The Gamebooklib Package*

Robert J Lee
latex@rjlee.homelinux.org

May 27, 2022

Abstract

This package provides macros and environments to allow the user to
typeset a series of cross-referenced, numbered “entries”, shuffled into random
order

1 Introduction

This package was written to allow the typesetting of gamebooks.

A gamebook is a book divided into “entries” (which may be a single paragraph
or longer), each with a sequentially numbered value. At the end of each entry, a
link to one or more other numbered entries is given; the reader selects one and
they follow through the story in this order. Gamebooks traditionally start at the
entry numbered “1” and some gamebooks have multiple endings, with the final
section representing a success or victory for the reader.

In particular, this package handles two technical challenges which are tricky to
solve with \LaTeX{} directly: writing the “entries” in order but presenting them in a
randomised order; and presenting footnotes at the end of each “entry”, numbered
in the order in which they appear, not the order in which they are written.

Aside from shuffling the paragraphs and fixing the handling of footnotes, this
package provides little help for the actual typesetting of gamebook entries. For
this, please consider using André Miede’s gamebook package, available on CTAN\footnote{https://www.ctan.org/pkg/gamebook}.
That package is orthogonal to this one; both may be used together.

Terminology

For the purposes of this document, the term entry means a numbered section of
text. This term helps describe the common format of a gamebook, while avoiding
confusion with the terms paragraph and a section, both of which already have a
clear definition.

The term gamebook means a book consisting of numbered sections, typically
presented in a non-linear order with a tree or graph of possible reading options.

\footnote{This document corresponds to Gamebooklib Gamebook, dated 2022/05/25.}
Usage

To load the class, use `\usepackage[options]{gamebook}`. The class options are described below: `[footnotes, jukebox]` is normally a good choice, although there are some limitations.

The user simply writes entries like this inside the document:

```latex
\begin{gentry}{codea}
This is the first entry
\end{gentry}

Turn to `\turnto{codeb}`
\begin{gentry}
This is the second and final entry.\par
To play again, turn to `\turnto{codea}`
\end{gentry}
```

% Output:
\thegentries

Here, two entries are defined, and then output.

The first entry is given a label of `gentry:codea`, and the `\turnto{codea}` in the second entry generates a link back to that label.

As `entry` is a common term likely to conflict with other environments, the environment to declare an entry has been named `gentry` (gamebook entry) instead.

Sometimes it is convenient, when writing an entry, to give it a fixed index in the text. The first optional argument in the second entry — `[123]` — fixes the entry at that position during shuffling. This is implemented by iterating over each entry and swapping it with that index. Duplicates are not currently checked for, but they would not be lost; if two entries are defined with the same fixed index, one will end up somewhere else in the text.

The first and last entries are automatically fixed without needing to specify the optional number.

Specifying the optional number as a value greater than or equal to the number of entries, or less than 2, is not recommended. (Yes, this example used 123, which is beyond the number of entries. This was done to show the syntax).

The second optional argument is a title to be displayed alongside the numerical value.

Finally, the command `\thegentries` will output all entries defined so far, handling shuffling and footnotes.

Users may also want to redefine `\gentryheader` to change how the entries are displayed. The initial version is quite basic and intended for debugging; users should use `\renewcommand` to change it to something more pleasant.

Here's one suggestion, using packages from CTAN. It works well if you have either a title, or at least 2 lines of text before the first paragraph break inside the
gentry.

It also requires near the start of the document:

\usepackage{lettrine}
\usepackage{etoolbox}
\usepackage{color}

And this anywhere before \thegentries:

\renewcommand{\LettrineFontHook}{%
  \color[gray]{0.5}\fontfamily{ppl}\fontseries{bx}}
\renewcommand{\gentryheader}[4]{%
  \lettrine[lines=3,lhang=0.2,loversize=0.2]\raisebox{0.1em}{#1}%
  \textbf{\Large\textbf{\raisebox{0.3em}{~#4}}%
    \ifstrempty{#1}{}{\linebreak\mbox{}~}}%
\ifstrempty{#1}{}{\linebreak\mbox{}~}}

On the last line, \mbox stops the \parindent space being ignored; the ~ brings it back up to the expected spacing.

Package options

These options can be specified as a comma-separated list to the \usepackage line.

footnote The footnote option causes \LaTeX{} to output footnotes at the end of each entry, or the end of each page, whichever comes first after the footnote mark.

