

L^AT_EX

Tables

SAYAK HALDAR

sayakhaldar@ymail.com

SHUVAM BOSANA

shuvambosana0705@gmail.com

SUBHAM BANERJEE

subhambansphs@gmail.com

Students of

Bengal Engineering and Science Univer-
sity, Shibpur

Abstract

Through this document, you will learn how to create tables in L^AT_EX using different packages

Contents

1	Introduction	4
2	The tabular environment:	4
2.1	Basic examples of creating tables using tabular environment:	6
2.2	Manually broken paragraphs in table cells:	12
2.3	Space between columns:	13
2.4	Space between rows:	14
2.5	Other environment inside table:	16
2.6	Defining multiple columns:	16
2.7	Column specification using $\gt\{\backslash\text{cmd}\}$ and $\lt\{\backslash\text{cmd}\}$:	17
2.8	@-expressions:	18
2.9	Aligning columns at decimal points using dcolumn:	19
2.10	Bold text and dcolumn:	20
2.11	Spanning:	21
2.11.1	Row spanning multiple columns:	21
2.11.2	Columns spanning multiple rows:	23
2.11.3	Spanning in both directions simultaneously:	24
2.12	Partial Vertical Lines:	26
2.12.1	Adding a partial vertical line to an individual cell:	26
2.12.2	Removing part of a vertical line in a particular cell:	26
2.13	Controlling table size:	27
2.13.1	Resize tables:	27
2.13.2	Changing font size:	28
2.14	Colors:	29
2.14.1	Alternate row colors in tables:	29
2.14.2	Colors of individual Cells:	31
2.15	Width and stretching:	31
2.15.1	The tabular* environment:	31
2.16	Maths inside table created by tabular environment:	32
3	The tabularx package and table creation:	34
3.1	Basic examples with X columns:	35
3.2	Customising the behaviour of tabularx:	35
3.2.1	Terminal Output:	35
3.2.2	Customization of the X column:	36

4	tabulary package and table creation:	37
4.1	Features of tabulary :	37
4.2	Example with tabulary:	38
5	Tabu package and table creation:	39
5.1	The tabu environment:	39
5.1.1	Basic example with tabu	39
5.1.2	Use and customization of X column:	41
5.1.3	Width and stretching of tables with tabu environment:	46
5.1.4	Mastering vertical space with tabu environment:	47
5.1.5	Using other environments within tabu:	49
5.1.6	Lines leaders and colors inside tabu:	50
5.1.7	Modifying the font and the alignment in one row:	59
5.1.8	Saving and restoring a tabu:	60
5.1.9	Some other features:	63
5.2	longtabu environment:	64
6	longtable package and table creation:	66
6.1	Useful commands and feature:	66
6.2	Chunk size:	67
6.3	Multicolumn entries:	68
7	supertabular package and table creation:	68
7.1	Useful features, commands & environments:	69
7.2	Weakpoints of supertabular:	70
7.3	Example:	70
8	ctable package and table creation:	72
8.1	Usages:	72
8.2	Options:	73
8.3	The width and maxwidth options:	78
8.4	Tables wider than the text width:	78
8.5	Other commands:	79
8.6	Examples:	80
8.6.1	Use of different options in the optional field of \ctable :	83
9	booktabs package and table creation:	96
9.1	Useful commands:	97
9.2	Booktabs and longtables:	98
9.3	Booktabs and ctable package:	98
9.4	Booktabs and and the colortbl package:	98

10 Footnotes in tables:	99
10.1 Creating footnotes with footnotes command:	99
10.2 Creating tablefootnote or tablenote using tablefootnote package: .	99
10.3 Creating footnotes with \footnotetext command:	101
10.4 Creating footnotes in other table environments and using other packages:	102
10.5 Use of threeparttable package in creating tablenotes:	104
11 Sideways tables:	106
12 Floats,figures and captions:	106
12.1 Floats:	106
12.1.1 Placement specifiers:	107
12.1.2 creating a list of tables:	107
12.2 Tables:	107
12.3 captions:	108
12.3.1 Examples:	108
12.4 Subfloats:	109
13 References:	110

1 Introduction

Tables are a common feature in academic writing, often used to summarise research results. Mastering the art of table construction in \LaTeX is therefore necessary to produce quality papers and with sufficient practise one can print beautiful tables of any kind.

There are several packages in \LaTeX which can provide you the facilities of creating beautiful tables like **tabularx**, **tabulary**, **dcolumn**, **ctable**, **booktabs**, **longtable**, **tabu**, **supertabular** etc. . . . \LaTeX also provides an environment named **tabular** to create tables by default (you don't have to include any additional packages for that).

Here, you will first learn the table creation with **tabular** environment. Then table creation using other packages will be discussed.

2 The tabular environment:

The tabular environment can be used to typeset tables with optional horizontal and vertical lines. \LaTeX determines the width of the columns automatically.

The first line of the environment has the form:

```
\begin{tabular}[pos]{table spec}
```

The table spec argument tells \LaTeX the alignment to be used in each column and the vertical lines to insert.

The number of columns does not need to be specified as it is inferred by looking at the number of arguments provided. It is also possible to add vertical lines between the columns here.

Table specifications	Meaning
l	left-justified column
c	centered column
r	right-justified column
continued to next page	

continued from previous page	
<code>p{'width'}</code>	paragraph column with text vertically aligned at the top
<code>m{'width'}</code>	paragraph column with text vertically aligned in the middle (requires array package)
<code>b{'width'}</code>	paragraph column with text vertically aligned at the bottom (requires array package)
<code> </code>	Vertical line
<code> </code>	Double vertical line

If you want to create such a table, you cannot type `'|'` & `'||'` directly. You have to use `$|` & `$|` commands for typing `'|'` & `'||'` respectively.
 $\$$:-This is generally called as math symbol. i.e for inserting any kind of math symbols you have to enclose that symbol with a opening $\$$ and a closing $\$$.

By default, if the text in a column is too wide for the page, LaTeX won't automatically wrap it. Using `p'width'` you can define a special type of column which will wrap-around the text as in a normal paragraph. You can pass the width using any unit supported by LaTeX, such as `'pt'` and `'cm'`, or command lengths, such as `\textwidth`.

The optional parameter `pos` can be used to specify the vertical position of the table relative to the baseline of the surrounding text. In most cases, you will not need this option. It becomes relevant only if your table is not in a paragraph of its own. You can use the following letters:

Letter	Meaning
<code>b</code>	bottom
<code>c</code>	center(default)
<code>t</code>	top

To specify a font format (such as bold, italic, etc.) for an entire column, you can add `>\format` before you declare the alignment. For example:

```
\begin{tabular}{>\bfseries}l c >\itshape}r }
```

will indicate a three column table with the first one aligned to the left and in bold font, the second one aligned in the center and with normal font, and the third aligned to the right and in italic. We will provide an example of it later.

In the first line you have pointed out how many columns you want, their alignment

and the vertical lines to separate them. Once in the environment, you have to introduce the text you want, separating between cells and introducing new lines. The commands you have to use are the following:

<code>&</code>	Column Seperator
<code>\\</code>	start new row (additional space may be specified after <code>\\</code> using square brackets, such as <code>[6pt]</code>)
<code>\hline</code>	horizontal line
<code>\newline</code>	start a new line within a cell (in a paragraph column)
<code>\cline{i-j}</code>	partial horizontal line beginning in column i and ending in column j

Note: Any white space inserted between these commands is purely down to ones' preferences

2.1 Basic examples of creating tables using tabular environment:

This example shows how to create a simple table in LaTeX. It is a three-by-three table, but without any horizontal or vertical lines.

```
\begin{tabular}{l c r }
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
```

If you use the given code, the output would be like:

1	2	3
4	5	6
7	8	9

Note: In the code, we did not use `\begin{center}` & `\end{center}`. Otherwise the table would be created after the written line”If you use the given code, the output would be like” and the table creation would be started from the middle of the next line. Also, if we give `\\` after the written line, the table creation would be started from the left side of the next line.

Expanding upon that by including some vertical lines:

```
\begin{tabular}{l | c | r }
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
```

If you use this code, the output would be:

1	2	3
4	5	6
7	8	9

Now, if you use two vertical lines as separation lines between the columns, you have to use the following command:

```
\begin{tabular}{l || c || r }
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
```

This will produce output like:

1	2	3
4	5	6
7	8	9

A more complicated example will be provided where we will use both horizontal lines and vertical lines to separate rows and columns respectively.

```
\begin{tabular}{|l|c|r|}
\hline
1&2&3\\
4&5&6\\
7&8&9\\
\hline
\end{tabular}
```

This will produce output like:

1	2	3
4	5	6
7	8	9

Now, we will illustrate an example where the leftmost column will be written in bold font and rightmost column will be written in the italics font. Remember the table where we specify the letters for the optional parameter **pos** and its role in creation of a table? The table has two columns -Letter and Meaning. We

will rewrite the table where the 'Letter' column will be written in bold font and 'Meaning' column will be written in italics font.

