emulated.

\@Next The macro \@Next increments the tempory counter by one, which will be the value of the next example:

```
\newcommand{\@Next}{\printtmpcounter{1}}
```

\@NNext The macro \@NNext increments the tempory counter by two, which will be the value of the example after next:

```
\newcommand{\@NNext}{\printtmpcounter{2}}
```

\@Last The macro \@Last does not change the value of the temporary counter, since this is the value of the previous example:

```
\newcommand{\@Last}{\printtmpcounter{0}}
```

\@LLast The macro \@LLast decreases the tempory counter by one, which will be the value of the example before last:

```
\newcommand{\@LLast}{\printtmpcounter{-1}}
```

\@LLLast The macro \@LLLast decreases the tempory counter by two, which will be the value of the example before the example before last:

```
\newcommand{\@LLLast}{\printtmpcounter{-2}}
```

5.7 Emulate linguex

\Next \NNext \Last \LLast If the user chooses to emulate the linguex commands, then this conditional uses the command names from linguex and defines them to expand to the macros defined in the previous subsection.

```
\ifthenelse{\boolean{emulatelinguex}}{%
    \newcommand{\Next}{\@Next}
    \newcommand{\NNext}{\@NNext}
    \newcommand{\Last}{\@Last}
    \newcommand{\LLast}{\@LLast}}{}
```

5.8 Emulate expex

\nextx \anextx \lastx \blastx \bblastx If the user chooses to emulate the linguex commands, then this conditional uses the command names from linguex and defines them to expand to the macros defined in the previous subsection.

```
\ifthenelse{\boolean{emulateexpex}}{%
    \newcommand{\nextx}{\@Next}
```

```
58          \newcommand{\anextx}{\@NNext}
59          \newcommand{\lastx}{\@Last}
60          \newcommand{\blastx}{\@LLast}
61          \newcommand{\bblastx}{\@LLLast}}{}
```

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.

Change History

v0.0

General: Initial version 1