Writing a LaTeX Article – Using Tables

T. Stitz *

Abstract

During module 4, attendees will include tables in a LaTeX document using the standard commands available with LaTeX as well as packages that improve appearance and simplify entry. External software developed to simplify table entry will be discussed briefly as well.

1 Tables with LaTeX

1.1 Tabular Environment

The basic structure of a table is constructed by using **tabular**[vertical position]{column formatting}. An example of a simple table is

Hardware		
Model	Name	Price
LUS26Z	Shear face mount joist hanger	$11.62
ABA66Z	Adjustable post base	$21.00
FB24Z	Fence bracket	$0.67

and the code to create this table is

```
\begin{tabular}{|c|l|r|}
  \hline
  \multicolumn{3}{|l|}{\bfseries Hardware}\\
  \hline\hline
  Model & Name & Price\\
  \hline
  LUS26Z & Shear face mount joist hanger & \$11.62\\
  \hline
  ABA66Z & Adjustable post base & \$21.00\\
  \hline
  FB24Z & Fence bracket & \$0.67\\
  \hline
\end{tabular}
```

First, notice that column data is separated by & and row end is marked as \\. Also, the declaration version boldface is used. The scope of this declaration is the cell only, so declarations are often used in tables. The *hline* command places horizontal lines between the rows. There is a double horizontal line separating the first and second rows.

Let's examine the column formatting. Lines are drawn between the columns using |. Data can be centered (c), left justified (l), or right justified (r). The first row is one cell that spans the entire table using the **multicolumn** {number of columns}{column alignment}{text} command. Don't forget to use the horizontal lines in the multicolumn alignment argument if you need them.

*Applied Sciences Librarian, Phone: 330-972-6192, Fax: 330-972-7033, E-mail: tstitz@uakron.edu

The data in our example lends to the tabular environment very well. Sometimes, the alignment is not as easy. Let's say that we want to align on the period, but there is not always two values after the decimal and there is an unknown number of digits before the decimal.

Hardware		
Model	Name	Length (in)
LUS26Z	Shear face mount joist hanger	5.25
ABA66Z	Adjustable post base	11.125
FB24Z	Fence bracket	3.5

The code to create this table is

```
\begin{tabular}{|c|l|r@{.}l|}
  \hline
  \multicolumn{4}{|l|}{\bfseries Hardware}\\
  \hline\hline
  Model & Name & \multicolumn{2}{l|}{Length (in)}\\
  \hline
  LUS26Z & Shear face mount joist hanger & 5 & 25\\
  \hline
  ABA66Z & Adjustable post base & 11 & 125\\
  \hline
  FB24Z & Fence bracket & 3 & 5\\
  \hline
\end{tabular}
```

In order to force the last column to center on the period, it is split into two columns. The first column contains the information before the decimal point and the second column contains the content after the decimal point. Instead of printing the decimal point directly, the argument @{.} is used, which forces a period to print between the third and fourth columns.

Now let's try a more complicated example. This table has a double horizontal line after the first column and before the last column. The first column is centered. The second and fourth are right justified and the third and fifth are left justified, where a colon the prints between the right and left justified columns using @{:}. The eighth column has a width equal to a quarter of the page width. Normally, the columns do not use word wrapping unless a column width is specified using the argument, p. Word wrapped text in a column is fullly justified by default. The code for this table is

```
\begin{tabular}{|c||r@{:}l||r@{:}l||p{0.25\textwidth}|}
  \hline
  \raisebox{-3ex}{\bfseries Days} & \multicolumn{2}{l||}{\raisebox{-3ex}{\bfseries Start Time}} & \multicolumn{2}{l||}{\
      raisebox{-3ex}{\bfseries End Time}} & \raisebox{-3ex}{\bfseries Class Name}\\[5ex]
  \hline\hline
  & 10 & 00AM & 10 & 50AM & \bfseries Class 1\\
  \cline{2-5}
  \raisebox{1.5ex}[0pt]{M} & 2 & 00PM & 4 & 50PM & \bfseries Lab 1\\
  \hline\hline
  & 8 & 00AM & 8 & 50AM & \bfseries Class 2\\
  \cline{2-5}
  \raisebox{1.5ex}[0pt]{T} & 9 & 00AM & 9 & 50AM & \bfseries Class 3\\
  \hline\hline
  W & 10 & 00AM & 10 & 50AM & \bfseries Class 1\\
  \hline\hline
  & 8 & 00AM & 8 & 50AM & \bfseries Class 2\\
  \cline{2-5}
  \raisebox{1.5ex}[0pt]{H} & 9 & 00AM & 9 & 50AM & \bfseries Class 3\\
  \hline
\end{tabular}
```

