The \texttt{getitems} package: gathering \texttt{item}'s from a list-like environment\textsuperscript{*}

Anders Hendrickson  
St. Norbert College, De Pere, WI, USA  
anders.hendrickson@snc.edu  
January 11, 2016

1 Overview

The \texttt{enumerate} and \texttt{itemize} environments of \LaTeX \TeX{} organize their contents through the use of the \texttt{item} command. Each entry in these lists is prefaced with the command \texttt{item}, making for very compact and easily readable source code. Package designers may find it useful to use the same syntax for their custom environments. The \texttt{getitems} package makes it easy to code such environments by parsing a string of tokens, separating them by the occurrence of \texttt{item’s}, and saving the contents as macros. Nested environments are handled correctly.

Moreover, some typesetting tasks naturally consist of a “header” followed by several related items; one example would be a multiple-choice question on a school examination. This package saves any \TeX{} tokens appearing before the first \texttt{item} as the zeroth item for special handling.

2 Usage

\texttt{\textbackslash gatheritems} To parse a string of text, such as the body of an environment, call

\begin{verbatim}
\texttt{\textbackslash gatheritems\{\texttt{(text to parse)}\}}.
\end{verbatim}

This will scan through the \texttt{(text to parse)}, dividing it at each \texttt{item} while respecting \TeX{} groupings and nested environments, and store the divided portions of text into memory.

\texttt{\textbackslash numgathereditems} The total number of items in the parsed text is stored in the \LaTeX{} counter \texttt{numgathereditems}.

\texttt{\textbackslash gathereditem} To retrieve a stored item, you may call \texttt{\textbackslash gathereditem\{\texttt{(item number)}\}}; the \texttt{(item number)} should expand to an arabic representation of a nonnegative

\textsuperscript{*}This document corresponds to \texttt{getitems.sty} v1.0, dated 2016/01/11.
integer. Any tokens occurring before the first \item may be retrieved with \gathereditem{0}.

Once the items are gathered, it will probably be necessary to loop through all of them. Of course a package author can do so manually, but \getitems provides a built-in way to do so by calling \loopthroughitemswithcommand\{⟨macro⟩\}. The \textit{⟨macro⟩} must be a control sequence taking exactly one argument; it will be called successively with the item text. For example,

\begin{verbatim}
gatheritems{%
Zero
\item One
\item Two
\item Three
}
\loopthroughitemswithcommand{\fbox}
\end{verbatim}

The result is the same as processing \fbox{One}, then \fbox{Two}, and finally \fbox{Three}. Note that \loopthroughitemswithcommand deliberately ignores the zeroth entry, which occurs before the first \item.

Typically the package author will create a custom macro to process each item. This macro may make use of the index of the loop, which is stored in the \LaTeX counter currentitemnumber. A conditional \texttt{ifgatherbeginningofloop} is also available, which only evaluates as true when processing the first item; it is thus functionally equivalent to \texttt{ifnum1=c@currentitemnumber}. The custom macro may take advantage of this to run special code for the first item only.

3 Example

An example using \getitems to create a custom environment may be informative. We use the \texttt{\NewEnviron} command from the \texttt{environ} package (automatically loaded by \getitems) to define a \texttt{question} environment; the body between the \texttt{\begin{question}} and \texttt{\end{question}} is available as \texttt{\BODY}.

\begin{verbatim}
def\doitem#1{\item #1\hfill $\Box$}
\NewEnviron{question}{%
\expandafter\gatheritems\expandafter{\BODY}%
\gathereditem{0}%
\begin{itemize}
\loopthroughitemswithcommand{\doitem}
\end{itemize}
}
\begin{question}
Who proved the unsolvability of the quintic?
Check the appropriate box.
• Abel \square
• Galois \square
• Lie \square
\end{question}
\end{verbatim}
This second example shows that nested environments are handled as expected.

\def\doitem#1{\item[$\Box$]
  \fbox{\parbox[t]{1.75in}{#1}}}\n\NewEnviron{question}{%  \expandafter\gatheritems\expandafter{\BODY}  \gathereditem{0}  \begin{itemize}  \loopthroughitemswithcommand{\doitem}  \end{itemize}  }\begin{question}  Who proved the unsolvability of the quintic? Check the appropriate box.  □ Abel  – Born August 5, 1802  – Died April 6, 1829  □ Galois  – Born October 25, 1811  – Died May 31, 1832  □ Lie  – Born December 17, 1842  – Died February 18, 1899\end{question}

4 Implementation

We need the \trimspaces package to remove excess spaces from the items we find. Although the \environ package is not used by \getitems itself, it will almost certainly be needed.

1 \RequirePackage{environ}
2 \RequirePackage{trimspaces}
3 \let\xa=\expandafter\gathereditem
4 \def\gathereditem#1{\csname getitems@item@#1\endcsname}
5 \setcounter{numgathereditems}{0}

\gatheritems The $k$th item found will be stored in the macro \getitems@item@($k$); the user can access it through the \gathereditem macro.

\numgathereditems We define the LaTeX counter \texttt{numgathereditems}.

