www.AcroTeX.net

AcroTeX.Net

The ran_toks Package

Randomizing the order of tokens

D. P. Story

Copyright © 2019 dpstory@acrotex.net
Prepared: December 30, 2019

www.acrotex.net
Version 1.2, 2019/12/28

The ran_toks Package

www.AcroTeX.net

Table of Contents

www.AcroTeX.net

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 `examples` folder contains three demonstration files:

1. `ran_toks.tex` reproduces the sample code of this manual.
2. `random_tst.tex` shows how to use `ran_toks` to randomize the *questions* of an exam document created by the `eqexam` package.
3. `random_tst_qz.tex` shows how to randomize choices of a multiple choice field in a `quiz` environment of the `exerquiz` package, when the choices contain verbatim text.
4. `mc-db.tex` is an `eqexam` file that draws from the database files `db1.tex`, `db2.tex`, `db3.tex`, and `db4.tex`, to construct the questions of the exam. The questions are drawn at random from the DB files. Refer to Section 5 for a few more details.

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

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

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

www.AcroTeX.net

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,

\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.

\useRanTok{*num*}
\useRTName{*name*}

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 ({}), for example, {\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", which is the exact same random order. To obtain a different order, re-execute the \ranToks command with the same arguments.[2] Doing just that, we obtain,

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

[2] \ranToks{myPals}{{Jim}{Richard}{Don}{Alex}{Tom}{J\"{u}rgen}} in this example.

"I have 6 pals, they are Jürgen, Alex, Don, Tom, Jim and Richard." A new order? An alternative to re-executing `\ranToks` is to use the `\reorderRanToks` command:

`\reorderRanToks{`*name*`}`

Now, executing `\reorderRanToks{myPals}` and compiling the sentence again yields, "I have 6 pals, they are Jürgen, Alex, Don, Tom, Jim and Richard." For most applications, re-randomizing the same token list in the same document is not very likely something you need to do.

The `\reorderRanToks{`*name*`}` rearranges the list of tokens associated with ⟨*name*⟩, which may not be what you want; the `\copyRanToks` command, on the other hand, makes a (randomized) copy of its first required argument ⟨$name_1$⟩ and saves it as ⟨$name_2$⟩, without effecting the order of ⟨$name_1$⟩.

`\copyRanToks{`⟨$name_1$⟩`}{`⟨$name_2$⟩`}`

Thus, if `\copyRanToks{myPals}{myPals1}` is executed, the token list name `myPals1` contains the names of my pals in another randomized order, while maintaining the same order of `myPals`.

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 or contains verbatim text. For this, it may be more convenient to use the `\bRTVToks`/`\eRTVToks` command pair. The syntax is

```
\bRTVToks{name}   % <- Begin token listing
\begin{rtVW}
    ⟨content_1⟩
\end{rtVW}
...
...
\begin{rtVW}
    ⟨content_n⟩
\end{rtVW}
\eRTVToks         % <- End token listing
```

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 (⟨$content_i$⟩) of each of these environments are written to the computer hard drive and saved under a different name

www.AcroTeX.net

(based on the parameter *name*). Later, using the `\useRanTok` commands, they are input back into the document in a random order.

`\RTVWHook` The `rtWVW` environment also writes the command `\RTVWHook` to the top of the file. Its initial value is `\relax`. It can be redefined using the convenience command `\rtVWHook{⟨arg⟩}`, which expands to `\def\RTVWHook{⟨arg⟩}`.

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

www.AcroTeX.net

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

The \reorderRanToks works for lists created by the \bRTVToks/\bRTVToks construct. If we say \reorderRanToks{myThoughts} and reissue the above list, we obtain,

1. 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?

2. Roses are red and violets are blue, I've forgotten the rest, have you too?

3. I gave up saying bad things like $#%%%^*%^&#$@# when I was just a teenager.

The command \copyRanToks works for list created by \bRTVToks/\bRTVToks as well.

On the \displayListRandomly command. In the enumerate example immediately above, the items in the list are explicitly listed as \item \useRanTok{1} and so one; an alternate approach is to use the command \displayListRandomly, like so,

```
\begin{enumerate}
    \displayListRandomly[\item]{myThoughts}
\end{enumerate}
```

The full syntax for \displayListRandomly is displayed next.

\displayListRandomly[⟨*prior*⟩][⟨*post*⟩]{*name*}

The action of \displayListRandomly is to expand all tokens that are listed in the *name* token list, each entry is displayed as ⟨*prior*⟩\useRanTok{i}⟨*post*⟩, where i goes from 1 to \nToksFor{*name*}. In the example above, *prior* is \item, but normally, its default is empty. The defaults for ⟨*prior*⟩ and ⟨*post*⟩ are both empty.