There are some limitations, described below. Support for other packages that affect footnotes is limited. If using hyperref, it is recommended to pass [hyperfootnotes=false] to avoid broken links.

jukebox Use the Jukebox Index shuffling algorithm\textsuperscript{2}. This is slightly slower, but tends to reduce the number of times you get a “turn to” instruction referencing the next (or previous) entry in the original order, by modifying the shuffle to ensure that adjacent gentries in the input have much less chance of being adjacent in the final document. jukebox requires a \LaTeX{} 2\epsilon that supports \numexpr, and a minimum of 6 gentries. If fewer than 6 gentry environments are supplied before \thegentries, this option will only log a warning in verbose mode.

The jukebox option guarantees no more adjacent entries than without the option, for a given seed value; it may not eliminate them completely unless the number of entries is large. The performance is linear to the number of entries.

noshuffle In general, it’s easier to write gamebooks in a more linear fashion, in which related entries are kept together. But this is much less fun to play, as it’s too easy for the reader to simply read the adjacent entries to decide what to do. For this reason,

\textsuperscript{2}https://github.com/robertjlee/jukeboxshuffle
this package shuffles the output entries by default. The first and last entries are never shuffled.

Sometimes, perhaps for proofreading, you may not want the entries to be shuffled. In this case, you can use the \texttt{noshuffle} option.

\textbf{verbose} This macro causes the package to output information messages about what it’s doing to the log file. This is not generally too useful, but it does include a mapping of the original paragraph indexes to their sorted positions, which may be useful to keep for handling proofreading corrections.

\textbf{endpage} Puts the last entry on a page on its own. This typically produces a better “winning” feel for reaching the last entry, but it can also produce documents with ugly spacing, so it’s recommended to try it each way and see which works better for your text.

\textbf{seed} It’s suggested that users specify a value for “seed” for stable builds, by adding the following before including this package: \texttt{\usepackage[seed=123]{lcg}}

\section*{Footnotes}

When typesetting a gamebook with footnotes, it is confusing if they migrate to the bottom of each page, as the footnote becomes visually detached from the entry to which it relates. Some effort has been taken to ensure that footnotes can be typeset at the bottom of the page on which the mark appears, or the bottom of each entry, whichever comes first.

However, this implementation has some limitations:

- Footnotes are not expanded until they are typeset.

- As a consequence, attaching one footnote to another with the use of \texttt{\footnotemark} within a \texttt{\footnotetext} argument will not advance the counter in time, so \texttt{\footnotetext} may not behave as you expect unless it is also set inside the first footnote’s text.

- Footnotes at the end of the page will not be broken across pages. Putting sufficient text into footnotes will cause the page to overflow.

- Where the footnote \texttt{mark} appears at the very end of a page, the footnote \texttt{text} may be set at the top of the subsequent page. Te\TeX{} is asked not to break the page there, but this influence is limited. It may be possible for the author to avoid this, eg by adding a rubber length to the interline spacing (it may be easier to simply choose a different \texttt{seed} value for the \texttt{lcg} package to reshuffle).

- If you have a lot of rubber space in the text, or variably-sized items, \TeX{} may expand the footnote at the end of the entry before it works out where to put the page break. In this case, the footnote will appear on the wrong page. The macro \texttt{\noentryfoot} is provided to allow the author to fix this case.
• Support for other footnotes packages may be limited and the behaviour of packages like \texttt{footnote}, \texttt{endnotes}, \texttt{footmisc}, \texttt{fnote}, \texttt{dblfnote} etc may be affected.

• If using the \texttt{hyperref} package, it is recommended to pass the option \texttt{hyperfootnotes=false} to that package, as the footnote links will be incorrect.

• The package uses \TeX\ \texttt{\mark}s to indicate which footnotes should appear on each page. Because \TeX\ supports only one set of marks, this would break any other package or usage of \texttt{\mark} while a gamebook was being output.

• Because \texttt{\mark} does not escape a floating environment, such as \texttt{\minipage}, it is likely that this footnote implementation will not work as expected if the gamebook or \texttt{\footnote} is set in a \texttt{\minipage} or similar.

• Each footnote is evaluated in a group. Just conceivably, this might affect the behaviour of some macros that affect the document beyond the footnote.

• We rely on some non-“public” macros, such as \texttt{\@mpfn} and \texttt{\@footnotetext} defined by \LaTeX\ standard document classes. Other document classes may override these, which would override this package’s footnotes too.

For this reason, the improved footnote handling is only enabled if you pass the \texttt{footnote} option, and only while the environment is active.

\section*{Implementation}

1 \texttt{\ProvidesPackage{gamebooklib}[2022/05/25 Gamebook by R Lee latex@rjlee.homelinux.org]}

We need \LaTeX\ \texttt{2\epsilon}, for the extra token registers.