\gatheritems The main control sequence of this package is \texttt{\gatheritems}. The naïve strategy is to use the delimiter mechanism of \LaTeX to split the text at the first
occurrence of the token “\texttt{item}.” We add \texttt{\getitems@relax} before, and “\texttt{item}\texttt{getitems@terminalitem}” after, the text to help us detect empty items and prevent errors after we have found all the genuine \texttt{item}’s.

\begin{verbatim}
\long\def\gatheritems#1\
\setcounter{getitems@begindepth}{0}\%
\setcounter{numgathereditems}{0}\%
\xa\long\xa\gdef\csname getitems@item@0\endcsname{}
\gatheritems@int\getitems@relax#1\item\getitems@terminalitem\getitems@endgatheritems
\xa\let\xa\gatheredheader\xa=\csname getitems@item@0\endcsname
\}
\end{verbatim}

The trouble with the naïve strategy is that it won’t handle nested environments correctly. To do that, we need to keep track of how deeply nested we are with the macro \texttt{\getitems@trackbegindepth}, defined below. That macro stores its results in the \LaTeX{} counter \texttt{getitems@begindepth}; a value of 0 indicates the top-level within the argument of \texttt{\gatheritems}.

\begin{verbatim}
\def\@getitems@terminalitem{\getitems@terminalitem}
\def\@dummy@relax{\getitems@relax}
\long\def\gatheritems@int#1\item#2\getitems@endgatheritems{\getitems@trackbegindepth{#1}\%
\ifnum\c@getitems@begindepth=0\relax
At this point we have gathered a complete \texttt{item}; we have not stopped accidentally at a sub\texttt{item}. The original \texttt{item} might have had no content, in which case \texttt{#1} will be simply “\texttt{\getitems@relax}, and we do nothing; otherwise we strip off the \texttt{\getitems@relax} and store those tokens in \texttt{\getitems@item@\langle\texttt{numgathereditems}\rangle}.

\begin{verbatim}
\def\getitems@test@i{#1}\%
\if\xchar\getitems@test@i\@dummy@relax
\else
\g@addto@macro
\csname getitems@item@\the\c@numgathereditems\endcsname
{\getitems@stripfirsttokenfrom#1\getitems@endstrip}\%
\fi
\end{verbatim}

Now we test whether we have reached the end of the text to be parsed. This is the case if \texttt{#2} is simply \texttt{\getitems@terminalitem}, and we stop the recursion. Otherwise there is at least one more \texttt{item} to process, so we increment \texttt{numgathereditems}, prepare \texttt{\getitems@item@\langle\texttt{k+1}\rangle}, and prepare to recurse.

\begin{verbatim}
\def\getitems@test@ii{#2}\%
\if\xchar\getitems@test@ii\@getitems@terminalitem
\let\getitems@next=\relax
\else
\stepcounter{numgathereditems}\%
\gdef\csname getitems@item@\the\c@numgathereditems\endcsname{}
\def\getitems@next{\gatheritems@int\getitems@relax#2\getitems@endgatheritems}\%
\fi
\end{verbatim}

\end{verbatim}

\begin{verbatim}
\def\getitems@next=\relax
\else
\stepcounter{numgathereditems}\%
\gdef\csname getitems@item@\the\c@numgathereditems\endcsname{}
\def\getitems@next{\gatheritems@int\getitems@relax#2\getitems@endgatheritems}\%
\fi
\end{verbatim}

\end{verbatim}
We are now in the case where \texttt{getitems@begindepth} \neq 0. This essentially means that the text in \#1 has more \texttt{\begin}'s than \texttt{\end}'s, so we have not read a complete \texttt{\item}; we stopped at an “\texttt{\item}” token within a sub-environment. We save the text gathered so far to \texttt{\getitems@item@⟨k⟩}, including the \texttt{\item} we parsed by mistake, and then call \texttt{\gatheritems@int} again to sweep up more tokens.

\begin{verbatim}
35  \xaxa\g@addto@macro
36  \xaxa\csname getitems@item@\the\c@numgathereditems\endcsname
37  \xa\{\getitems@stripfirsttokenfrom#1\getitems@endstrip\%
38  \xag@addto@macro\csname getitems@item@\the\c@numgathereditems\endcsname{\item}\%
39  \def\getitems@next{\gatheritems@int\getitems@relax\#2\getitems@endgatheritems}\
40  \fi
41  \getitems@next
42 \}
\end{verbatim}

This next macro is used by \texttt{\gatheritems@int} to strip off a dummy \texttt{\getitems@relax} token from the beginning of its first parameter.

\begin{verbatim}
43 \long\def\getitems@stripfirsttokenfrom#1#2\getitems@endstrip{#2}
\end{verbatim}

Here is the code used to track the depth of nesting of \texttt{\begin}'s in a text.

\begin{verbatim}
44 \newcounter{getitems@begindepth}
45 \long\def\getitems@trackbegindepth#1{\
46  \getitems@trackbegindepth@int#1\getitems@terminalbegindepth\getitems@endtrackbegindepth
47 }
48 \def\@getitems@begin{\begin}\
49 \def\@getitems@end{\end}\
50 \def\@getitems@terminalbegindepth{\getitems@terminalbegindepth}\
51 \long\def\getitems@trackbegindepth@int#1#2\getitems@endtrackbegindepth{\
52  \def\getitems@test@i{#1}\
53  \ifx\getitems@test@i\@getitems@begin\
54     \advance\c@getitems@begindepth by 1\relax
55  \else
56     \ifx\getitems@test@i\@getitems@end\
57        \advance\c@getitems@begindepth by -1\relax
58     \fi
59  \fi
60  \def\getitems@test@ii{\#2}\
61  \trim@spaces@in\getitems@test@ii
62  \ifx\getitems@test@ii\@getitems@terminalbegindepth\
63     \let\getitems@trackbegindepth@next=\relax
64  \else
65     \def\getitems@trackbegindepth@next{\getitems@trackbegindepth@int\#2\getitems@endtrackbegindepth}\
66     \fi
67  \fi
68  \getitems@trackbegindepth@next
69 }
\end{verbatim}

\texttt{\loopthroughitemswithcommand} Finally, we define the user-level command to loop through the gathered items from 1 through \texttt{\numgathereditems}.

\begin{verbatim}
70 \newif\ifgetbeginningofloop
71 \newcounter{currentitemnumber}
\end{verbatim}

\texttt{\loopthroughitemswithcommand}