The xpeekahead Package
Version 1.1

Alceu Frigeri*

October 2025

Abstract

This package offers a few commands aiming at peeking ahead environments and commands in simple cases. It's based on *expl3* and a question at stackexchange [2].

Contents

1 Introduction

expl3 offers a solid base for programmatically peeking ahead (with the many `\peek_` commands), nevertheless some constructions might be extensive and, at times, tricky. This package is focused in two cases:

- detecting the first token (perhaps a command) past the end of the current one, ignoring all spaces, blanks, new lines.
- same, but past the end of an environment (the tricky part)

This should be enough in most cases where one wants to fine tune formatting, e.g. spacing, based on what follows.

Two sets of commands are defined, one to be used in a *expl3* package (see 2), and another for use in a LaTeX 2_ε code regimé (see 3).

> ***Note:*** In fact, given `\peek_regex:` flexibility, it is possible to construct a regular expression that will look past a single/few tokens, in which case one is probably best served with the *expl3* `\peek_` functions.
>
> ***Note:*** The LaTeX 2_ε commands at 3 are just aliases to some of the *expl3* commands at 2.

*https://github.com/alceu-frigeri/xpeekahead

2 Expl3 Commands

When peeking ahead, one needs a regular expression to match against, in the commands 2.2 there is the option of given said regular expression "directly" (best in most cases) or through a pre-defined command (better, if, for instance, the very same command has to be re-used many times). Besides that, one can "pre-compile" a regular expression (defining a regex variable, as perl *expl3* convention). In the commands below, ⟨regex⟩ stands for the "direct form", and ⟨pre-regex⟩ stands for the "pre-compiled" variables. Lastly the commands ending with a TF aren't *expl3* true conditionals (you can't `\prg_generate_conditional_variant:` from those).

> ***Note:*** The first release had some unfortunate parameter's naming (using `nn` instead of `TF`), those commands are now deprecated and will generate a warning if used.

2.1 Defining Action Commands

`\xpeekahead_set:NnTF` `\xpeekahead_set:NnTF` {⟨cmd⟩} {⟨regex⟩} {⟨if-true⟩} {⟨if-false⟩}
`\xpeekahead_gset:NnTF` `\xpeekahead_gset:NnTF` {⟨cmd⟩} {⟨regex⟩} {⟨if-true⟩} {⟨if-false⟩}
`\xpeekahead_set:NNTF` `\xpeekahead_set:NNTF` {⟨cmd⟩} {⟨pre-regex⟩} {⟨if-true⟩} {⟨if-false⟩}
`\xpeekahead_gset:NNTF` `\xpeekahead_gset:NNTF` {⟨cmd⟩} {⟨pre-regex⟩} {⟨if-true⟩} {⟨if-false⟩}

updated: 2025/10/07

These will create a new command ⟨cmd⟩, the peeked ahead token(s) will be compared with ⟨regex⟩ and ⟨if-true⟩ or ⟨if-false⟩ will be left on the input stream, before the peeked ahead token(s). For instance, `\c {begin}\cB {envx}` will match `\begin {envx}`.
The `\xpeekahead_set:` will create the new command in the current group, whilst `\xpeekahead_gset:` will create it grobally

> ***Note:*** ⟨regex⟩ can be any valid Regular Expression, as described in [1], in particular take a look on the section *Matching exact tokens*.
>
> ***Warning:*** These commands won't check if ⟨cmd⟩ is already defined, and will overwrite any previous definition.

2.2 Peeking Ahead

`\xpeekahead_cmd_peek:N` `\xpeekahead_cmd_peek:N` {⟨cmd⟩}
`\xpeekahead_cmd_peek:nTF` `\xpeekahead_cmd_peek:nTF` {⟨regex⟩} {⟨if-true⟩} {⟨if-false⟩}
`\xpeekahead_cmd_peek:NTF` `\xpeekahead_cmd_peek:NTF` {⟨pre-regex⟩} {⟨if-true⟩} {⟨if-false⟩}

updated: 2025/10/07

Those are for the most simple cases, where `\xpeekahead_cmd_peek:N` or `\xpeekahead_cmd_peek:nTF` will be placed at the very end of a command definition. ⟨regex⟩, ⟨if-true⟩ and ⟨if-false⟩ are the same as defined in 2.1 and ⟨cmd⟩ is any command defined using `\xpeekahead_(g)set:`.