Letter	<i>Meaning</i>
b	<i>bottom</i>
c	<i>center(default)</i>
t	<i>top</i>

The table is created by the following command:

```
\begin{center}
\begin{tabular}{|>\bfseries l|>\itshape r|}
\hline
Letter&Meaning\\\hline
b&bottom\\\hline
c&center(default)\\\hline
t&top\\\hline
\end{tabular}
\end{center}
```

Now, we will illustrate an example so you could understand the use of p{'width'}. Remember the table where we mention table specifications and its use. Now, if you look at the given code, you will see that we use p{10cm}. In the next example we will just replace the p{10cm} with r and you will see the difference.

```
\begin{center}
\begin{tabular}{|l|p{10cm}|}
\hline
Table specifications&Meaning\\\hline
l&left-justified column
\\\hline
c&centered column\\\hline
r&right-justified column\\\hline
p\{'width'\}&\paragraph column with text vertically aligned at
the top\\\hline
m\{'width'\}&\paragraph column with text vertically aligned in
the middle (requires array package)\\\hline
\end{tabular}
\end{center}
```

continued to next page

```

b\{'width'\}&paragraph column with text vertically aligned at
the bottom (requires array package)\\hline
$\mid$&Vertical line\\hline
$\parallel$&Double vertical line\\hline
\end{tabular}
\end{center}

```

This will produce output like:

Table specifications	Meaning
l	left-justified column
c	centered column
r	right-justified column
p{'width'}	paragraph column with text vertically aligned at the top
m{'width'}	paragraph column with text vertically aligned in the middle (requires array package)
b{'width'}	paragraph column with text vertically aligned at the bottom (requires array package)
	Vertical line
	Double vertical line

Now, if we change the **p{10cm}** with r(right justified column) in this code, this will produce output like:

Table specifications	Meaning
l	left-justified column
c	centered column
r	right-justified column
p{'width'}	paragraph column with text vertically aligned at the top
m{'width'}	paragraph column with text vertically aligned in the middle (requires array package)
b{'width'}	paragraph column with text vertically aligned at the bottom (requires array package)
	Vertical line
	Double vertical line

Note: This problem is caused since \LaTeX have some problems with the Text wrapping in tables issue. It is that \LaTeX will not wrap text in cells, even if it overruns the width of the page.

Now, we will illustrate the difference between $\text{\texttt{p\{width\}}}$, $\text{\texttt{m\{width\}}}$ and $\text{\texttt{b\{width\}}}$ by means of examples.

First, we will use the $\text{\texttt{p\{width\}}}$ specification.

```
\begin{center}
\begin{tabular}{|p{4cm}|p{10cm}|}
\hline
Table specifications&Meaning\\\hline
l&left-justified column
\\\hline
c&centered column\\\hline
r&right-justified column\\\hline
p\{'width'\}&paragraph column with text vertically aligned at
the top\\\hline
m\{'width'\}&paragraph column with text vertically aligned in
the middle (requires array package)\\\hline
b\{'width'\}&paragraph column with text vertically aligned at
the bottom (requires array package)\\\hline
$\mid$&Vertical line\\\hline
$\parallel$&Double vertical line\\\hline
\end{tabular}
\end{center}
```

This will produce output like:

Table specifications	Meaning
l	left-justified column
c	centered column
r	right-justified column
p{'width'}	paragraph column with text vertically aligned at the top
m{'width'}	paragraph column with text vertically aligned in the middle (requires array package)
b{'width'}	paragraph column with text vertically aligned at the bottom (requires array package)
	Vertical line
	Double vertical line

Now,if we change **p{4cm}** with **m{4cm}** and **p{10cm}** with **m{10cm}** in the given code, it will produce output like:

Table specifications	Meaning
l	left-justified column
c	centered column
r	right-justified column
p{'width'}	paragraph column with text vertically aligned at the top
m{'width'}	paragraph column with text vertically aligned in the middle (requires array package)
b{'width'}	paragraph column with text vertically aligned at the bottom (requires array package)
	Vertical line
	Double vertical line

And finally if we use **b{4cm}** and **b{10cm}** instead of **p{4cm}** and **p{10cm}** respectively:

Table specifications	Meaning
l	left-justified column
c	centered column
r	right-justified column
p{'width'}	paragraph column with text vertically aligned at the top
m{'width'}	paragraph column with text vertically aligned in the middle (requires array package)
b{'width'}	paragraph column with text vertically aligned at the bottom (requires array package)
	Vertical line
	Double vertical line

Do you understand the difference between these three examples? If not, then notice the **table specification** column carefully.

Now, we will illustrate an example to make you learn the use of `\cline{i-j}`, i.e the use of partial horizontal line beginning in column i and ending in column j. If we use the following commands:

```
\begin{tabular}{|r|l|}
\hline
7C0 & hexadecimal \\
3700 & octal \\
11111000000 & binary \\
\hline
1984 & decimal \\
\hline
\end{tabular}
```

This will produce output like:

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

2.2 Manually broken paragraphs in table cells:

Sometimes it is necessary to not rely on the breaking algorithm when using the p specifier, but rather specify the line breaks by hand. In this case it is easiest to use a `\parbox`:

```
\begin{tabular}{|c|c|}\hline
  boring cell content & \parbox[t]{4cm}{rather long par
  new par}
  \\ \hline
\end{tabular}
```

The output would be like:

boring cell content	rather long par new par
---------------------	----------------------------

Using this `\parbox`, what we have done that we make sure when the length of the text "rather long par new par" will cross 4cm, the linebreak will occur automatically. However, like we discussed before that the line breaking could be done by using `\\` or `\newline`.

2.3 Space between columns:

To tweak the space between columns (LaTeX will by default choose very tight columns), one can alter the column separation:

`\setlength{\tabcolsep}{length unit}`. The default value is 6pt.

Now, we are going to illustrate two examples. We first use the normal column separation between two columns, then we will use the column separation between two columns as 10pt.

```
\begin{center}
\begin{tabular}{|l|c|r|}\hline
b&bottom\\ \hline
c&center (default)\\ \hline
t&top\\ \hline
\end{tabular}
\end{center}
```

It will produce output like:

b	bottom
c	center (default)
t	top

Now, if we change the column separation between two columns as 10pt:

```
\begin{center}
\setlength{\tabcolsep}{10pt}
\begin{tabular}{|l|c|r|}\hline
b&bottom\\\hline
c&center (default)\\\hline
t&top\\\hline
\end{tabular}
\end{center}
```

It will produce output like:

b	bottom
c	center (default)
t	top

Now, see the difference between these two tables.

2.4 Space between rows:

One way is to redefine the `\arraystretch` command to set the space between rows:

```
\renewcommand{\arraystretch}{1.5}
```

Default value is 1.0.

An alternative way to adjust the rule spacing is to add `\noalign{\smallskip}` before or after the `\hline` and `\cline{i-j}` commands:

```
\begin{tabular}{| l | l | r | }
\hline\noalign{\smallskip}
\multicolumn{2}{c}{Item} \\
\cline{1-2}\noalign{\smallskip}
```

continued to next page

```

Animal & Description & Price (\$) \\
\noalign{\smallskip}\hline\noalign{\smallskip}
Gnat  & per gram & 13.65 \\
      & each      & 0.01 \\
Gnu   & stuffed  & 92.50 \\
Emu   & stuffed  & 33.33 \\
Armadillo & frozen & 8.99 \\
\noalign{\smallskip}\hline
\end{tabular}

```

It will produce output like:

Item		
Animal	Description	Price (\$)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

You may also specify the skip after a line explicitly using glue after the line terminator:

```

\begin{tabular}{ll}
\hline
Mineral & Color \\[1cm]
Ruby    & red \\
Sapphire & blue \\
\hline
\end{tabular}

```

Mineral	Color
---------	-------

It will produce output like:

Ruby	red
Sapphire	blue

2.5 Other environment inside table:

If you use \LaTeX environment inside table cells, like **verbatim** or **enumerate**, you might encounter errors.

```
\begin{tabular}{c c}
    \hline
    \begin{verbatim}
code
\end{verbatim}
& description
\\ \hline
\end{tabular}
```

If you write this type of codes, you will get error.