In this table, there is a new horizontal line command. The *cline* command is used to draw a horizontal line between columns 2 and 5. For the first row some entries span multiple columns, the height is increased by using an optional argument, [5ex], at the end of the first line. The *raisebox* command was used to vertically center the heading text by lowering the line by 3ex; otherwise, the text would appear at the top. The output of this code is

Days	**Start Time**	**End Time**	**Class Name**
M	10:00AM	10:50AM	**Class 1**
	2:00PM	4:50PM	**Lab 1**
T	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**
W	10:00AM	10:50AM	**Class 1**
H	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**

If you have a cell that spans an even number of multiple rows, you can use the *raisebox* command too. Usually if there is an odd number of rows, the text can be placed in the center row to vertically center it. Notice there is an option of 0pt used for the *raisebox* command. If you do not select this option, it increases the height of the row. This did not cause a problem for the first row because the height is larger than the height of the line plus the box. An example of the increased whitespace for the row is shown.

T	8:00AM	8:50AM	**Class 3**
	9:00AM	9:50AM	**Class 4**

1.2 Table Environment

The *table* environment allows tables to be floated to optimize space and it enables automatic table numbering with the author's caption. A caption that is shorter than one line will be centered. In those cases, the tabular environment requires centering to align properly with the caption.

A caption can be placed below or above a table by declaring before or after the *tabular* environment. Sometimes the caption will be too close to the table when placing the caption above it. Use **vspace**{} when that happens. Options for placement of the table include

- where it occurs within the text (*h*),
- at the top of the page (*t*),
- at the bottom of the page (*b*), or
- on a special page (*p*).

The placement given by the optional argument should be viewed as a suggestion instead of a requirement due to the engine's optimization algorithm. It is still not guaranteed, but it will be more likely to follow your suggestion when using ! in conjunction with these options. The exclamation point will override some normal spacing and number restrictions. A float will not appear on a page previous to where it occurs in the code. The *table* environment is as shown.

```
\begin{table}[h]
  \caption{Class Schedule}
  \centering
  ... our tabular example
\end{table}
```

Table 1: Class Schedule

Days	Start Time	End Time	Class Name
M	10:00AM	10:50AM	**Class 1**
	2:00PM	4:50PM	**Lab 1**
T	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**
W	10:00AM	10:50AM	**Class 1**
H	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**

Our table now appears as shown in Table 1.

There is a special version of the *table* environment, *table** as well. When the document contains two columns, this environment is used to span both columns. Generally, tables print in sequence; however, it is possible to print out of sequence when using *table**. For more information about floats, use Kopka and Daly (2004) or Mittelbach, Goossens, Braams, Carlisle, and Rowley (2004).

2 Table Packages

There are packages that LaTeX users have developed to enable easier entry and formatting of tables. Several will be discussed in this section.

2.1 Caption Package

The *caption* package enables easy formatting of the caption. Caption format, such as font and position, can be adjusted using this package. The command to include the package is of the form **usepackage**[key=value]{caption}

The formatting can be set for the entire document using the optional arguments when loading the package. Use the *captionsetup* command within the *table* environment to scope the format to a particular table. As stated previously, the caption is automatically centered when using the *table* environment. By using the argument, *singlelinecheck=false*, it can be overridden.

You can change the label separator to a period, the font size to footnote size, and the label caption to bold and italic by using **captionsetup**{singlelinecheck=false,labelsep=period,font=footnotesize,labelfont={bf,it}} Our table would now appear as shown in Table 2.

2.2 Array Package

The *array* package extends the column definitions available with the *tabular* environment. At a minimum, this package will save you much typing. A few important features will be demonstrated in this section. With the use of $>\{\}$ before a column, code can be inserted to format the entire column. For example, you no longer need to type the boldface declaration in every entry for the entire column. It is definitely preferable to type the command one time to typing the command twenty-five times. The code would be

begin{tabular}{|c||r@{:}l||r@{:}l||>{**bfseries**}p{0.25**textwidth**}|}

In addition, using $>\{\}$ does not change column spacing like @{}. Our table would now appear as shown in Table 3.