2 \texttt{\NeedsTeXFormat{LaTeX2e}[1994/06/01]}

\subsection*{verbose}

The package option \texttt{verbose} enables detailed logging. Logging is via the macro \texttt{\gamebook@info}, which throws away detail messages unless the \texttt{verbose} option is given.

3 \texttt{\newcommand{\gamebook@info}[1]{}%}

4 \texttt{\DeclareOption{verbose}{%}

5 \texttt{\renewcommand{\gamebook@info}[1]{\PackageInfo{gamebooklib}{#1}}%}

6 \texttt{\gamebook@info{Gamebook Library package is logging}}%}

\subsection*{endpage}

The \texttt{endpage} option puts the last entry on its own page. This can work better when the last entry is about a page long, and also the final “winning” entry of the gamebook.

7 \texttt{\newcommand{\gamebook@beforelast{}}}

8 \texttt{\DeclareOption{endpage}{%}

9 \texttt{\renewcommand{\gamebook@beforelast{\eject}}%}

10 }%
The jukebox option defines the \gamebook@jukebox macro; while the macro does nothing, \ifcsname can then be used to determine if the option was set.
\begin{verbatim}
\DeclareOption{jukebox}{% 
  \newcommand\gamebook@jukebox{}% 
  \gamebook@info{Gamebook Library to perform jukebox index reshuffle pass}% 
}\end{verbatim}

The footnote option enables our footnote processing, to throw out footnotes at the end of each entry so that they don’t appear to be against subsequent entries for that page.

This is generally recommended, unless you have a reason to turn it off (such as a conflicting package). It’s disabled by default because it could cause unexpected faults.
\begin{verbatim}
\def\if@gamebook@footnotes{\iffalse}
\DeclareOption{footnote}{% 
  \gdef\if@gamebook@footnotes{\iftrue}% 
  \gamebook@info{Gamebook Library footnotes per gamebook entry}% 
}\end{verbatim}

All unknown options are passed to lcg, as it’s our only dependency with options.
\begin{verbatim}
\DeclareOption*{% 
  \PassOptionsToClass{\CurrentOption}{lcg}% 
}\end{verbatim}

We need to capture environment contents
\begin{verbatim}
\RequirePackage{environ}%
\RequirePackage{macroswap}%
\RequirePackage{ifthen}%
\RequirePackage{silence}%
\end{verbatim}

Macroswap: used to swap the commands that evaluate the macros
\begin{verbatim}
\RequirePackage{macroswap}%
\end{verbatim}

Ifthen makes branching and loops a little easier
\begin{verbatim}
\RequirePackage{ifthen}%
\end{verbatim}

LCG: random numbers for the shuffle
\begin{verbatim}
\RequirePackage{lcg}%
\end{verbatim}

Silence is used to suppress a warning from lcg that gamebook is not wasting counter registers. debrief ensures that the user is at least told that warnings were suppressed.

To see the warnings, put the following line before \usepackage{gamebook}:
\begin{verbatim}
\usepackage[debrief,showwarnings]{silence}:
\end{verbatim}
\begin{verbatim}
\RequirePackage[debrief]{silence}%
\end{verbatim}
Let’s count the entries we’re reading in so we can build up token registers.

\newcounter{gentryctr}\setcounter{gentryctr}{0}\newcommand{\gentry}[1][\{}{\@gentry{#1}}\gentry just makes the first argument mandatory.

NB: If you define two entries with the same code, \LaTeX{} will print out a "multiply defined" label warning.

\newenvironment{@gentry}[2]{\xdef\gentryidx{#1}\xdef\gentrycode{#2}\@@gentry}{\ignorespacesandallpars\global\let\gentryidx\@undefined\global\let\gentrycode\@undefined\global\let\gentryidxu\@undefined\global\let\gentryidxs\@undefined}

\newcommand{\@@gentry}[1][\{}{\def\gentrytitle{#1}\stepcounter{gentryctr}\ifthenelse{\equal{\gentryidx}{}}{}{\expandafter\xdef\csname fixedat\arabic{gentryctr}\endcsname{\gentryidx}\expandafter\global\expandafter\newtoks\expandafter{\csname paratok\arabic{gentryctr}\endcsname}{ }\Collect@Body\gentry@store}}

\newenvironment{gentry}[1][\{}{\begin{gentry}\begin{gentrystore}}\end{gentrystore}\end{gentry}}{\end{gentry}\begin{gentrystore}}

\gentryidx This can be used inside a \gentry{}; it expands to the first optional argument of the \gentry{} environment, which is either blank or the requested fixed index of the entry. To get the actual shuffled index, use \gentryidxu.

\gentrycode This can be used inside a \gentry{}; it expands to the first mandatory argument of the \gentry{} environment, which is the code for this entry (without the \gentry{} prefix).