To solve this problem, change column specifier to "paragraph" (p, m or b).

2.6 Defining multiple columns:

It is possible to define many identical columns at once using the `*{"num"}{"str"}` syntax. This is particularly useful when your table has many columns.

Here is a table with six centered columns flanked by a single column on each side:

```
\begin{tabular}{l*{8}{c}r}
\hline
\#&Team&GP&W&D&L&GF&GA&GD&PTS\\ \hline
1&Barcelona&17&15&1&1&49&12&37&46\\
2&Atletico Madrid&17&15&1&1&46&11&35&46\\
3&Real Madrid&17&13&2&2&49&21&28&41\\
4&Athletic&17&10&3&4&26&21&5&33\\ \hline
\end{tabular}
```

It will produce output like:

#	Team	GP	W	D	L	GF	GA	GD	PTS
1	Barcelona	17	15	1	1	49	12	37	46
2	Atletico Madrid	17	15	1	1	46	11	35	46
3	Real Madrid	17	13	2	2	49	21	28	41
4	Athletic	17	10	3	4	26	21	5	33

2.7 Column specification using $\>\{\backslash\text{cmd}\}$ and $\<\{\backslash\text{cmd}\}$:

The column specification can be altered using the array package. This is done in the argument of the tabular environment using $\>\{\backslash\text{command}\}$ for commands executed right before each column element and $\<\{\backslash\text{command}\}$ for commands to be executed right after each column element. As an example: to get a column in math mode enter: $\backslash\text{begin}\{\text{tabular}\}\{\>\{\backslash\text{c}\}\{\backslash\text{c}\}\}$. Another example is changing the font: $\backslash\text{begin}\{\text{tabular}\}\{\>\{\backslash\text{small}\}\text{c}\}$ to print the column in a small font.

The argument of the $\>$ and $\<$ specifications must be correctly balanced when it comes to $\{$ and $\}$ characters. This means that $\>\{\backslash\text{bfseries}\}$ is valid, while $\>\{\backslash\text{textbf}\}$ will not work and $\>\{\backslash\text{textbf}\}$ is not valid. If there is the need to use the text of the table as an argument (for instance, using the $\backslash\text{textbf}$ to produce bold text), one should use the $\backslash\text{bgroup}$ and $\backslash\text{egroup}$ commands: $\>\{\backslash\text{textbf}\backslash\text{bgroup}\}\text{c}\<\{\backslash\text{egroup}\}$ produces the intended effect. This works only for some basic LaTeX commands. For other commands, such as $\backslash\text{underline}$ to underline text, it is necessary to temporarily store the column text in a box using lrbox . First, you must define such a box with $\backslash\text{newsavebox}\{\backslash\text{boxname}\}$ and then you can define:

```
>\begin{lrbox}\boxname }%
l%
<\end{lrbox}%
  \underline{\unhbox\boxname }%
}
```

This stores the text in a box and afterwards, takes the text out of the box with $\backslash\text{unhbox}$ (this destroys the box, if the box is needed again one should use $\backslash\text{unhcopy}$ instead) and passing it to $\backslash\text{underline}$. (For LaTeX2e, you may want to use $\backslash\text{usebox}\{\backslash\text{boxname}\}$ instead of $\backslash\text{unhbox}\backslash\text{boxname}$.)

This same trick done with $\backslash\text{raisebox}$ instead of $\backslash\text{underline}$ can force all lines in a table to have equal height, instead of the natural varying height that can occur when e.g. math terms or superscripts occur in the text.

Here is an example showing the use of both $\text{p}\{\dots\}$ and $\>\{\backslash\text{centering}\}$:

```
\begin{tabular}{>{\centering}p{3.5cm}<{\centering}p{3.5cm} }
Geometry & Algebra
\tabularnewline
\hline
Points & Addition
\tabularnewline
Spheres & Multiplication
\end{tabular}
```

It will produce output like:

	Geometry	Algebra
	Points	Addition
	Spheres	Multiplication

Note the use of `\tabularnewline` instead of `\\` to avoid a Misplaced `\noalign` error.

2.8 @-expressions:

The column separator can be specified with the `@{...}` construct.

It typically takes some text as its argument, and when appended to a column, it will automatically insert that text into each cell in that column before the actual data for that cell. This command kills the inter-column space and replaces it with whatever is between the curly braces. To add space, use `@{\hspace{"width"}}`.

Admittedly, this is not that clear, and so will require a few examples to clarify. Sometimes, it is desirable in scientific tables to have the numbers aligned on the decimal point. This can be achieved by doing the following:

```
\begin{tabular}{r@{.}l}
3 & 14159 \\
16 & 2 \\
123 & 456 \\
\end{tabular}
```

It will produce output like:

	3.14159
	16.2
	123.456

The space-suppressing qualities of the `@`-expression actually make it quite useful

for manipulating the horizontal spacing between columns. Given a basic table, and varying the column descriptions:

```
\begin{tabular}{|l|l|}
\hline
stuff & stuff \\ \hline
stuff & stuff \\
\hline
\end{tabular}
```

It will produce output like:

stuff	stuff
stuff	stuff

Now, if we replace the `{|l|l|}` with `{|@{}l|la{}}|}`... (1), `{|@{}l@{}|l@{}|}` (2) & `{|@{}l@{}|@{}l@{}|}` (3) respectively, we will get outputs like:

stuff	stuff
stuff	stuff

stuff	stuff
stuff	stuff

&

stuff	stuff
stuff	stuff

for (1),(2) & (3) respectively.

2.9 Aligning columns at decimal points using `dcolum`:

Instead of using **@-expressions** to build columns of decimals aligned to the decimal point (or equivalent symbol), it is possible to center a column on the decimal separator using the **dcolum** package, which provides a new column specifier for floating point data. A simple way to use **dcolum** is as follows.

```
\newcolumntype{d}[1]{D{.}{\cdot}{#1} }
%the argument for d specifies the maximum number of decimal
%places
\begin{tabular}{l r c d{1} }
Left&Right&Center&\mathrm{Decimal}\\
1&2&3&4\\
11&22&33&44\\
1.1&2.2&3.3&4.4\\
\end{tabular}
```

It will produce output like:

	Left	Right	Center	Decimal
	1	2	3	4
	11	22	33	44
	1.1	2.2	3.3	4.4

A negative argument provided for the number of decimal places in the new column type allows unlimited decimal places, but may result in rather wide columns. Rounding is not applied, so the data to be tabulated should be adjusted to the number of decimal places specified. Note that a decimal aligned column is typeset in math mode, hence the use of `\mathrm` for the column heading in the example above. Also, text in a decimal aligned column (for example the header) will be right-aligned before the decimal separator (assuming there's no decimal separator in the text). While this may be fine for very short text, or numeric column headings, it looks cumbersome in the example above. A solution to this is to use the `\multicolumn` command described below, specifying a single column and its alignment. For example to center the header `Decimal` over its column in the above example, the first line of the table itself would be `Left&Right&Center&\multicolumn{1}{c}{\mathrm{Decimal}}\end{table}`

2.10 Bold text and dcolumn:

To draw attention to particular entries in a table, it may be nice to use bold text. Ordinarily this is easy, but as `dcolumn` needs to see the decimal point it is rather harder to do. In addition, the usual bold characters are wider than their normal counterparts, meaning that although the decimals may align nicely, the figures (for more than 2-3 digits on one side of the decimal point) will be visibly misaligned. It is however possible to use normal width bold characters and define a new bold column type, as shown below.

```
\usepackage{dcolumn}
%here we're setting up a version of the math fonts with normal
%x-width
\DeclareMathVersion{nxbold}
\SetSymbolFont{operators}{nxbold}{OT1}{cmr}{b}{n}
\SetSymbolFont{letters}{nxbold}{OML}{cmm}{b}{it}
\SetSymbolFont{symbols}{nxbold}{OMS}{cmsy}{b}{n}
\begin{document}
\makeatletter
\newcolumntype{d}{D{.}{.}{-1}} %decimal column as before
%wide bold decimal column
\newcolumntype{B}[3]{>\boldmath\DC@{#1}{#2}{#3}}c<\DC@end} }
%normal width bold decimal column
```

continud to next page

```

\newcolumntype{Z}[3]{>\mathversion{nxbold}\DC@{#1}{#2}{#3} }c
<{\DC@end} }
\makeatother
\begin{tabular}{l l d}
Type & M & \multicolumn{1}{c}{N} \\
Normal & 1 & 22222.222 \\
Bold (standard)&10 & \multicolumn{1}{B{.}{.}{-1} }{22222.222} \\
Bold (nxbold)&100 & \multicolumn{1}{Z{.}{.}{-1} }{22222.222} \\
\end{tabular}
\end{document}

```

	Type	M	N
It will produce output like:	Normal	1	22222.222
	Bold (standard)	10	22222.222
	Bold (nxbold)	100	22222.222

2.11 Spanning:

2.11.1 Row spanning multiple columns:

The command for this looks like this: `\multicolumn{'num_cols'}{'alignment'}{'contents'}`. **num_cols** is the number of subsequent columns to merge; alignment is either **l**, **c**, **r**, or to have text wrapping specify a width like: **p{5.0cm}** . And **contents** is simply the actual data you want to be contained within that cell.

