My Way

July 26, 2006

Creating tables using CSV (comma–separated values)

Mojca Miklavec

ConTEXt offers good support for creating complex tables (Natural Tables, tabulate, table, tables, linetable), see http://wiki.contextgarden.net/Tables_Overview, but creating simple tables is still cumbersome in the TEX word. The `database` module should simplify input of tables (where no row– or column–merging is required): instead of writing lengthy and cumbersome `\bTR\bTD`–s or `\NC\NR`–s you can now separate the rows with newlines and columns with commas, spaces, tabs or other character(s) of your choice.

1 Motivation

Writing and manipulating tables in applications such as Excel© is a childplay. But for writing a table in TEX environment you first need to study 10 pages of manual and even if you already have a table in some plain text file, office application or on a web page, you still need to add dozens of commands to tell TEX how to typeset it.

I asked Hans if there was no simpler way to do it. And the answer was:

```
\startseparatedlist[NaturalTable]
Of,course
  ,it is!
\stopseparatedlist
```

Of	course
	it is!

2 Defining a new "data parser": \defineseparatedlist

In order to turn the above code into a table and to save you some typing, the following definition was provided in the module:

```
\defineseparatedlist
   [NaturalTable]
   [separator=comma,
    before=\bTABLE,after=\eTABLE,
    first=\bTR,last=\eTR,
    left=\bTD,right=\eTD]
```

`NaturalTable` is the name of the list, `comma` (which could also be written as `{,}`) means that each comma will start a new column, while the other six parameters define the rules for typesetting the data.

The syntax of `\defineseparatedlist` is as follows:

```
\defineseparatedlist [..1..] [..,..2=..,..]

1   IDENTIFIER

2   separator  =  comma  space  tab  TEXT
    quotechar  =  TEXT
    before     =  COMMAND  TEXT
    after      =  COMMAND  TEXT
    first      =  COMMAND  TEXT
    last       =  COMMAND  TEXT
    left       =  COMMAND  TEXT
    right      =  COMMAND  TEXT
    command    =  COMMAND
    setups     =  IDENTIFIER
```

separator

Character(s) separating the data cells. There are currently three predefined values: `comma` (the default one), `space`[1] and `tab` which is a bit special[2], but you can use 'any' other character(s) as long as they don't have some special meaning. `separator=X` will thus start a new column each time when the character 'X' is encountered.

quotechar[3]

Triggers literate handling of the cell content, usually it is double quote ("). It is mostly meant to be used for parsing proper CSV[4] data.

```
\defineseparatedlist
  [CSV]
  [separator={,},
   quotechar={"},
   before={\starttabulate[|r|c|l|]},after=\stoptabulate,
   first=\NC,last=\NR,
   left=,right=\NC]

\startCSV
some data,&,"a comma, hidden inside a quote"
quoted quotes,""""","need lots of ""quotes"""
\TeX\ commands,are $\lnot$,processed
UTF-8,should ¬,be a problem
\stopCSV
```

some data	&	a comma, hidden inside a quote
quoted quotes	"	need lots of "quotes"
\TeX \ commands	are $\lnot $	processed
UTF-8	should ¬	be a problem

Do I see a space after TeX? Well, forget it. It's not that important.

Note: According to CSV specification, content of a single cell could span across multiple rows (preserving newlines) if quoted properly. This won't work here, at least not until pdfluaTEX is out.

---

[1] It's there just to justify the effort put into the introduction of a new keyword ;)
`separator={ }` works just as well

[2] TEX usually doesn't distinguish between space and tab unless it's explicitely instructed to do so

[3] Taco's favourite!

[4] comma-separated values, as already noted in the title

before/after, first/last, left/right

keyword	used for	examples of possible arguments		
before	beginning of table	\bTABLE	\starttable	\starttabulate
after	end of table	\eTABLE	\stoptable	\stoptabulate
first	beginning of row	\bTR	\NC or \VL	
last	end of row	\eTR	\NR	
left	beginning of cell	\bTD		
right	end of cell	\eTD	\NC	

Note that for \starttable and \starttabulate you also need to specify the pattern, such as before=\starttable[|l|l|l|] for three left-aligned columns. In contrast to natural tables where the number of columns is able to adapt itself according to the data, you have to watch out here, so that you provide the exact number of columns, otherwise you may run into troubles.

command

Instead of creating a table, you can also provide your own command accepting the same number of parameters as the number of columns in the data. If non-empty, the module will ignore any settings for before/after, first/last and left/right and use the supplied command instead.

Suppose the you wanted to print addressed on envelopes to send your magazine to some TEX user groups. You would first define a command to print the envelope:

```
\def\SendMe#1#2#3#4{\framed
   [align={flushleft,lohi},
    width=4cm,
    height=2.5cm]{#1\crlf#2\crlf\crlf\uppercase{#3\crlf#4}}}
```

An alternative to using \SendMe{name}{adress}{post office}{country} for each entry is now to \defineseparatedlist for the whole list:

```
\defineseparatedlist[Address][separator={;},command=\SendMe]
\startAddress
NTG;Maasstraat 2;NL-5836 BB Sambeek;The Netherlands
Dante-e.V.;Postfach 101840;D-69008 Heidelberg;Germany
\stopAddress
```

NTG Maasstraat 2 NL-5836 BB SAMBEEK THE NETHERLANDS
Dante e.V. Postfach 101840 D-69008 HEIDELBERG GERMANY

setups

Until I figure out how to explain it, I hope that the example below will be descriptive enough to give you an idea how to use it.

