dvisirule package v1.1
(sitabular environment)

jiro1010senju at gmail dot com

2022/06/27

Superimpose the hidden/covered \hline and \vline in a LaTeX tabular/colortbl environment.

1 Motivation

Over a decade ago, I need to write down some latex tables in color. Not so many colors used, just two background colors. One is white as a usual background, and the other is gray which is to be easier to distinguish the preceding and succeeding rows from each other. In other words, the tables have two different row colors one after the other.

Fortunately we have \rowcolors from xcolor.sty and it is really helpful, except one thing. The problem is the lines (\hline, \cline and others) are not shown in the PDF viewer. Technically speaking, if we zoom in/out, then the lines appear, but not always. If I change the thickness of the lines, I can always see them. But it is too ugly. It seems highly depending on the monitor resolution and the PDF viewer's internal calculation of the coordinates. Even if I set the width of the lines to 4pt, it can be only one pixel on the screen. It is totally up to the environment of the viewer.

To address this issue, I guessed the cause is the order of drawing lines and painting the row background color. E.g. drawing lines first, and then painting the background color next. In that case, the background color may hide the lines due to the internal rounding-up/down the calculated coordinates.

Some TeXnicians may be able to modify the tabular/colortbl environment, and postpone drawing the lines after completing the background color. But I don't have such TeXnique. I had considered a few ways to get what I want, and my solution is to draw the lines again at the end of the page by a new dviware.

2 Basic approach

1. In .tex (actually .sty), embed the markers to represent the begin/end of a line.
2. A new dviware searches the embedded markers, if found, remembers the coordinates in the DVI page of them, and the offset in the DVI file.

3. When the dviware reaches the end of the page, it copies the DVI instructions from the marker to the end of the page. I mean appending the line instructions to the page. And replace the the markers by NOP at the original position, to keep silent other dviwares.

Now the lines are re-drawn over the row background color, and we can always see them clearly.

3 Compile & Install

```
$ cd $this_dir/src
$ configure -q
$ cd ..
$ make -s Dir=/tmp
```

Then you will get these files.

```
/tmp/dvisirule
/tmp/dvisirule-bin
/tmp/dvisirule-expg.mk
/tmp/dvisirule-marker.awk
/tmp/dvisirule-pgnum.awk
/tmp/dvisirule.sty
```

Now you can install them by

```
$ make InstallBase=/tmp/texmf-dist
```

Here `${InstallBase}` is a make-variable which is referred by other make-variables.

```
InstallBin ?= ${InstallBase}/bin
InstallLib ?= ${InstallBase}/lib
InstallSty ?= ${InstallBase}/lib/texinputs
```

and the files will be installed to these dirs.

```
${InstallBin}/dvisirule

${InstallLib}/dvisirule-bin
${InstallLib}/dvisirule-expg.mk
${InstallLib}/dvisirule-marker.awk
${InstallLib}/dvisirule-pgnum.awk

${InstallSty}/dvisirule.sty
```

Please note that these installed dirs have to be recognized by `Kpathsea` since the main `sh` script `dvisirule` searches these sub-files by `kpsewhich(1)`.

4 Usage

`dvisirule.sty` is provided, and `sitabular` environment is a wrapper of `tabular` (including `colortbl`. You should include `xcolor.sty` or something BEFORE this pkg).

The tabular syntax are kept, so you can use it by simply replacing `tabular` by `sitabular`. But it is not a main part of this package. `sitabular` does only embedding the markers and stores the page number to an external file. The main part of this packages is `dvisirule` (shell script) and the internal command `dvisirule-bin` (C program). So the usage here is not only including `dvisirule.sty` and `\begin{sitabular}`, but also run `dvisirule` after LaTeX compilation.

```
(a.tex)
\usepackage{dvisirule}

\begin{sitabular} ... \end{sitabular}

latex a.tex
cp -p a.dvi a.dvi.save
dvisirule a.dvi a-si.dvi
mv a-si.dvi a.dvi
```

You don't want run `dvisirule` command unconditionally? Good. `dvisirule.sty` creates and writes a file the page numbers which contain the line to be superimposed, so when the size of that file is zero, you can skip `dvisirule` command.

```
latex a.tex
if [ -s a.sirule ]
then
      cp -p a.dvi a.dvi.save
      dvisirule a.dvi a-si.dvi
      mv a-si.dvi a.dvi
fi
```