An example: we want to create the Team sheet of Indian playing xi(cricket) of the third match which was played at Eden Park, Auckland 25 January 2014(no offence,cricket is my favourite sports):

```

\begin{center}
\begin{tabular}{|l|l|l| } \hline
\multicolumn{2}{|l|}{Team sheet} \\ \hline
Batsman&S Dhawan \\ \hline
Batsman&RG Sharma \\ \hline
Batsman&V Kohli \\ \hline
Batsman&AM Rahane \\ \hline
\end{tabular}
\end{center}

```

```

Batsman&SK Raina\\ \\hline
Wicketkeeper&MS Dhoni\\ \\hline
Allrounder&R Ashwin\\ \\hline
Allrounder&RA Jadeja\\ \\hline
Bowler&B Kumar\\ \\hline
Bowler&Mohammed Shami\\ \\hline
Bowler&VR Aaron\\ \\hline
\\end{tabular}
\\end{center}

```

It will produce output like:

Team sheet	
Batsman	S Dhawan
Batsman	RG Sharma
Batsman	V Kohli
Batsman	AM Rahane
Batsman	SK Raina
Wicketkeeper	MS Dhoni
Allrounder	R Ashwin
Allrounder	RA Jadeja
Bowler	B Kumar
Bowler	Mohammed Shami
Bowler	VR Aaron

Note: If we use `\multicolumn{2}{|c|}{Team sheet}` or `\multicolumn{2}{|r|}{Team sheet}`

instead of

`\multicolumn{2}{|l|}{Team sheet}`

the output will be same. Because, in l(left-justified column),c(centered column) and r(right-justified column), width specification is not needed and when we specify :

`\multicolumn{2}{|l|}{Team sheet}`

we are specifying a different entity. That means we are instructing L^AT_EX that we want to merge two columns whose width are not specified. and we need two vertical column separator around the merged cell. Suppose, you have to merge one l(left-justified column) and one r(right-justified column) then if you use :

contd.

```
\multicolumn{2}{|l|}{written portion}
```

L^AT_EX will not show you any error and two columns will be perfectly merged. But if you want to merge one l left justified column and p{width} column then you have to mention a length like:

```
\multicolumn{2}{|p{width}|}{written portion}
```

mentioning `\multicolumn{2}{|l|}{written portion}` won't work for you now. And you have to predict the right length. Things become a little easier when you have to combine two columns which lengths are specified. Suppose, you have created a table with two columns. First column is specified as `p{length1}` and second column is specified as `p{length2}` then for merging these two cells you have to specify like this:

```
\multicolumn{2}{|p{combined width}|}{written portion}
```

where combined width =length1+length2

2.11.2 Columns spanning multiple rows:

Here, you need to use **multirow** package. After loading the mentioned package, it provides the command needed for spanning rows:

```
\multirow{'num_rows'}{'width'}{'contents'}.
```

The arguments are pretty simple to deduce (* for the width means the content's natural width).

An example: Here, we are going to create the team sheet of **Leeds United** of 2003-04 season:

```
\begin{tabular}{|l|l|l|l| }
\hline
\multicolumn{3}{|c| }{Team sheet} \\
\hline
Goalkeeper & GK & Paul Robinson \\
\hline
\multirow{4}{*}{Defenders} & LB & Lucus Radebe \\
& DC & Michael Duberry \\
& DC & Dominic Matteo \\
& RB & Didier Domi \\
\hline
```

continued to next page


```

\multirow{3}{*}{Midfielders} & MC & David Batty \\
& MC & Eirik Bakke \\
& MC & Jody Morris \\ \hline
Forward & FW & Jamie McMaster \\ \hline
\multirow{2}{*}{Strikers} & ST & Alan Smith \\
& ST & Mark Viduka \\
\hline
\end{tabular}

```

It will produce output like:

Team sheet		
Goalkeeper	GK	Paul Robinson
Defenders	LB	Lucas Radebe
	DC	Michael Duberry
	DC	Dominic Matteo
	RB	Didier Domi
Midfielders	MC	David Batty
	MC	Eirik Bakke
	MC	Jody Morris
Forward	FW	Jamie McMaster
Strikers	ST	Alan Smith
	ST	Mark Viduka

The main thing to note when using `\multirow` is that a blank entry must be inserted for each appropriate cell in each subsequent row to be spanned.

If there is no data for a cell, just don't type anything, but you still need the "&" separating it from the next column's data. The astute reader will already have deduced that for a table of columns, there must always be ampersands in each row (unless `\multicolumn` is also used).

2.11.3 Spanning in both directions simultaneously:

```

\begin{tabular}{cc|c|c|c|c|l}
\cline{3-6}

```

```

& & \multicolumn{4}{c|}{Primes} \\ \cline{3-6}
& & 2 & 3 & 5 & 7 \\ \cline{1-6}
\multicolumn{1}{|c|}{\multirow{2}{*}{Powers}} & & & & & \\
\multicolumn{1}{|c|}{504} & 3 & 2 & 0 & 1 & \\ \cline{2-6}
\multicolumn{1}{|c|}{} & & & & & \\
\multicolumn{1}{|c|}{540} & 2 & 3 & 1 & 0 & \\ \cline{1-6}
\multicolumn{1}{|c|}{\multirow{2}{*}{Powers}} & & & & & \\
\multicolumn{1}{|c|}{gcd} & 2 & 2 & 0 & 0 & min \\ \cline{2-6}
\multicolumn{1}{|c|}{} & & & & & \\
\multicolumn{1}{|c|}{lcm} & 3 & 3 & 1 & 1 & max \\ \cline{1-6}
\end{tabular}

```

It will produce output like:

		Primes			
		2	3	5	7
Powers	504	3	2	0	1
	540	2	3	1	0
Powers	gcd	2	2	0	0
	lcm	3	3	1	1
		min			
		max			

The command `\multicolumn{1}{}` is just used to draw vertical borders both on the left and on the right of the cell. Even when combined with `\multirow{2}{*}{...}`, it still draws vertical borders that only span the first row. To compensate for that, we add `\multicolumn{1}{}` in the following rows spanned by the multirow. Note that we cannot just use `\hline` to draw horizontal lines, since we do not want the line to be drawn over the text that spans several rows. Instead we use the command `\cline{2-6}` and opt out the first column that contains the text "Powers".

Here is another example exploiting the same ideas to make the familiar and popular "2x2" or double dichotomy:

```

\begin{tabular}{r|c|c|}
\multicolumn{1}{r}{}
& \multicolumn{1}{c}{noninteractive}
& \multicolumn{1}{c}{interactive} \\ \cline{2-3}
massively multiple & Library & University
\end{tabular}

```

```
\cline{2-3}
one-to-one & Book & Tutor \\
\cline{2-3}
\end{tabular}
```

It will produce like:

	noninteractive	interactive
massively multiple	Library	University
one-to-one	Book	Tutor

2.12 Partial Vertical Lines:

This is an indirect application of `\multicolumn` command.