Table 2. Class Schedule

Days	Start Time	End Time	Class Name
M	10:00AM	10:50AM	**Class 1**
	2:00PM	4:50PM	**Lab 1**
T	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**
W	10:00AM	10:50AM	**Class 1**
H	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**

Table 3. Class Schedule

Days	Start Time	End Time	Class Name
M	10:00AM	10:50AM	**Class 1**
	2:00PM	4:50PM	**Lab 1**
T	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**
W	10:00AM	10:50AM	**Class 1**
H	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**

Two additional arguments for the *tabular* environment are available with the array package: m and b. Each option works like p except for vertical alignment. The text is aligned at the top line using p, the middle using m, and the bottom using b. These arguments work only when the text is long enough to use word wrapping; therefore, we will create our own command for easier entry of our column headings. The command will be **newcommand**{**theading**}[3]{**multicolumn**{#1}{#2}{**raisebox**{−3ex}{**bfseries** #3}}} Now, we will enter **theading**{1}{|c||}{Days}, which is a little easier.

When declaring column width with p, m, and b, the cells will be word wrapped and fully justified. Often, a column will have a better appearance when it is left justified. Using **raggedright****arraybackslash**, the text will be left justified the column. The command, **arraybackslash**, must be used whenever aligning column text using this method; otherwise, there will be difficulties when \\ is used and there could be errors when compiling your document. Table 4 is an example of a left justified, word wrapped table.

Table 4. Class Schedule

Days	Start Time	End Time	Class Name
M	10:00AM	10:50AM	**Class 1 Note: The prerequisite is Class A.**
	2:00PM	4:50PM	**Lab 1**
	5:20PM	6:25PM	**Class 2**

2.3 Multirow Package

LaTeX has a command to span columns; however, it does not have a command to span rows. The *multirow* package can be used for this purpose by using **multirow**{number of rows}{width}{text}

Most of the time, you will use * for the width argument to use natural width. You could use the *multirow* command instead of the *raisebox* command when text spans multiple rows. If you use word wrapping, the *multirow* package does not work well. In this case, it would be better to use the *raisebox* command. You could create your own command for easier entry in this case too. Table 5 appears exactly the same and uses the *multirow* package instead of the *raisebox* command.

Table 5. Class Schedule

Days	**Start Time**	**End Time**	**Class Name**
M	10:00AM	10:50AM	**Class 1**
	2:00PM	4:50PM	**Lab 1**
T	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**
W	10:00AM	10:50AM	**Class 1**
H	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**

2.4 Colortbl Package

The *colortbl* package enables background color for rows, columns, and cells of a table. Since we use the *xcolor* package for text color, we can use the optional argument *table* to include the *colortbl* package. The *colortbl* package works well with $>\{\}$, but not all commands work properly with $@\{\}$.

Using our table example, we could set the first column to bold and red text, the last column to a background color of yellow, and the first row to a background color of dark gray with white text. If you use the dvipnames option with the *xcolor* package, it expands the number of colors that you can use. Basically, you can use all the colors defined by dvips (common graphics driver used when creating DVI files). Using these options enables the use of the color “BrickRed” in our example.

```
\usepackage[dvipsnames,table]{xcolor}
...
\renewcommand{\theading}[3]{\multicolumn{#1}{#2}{\raisebox{-3ex}{\color{white}\bfseries #3}}}
\begin{tabular}{|>{\bfseries\color{BrickRed}}c||r@{:}l||r@{:}l||>{\bfseries\columncolor{Yellow}}p{0.25\textwidth}|}
...
\rowcolor[gray]{.3}
\theading{1}{|c|}{Days} & \theading{2}{|l|}{Start Time} & \theading{2}{|l|}{End Time} & \theading{1}{p{0.25\textwidth}|}{
    Class Name}\\[5ex]
```

Notice that I eliminated double vertical lines in the table headings. The space between the lines would appear white. There are commands provided by the *colortbl* package to apply color to this space, but this solution is easier – as long as it look fine. Our table now appears as shown in Table 6.

2.5 Footnotes for Tables

There are a few options when incorporating footnotes with a table. The *threeparttable* package enables the author to easily add footnotes that display below the table. The basic structure when using *threeparttable* is

Table 6. Class Schedule

Days	Start Time	End Time	Class Name
M	10:00AM	10:50AM	**Class 1**
	2:00PM	4:50PM	**Lab 1**
T	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**
W	10:00AM	10:50AM	**Class 1**
H	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**

```
\begin{table}[h!]
  \captionsetup{singlelinecheck=false,labelsep=period,font=footnotesize,labelfont={bf,it}}
  \caption{Class Schedule}
  \begin{threeparttable}
    \begin{tabular}{|>{\bfseries\color{BrickRed}}c||r@{:}l|r@{:}l||>{\bfseries\columncolor{Yellow}}p{0.25\textwidth}|}
      ...
      \multirow{2}{*}{M} & 10 & 00AM & 10 & 50AM & Class 1\tnote{a}\\
      ...
    \end{tabular}
    \begin{tablenotes}
      \item[a]{Be 15 min. early on first day of laboratory to fill out necessary paperwork.}
    \end{tablenotes}
  \end{threeparttable}
\end{table}
```

Our table now appears as shown in Table 7.

