AcroTEX.Net

The ran_toks Package

Randomizing the order of tokens

D. P. Story

www.acrotex.net
Version 1.0

Table of Contents

1 **Introduction** 3

2 **The Preamble and Package Options** 3

3 **The main commands and environments** 3

3.1 The \ranToks command command 3

3.2 The \bRTVToks/\eRTVToks pair of commands 5

4 **Additional arguments and commands** 6

1. Introduction

This is a short package for randomizing the order of tokens. The package is long overdue; users of **AeB** and of eqexam have long asked for a way to randomize the order of the problems in a test or quiz, or anything for that matter.

The example file for this package is `ran_toks.tex` found in the `examples` folder. The file reproduces the sample code of this manual.

2. The Preamble and Package Options

The preamble for this package is

```
\usepackage{ran_toks}
```

The package itself has no options.

The requirements for ran_toks are the verbatim package (part of the standard LaTeX distribution, and the macro file `random.tex`, by Donald Arseneau

3. The main commands and environments

There are two styles for defining a series of tokens to be randomized, using either the `\ranToks` command or the `\bRTVToks`/`\eRTVToks` pair. Each of these is discussed in the next two subsections.

3.1. The \ranToks command command

The `\ranToks` command was the original concept; declare a series of tokens to be randomized.

```
\ranToks{name}{%
    {token_1}
    {token_2}
    ...
    {token_n}
}
```

were *token_k* is any non-verbatim content;[1] each token is enclosed in braces (`{}`), this is required. The *name* parameter is required, and must be unique for the document; it is used to build the names of internal macros. Of course several such `\ranToks` can be used in the document, either in the preamble or in the body of the document. Multiple `\ranToks` commands must have a different *name* parameter.

After a `\ranToks` command has been executed, the number of tokens counted is accessible through the `\nToksFor` command,

[1] Any token that can be in the argument of a command.

```
\nToksFor{name}
```

The one argument is *name*, and will expand to the total number of tokens listing as argument in the \ranToks command by the same name.

The \ranToks command does not display the randomized tokens, for that the command \useRanTok is used.

```
\useRTName{name}
\useRanTok{1}
\useRanTok{2}
...
\useRanTok{n}
```

The argument of \useRanTok is a positive integer between 1 and \nToksFor{*name*}, the number of tokens declared by \ranToks, inclusive. There is no space created following the \useRanTok command, so if these are to be used "inline", enclose them in braces ({}), eg, {\useRanTok{1}}. The use of \useRTName is optional unless the listing of the \useRanTok commands is separated from the \ranToks command that defined them by another \ranToks command of a different name. That should be clear!

Consider this example.

```
\ranToks{myPals}{%
    {Jim}{Richard}{Don}
    {Alex}{Tom}{J\"{u}rgen}
}
```

I have 6 pals, they are Alex, Jim, Tom, Jürgen, Don and Richard. (Listed in the order of best friend to least best friend.) The verbatim listing is,

```
I have {\nToksFor{myPals}} pals, they are \useRanTok{1},
\useRanTok{2}, \useRanTok{3}, \useRanTok{4}, {\useRanTok{5}}
and \useRanTok{6}.
```

Notice that \useRanToks are not enclosed in braces for 1–4 because they are each followed by a comma; the fifth token, {useRanTok{5}}, is enclosed in braces to generate a space following the insertion of the text.

Repeating the sentence yields, "I have 6 pals, they are Alex, Jim, Tom, Jürgen, Don and Richard" the exact same random order. To obtain a different order, re-execute the \ranToks command with the same arguments. Doing just that, we obtain, "I have 6 pals, they are Jürgen, Alex, Don, Tom, Jim and Richard." A new order? For most applications, re-randomizing the same token list in the same document is not very likely something you need to do.

My original application for this, the one that motivated writing this package at long last, was the need to arrange several form buttons randomly on the page. My point is that the listing given in the argument of \ranToks can pretty much be anything that is allowed to be an argument of a macro; this would exclude verbatim text created by \verb and verbatim environments.

3.2. The \bRTVToks/\eRTVToks pair of commands

Sometimes the content to be randomized is quite large and contain verbatim text. For this, it may be more convenient to use the \bRTVToks/\eRTVToks command pair. The syntax is

```
\bRTVToks{name}
\begin{rtVW}
    content
\end{rtVW}
...
...
\begin{rtVW}
    content
\end{rtVW}
\eRTVToks
```

The \bRTVToks{*name*} command begins the (pseudo) environment and is ended by \eRTVToks. Between these two are a series of rtVW (random toks verbatim write) environments. When the document is compiled, the contents of each of these environments is written to the disk drive and saved under a different name (based on the parameter *name*). Later, using the \useRanTok commands, they are input back into the document in a random order.

The use of \useRTName and \useRanTok were explained and illustrated in the previous section. Let's go to the examples,

```
\bRTVToks{myThoughts}
\begin{rtVW}
\begin{minipage}[t]{.67\linewidth}
Roses are red and violets are blue,
I've forgotten the rest, have you too?
\end{minipage}
\end{rtVW}
\begin{rtVW}
\begin{minipage}[t]{.67\linewidth}
I gave up saying bad things like
\verb$#%%%^*%^&#$@# when I was just a teenager
\end{minipage}
```

```
\end{rtVW}
\begin{rtVW}
\begin{minipage}[t]{.67\linewidth}
I am a good guy, pass it on The code for this last sentence is,
\begin{verbatim}
%#$% I am a good guy, pass it on ^&*&^*
\end{verbatim}
How did that other stuff get in there?
\end{minipage}
\end{rtVW}
\eRTVToks
```

OK, now, let's display these three in random order. Here we place them in an `enumerate` environment.