2.12.1 Adding a partial vertical line to an individual cell:

```
\begin{tabular}{l c r }
\hline
1 & 2 & 3 \\ \hline
4 & 5 & \multicolumn{1}{r|}{6} \\ \hline
7 & 8 & 9 \\ \hline
\end{tabular}
```

This will produce output like:

1	2	3
4	5	6
7	8	9

2.12.2 Removing part of a vertical line in a particular cell:

```
\begin{tabular}{| l | c | r | }
\hline
```

```

1 & 2 & 3 \\ \hline
4 & 5 & \multicolumn{1}{r}{6} \\ \hline
7 & 8 & 9 \\ \hline
\end{tabular}

```

This will produce output like:

1	2	3
4	5	6
7	8	9

2.13 Controlling table size:

2.13.1 Resize tables:

The **graphicx** packages features the command `\resizebox{width}{height}{object}` which can be used with `tabular` to specify the height and width of a table. The following example shows how to resize a table to 8cm width while maintaining the original width/height ratio.

```

\usepackage{graphicx}
% ...
\resizebox{8cm}{!} {
  \begin{tabular}...
  \end{tabular}
}

```

Alternatively you can use `\scalebox{ratio}{object}` in the same way but with ratios rather than fixed sizes:

```

\usepackage{graphicx}
% ...
\scalebox{0.7}{
  \begin{tabular}...
  \end{tabular}
}

```

2.13.2 Changing font size:

A table can be globally switched to a different font size by simply adding the desired size command (here: `\footnotesize`) in the table scope, which may be after the `\begin{table}` statement if you use floats, otherwise you need to add a group delimiter.

In the case of using tabular environment:

```
{\footnotesize
  \begin{tabular}{| r | r || c | c | c |}
    % ...
  \end{tabular}
}
```

The code structure would be like this.

And in the case where **tabular** environment is wrapped inside **table** environment the code structure would be like:

```
\begin{table}[h]\footnotesize
  \caption{Performance at peak F-measure}
  \begin{tabular}{| r | r || c | c | c |}
    % ...
  \end{tabular}
\end{table}
```

Alternatively, you can change the default font for all the tables in your document by placing the following code in the preamble:

```
\let\oldtabular\tabular
\renewcommand{\tabular}{\footnotesize\oldtabular}
```

As you can see, we are trying to redefine the default font size of the **tabular** environment here.

2.14 Colors:

2.14.1 Alternate row colors in tables:

The xcolor package provides the necessary commands to produce tables with alternate row colors, when loaded with the table option. The command:

`\rowcolors{<"starting row">}{<"odd color">}{<"even color">}`

has to be specified right before the tabular environment starts.

```
\begin{center}
\rowcolors{1}{Aquamarine}{white}
\begin{tabular}{lll}
odd    & odd    & odd \\
even   & even   & even \\
odd    & odd    & odd \\
even   & even   & even \\
\end{tabular}
\end{center}
```

This will produce output like:

odd	odd	odd
even	even	even
odd	odd	odd
even	even	even

As you can see in the code, that we mention the starting row number as 1. However, if we mention the starting row number as 2 instead of 1 in that code, the first two rows will be of color white.

The command `\hiderowcolors` is available to deactivate highlighting from a specified row until the end of the table. Highlighting can be reactivated within the table via the `\showrowcolors` command. If while using these commands you experience "misplaced `\noalign` errors" then use the commands at the very beginning or end of a row in your tabular.

Now, we will provide you another example to illustrate the process of using `\hiderowcolors` and `\showrowcolors` commands.

```

\begin{center}
\rowcolors{1}{ForestGreen!40}{DarkOrchid!30}
\begin{tabular}{lll}
odd      & odd    & odd \\
even     & even   & even \\
odd      & odd    & odd \\
even     & even   & even \\
odd      & odd    & odd \\
even     & even   & even \\
\end{tabular}
\end{center}

```

This will produce output like:

odd	odd	odd
even	even	even
odd	odd	odd
even	even	even
odd	odd	odd
even	even	even

Now, suppose, you want to hide the rowcolor for *only* the third row, then you have to use the following code:

```

\begin{center}
\rowcolors{1}{ForestGreen!40}{DarkOrchid!30}
\begin{tabular}{lll}
odd      & odd    & odd \\
even     & even   & even \\
\hiderowcolors odd      & odd    & odd \\
\showrowcolors even     & even   & even \\
odd      & odd    & odd \\
even     & even   & even \\
\end{tabular}
\end{center}

```

It will produce output like:

odd	odd	odd
even	even	even
odd	odd	odd
even	even	even
odd	odd	odd
even	even	even

2.14.2 Colors of individual Cells:

As above this uses the **xcolor** package.

A sample code structure:

```
% Enter this in the cell you wish to color a light grey.
% NB: the word 'gray' here denotes the grayscale color scheme,
%not the color grey.
%'0.9' denotes how dark the grey is.
\cellcolor[gray]{0.9}
% The following will color the cell red.
\cellcolor{red}
```

2.15 Width and stretching:

This cant be done in **tabular** environment, but with the help of **tabular*** environment which is a slightly modified version of the old **tabular** environment, we can do it.

2.15.1 The **tabular*** environment:

This is basically a slight extension on the original **tabular** version, although it requires an extra argument (before the column descriptions) to specify the preferred width of the table.

```
\begin{tabular*}{0.75\textwidth}{ | c | c | c | r | }
\hline
label 1 & label 2 & label 3 & label 4 \\\
```

continued to next page


```

\hline
item 1 & item 2 & item 3 & item 4 \\
\hline
\end{tabular*}

```

This will produce output like:

label 1	label 2	label 3	label 4	
item 1	item 2	item 3	item 4	

However, that may not look quite as intended. The columns are still at their natural width (just wide enough to fit their contents) while the rows are as wide as the table width specified. If you do not like this default, you must also explicitly insert extra column space. LaTeX has rubber lengths, which, unlike others, are not fixed. L^AT_EX can dynamically decide how long the lengths should be. So, an example of this is the following.

```

\begin{tabular*}
{0.75\textwidth}{@{\extracolsep{\fill}} } | c | c | c | r | }
\hline
label 1 & label 2 & label 3 & label 4 \\
\hline
item 1 & item 2 & item 3 & item 4 \\
\hline
\end{tabular*}

```

This will produce output like:

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

2.16 Maths inside table created by tabular environment:

We often need to insert math symbols within a table. It can be inserted using math mode (i.e. enclosing it within a pair of $\$$ s) for a particular cell. But, suppose you want to create a table about the commands for inserting mathematical symbols in L^AT_EX. In that kind of table, you need a column of mathematical symbols. For that type of cases, L^AT_EX provides you some features using which you can do it easily.

```

\begin{center}
\begin{tabular}{| >{\$}l<{\$} | >{\$}c<{\$} | >{\$}c<{\$} | >{\$}r<{\$}|}
\hline
Symbol&Command&Symbol&Command
\\ \hline
\le&\textbackslash{}le&\ge&\textbackslash{}ge
\\ \hline
\neq&\textbackslash{}neq&\sim&\textbackslash{}sim
\\ \hline
\ll&\textbackslash{}ll&\gg&\textbackslash{}gg
\\ \hline
\doteq&\textbackslash{}doteq&\simeq&\textbackslash{}simeq
\\ \hline
\subset&\textbackslash{}subset&\supset&\textbackslash{}supset
\\ \hline
\subseteq&\textbackslash{}subseteq&\supseteq&\textbackslash{}supseteq
\\ \hline
\approx&\textbackslash{}approx&\equiv&\textbackslash{}equiv
\\ \hline
\propto&\textbackslash{}propto&\perp&\textbackslash{}perp
\\ \hline
\in&\textbackslash{}in&\neq&\textbackslash{}neq
\\ \hline
\mid&\textbackslash{}mid&\parallel&\textbackslash{}parallel
\\ \hline
\end{tabular}
\end{center}

```

It will produce output like:

<i>Symbol</i>	<i>Command</i>	<i>Symbol</i>	<i>Command</i>
\leq	<i>nle</i>	\geq	<i>nge</i>
\neq	<i>nneq</i>	\sim	<i>nsim</i>
\ll	<i>nll</i>	\gg	<i>ngg</i>
$\dot{=}$	<i>ndoteq</i>	\simeq	<i>nsimeq</i>
\subset	<i>nsubset</i>	\supset	<i>nsupset</i>
\subseteq	<i>nsubseteq</i>	\supseteq	<i>nsupseteq</i>
\approx	<i>napprox</i>	\equiv	<i>nequiv</i>
\propto	<i>npropto</i>	\perp	<i>nperp</i>
\in	<i>nin</i>	\ncong	<i>nne</i>
	<i>nmid</i>	\parallel	<i>nparallel</i>

As you can see, you need not to insert math mode for each cells of a particular column any more. Because use of **array** package provides you advanced column specification options: $\>\{\backslash\textbf{command}\}$: for commands executed right before each column element.

and

$\<\{\backslash\textbf{command}\}$: for commands to be executed right after each column element.