The optional arguments. When only one optional argument is present, if is interpreted as ⟨*prior*⟩. To obtain a ⟨*post*⟩ with no ⟨*prior*⟩ use the syntax,

```
\displayListRandomly[][⟨post⟩]{⟨name⟩}
```

Within *each optional argument*, the four commands \i, \first, \last, and \lessone are (locally) defined. The \i command is the index counter of the token currently being typeset; \first is the index of the first item; \last is the index of the last item; and \lessone is one less than \last. The two optional arguments and the four commands may use to perform logic on the token as it is being typeset. For example:

www.AcroTeX.net

```
List of pals: \displayListRandomly
    [\ifnum\i=\last and \fi]
    [\ifnum\i=\last.\else, \fi]{myPals}
```

yields,

List of pals: Jürgen, Alex, Don, Tom, Jim, and Richard.

The optional arguments are wrapped to the next line to keep them within the margins, cool.

The example above shows the list of my pals with an Oxford comma. How would you modify the optional argument to get the same listing without the Oxford comma? (Jürgen, Alex, Don, Tom, Jim and Richard.) Hint: a solution involves the other command `\lessone`.

4. Additional arguments and commands

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

`\useRanTok[`*name*`]{`*num*`}`

The optional first parameter specifies the *name* of the list from which to draw a random token; *num* 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, Roses are red and violets are blue, I've forgotten the rest,
have you too?

The verbatim listing is

```
To illustrate: \useRanTok[myPals]{1}, \useRanTok[myThoughts]{2}
```

The typeset version looks a little strange, but recall, the text of `myThoughts` were each put in a `minipage` of width `.67\linewidth`. Without the `minipage`, the text would wrap around normally.

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

`\rtTokByNum[`*name*`]{`*num*`}`

This command expands to the token declared in the list named *name* that appears at the *num* 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*`}`.

Turning off randomization. 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} }
\ranToksOn
```

As a check, executing '`\useRanTok{3}` = `\rtTokByNum{3}` = 3' yields '3 = 3 = 3'? As anticipated.

To create a non-randomized list of tokens that already have been created (and randomized), use `\copyRanToks`:

```
\ranToksOff\copyRanToks{myPals}{myOriginalPals}\ranToksOn
```

Then, using `\displayListRandomly` in a clever way,

```
\displayListRandomly[\ifnum\i=\last\space and \fi(\the\i)~]
    [\ifnum\i=\last.\else,\fi\space]{myOriginalPals}
```

we obtain: (1) Jim, (2) Richard, (3) Don, (4) Alex, (5) Tom, and (6) Jürgen. The original list for `myPals` remains unchanged: (1) Jürgen, (2) Alex, (3) Don, (4) Tom, (5) Jim, and (6) Richard.

The `\useRanTok` command—whether it operates on a randomized token list or not—behaves similarly to an array. Thus, if we wanted the extract the third entry of the non-randomized token list (array) `myOriginalPals`, we do so by expanding the command `\useRanTok[myOriginalPals]{3}` to produce Don.

Document preparation. 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 auxiliary file. The package writes to a 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.

www.AcroTeX.net

Controlling the 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 command pair 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.

www.AcroTeX.net

www.AcroTeX.net

5. Commands to support a DB application

One user wanted to create exams using eqexam, but wanted to randomly select questions from a series of 'database' files. My thought was that ran_toks would do the job for him. After setting up a demo for him, I added the new command \useTheseDBs to ran_toks:

```
\useTheseDBs{⟨db1⟩,⟨db2⟩,...,⟨dbn⟩}
\useProbDBs{⟨db1⟩,⟨db2⟩,...,⟨dbn⟩}
```

The argument of \useTheseDBs is a comma-delimited list of file names. Each file name contains a \bRTVToks/\eRTVToks construct. Within this pair are rtVW environments, as described in Section 3.2. The \useTheseDBs command inputs the files listed in its comma-delimited argument; a warning is emitted if one or more of the files are not found. The default extension is .tex, \useTheseDBs{db1,db2} inputs the files db1.tex and db2.tex, if they exist, while \useTheseDBs{db1.def,db2.db} inputs the files db1.def and db2.db, if they exist. The command \useProbDBs is an alias for \useTheseDBs.

The placement of \useTheseDBs is anywhere prior to the insertion of the problems into the document, usually in the preamble.

Refer to the demonstration file mc-db.tex for an example.

Now, I simply must get back to my retirement. ☺