1. I gave up saying bad things like $#%%%^*%^&#$@# when I was just a teenager
2. I am a good guy, pass it on! The code for this last sentence is,

   ```
   %#$% I am a good guy, pass it on! ^&*&^*
   ```

   How did that other stuff get in there?
3. Roses are red and violets are blue, I've forgotten the rest, have you too?

The verbatim listing of the example above is

```
\begin{enumerate}
    \item \useRanTok{1}
    \item \useRanTok{2}
    \item \useRanTok{3}
\end{enumerate}
```

4. Additional arguments and commands

The syntax given earlier for \useRanTok was not completely specified. It is

\useRanTok[*name*]{*number*}

The optional first parameter specifies the *name* of the list from which to draw a random token; the *number* is the number of the token in the range of 1 and \nToksFor{*name*}, inclusive. The optional argument is useful in special circumstances when you want to mix two random lists together.

To illustrate: Jürgen, I gave up saying bad things like $#%%%^*%^&#$@# when I
was just a teenager

The verbatim listing is

```
To illustrate: \useRanTok[myPals]{1}, \useRanTok[myThoughts]{1}
```

This is a rather bad illustration. Every time this document is compiled there is a different page break on page 6. This must be used wisely.

The original order of the list of tokens is not lost, you can retrieve them using the command \rtTokByNum,

```
\rtTokByNum[name]{number}
```

This command expands to the token declared in the list named *name* that appears at the *number* place in the list. (Rather awkwardly written.) For example, my really best pals are Don and Alex, but don't tell them. The listing is,

```
For example, my really best pals are {\rtTokByNum[myPals]{3}}
and \rtTokByNum[myPals]{4}, but don't tell them.
```

In some sense, \rtTokByNum[*name*] acts like a simple array, the length of which is \nToksFor{*name*}, and whose k^{th} element is \rtTokByNum[*name*]{k}.

The randomization may be turned off using \ranToksOff or turned back on with \ranToksOn.

```
\ranToksOff
\ranToksOn
```

This can be done globally in the preamble for the whole of the document, or in the body of the document just prior to either \ranToks or \bRTVToks. For example,

```
\ranToksOff
\ranToks{integers}{ {1}{2}{3}{4} }
\ranToksOff
```

This should make \useRanTok{3} $= $ \rtTokByNum{3} $= 3$; executing this code, we get, $3 = 3 = 3$? As anticipated.

The command \ranToksOff is probably best in the preamble to turn off all randomization while the rest of the document is being composed.

The ran_toks package writes an auxiliary file named \jobname_rt.sav, below represents two typical lines in this file.

```
1604051353 % initializing seed value
5747283528 % last random number used
```

The first line is the initializing seed value used for the last compilation of the document; the second line is the last value of the pseudo-random number generator used in the document.

Normally, the pseudo-random number generator provided by `random.tex` produces a new initial seed value every minute. So if you recompile again before another minute, you'll get the same initial seed value.

To obtain a new initial seed value each time you compile, place `\useLastAsSeed` in the preamble.

```
\useLastAsSeed
```

When the document is compiled, the initial seed value taken as the second line in the `\jobname\_rt.sav` file, as seen in the above example. With this command in the preamble, a new set of random numbers is generated on each compile. If the file `\jobname_rt.sav` does not exist, the generator will be initialized by its usual method, using the time and date.

The command `\useThisSeed` allows you to reproduce a previous pseudo-random sequence.

```
\useThisSeed{init_seed_value}
```

This command needs to be placed in the preamble. The value of *`init_seed_value`* is an integer, normally taken from the first line of the `\jobname\_rt.sav` file.

When creating tests (possibly using eqexam), the problems, or contiguous collections of problems, can be randomly ordered using the \bRTVToks/\eRTVToks paradigm. For example, suppose there are two classes and you want a random order (some of) the problems for each of the two classes. Proceed as follows:

1. Compile the document, open `\jobname_rt.sav`, and copy the first line (in the above example, that would be 1604051353).
2. Place `\useThisSeed{1604051353}` in the preamble. Compiling will bring back the same pseudo-random sequence very time.
3. Comment this line out, and repeat the process (use `\useLastAsSeed` to generate new random sequences at each compile) until you get another distinct randomization, open `\jobname_rt.sav`, and copy the first line again, say its 735794511.
4. Place `\useThisSeed{735794511}` in the preamble.
5. Label each

```
%\useThisSeed{1604051353} % 11:00 class
%\useThisSeed{735794511}  % 12:30 class
```

To reproduce the random sequence for the class, just uncomment the random seed used for that class.

If you are using eqexam, the process can be automated as follows:

```
\vA{\useThisSeed{1604051353}} % 11:00 class
\vB{\useThisSeed{735794511}}  % 12:30 class
```

Again, this goes in the preamble.

Now, I simply must get back to my retirement.