3 The tabularx package and table creation:

The **tabularx** package provides the environment named **tabularx** , which takes the same arguments as **tabular***, but modifies the widths of certain columns, rather than the inter column space, to set a table with the requested total width. The columns that may stretch are marked with the new token X in the preamble argument.

This also requires the loading of **array** package.

You have to mention the width and also have to mention optional arguments in the preamble when are starting **tabularx** environment by the command:

```
\begin{tabularx}{<width>}{<preamble>}
```

The arguments of tabularx are essentially the same as those of the standard tabular* environment. However rather than adding space between the columns to achieve the desired width, it adjusts the widths of some of the columns. The columns which are acted by the tabularx environment should be denoted with

the letter X in the preamble argument. The X column specification will be converted to p<some value> once the correct column width has been calculated.

3.1 Basic examples with X columns:

```
\begin{tabularx}{\textwidth}{|X|X|X|X| }
\hline
label 1 & label 2 & label 3 & label 4 \\
\hline
item 1 & item 2 & item 3 & item 4 \\
\hline
\end{tabularx}
```

It will provide output like:

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

3.2 Customising the behaviour of tabularx:

3.2.1 Terminal Output:

\tracingtabularx: If this declaration is made, say in the document preamble, then all following tabularx environments will print information about column widths as they repeatedly reset the tables to find the correct widths. As an alternative to using the **\tracingtabularx** declaration, either of the options **infoshow** or **debugshow** may be given, either in the **\usepackage** command that loads tabularx, or as a global option in the **\documentclass** command.

With the help of **tabularx** package, one can do many things with **X column**.

3.2.2 Customization of the X column:

With the help of **tabularx** package, you can use the following specifications for the **X column** like By default the X specication is turned into **p{<some value>}**. Such narrow columns often require a special format, this may be achieved using the >syntax of array.sty. So for example you may give a specication of **>{\small}X**

- \arraybackslash:** Another format which is useful in narrow columns is ragged right, however L^AT_EX's **\raggedright** macro redefines **** in a way which conflicts with its use in a tabular **\arraybackslash** or array environments. For this reason this package introduces the command **\arraybackslash**, this may be used after a **\raggedright**, **\raggedleft** or **\centering** declaration. Thus a tabularx preamble may specify **>\raggedright\arraybackslashX**.
- \newcolumntype:** These preamble specications may of course be saved using the command, **\newcolumntype**, defined in array.sty. Thus we may say **\newcolumntype{Y}{>{\small\raggedright\arraybackslash}X}** and then use **Y** in the **tabularx** preamble argument.
- \tabularxcolumn:** The X columns are set using the p column which corresponds to **\parbox[t]**. You may want them set using, say, the m column, which corresponds to **\parbox[c]**. It is not possible to change the column type using the >syntax, so another system is provided. **\tabularxcolumn** should be defined to be a macro with one argument, which expands to the tabular preamble specication that you want to correspond to X. The argument will be replaced by the calculated width of a column. The default is **\newcommand{\tabularxcolumn}[1]{p{#1}}**. So we may change this with a command such as: **\renewcommand{\tabularxcolumn}[1]{>{\small}m{#1}}**

An example:

This will make you learn the use of these commands:

```
\newcolumntype{R}{>{\raggedleft\arraybackslash}X}%
\begin{tabularx}{\textwidth}{|l|R|l|R| }
  \hline
  label 1 & label 2 & label 3 & label 4 \\
  \hline
  item 1  & item 2   & item 3   & item 4   \\
  \hline
\end{tabularx}
```

This will provide output like:

label 1	label 2	label 3	label 4
item 1	item 2	item 3	item 4

Note: Do not try to use `\multicolumn` with the X column.

4 tabulary package and table creation:

This package provides a new environment of creating tables. It is **tabulary**. It looks like:

```
\begin{tabulary}{<length>}{<preamble>}
\end{tabulary}
```

4.1 Features of tabulary:

- **LCRJ:** These new ‘uppercase’ column types are only activated in the tabulary environment. In order to make the total table width equal to length the LCRJ columns are converted to p columns (with `\raggedright`, `\centering`, or `\raggedleft` or normal justification respectively applied). The width of these converted columns is proportional to the natural width of the longest entry in each column.

- **\tymin setting:** To stop very narrow columns being too ‘squeezed’ by this process any columns that are narrower than **\tymin** are set to their natural width. This length may be set with **\setlength** and is arbitrarily initialised to 10 pt. (If you know that a column will be narrow, it may be preferable to use, say, c rather than C so that the tabulary mechanism is never invoked on that column.)
- **\tymax setting:** one very large entry can force its column to be too wide. So to prevent this, all columns with natural length greater than **\tymax** are set to the same width (with the proportion being taken as if the natural length was equal to **\tymax**). This is initially set to twice the text width..
- **\tyformat setting:** Narrow p columns are sometimes quite hard to set, and so you may redefine the command to be any declarations to make just after the **\centering** or **\ragged.** . . declaration. By default it redefines **\everypar** to insert a zero space at the start of every paragraph, so the first word may be hyphenated.
- **Use of \multicolumn command:** Here, in the tabulary environment, you can use **\multicolumn** command but if the multicolumn text turns out to be longer than the final calculated widths of the columns that it spans, then the final table will be too wide..
- **Use of \verb command:** \verb does not work here.

4.2 Example with tabulary:

Here, we are going to provide a basic example with tabulary. We will use the table stretching feature as well as some of the new column specifiers (like **L,C**):

```
\begin{center}
\begin{tabulary}{0.7\textwidth}{LCL}
  Short sentences      & \# & Long sentences
\hline
  This is short.      & 173 & This is much loooooooooonger,
because there are many more words.  \\
  This is not shorter. & 317 & This is still loooooooooonger,
because there are many more words.  \\
\end{tabulary}
\end{center}
```

It will produce output like:

Short sentences	#	Long sentences
This is short.	173	This is much loooooooooonger, because there are many more words.
This is not shorter.	317	This is still loooooooooonger, because there are many more words.

5 Tabu package and table creation:

The **tabu** package is the most powerful package to create tables. With it, you can do the basic things like using the basic column specifiers, giving the column separators, row separators as well as you can use and customize the X column, specify the width of the table, customize the column separator and row separators, use other environments inside tables, use math symbols and related environments inside tables, alternate the color of the table rows, create a long table across several pages etc. Now, we will discuss those features one by one.

5.1 The tabu environment:

The environment provided by the **tabu** package to typeset normal tables is **tabu**.

5.1.1 Basic examples with tabu using l,c,r,p{width},m{width},b{width}:

With the **tabu** environment, you can use all the specifiers which were used with the **tabular** environment. Like if you write the following code:

```
\begin{center}
\begin{tabu}{|l|c|r|}
\hline
item 1&item 2&item 3\\\hline
item 4&item 5&item 6\\\hline
\end{tabu}
\end{center}
```


It will produce output like:

item 1	item 2	item 3
item 4	item 5	item 6

You can also use the width specified columns like the columns denoted by **p{width}**, **m{width}** and **b{width}** . Here's an example:

```
\begin{center}
\begin{tabu}{|l|p{10cm}|}
\hline
Table specifications&Meaning\\\hline
l&left-justified column
\\\hline
c&centered column\\\hline
r&right-justified column\\\hline
p{'width'}&paragraph column with text vertically aligned at
the top\\\hline
m{'width'}&paragraph column with text vertically aligned in
the middle (requires array package)\\\hline
b{'width'}&paragraph column with text vertically aligned at
the bottom (requires array package)\\\hline
$\mid$&Vertical line\\\hline
$\parallel$&Double vertical line\\\hline
\end{tabu}
\end{center}
```

It will produce output like:

Table specifications	Meaning
l	left-justified column
c	centered column
r	right-justified column
p{'width'}	paragraph column with text vertically aligned at the top
m{'width'}	paragraph column with text vertically aligned in the middle (requires array package)
b{'width'}	paragraph column with text vertically aligned at the bottom (requires array package)
	Vertical line
	Double vertical line

5.1.2 Use and customization of X column:

Like the **tabularx** package, **tabu** package also provides you the facilities to use X column. But with the help of **tabu** you can do more things with **X** columns.

Specifying column width with X column: Width specification as well as width specification ratios can optionally be given to X columns:

Like: **X[2.5]X[1]** This means that the first X column will be two and a half wider than the second one or that the first X column width will be $5/7$ of the whole tabular width. Same type of tables(similarity in width ratio) can be created by mentioning **X[2.5]X** or **X[5]X[2]**.