`\xpeekahead_env_set:nN` `\xpeekahead_env_set:nN` {⟨env-name⟩} {⟨cmd⟩}
`\xpeekahead_env_set:nnTF` `\xpeekahead_env_set:nnTF` {⟨env-name⟩} {⟨regex⟩} {⟨if-true⟩} {⟨if-false⟩}
`\xpeekahead_env_set:nNTF` `\xpeekahead_env_set:nNTF` {⟨env-name⟩} {⟨pre-regex⟩} {⟨if-true⟩} {⟨if-false⟩}

updated: 2025/10/07

Those are for the cases where one wants to detect what comes after the end of an environment. In this case, those commands are to be placed in the "beginning part" of an environment definition. Note that the peek ahead command will be injected past the end of the environment, meaning any local assignment made inside of the environment won't be accessible.
⟨regex⟩, ⟨if-true⟩ and ⟨if-false⟩ are the same as defined in 2.1 and ⟨cmd⟩ is any command defined using `\xpeekahead_(g)set:`.

> ***Important:*** The ⟨env-name⟩ HAS TO BE the name of the environment being defined. It will be used by the injected function to evaluate if it should peek ahead or not (in case it's called in the context of another (inner) environment).
>
> ***Note:*** Those commands are reentrant safe, meaning, the resulting environment can be nested as needed.
>
> ***Warning:*** The peek ahead command injection assumes that the macro "end " (with an space at the end of it) doesn't change (thanks David, [2]). Since the macro capture is done at the first call to `\xpeekahead_env_set:` it should be safe.

3 LaTeX2e Commands

When peeking ahead one needs a regular expression to match against, in the commands 3.2 there is the option of given said regular expression "directly" (best in most cases) or through a pre-defined command (better, if, for instance, the very same regular expression has to be re-used many times).

3.1 Defining Action Commands

`\xpeekSetCmd` `\xpeekSetCmd {⟨cmd⟩}{⟨regex⟩}{⟨if-true⟩}{⟨if-false⟩}`
`\xpeekSetCmdGlobal` `\xpeekSetCmdGlobal {⟨cmd⟩}{⟨regex⟩}{⟨if-true⟩}{⟨if-false⟩}`

These will create a new command ⟨cmd⟩, the peeked ahead token(s) will be compared with ⟨regex⟩ and ⟨if-true⟩ or ⟨if-false⟩ will be left on the input stream, before the peeked ahead token(s). For instance, `\c {begin}\cB {envx}` will match `\begin {envx}`.
The `\xpeekSetCmd` will create the new command in the current group, whilst `\xpeekSetCmdGlobal` will create it grobally

> ***Note:*** ⟨regex⟩ can be any valid Regular Expression, as described in [1], in particular take a look on the section *Matching exact tokens*.
>
> ***Warning:*** These commands won't check if ⟨cmd⟩ is already defined, and will overwrite any previous definition.

3.2 Peeking Ahead

`\xpeekTokCmd` `\xpeekTokCmd {⟨cmd⟩}`
`\xpeekTok` `\xpeekTok {⟨regex⟩}{⟨if-true⟩}{⟨if-false⟩}`

Those are for the most simple cases, where `\xpeekTokCmd` or `\xpeekTok` will be placed at the very end of a command definition. ⟨regex⟩, ⟨if-true⟩ and ⟨if-false⟩ are the same as defined in 3.1 and ⟨cmd⟩ is any command defined using `\xpeekSetCmd` or `\xpeekSetCmdGlobal`.

`\xpeekEnvCmd` `\xpeekEnvCmd {⟨env-name⟩}{⟨cmd⟩}`
`\xpeekEnv` `\xpeekEnv {⟨env-name⟩}{⟨regex⟩}{⟨if-true⟩}{⟨if-false⟩}`

Those are for the cases where one wants to detect what comes after the end of an environment. In this case, those commands are to be placed in the "beginning part" of an environment definition. Note that the peek ahead command will be injected past the end of the environment, meaning any local assignment made inside of the environment won't be accessible.
⟨regex⟩, ⟨if-true⟩ and ⟨if-false⟩ are the same as defined in 3.1 and ⟨cmd⟩ is any command defined using `\xpeekSetCmd` or `\xpeekSetCmdGlobal`.

> ***Important:*** The ⟨env-name⟩ HAS TO BE the name of the environment being defined. It will be used by the injected function to evaluate if it should peek ahead or not (in case it's called in the context of another (inner) environment).
>
> ***Note:*** Those commands are reentrant safe, meaning, the resulting environment can be nested as needed.
>
> ***Warning:*** The peek ahead command injection assumes that the macro "end " (with an space at the end of it) doesn't change (thanks David, [2]). Since the macro capture is done at the first call to `\xpeekEnv` or `\xpeekEnvCmd` it should be safe.