\@@gentry then reads in the optional title argument, storing it in the \gentrytitle macro to supply the unsorted index number and the current entry’s code respectively.

\newcommand{\@@gentry}[1][\{}{\def\gentrytitle{#1}\stepcounter{gentryctr}\ifthenelse{\equal{\gentryidx}{}}{}{\expandafter\xdef\csname fixedat\arabic{gentryctr}\endcsname{\gentryidx}\expandafter\global\expandafter\newtoks\expandafter{\csname paratok\arabic{gentryctr}\endcsname}{ }\Collect@Body\gentry@store}}

\gentryidxu This was supposed to discard any blank lines at the end of the \gentry{} environment, but it still left odd spacing in the output somehow, and sometimes produced weird error messages about \inaccessible{} and \head. Removed for now.
Store the collected environment contents for `\theentries` to output: This uses the token register \texttt{gentry\textsubscript{N}}, where \texttt{N} is the unsorted index of this entry. Later, we'll change the values of the \texttt{N} bit of \texttt{gentry\textsubscript{N}} macros when we shuffle (the underlying token registers stay the same).

This relies on `\refstepcounter{\langle gentryctr \rangle}` being expanded before this macro. The label \texttt{gentry:}\texttt{gentrycode} is thus set to the current index of the final output entry.

\newcommand{\gentry@store}[1]{%}
\edef\head{%
  \begingroup
  \bgroup\gentryheader{\arabic{gentryctr}}{\gentryidx}{\gentrycode}{\gentrytitle}%
  \label{gentry:}\gentrycode%}
\global\expandafter\csname paratok\arabic{gentryctr}\endcsname=%
Output the header, the environment token list, flush any footnotes (if applicable) then the inter-gentry footer.
\expandafter{\head #1%
\outputfootnotes\@endgentry%
\gentryfooter\endgroup}%
This macro is executed inside the \texttt{gentry} group and sets up the commands to be run to allow footnotes. It is only expanded if `\if@gamebook@footnotes` is true, i.e. the \texttt{footnotes} option is given.
\newcommand{\gentry@footnotespergentry}{%}
This macro shuffles the entries as required, then expands to them in the correct order.
\newcommand{\theentries}{%}
This command is in a group so that the output routine resets. The expansion of \texttt{\gentry@footnotespergentry} only happens if the \texttt{footnote} option was given; it sets up the output routine while the gamebook is running.
\begingroup%
\if@gamebook@footnotes\gentry@footnotespergentry\fi%
To begin, let's record the number of entries. This may come in useful. Note that you can use this macro in the \texttt{gentry} environment, because that's not been expanded yet.
\xdef\gentrycount{\arabic{gentryctr}}%

The next thing is to perform some surgery on LCG. This cuts out an annoying warning, hopefully more reliably than replacing the definition of `\p@stkeysr@nd`. The warning is simply that this package doesn't waste another counter every time it changes the random limits (which happens a lot during the Fisher-Yates shuffle):
\WarningFilter{lcg}{Using an already existing counter rand}%
The output routine and end-of-page footnotes

The idea is to keep footnotes always on the same page as their mark where possible. E\TeX does lots of fun things with the output routine, which we want to keep. So grab a copy of whatever the code is currently doing:

Here I’m using the \edef trick to expand \theoutput into a token register, because using a macro causes a weird error about an “{” after “\the”.

\edef\mytmp@{\noexpand\gentry@oldoutput={\the\output}}\mytmp@

For the footnotes, if we reach the end of a page without outputting them, we need to flush them.

\output is the output routine. It takes the page built up in \box255, annotates it with headers, footers etc, then ships it out. For our purposes, we only need to append the footnotes to the bottom of \box255.

\if@gamebook@footnotes\output={%  
  \def\gentry@deferoutput{\the\gentry@oldoutput}%  
  The \outputpenalty tells us why the output routine was called; generally, it’s invoked whenever a new floatable environment is generated, or when a page is full. Anything less than -1000 means that the page was filled, so we should add any footnotes only in this case.
  \ifnum\outputpenalty<\@M%  
    \if\gentryshouldoutput0%  
      \unvbox255\def\gentry@deferoutput{}%  
    \else%  
      \expandafter\ifcsname footnotetoks\botmark\endcsname%  
        \expandafter\if\expandafter\relax\expandafter%  
          \detokenize\expandafter{\csname footnotetoks\botmark\endcsname}\relax\else%  
            \global\setbox255=\vbox to \vsize{%  
              \unvbox255\vfill\outputfootnotes@endpage}%  
          \fi\fi%  
    \fi\fi%  
  \gentry@deferoutput%  
  \the\gentry@oldoutput%  
  \fi%  
}

The Shuffling Algorithm

The basic shuffling algorithm is to first shuffle all entries, except for those marked with a fixed index, then to go through the fixed-index entries in order and swap them into their final place.