Negative width coefficients can also be given to X columns: ex. **X[-2.5]X[1]** or **X[-2.5]X** or **X[-5]X[2]** In this case, the first X column will be at most two and a half wider than the second one, and if the natural width of the first X column is finally less than $2.5 \times$ (the width of the second column) then it will be narrowed down to this natural width.

Horizontal and vertical alignment specification with X column: Now, obviously you are wondering about the specifications like r,m,p,c etc

(**r**, **c**, **l**, or **j**, and **R**, **C**, **L** or **J**) are alignment specifiers and **p**,**m** or **b** are column specifiers.

Modifier	Command	Default
l, c, r, j, L, C, R, J	left, centered, right, justified	j
p, m, b	X column is converted into p, m or b column	p

L,C,R,J are originally column specifiers provided by **tabulary** package. But this can also be used with **tabu** environment.

Creating math columns with X column: This can be done by using two types of specification:

Modifier	Command	Default
\$	X[\$]is a shortcut for: $X<{\$}$	
\$\$	X[\$\$]is a shortcut for: $X<{\$}$ $X<{\$}$	

Note: `\displaystyle` switches to displaymath or equation environment type-setting (math mode).

Examples: Examples of creating tables with those modifications of tabu columns:

```
\begin{tabu}{|X[$]|X[$]|}\hline
\alpha & \beta \\\hline
\gamma & \delta + \epsilon + \zeta + \eta + \theta\\\hline
\end{tabu}
```

This will produce output like:

α	β
γ	$\delta + \epsilon + \zeta + \eta + \theta$

Additionally, you can provide other specifications with `X[$]`. Like if you want the text at the middle portion of a table, you could use c(centered column specification) with it:

```
\begin{tabu}{|X[$c]|X[$c]|}\hline
\alpha & \beta \\\hline
\gamma & \delta + \epsilon + \zeta + \eta + \theta\\\hline
\end{tabu}
```

It will produce output like:

α	β
γ	$\delta + \epsilon + \zeta + \eta + \theta$

You can also specify the width ration with it:

```
\begin{tabu}{|X[1$c]|X[2.5$c]|}\hline
\alpha & \beta \\\hline
\gamma & \delta + \epsilon + \zeta + \eta + \theta\\\hline
\end{tabu}
```

It will provide output like:

α	β
γ	$\delta + \epsilon + \zeta + \eta + \theta$

Similar type of things can also be done with **X[\$\$]** columns.

Embedding sunitx S columns inside X columns: A S column from siunitx can be embedded into a X column of **tabu**.

Examples: We are now going to provide the corresponding examples:

```
\newcolumntype Y{S[group-four-digits=true ,
round-mode=places,
round-precision=2,
round-integer-to-decimal=true ,
per-mode=symbol,
detect-all]}
\tabucolumn Y
\begin{tabu}{|Y|Y|c|}\hline
12.324 & 745.32 & & . . . \\\
21.13 & 0 & & . . . \\\
213.3245 & 12.342 & & . . . \\\
2143.12 & 324.325 & & . . . \\\hline
\end{tabu}
```

This will provide output like:

12.32	745.32	. . .
21.13	0.00	. . .
213.32	12.34	. . .
2 143.12	324.33	. . .

Note: For using this, you have to load **siunitx** package. You have to include table inputs according to the specifications.

Now, notice that there are two Y columns and their lengths are different. Because, siunitx S columns are not embedded with X columns here. For doing this X column must be centered.

The code structure for embedding S columns within X columns is like the following:

```
X[c]{S[S column specifications]}
```

Additional things can be done with X columns:

- You can use **multicolumn** with X columns with the help of this package. But there are some rules of using it.

The process of `\multicolumn` implies the TeX primitive `\omit` which discards the tabular preamble for the spanned columns. Discarding the preamble means discarding the information about the widths of the columns. This explains why the following example does not work properly:

```
\begin{tabu}{|X|X|X[2]|} \hline
\multicolumn2{|c|}{Hello} & World \\\hline
\end{tabu}
```

The correct result can be obtained by the mean of a phantom line, that will remain invisible unless your preamble contains special @ or ! columns that prints some text:

```
\begin{tabu}{|X|X|X[2]|} \hline
\multicolumn2{|c|}{Hello} & World \\\hline
\tabuphantomline
\end{tabu}
```

The first code will produce error like : The dimension is too large... and the second code will produce output like:

Hello	World
-------	-------

- You can use X columns with **{tabu} spread <dimen>** (it will be discussed later).
- tabu X columns can contain any type of **tabular**, **tabular***, **tabularx** or **tabu** without special care about the syntax. tabu can also be put inside **tabular**, **tabular*** and **tabularx**. As long as tabu with X columns has a default target, nesting tabu with X columns is easy. Furthermore, the default global alignment of a nested tabu is t (for top) while the default global alignment of a tabu in a paragraph is c (for centered).
- The “algorithm” (or the arithmetic) to get the target width for tabu X columns is the same as the one used by tabularx. **\hfuzz** is the “tolerance” for the whole tabular width. We use ε -**T_EX** **\dimexpr** instead of T_EX primitives (with round/truncate bias correction).
- Convergence to the target width is optimised: the **\halign** preamble is not re-built at each trial, but only expanded again, until the target is reached. Though optimized, the process is the same as the one implemented for tabularx and in particular the content of the tabu environment is collected as soon as a tabu X column is found in the preamble. This implies restrictions on catcode modifications and verbatim text inside a tabu with X columns.
- If the width of the whole tabular is not specified with “tabu to” it is considered to be **\linewidth**. The linegoal package option makes the default width equal to **\linegoal**.

Compilation must then be done with pdfT_EX either in pdf or dvi mode, and package linegoal is loaded. **\linegoal** requires pdfT_EX for its **\pdfsavepos** primitive and the zref-savepos: if the tabu is not alone in its paragraph ie.if the target is not **\linewidth**, then two compilations (or more) are required to get the correct target.

Default target for nested tabu environments is always **\linewidth**, which equals to the column width inside **p**, **m**, **b** and **X** columns.

- As long as the **\halign** content is expanded more than once, protections against counters incrementation, whatsits (write) index entries, footnotes etc.. are set up: the mechanism of tabularx is reimplemented and enhanced for tabu X columns. **\tabuDisableCommands** can be used to neutralize the expansion of additional macros during the trials.

5.1.3 Width and stretching of tables with tabu environment:

You can additionally specify the width of the table, by using the following two specification:

`\begin{tabu}to <dimen>` It is like tabular but the inter-columns space is given a stretchability of 1fil, in other words `@{\extracolsep {0pt plus 1fil}}` is inserted by default at the beginning of the tabular preamble, unless another value for `\extracolsep` is specified. Therefore “tabu to” fills in width the specified `<dimen>`.

`\begin{tabu} spread<dimen>` does a tabular whose width is `<dimen>` wider than its natural width. `@{\extracolsep {0pt plus 1fil}}` is inserted by default if `<dimen>>0`

X columns with “tabu spread”: The tabu spread command is like:

```
\begin {tabu} spread <dimen> [pos] {tabular preamble}
```

does a tabular whose width is `<dimen>` wider than its natural width. `@{\extracolsep {0pt plus 1fil}}` is inserted by default if `<dimen>>0`.

tabu X columns can be used with “tabu spread” to adjust the column widths of tabulars that contain only small pieces of text. The question is: how to make a tabular the width of the line, with 6 columns; the columns 1, 2, 5 and 6 are of equal widths and the widths of columns 3 and 4 are only one half. As possible solution:

```
\begin{tabu} to\linewidth{|X[2]|X[2]|X|X|X[2]|X[2]|} \hline
1 & 2 & 3 & 4 & 5 & 6 \\\hline
\end{tabu}
```

Now, if you see the output:

1	2	3	4	5	6
---	---	---	---	---	---

Notice that if you are getting a different output by using the same code, use `\tabureset` command before writing the code. This is clear all your previous declarations which changes the standard parameters of `tabu`.