4 Examples

To keep things simple, in the examples below LaTeX 2$_\varepsilon$ syntax will be used, since they are just aliases to their *expl3* counter parts.

4.1 Peeking Ahead Simple Commands

Note that, in this first example, the `\xpeekTok` and `\xpeekTokCmd` were the last commands on commands `\cmdA` and `\cmdB`.

Of course, in a more real case, one will (instead of adding some conditional text as in those examples) perhaps adjust vertical spacing and/or setting auxiliary variables, etc.

```
  %% This will match either \cmdA or \cmdC
\xpeekSetCmd{\detectAC}
  {\c{cmdA} | \c{cmdC}}
  {\hspace{5mm} A or C will be next\par}
  {\hspace{5mm} something else\par}

  %% This will match only \cmdB
\xpeekSetCmd{\detectB}
  {\c{cmdB}}
  {\hspace{5mm} B will be next\par}
  {\hspace{5mm} something else\par}
```

```
  % using the pre-defined \detectAC
\NewDocumentCommand{\cmdA}{m}
  {\par .. command A (#1).. \par\xpeekTokCmd{\detectAC}}

  % given the matching regular expression directly
\NewDocumentCommand{\cmdB}{m}
  {\par .. command B (#1).. \par\xpeekTok{\c{cmdC}}{\hspace{3mm}C will be next\par}{\hspace{3mm}
      something else\par}}

\NewDocumentCommand{\cmdC}{m}
  {\par .. command C (#1).. \par}
```

```
  \cmdA{some par for A}

  \cmdA{some par for A, again}

  \cmdB{some for B}

  \cmdC{some for C}
```

.. command A (some par for A)..
A or C will be next
.. command A (some par for A, again)..
something else
.. command B (some for B)..
C will be next
.. command C (some for C)..

4.2 Peeking Ahead Environments

Note that all one need to detect the beginning of an environment is to recognise the sequence `\c{begin}` followed by whatever environment one wants to detect:

```
  %% This will match the begin of two environments: \begin{envA} or \begin{envC}
\xpeekSetCmd{\detectEnvAC}
  {\c{begin}(\cB{envA}|\cB{envC})}
  {\hspace{5mm} envA or envC will be next\par}
  {\hspace{5mm} something else\par}

  %% This will match only \begin{envB}
\xpeekSetCmd{\detectEnvB}
  {\c{begin}\cB{envB}}
  {\hspace{5mm} B will be next\par}
  {\hspace{5mm} something else\par}
```

```
\NewDocumentCommand{\cmdD}{m}
  {
    \par .. command D (#1) .. \par
    \xpeekTok{\c{begin}\cB{envA}}
      {\hspace{3mm}env A will be next\par}
      {\hspace{3mm}something else\par}
  }

\NewDocumentEnvironment{envA}{}
  {
    \par beginning of environment A\par

    %this can be put anywhere in the beginning block... no need to be the last command.
    \xpeekEnvCmd{envA}
      {\detectEnvAC}
  }
  {
   \par end of environment A\par
  }

\NewDocumentEnvironment{envB}{}
  {
    \par beginning of environment B\par

    %this can be put anywhere in the beginning block... no need to be the last command.
    \xpeekEnv{envB}
      {\c{begin}\cB{envC}}
      {\hspace{3mm}env C will be next\par}
      {\hspace{3mm}something else\par}
  }
  {
   \par end of environment B\par
  }


\NewDocumentEnvironment{envC}{}
  {
    \par beginning of environment C\par
  }
  {
   \par end of environment C\par
  }
```

```
\cmdD{some par for D}

\begin{envA}
  some text
\end{envA}

\begin{envC}
  some text
\end{envC}


\begin{envB}
  some text
\end{envB}

\begin{envB}
  some text
\end{envB}
```

.. command D (some par for D) ..
env A will be next
beginning of environment A
some text
end of environment A
envA or envC will be next
beginning of environment C
some text
end of environment C
beginning of environment B
some text
end of environment B
something else
beginning of environment B
some text
end of environment B
something else

References

[1] The LaTeX3 Project. *The LaTeX3 Interfaces*. 2025. URL: https://mirrors.ctan.org/macros/latex/required/l3kernel/interface3.pdf (visited on 09/14/2025).

[2] David Purton. *peek ahead in expl3*. 2025. URL: https://tex.stackexchange.com/a/745603/207840 (visited on 09/14/2025).