The original version of this package had a bug relating to multiple fixed-index entries (now fixed). In short, let A, B, and C be indices; if A < B and (unshuffled) entry number A was fixed at (shuffled) location B, while (unshuffled) entry number B was fixed at (shuffled) location C, so during the “shuffle,” A would be swapped with B, then B would be swapped with C, resulting in A appearing at C in the text, not B as requested. Because the unshuffled index doesn’t appear in either
the source or output document, this could be difficult to diagnose; the author
simply saw one of their entries ending up in the wrong place.

\TeX has well beyond 255 token registers these days, so don’t bother to check
that limit.

The LCG package provides a suitable pseudo-random number generator. What
we want is a repeatable series of disparate numbers, not an especially random one.

1. Work out how many entries there are \((N)\). Provided \texttt{\textbackslash thegentries} is called
at the end, this is just the value of \texttt{\textbackslash gentryctr}

2. Declare a set of token registers named \texttt{\paratoks_n}, where \(n\) is each integer
1\ldots N inclusive. These will hold the contents of the entry.

3. Declare a set of macros named \texttt{\paraidx_n}, where \(n\) is each integer 1\ldots N
inclusive, each initialised to \(n\). These will hold the number of the entry.

4. Shuffle elements \(\{2 : N - 1\}\), in that array. For \(i = 2\) through \(N - 2\)
   \begin{enumerate}
   
   \item Let \(R\) be a random number between \(i\) and \(N - 1\) inclusive
   \item If \(R \neq i\) then swap macros \texttt{\textbackslash gentrytoks_R} and \texttt{\textbackslash gentrytoks_i}
   \item If \(R \neq i\) then swap macros \texttt{\paraidx_R} and \texttt{\gentryidx_i}
   \end{enumerate}

5. If a jukebox index sort is requested, perform an optimisation pass (see below)

6. For \(i = 1 : n\), output token reg \(i\)

Define macros \texttt{\textbackslash csname paraIdx_n\textbackslash endcsname} containing the arabic original
gentry number to be put out on the \(n\)th output gentry.

\begin{verbatim}
\gentrycount
\setcounter{gentryctr}{0}%
\ifthenelse{\gentrycount<3}{}{%SHUFFLE START
\edef\gentryidxu{\arabic{gentryctr}}%
\expandafter\xdef\csname paraIdx\gentryidxu\endcsname{\gentryidxu}%
\typeout{DEFINED paraIdx\gentryidxu}%
\stepcounter{gentryctr}%}
\end{verbatim}

\gentryidxu The \texttt{\gentryidxu} macro can be used inside a \texttt{\gentry} to obtain the current arabic
shuffled index of the entry.

\gentryidxs The \texttt{\gentryidxs} macro can be used inside a \texttt{\gentry} to obtain the arabic original
unshuffled index of the entry (the first \texttt{\gentry} is 1, and this counter resets after
each expansion of \texttt{\textbackslash thegentries}).
First, shuffle everything that isn’t fixed down. Don’t renumber para 1 or \gentrycount; Fisher-Yates-shuffle the rest NB: we stop at \gentrycount−2, because \gentrycount−1 would only shuffle with itself.

If this is to be swapped with a fixed position, skip it

Roll the dice. If we’ve hit an entry with fixed position, we must skip it, or it would end up being swapped out into fixedat\arabic{rand} instead.

Now move fixed entries into their final place:

Edge case: It’s possible that we also have fixedat\mydest; in which case that would be messed up with the reshuffling. So we need to rename that to fixedat\gentryidxu as well, then reprocess this index

113 \expandafter\ifcsname fixedat\gentryidxu\endcsname
114 \gamebook@info{MOVING FIXED GAMEBOOK ENTRY INTO PLACE: \gentryidxu -> \mydest}\%
115 \macroswap{paraIdx\gentryidxu}{paraIdx\expandafter\csname fixedat\gentryidxu\endcsname}\%
116 \fi\%
117 \fi\%
118 %
119 Now move fixed entries into their final place:
120 \setcounter{gentryctr}{2}\%
121 \whiledo{\not{\value{gentryctr}\\stoppoint}}{%
122 \edef\gentryidxu{\arabic{gentryctr}}\%
123 \expandafter\ifcsname fixedat\gentryidxu\endcsname\%
124 \gamebook@info{MOVING FIXED GAMEBOOK ENTRY INTO PLACE: \gentryidxu -> \mydest}\%
125 \macroswap{paraIdx\gentryidxu}{paraIdx\arabic{rand}}\%
126 \fi\%
127 %
128 %
129 %
130 %
if we are doing a jukebox shuffle, remember which final entries are fixed, so they
don’t get moved.