Table 7. Class Schedule

Days	Start Time	End Time	Class Name
M	10:00AM	10:50AM	**Class 1**[a]
	2:00PM	4:50PM	**Lab 1**
T	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**
W	10:00AM	10:50AM	**Class 1**
H	8:00AM	8:50AM	**Class 2**
	9:00AM	9:50AM	**Class 3**

[a] Be 15 min. early on first day of laboratory to fill out necessary paperwork.

2.6 Some Packages for Large Tables

There are situations when a table is larger than can fit on one page in portrait mode. One popular package to enable a table to span multiple pages is *longtable*. It is possible to produce a table in landscape mode as well. Use of the *rotating* package enables a landscape table using the *sidewaystable* environment.

2.7 Datatool Package

The *datatool* package allows database type operations to be performed on a text file. There are many operations available, but this module is going to focus on pulling data from an existing text file to display as a table. To load the text file, use

DTLloaddb[options]{database name}{file name}

If the data does not contain a header row, *keys* can be set with the optional argument. By default, the separator is a comma; however, these values can be adjusted using commands. If there are special characters in your data, it might be beneficial to use **DTLloadrawdb**. The data read from the text file can be displayed in a *tabular* or *longtable* environment using the *DTLdisplaydb* or*DTLdisplaylongdb* command, respectively. This example uses the ascii.csv file

DTLloaddb{ascii}{ascii.csv}
DTLdisplaylongdb[caption={ascii table},contcaption={ascii table cont.}]{ascii}

Table 8: ascii table

Char	**Dec**	**Oct**	**Hex**
(nul)	0	0	0x00
(soh)	1	1	0x01
(stx)	2	2	0x02
(etx)	3	3	0x03
(eot)	4	4	0x04
(enq)	5	5	0x05

Page 1

Table 8: ascii table cont.

Char	**Dec**	**Oct**	**Hex**
(ack)	6	6	0x06
(bel)	7	7	0x07
(bs)	8	10	0x08
(ht)	9	11	0x09

Page 2

In the datafile, I included the appropriate commands in the data instead of using **DTLloadrawdb**. You should only use **DTLloadrawdb** if all the exceptions in your data are covered by the command. If you want more control of how the data is typeset, you might want to display using *DTLforeach* command. It requires more code, but you can apply formatting to table columns as in this example.

```
\begin{longtable}{llll}
  \caption{ascii table} \\
  \bfseries Char & \bfseries Dec & \bfseries Oct & \bfseries Hex \\
  \endfirsthead
  \multicolumn{4}{c}{\tablename\ \thetable: ascii table cont.}\\
  \bfseries Char & \bfseries Dec & \bfseries Oct & \bfseries Hex \\
  \endhead
  \DTLforeach{ascii}{\symbol=Char,\decBase=Dec,\octBase=Oct,\hexBase=Hex}{\bfseries \symbol & \decBase & \
      octBase & \hexBase\\}
\end{longtable}
```

Our table appears the same except the first column will be bold.

3 Other Software

Some editors, such as TeX Shop, have macros that enable users to paste from spreadsheets and the macro will create a simple tabular environment in a LaTeX document. There are some software tools that interface with other external programs and create a LaTeX table. There are a few for Microsoft Excel. Excel2latex is an example of one written using VBA and installed as an add on for Microsoft Excel. There is a tool for Open Office as well.

There is also software that enables users to connect to certain databases in a LaTeX document. Latexdb is available for Linux and it interfaces with MySQL and PostgreSQL databases. When using the database commands available with latexdb, the "latexdb" command is used instead of "latex" when creating a .dvi file. The program is written in Python; thus, Python is needed to run it. There is another program written in Ruby to perform the same function that could be used on a Windows system. To use it, Ruby must be installed.

References

Kopka, H., & Daly, P. W. (2004). *Guide to LaTeX* (4th ed.). Boston: Addison-Wesley. Retrieved from http://proquest.safaribooksonline.com/9780321617736

Mittelbach, F., Goossens, M., Braams, J., Carlisle, D., & Rowley, C. (2004). *The LaTeX companion* (2nd ed.). Boston: Addison-Wesley. Retrieved from http://proquest.safaribooksonline.com/0201362996