5 Demo

```
\definecolor{BGeven}{gray}{.8}
\rowcolors*{1}{}{BGeven}

\newcommand{\tblA}[2]{%
  \begin{#1}{|c|}
    \noalign{\global\rownum=0}
    \showrowcolors
    \hline
    #2\\ \hhline{=}
    Can\\ \hline
    you\\ \hline
    see\\ \hline
    all\\ \hline
    lines\,?\\ \hline
    Always\,?\\ \hline
    Try zooming in\slash out in your viewer.\\ \hline
    \hiderowcolors
  \end{#1}
}

\tblA{tabular}{WITHOUT \texttt{dvisirule}}%
\tblA{sitabular}{WITH \texttt{dvisirule}}%
```

WITHOUT `dvisirule`
Can
you
see
all
lines ?
Always ?
Try zooming in/out in your viewer.

WITH `dvisirule`
Can
you
see
all
lines ?
Always ?
Try zooming in/out in your viewer.

Tested on TeX Live 2019.

6 Limitation

`\hline` supports `\hline\hline` sequence, which means if followed by another `\hline`, it inserts `\vskip\doublerulesep`. This feature doesn't work in `sitabular`. It results a single thick `\hline`. Use `\hhline` (from `hhline.sty`) instead.

7 `dvisirule` script and sub-programs

The main script runs `dvitype` and other commands internally. You may want to replace some commands by other language specific variants. For `dvitype` command, you can set a environment variable, `$DVITYPE`. If it is set, the main script uses `$DVITYPE` instead of `dvitype`. Here is an example.

```
DVITYPE=pdvitype
export DVITYPE
dvisirule a.dvi
```

There are two other commands in the same group, `dviselect` (`$DVISELECT`) and `dviconcat` (`$DVICONCAT`).

Also the main script refers to `$TMPDIR` environment variable. It specifies a directory where the extracted DVI page(s) are stored temporarily. If it is not set, `/tmp` is used.

7.1 `dvisirule` — main shell script

Syntax:

```
dvisirule [option] in.dvi [out.dvi]
option:
-j jobs
   number of jobs in parallel.
   a job means handling the extracted page.
   default: the number of processors online
-t dvitype-cmd
   'dvitype' command.
   default: dvitype or $DIVTYPE
-c dviconcat-cmd
   'dviconcat' command.
   default: dviconcat or $DIVCONCAT
-s dviselect-cmd
   'dviselect' command.
   default: dviselect or $DIVSELECT
-l library-dir
   library path where dvisirule-bin, \*.awk and \*.mk
   are located.
   default: a dir kpsewhich(1) returned.
if you set "library-dir" by this option, then
"kpsewhich -path \"library-dir\"" is used.
Also this command creates a temporary directory under $TMPDIR
or /tmp. For details, refer to dvisirule.pdf.
```

1. Accepts `.dvi`, and searches `.sirule`. If `.sirule` doesn't exist, exits with an error. If it exists but the size is zero, then exits with a success.
2. By an AWK script, `dvisirule-pgnum.awk`, splits `.dvi` per page, dividing those who contains `sitabular` and who doesn't. Sequential pages who doesn't contain `sitabular` are grouped into a single file.
3. (for each extracted page which contains `sitabular`)

   By an AWK script, `dvisirule-marker.awk` finds and extracts the markers from the page and the DVI line instructions from it.
4. (for each extracted page which contains `sitabular`)

   By a binary program `dvisirule-bin`, copies the line instructions surrounded by the markers to the end of the page, and replaces the markers and the line instructions at the original position (in the DVI page) by NOP instruction, to stop other dviwares complain about the marker.
5. Concatenate all pages, and make the final `.dvi`.

7.2 `dvisirule-bin` — sub binary

This command has two inputs, one is a DVI file and the other is the output from `dvisirule-marker.awk`. The AWK script parses an output of `dvitype` for a single DVI page, and extract and prints SETRULE/PUTRULE instructions surrounded by the markers along with the color, the coordinates, and the offset in the DVI file.

`dvisirule-bin` command parses the output of the AWK script, and copies those SETRULE/PUTRULE instructions to the end of the page. The markers and the instructions at the original offset are replaced by NOP instruction, so that other dviwares don't complain about that such as "Unknown special" or something. At last, `dvisirule-bin` creates and writes a new DVI file.