The jukebox shuffle requires an extra pass. This must come after moving fixed
entries into their final place, to allow us to compare the initial indicies. We make
a reasonable effort:

- t is the current index
- u is the next index
- r is a random index after u (make 3 attempts to find a non-fixed r)
- if abs(t − u) = 1 and r is not fixed, then swap u and r if u is not fixed; otherwise:
  - if abs(t − u) = 1 and r is not fixed, then swap t and r if t is not fixed;
  - otherwise, give up.
Now we can output the \texttt{gentry} token registers to let \LaTeX{} do its thing:

\begin{verbatim}
\setcounter{gentryctr}{0}
\gamebook@info{Outputting \gentrycount{} gamebook entries}
\whiledo{\value{gentryctr}<\gentrycount{} }{
  \stepcounter{gentryctr}
  \edef\gentryidxu{\arabic{gentryctr}}
  \edef\gentryidxs{\csname paraIdx\gentryidxu\endcsname}
  \gamebook@info{Output gentry \gentryidxu{} of \gentrycount{}, original idx \gentryidxs}
  \the\csname paratok\gentryidxs\endcsname
}
\end{verbatim}

Finally, we clear the registers and reset the counter in case we want to start again (NB: fixedto is scope to the current block only, so no need to clear that).

\begin{verbatim}
\setcounter{gentryctr}{1}
\end{verbatim}
\whiledo{\not{\value{gentryctr} \gt \gentrycount}}{%
  \edef\gentryidxu{\arabic{gentryctr}}%
  \expandafter\global\expandafter\let\csname paratok\gentryidxu\endcsname\@undefined%
  \expandafter\ifcsname fixedat\gentryidxu\endcsname%
    \expandafter\global\expandafter\let\csname fixedat\gentryidxu\endcsname\@undefined%
  \fi
  \stepcounter{gentryctr}%
}\setcounter{gentryctr}{0}%
\eject%
\endgroup%

The final \eject ensures that the output routine has flushed as many pages as it can, before the output routine is reset again.

}{{⟨counterIdx⟩}{⟨fixedIdx⟩}{⟨code⟩}{⟨title⟩}}

This macro is called before outputting the header. Its job is to format whatever header information the user wants to see on each entry; generally, this will be the page number.

It takes 4 parameters:

1. The unshuffled index value (to help an author find an entry in the original text)
2. The fixed index value, if any; otherwise an empty argument
3. The user-entered unique code for this entry
4. The user-supplied title, if any; otherwise an empty argument

The value of the output index can be obtained with \arabic{gentryctr}

\gentryheader

\gentryshouldoutput This macro may be called from the output routine, and can be used to suppress page breaks. It was added because it proves fairly easy to write custom divider routines that can produce blank pages. If it expands to the number 1, then the page will be output; if it expands to 0, it will not be.

For example, the following will prevent pages that are less than 80% full.

\renewcommand{\gentryshouldoutput}{%\ifdim\pagetotal>0.8\pagegoal\relax1\else 0\fi}
Caution: if this macro continually tests false, then material will eventually be
discarded from the main vertical list to ensure that \TeX can complete the output of
the document. If you redefine this macro, make sure to check out output carefully
for missing text.

\newcommand{gentryshouldoutput}{1}

This takes no arguments and is simply expanded after the entry is typeset.
The default adds some vertical space and a simple separator.

\newcommand{gentryfooter}{%
\par\vspace{2em}\centerline{---}\vspace{2em plus 1in}\par%
}%

This is a technique described on StackExchange\(^3\).

This is used to ensure that extra space at the start and end of the \texttt{gentry}
environment is ignored.

\def{ignorespacesandallpars@}{%\@ifnextchar\par{%\expandafter{ignorespacesandallpars\@gobble}{}%}

\langle \text{token register} \rangle \text{ Token registers to hold each footnote.}

Each footnote is held in its own token register, so that we can control which
footnotes appear on each page.

\LaTeX would normally use saveboxes to store footnotes, but I prefer to hold off
on expanding them until we know which page they’re to be expanded on, which
saves some difficulties (and perhaps creates others).

To start with, modify \LaTeX’s \texttt{@makefnmark} to output a mark for the footnote,
holding its index. This tells the output routine which footnotes to include at the
end of the page. The \texttt{in@out} macro is only defined when outputting footnotes,
so suppresses this mark when we don’t need it.