Some files come with comments (usually lines starting with #). To ignore such lines, the following recipe might help you:

```
\unprotect
\startsetups CSV:unix
  \catcode`\#=\@@comment
\stopsetups
\protect

\defineseparatedlist[CSV][setups=unix,...]
```

3 Recycling: \setupseparatedlist

If you want to use space instead of comma as a separator in a list that is already defined, all you have to do is to

```
\setupseparatedlist[NaturalTable][separator=space]
\startseparatedlist[NaturalTable]
setup an\ existing {separated list}
and watch      for\space the\space spaces.
\stopseparatedlist
```

setup	an existing	separated list
and	watch	for the spaces.

```
\setupseparatedlist [.1.] [..,.2.,..]

1   IDENTIFIER
2   inherits from \defineseparatedlist
```

4 Using: \startseparatedlist

Once you have sucessfully defined a separated list called *NAME*, there are basically three ways to use it:

- \startseparatedlist[*NAME*] ... \stopseparatedlist
- \start*NAME* ... \stop*NAME*
- \processseparatedfile[*NAME*][*filename*]

Sadly enough this doesn't work (it must be my mistake somewhere):

```
\showsetup{startseparatedlistname}
\showsetup{processseparatedfile}
```

unknown setup 'startseparatedlistname'

```
\processseparatedfile [.1.] [.2.]

1   IDENTIFIER
2   FILE
```

Some time ago Willi sent me some data about the decreasing number of cows in Holland[5] in an Excel table. I copy-pasted the content into a simple text editor (so that tabs were placed between single cells) and commented out the first two lines[6]. The arrows are there just to visualize tabs.

```
# Number of cows in Holland
# Year → Total → Milking → Pregnant
1995 → 1709 → 1449 → 260
1997 → 1606 → 1387 → 219
1999 → 1520 → 1307 → 212
2001 → 1496 → 1345 → 151
2003 → 1492 → 1324 → 169
2005 → 1421 → 1263 → 158
```

[5] one unit meaning 1000 cows

[6] I wanted to plot the data with another program which didn't know what to do with words when it should plot numbers

Let's first define the appropriate *separatedlist*:

```
\defineseparatedlist
   [TSV]              % tab-separated values
   [separator=tab,
    before=,after=, % we'll place them explicitely
    first=\bTR,last=\eTR,
    left=\bTD,right=\eTD,
    setups=unix]
```

We might want to use boldface and background color for the first row. We also have to begin the table explicitly because we didn't set any command to start and stop the table[7].

```
\setupTABLE[r][1][style=bold,background=color,backgroundcolor=gray]
\bTABLE
% Header
\startTSV
Year Total Milking Pregnant
\stopTSV
% Content
\processseparatedfile[TSV][\jobname-TSV-example.tmp]
\eTABLE
```

Year	Total	Milking	Pregnant
1995	1709	1449	260
1997	1606	1387	219
1999	1520	1307	212
2001	1496	1345	151
2003	1492	1324	169
2005	1421	1263	158

5 Known bugs

- Recent versions of the module introduced some problems with UTF-8 character handling in normal mode (with `quotechar` it works OK). Example:

```
\startseparatedlist[NaturalTable]
č,š,ž
\stopseparatedlist
```

[7] If we did, we couldn't join the data from two different sources: we provide the header line explicitly and use a file as source of the data.

Other 8-bit regimes work OK.

- blank cells have problems at the end of line:

```
\startseparatedlist[NaturalTable]
a,b
c,
\stopseparatedlist
```

6 Wishlist / TODO

- `\defineseparatedlist[name][nameofotherlist]` to inherit properties
- selecting columns (and rows?)
  A handy feature would be something like `usecolumns={1-3,5}`, which would select only the columns 1, 2, 3 and 5 and:
  – ignore redundant information (unneeded columns/too long lines),
  – "add" empty cells if the data line would be too short.
  An example of a valid definition would thus be:

```
\defineseparatedlist
  [Address]
  [separator={;},
   command=\SendMe,
   usecolumns={1-4}]
\startAddress
NTG;Maasstraat 2;NL-5836 BB Sambeek;The Netherlands;ignored data
Dante-e.V.;Postfach 101840;D-69008 Heidelberg
\stopAddress
```

  Comments at the end of the first row would be ignored, and though leaving fields out doesn't really belong to good (programming) habits, the second line with one semicolon missing will pretend as if the field with Country would be present and blank. Without `usecolumns={1-3}` an error would be raised in such case.
- special treatment of header lines (I'm not sure yet how exactly this should work.)