You may be wondering is the AWK script really necessary? Why doesn't `dvisirule-bin` parse the DVI file by itself? Good point. It's just because I'm lazy. I can understand if a single binary operates all these processing, then the required cost (including CPU time) will be smaller. The performance will not be bad. But it is a run-rime performance. The lazy of me wants the develop-time performance too.

Roughly speaking, splitting and parsing for each DVI page, and copying the instructions should be done concurrently. If all were implemented by C including multi-threading, the technical hurdle would be rather high. So I chose scripting for splitting and parsing. It can be run concurrently by `"make -j NCPU"`. Since DVI is a binary format, I had to implement it by C. It is OK, and the single-threaded C program can be run in parallel by `"make -j NCPU"` too. This is my laziness.

8 Implementation

8.1 `.sirule` file

`\si@RulePageW` `\si@RulePage` Create a file and store the absolute page numbers which contain `sitabular`.

```
\RequirePackage{zref-abspage}
%
\newwrite\si@RulePageW
\immediate\openout\si@RulePageW=\jobname.sirule%\relax
\newcommand{\si@RulePage}{%\immediate
  \write\si@RulePageW{\theabspage}%
}
```

8.2 marker

\si@bol
\si@eol

Begin/End markers of a line.

```
\newcounter{si@rule}
\newcommand{\si@bol}{%
  \stepcounter{si@rule}%
  \special{sirule BOL\space\thesi@rule}%
  \si@RulePage%
}
\newcommand{\si@eol}{%
  \si@RulePage%
  \special{sirule EOL\space\thesi@rule}%
  \addtocounter{si@rule}{-1}%
}
```

8.3 `\hline` and `\vline`

\si@hline
\si@vline

Originally `\hline` issues `\futurelet` internally to support `\hline\hline` sequence. But `\si@hline` issues `\si@eol` just after the original `\hline`. So `\hline\hline` would not work expectedly. Use `\hhline` (`hhline.sty`) instead.

```
\let\si@Ohline=\hline
\newcommand{\si@hline}{%
  \noalign{\si@bol}%
  \si@Ohline%
  \noalign{\si@eol}%
}
\let\si@Ovline=\vline
\newcommand{\si@vline}{%
  \si@bol%
  \si@Ovline%
  \si@eol%
}
```

8.4 `\hhline` if loaded

\si@hline
\si@vline

```
\newif\ifsi@hhline
\@ifundefined{hhline}{}{%
  \global\si@hhlinetrue%
  \global\let\si@Ohhline=\hhline%
```

```
  \newcommand{\si@hhline}[1]{%
    \noalign{\si@bol}%
    \si@Ohhline{#1}%
    \noalign{\si@eol}%
  }%
}
```

8.5 `sitabular` environment

`sitabular`

```
\@ifundefined{newcolumntype}{%
  \global\let\newcolumntype=\@gobbletwo%
}{}
\let\si@Otabular=\tabular
\let\endsi@Otabular=\endtabular
\newenvironment{sitabular}[1]{%
  \def\hline{\si@hline}%
  \ifsi@hhline\def\hhline{\si@hhline}\fi%
  \newcolumntype{|}{!{\si@vline}}%
  \begin{si@Otabular}{#1}%
}{%
  \end{si@Otabular}%
}
```

8.6 `silongtable` environment if loaded

`silongtable`

```
\@ifclassloaded{longtable}{}{%
  \global\let\si@OLT@hline=\LT@hline%
  \newcommand{\si@LT@hline}{%
    \noalign{\si@bol}%
    \si@OLT@hline%
    \noalign{\si@eol}%
  }%
  \let\si@Olongtable=\longtable%
  \let\endsi@Olongtable=\endlongtable%
  \newenvironment{silongtable}[1]{%
    \def\LT@hline{\si@LT@hline}%
    \def\si@Ohline{\hline}%
    \def\hline{\si@hline}%
    \ifsi@hhline\def\hhline{\si@hhline}\fi%
    \newcolumntype{|}{!{\si@vline}}%
    \begin{si@Olongtable}{#1}%
  }{%
    \end{si@Olongtable}%
  }%
}
```