\g@addto@macro{@makefnmark}{%\ifcsname in@out\endcsname\else%\mark{\arabic{\@mpfn}}%\fi%}

The \texttt{@footnotetext@save} is a convenience that takes the the footnote counter
value in the first argument, the token register as a second argument, and the token
list (footnote text) as the third.

\g@addto@macro{gentry@footnotespergentry}{%\newcommand{@footnotetext@save}[3]{%\global{newtoks#2{}}\global#2={\noindent@footnotemark{}#3}}%}

\(^3\)https://tex.stackexchange.com/questions/179016/
ignore-spaces-and-pars-after-an-environment#179034
\@footnotetext@save unpacks the arguments for \@footnotetext@save. This one uses the value of the \@mpfn counter to unpack the footnote counter, the token register (full csname) as the first argument, and the token list (footnote text) as the second.

\@addto@macro{\gentry@footnotespergentry}{% 
\newcommand{\@footnotetext@save}[2]{% \edef\@tmp{\expandafter\arabic{\@mpfn}}% \expandafter\@footnotetext@save\expandafter{\@tmp}{#1}{#2}% }}%

Footnotes are built in a group, so that \in@out can be defined locally to indicate that footnotes are being built (which stops spurious \mark).

\@addto@macro{\gentry@footnotespergentry}{% 
\renewcommand{\@footnotetext}[1]{% \begingroup% \def\in@out{}% flag that we’re building footnotes \edef\@tmp{\expandafter\csname footnotetoks\arabic{\@mpfn}\endcsname}% \expandafter\@footnotetext@save\expandafter{\@tmp}{#1}% % This fixes up the use of \cs{footnotemark} within footnotes: #1% \endgroup% }}%

\outputfootnotes\{⟨maxIdx⟩\}
Command to output the footnotes, which are just in \footnotetoks N for now.
An end user can call this at any point in the text to set out the footnotes.
This macro takes one argument, being the maximum index of footnotes to output, inclusive. Footnotes after this index will be excluded. If not provided, the value provided by the counter \@mfpn, which is the default \LaTeX counter, will be used.

\newcounter{fncounter} %
\newcommand{\outputfootnotes}[1]{% \begingroup% \def\in@out{}% flag that footnotes are outputting; suppresses marks \setcounter{fncounter}{1}% Called in vertical mode, and don’t want to throw a page break.
I’m not sure how \LaTeX renders the footnote rule exactly, so I’m just using my own.
\def\footnote@rule{% \leftskip=0pt\rightskip=0pt\interlinepenalty=1000% \vspace{1pt plus 2pt minus 0.5pt}\
\hspace{-0.5in}\rule{1.5in}{0.4pt}\%}
Invoke the output routine. This attempts to stop the output routine being invoked while we’re adding the footnote rule, which could cause a blank footnote to appear.
Instead, it means some footnotes could appear on the subsequent page. To guard against this, footnotes are set into a \vbox to prevent page breaking.

The macro \footnote@rule is output before each footnote. The first time, it outputs a divider rule; subsequently, it just throws a new paragraph to keep footnotes on separate lines.

The next line comes from the TUGboat suggestions, and protect against various user changes

\begin{verbatim}
\outputpenalty=\@MM\break%
\vbox{% 
\whiledo{\not{\value{fncounter}>#1}}{% 
  We are now looping over all possible entry numbers, in order. Some will already have been output, but we check them all anyway (it doesn’t take much time).

  First, we check if the macro exists. If it does, it must contain the value of the token register.

  \expandafter\ifcsname footnotetoks\arabic{fncounter}\endcsname% 
  \edef\tmp@@{\csname footnotetoks\arabic{fncounter}\endcsname}% 
  This \detokenize black magic tests if a token reg is actually empty\footnote{4}
  \expandafter\if\expandafter\relax\expandafter% 
  \detokenize\expandafter{\the\tmp@@}\relax\else% 
  \footnote@rule\gdef\footnote@rule{\ifvmode\else\par\fi}% 
  Actually output the footnotes.

  Having output the footnote, clear the token register (to save memory), then use \let to clear the definition of the macro. This ensures that we don’t try to output the same footnote twice (at page end and entry end).

  \expandafter\the\tmp@@% 
  \global\expandafter\tmp@@={}% 
  \expandafter\let\tmp@@\@undefined% 
  \fi\fi% 
  \stepcounter{fncounter}% 
}\endgroup% 
}\end{verbatim}

LaTeX works by building up a little more than a page, then calling the output routine. The output routine then decides where to put the page end from the built-up material. If there is a footnote mark, it could come after the page end, so we can’t rely on the fact that there’s footnote register to determine if we should output footnotes or not. The edge case is: if the footnote mark is held back for the next page, the footnote text would appear on the footer of the page where being built when the mark was expanded, which is the page before the footnote.

\footnote{4https://tex.stackexchange.com/questions/263733/whats-the-best-practice-way-to-test-whether-parameter-is-empty}
One fix for this is to have each footnote mark in the main body of the text output a `\mark` containing the footnote’s counter value, and output the footnotes only to `\botmark`, which is the last `\mark` actually typeset on the page.

(see https://www.tug.org/TUGboat/Articles/tb11-4/tb30salomon.pdf, page 598, to read around footnotes).

This macro is called at the end of each `\gentry`. It switches into vertical mode (as ending `gentry` doesn’t throw a `\break` or line end by itself), then we output all footnotes so far (by reading the footnote counter `\@mpfn`, as this is “sequential”).

```latex
\newcommand{\outputfootnotes@endgentry}{%\if@gamebook@footnotes%\nopagebreak\ifhmode\fi% get into vertical mode\nopagebreak\outputfootnotes{\arabic{\@mpfn}}%\fi%}%
```

This method simply suppresses footnotes at the end of the entry within the current group. This forces footnotes within the current group to be printed at the bottom of the page.

This is useful in the case where \LaTeX\ expands the footnote at the end of the entry, then decides to put the page split between the footnote mark and the text.

```latex
\newcommand{\noentryfoot}{\def{\outputfootnotes@endgentry}{}}%
```

This is called by the output routine at the end of each page. If `\botmark` has a value, then we can output all footnotes up to that index.

```latex
\@addto@macro{\gentry@footnotespergentry}{%\newcommand{\outputfootnotes@endpage}{%\expandafter{\if\expandafter{relax}\expandafter{\relax{\expandafter{\if{\expandafter{\relax}\if{\relax}##1\relax}}}\relax\else%\outputfootnotes{##1}\relax\else%\fi%}}}%}}%
```

### Change History

**v1.0**

<table>
<thead>
<tr>
<th>General: Initial Release</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>endpage</code>: Endpage option</td>
<td>4</td>
</tr>
<tr>
<td><code>footnote</code>: Footnote option</td>
<td>3</td>
</tr>
<tr>
<td><code>noshuffle</code>: Noshuffle option</td>
<td>3</td>
</tr>
<tr>
<td><code>verbose</code>: Verbose option</td>
<td>4</td>
</tr>
</tbody>
</table>

**v1.1**

<table>
<thead>
<tr>
<th>General: Bugfix: edge case with clashing fixed-index entries</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gentryidxs</code>: Access original index</td>
<td>10</td>
</tr>
</tbody>
</table>

**v1.2**

| General: Bugfix: footnotes set justified on last line | 1 |

**v1.3**

<table>
<thead>
<tr>
<th><code>gentryshoudoutput</code>: Suppress short pages option</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>jukebox</strong>: Feature: “jukebox”</td>
<td>3</td>
</tr>
</tbody>
</table>
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.

Symbols
\@0gentry 41, 48
\@footnotemark 249
\@footnotetext 247, 254
\@footnotemark@save 252, 261
\@gentry 37
\@makefnmark 241
\@mpfn 243, 253, 260, 298
\@0 204, 225
\chgrand 108, 118, 165
\curpos 153, 157, 159, 164, 174
\eject 9, 221
\endpage 1, 2
environments:
\gentry 37
\F
\footnote 1, 16
\footnotes@rule 271, 284
\G
\gamebook@beforelast 7, 9, 200
\gamebook@info 3, 5, 6, 13, 19, 24, 112, 115, 121, 132, 148, 163, 170, 174, 177, 181, 188, 192, 196, 204, 208
\gamebook@jukebox 12
\gentry 37
\gentry@deferoutput 79, 82, 91
\gentry@footnotespergentry\in@out 259, 260
\jukebox \l 11
\M
\macros 122, 133, 136, 178, 183
\mydest 130, 132, 135–137, 141
\N
\nextidx 117, 118
\nextpos 156, 157, 159, 164, 175
\noentryfoot 18, 301
\noshuffle \l 21
\O
\output 77, 78
\outputfootnotes 16, 267, 298, 306
\outputfootnotes@endgentry 18, 67, 295, 301
\outputfootnotes@endpage 18, 88, 303
\P
\pdfinfo 157, 158, 161
\S
\sdiff 159–161
\seed \l 26
\space 164, 174, 175, 177, 181, 196
\stoppoint 106, 108, 109, 127, 151
\T
\thediff 161, 162
\thegentries 8, 71
\typeout 100
\V
\verbose \l 3