But the text in each cell is very short: one single character, and you prefer the table to be tight, but don't know the exact width of the whole (then just set the dimension of the spread as 0pt):

```
\begin{tabu} spread 0pt{|X[2]|X[2]|X|X|X[2]|X[2]|} \hline
1 & 2 & 3 & 4 & 5 & 6 \\\hline
\end{tabu}
```

It will produce output like:

1	2	3	4	5	6
---	---	---	---	---	---

But now it's definitely too narrow, then give it some more space:

```
\begin{tabu} spread 2in{|X[2]|X[2]|X|X|X[2]|X[2]|} \hline
1 & 2 & 3 & 4 & 5 & 6 \\\hline
\end{tabu}
```

It will produce output like:

1	2	3	4	5	6
---	---	---	---	---	---

In the preamble, if you use `@{}` like the following declaration of preamble:

```
\begin{tabu} spread 3cm{@{ }X[9]X[4]|X|}
```

The margin will be removed. (this is the use of `@{ }` command)

5.1.4 Mastering vertical space with `tabu` environment:

Use of `\tabulinesep` command:


```

\tabulinesep=<dimen>
\tabulinesep=^<dimen>
\tabulinesep=_<dimen>
\tabulinesep=^<dimen>_<dimen>
\tabulinesep=_<dimen>^<dimen>

```

\tabulinesep sets the minimal vertical space allowed between the cell content and the cell border. The macro may be prefixed by **\global** (even inside a **\noalign** group).

It is possible to set the “top limit” (a T_EX dimension called **\abovetabulinesep**) and the “bottom limit” independently with the syntaxes:

```

\tabulinesep =^<dimen> sets \abovetabulinesep
\tabulinesep =_<dimen> sets \belowtabulinesep
\tabulinesep =_<dimen>^<dimen> sets \belowtabulinesep and \abovetabulinesep

```

These parameters can be used in text and math modes to give more vertical space between lines, especially when using math formulae. **\tabulinesep** is a soft parameter, and leads to rows which do not share the same height.

Now, we are going to discuss about **\extrarowsep** command:

Use of \extrarowsep command:

```

\extrarowsep=<dimen>
\extrarowsep=^<dimen>
\extrarowsep=_<dimen>
\extrarowsep=^<dimen>_<dimen>
\extrarowsep=_<dimen>^<dimen>

```

\extrarowsep is an extra vertical space which is added to each row, unconditionally. **array.sty** provides the T_EX dimension **\extrarowheight** and **tabu** provides **\extrarowdepth** in addition.

As a result, the rows can share the same height/depth but the spacing is not dynamic. **\tabulinesep** can be used even with positive values for **\extrarowsep**, for **tabu** inserts only one strut per row and vertical spacing computations are

possible in all cases.

The macro can be prefixed by `\global` as well, even inside a `\noalign` group.

Set `\extrarowheight` and `\extrarowdepth` to different values, with the syntaxes:

```
\extrarowsep = ^<dimen> sets \extrarowheight
                        \extrarowdepth is unchanged
\extrarowsep = _<dimen> sets \extrarowdepth
                        \extrarowheight is unchanged
\extrarowsep = _<dimen> ^<dimen> sets \extrarowdepth and
\extrarowheight.
```

5.1.5 Using other environments within tabu:

Though the content of the `tabu` environment is collected for measuring purpose, it is possible to insert some environment specially verbatim material with the `tabu` variant of the environment. The content is then carefully collected and re-scanned (with `\scantokens`). During the process, the `@ letter` is read with the category code it has been given at the entry inside the environment (it is possible to say `\makeatletter` before `\begin{tabu*}`).

Example: It is possible to insert Verbatim material with some `\csname` control sequences `\endcsname` inside a `tabu` and inside `X` columns. Negative coefficients work well too, adjusting the width of the `X` column to the natural width if it is finally less than the width computed with the absolute value of the coefficient.

A complete Verbatim environment is also admissible.

But you must use the star form of the environment: `tabu` which uses `\scantokens`. Verbatim environments must be put alone on their lines (in the input file) for nothing is allowed after `\begin{Verbatim}` or `\end{Verbatim}`.

Another point to know is that `\begin` and `\end` control sequences should match otherwise, you must enclose the Verbatim environment inside braces.

This is related to the fact that `tabu` collects its body, and looks for matching pairs of `\begin ... \end !`

tabu is useless when nested inside another tabular. The star form of the environment should be used only for the outermost table ! Comments are removed, unless the % character is given a category code of 12 (or 11) before the entry inside the environment.

An example of inserting verbatim environment inside tabu environment:

```
\makeatletter \@makeother\%
\begin{tabu*} spread 0pt {|X[-1]X|} \tabucline -
This is a small \Verb+\Verbatim+\par
insertion
&
\begin{Verbatim}[listparameters={\topsep=-\ht\strutbox}]
& this is a complete % with some comments
Verbatim environment % every now and then
\end{Verbatim}
\\ \hline
\end{tabu*}
```

It will produce output like:



This is a small \Verbatim	& this is a complete % with some comments
insertion	Verbatim environment % every now and then

5.1.6 Lines leaders and colors inside tabu:

Vertical lines: | **has an optional parameter:** Inside tabu environment, the vertical line marker | has an optional argument which is the width of the vertical rule. The default width remains `\arrayrulewidth` of course. The optional argument for | can also contain the name of a color. color names are only possible, not a color specification by the mean of a color model. The width of the line if specified, must come before the color name and... as for X columns parameters, commas are optional.


Examples:

```
\begin{tabu}{|[5pt]|c|c|[5pt]|}
Hello & World
\end{tabu}
```

This will produce output like:  Hello | World 

Another example:

```
\begin {tabu}{|[|[5pt red]|c|c|[5pt Indigo]|}  
Hello & World  
\end {tabu}
```

It will produce output like:  Hello | World 

Multiple `\firstline` and `\lastline`:

```
\firstline [extratabsurround] make multiple lines !  
\firstline [extratabsurround]\hline  
\lastline [extratabsurround]  
\lastline [extratabsurround]\hline
```

`\firstline` and `\lastline` are defined in `array.sty` (i.e in **array** package) and can be used to preserve the alignment of text, when using horizontal lines. Besides, the optional argument can be used to change (locally) the `\extratabsurround` dimension.

With `tabu` you can make double, triple (or more) `\firstline` or `\lastline` as in:

- First, with top alignment (in **tabu** environment we can specify its alignment as b, t or m as optional parameter):

```
Tables  
\begin {tabu}[t]{c}  
with no\\ line \\ commands \\ used  
\end {tabu}
```

versus

```

Tables
\begin {tabu}[t]{|c|}
\firsthline \hline \hline \hline
with some \\ line \\ commands \\
\lasthline \hline \hline \hline
\end {tabu} used.

```

The output of the first code is:

with no line commands used	Versus	<table border="1"> <tr><td>with some line commands</td></tr> </table> used.	with some line commands
with some line commands			

- The same example with bottom alignment:

```

Tables
\begin {tabu}[b]{c}
with no\\ line \\ commands \\ used
\end {tabu}

```

versus

```

Tables
\begin {tabu}[b]{|c|}
\firsthline \hline \hline \hline
with some \\ line \\ commands \\
\lasthline \hline \hline \hline
\end {tabu} used.

```

Here, you can see the difference between two codes:

with no line commands used	versus	<table border="1"> <tr><td>with some line commands</td></tr> </table> used.	with some line commands
with some line commands			

But if you will use the optional parameter `\extratabsurround` the command `\firsthline[\extratabsurround]\firsthline\firsthline\firsthline`

can be used as `\firstline\hline\hline\hline`. Same thing can also be done for `\lastline`.

i.e if you the following code instead of the given code where we used multiple `\hline`:

```
Tables
\begin {tabu}[b]{|c|}
\firstline[\extratabsurround] \firstline \firstline
\firstline
with some \line \ commands \line
\lastline[\extratabsurround] \lastline \lastline
\lastline
\end {tabu} used.
```

It will produce output like:

Tables	<div> <div>with some line commands</div> </div>	used.
--------	---	-------

More line for styles:

`\taburulecolor` command:

```
\taburulecolor {<rule color>}
\taburulecolor |<double rule sep color>|{<rule color>}
```

`\taburulecolor` sets (in a “locally-global” way) the color to be used for `\hline`, `\firstline`, `\lastline` and also vertical lines if the standard line style is used (the standard line style is active after `\tabulinestyle {}` or after `\tabureset`).

The optional parameter enclosed by vertical bars: `|<double rule sep color>|` is the color to set between two adjacents rules. If not specified, double (or triple...) rules are separated by a vertical space (`\vskip`).

An example: Here, we are going to provide an example of that:

```

\taburulecolor|lime|{DarkSlateBlue}
\arrayrulewidth=1mm \doublerulesep=2mm
Here is
\begin{tabu} spread 0pt{X[- 1]}
\firsthline \hline
a tabu \\
environment \\
made with \\ \lasthline[5mm] \hline \hline
\end{tabu} Tabu package ! \ par
And the next paragraph follows . . .

```

It will provide output like:

```

      _____
      |a tabu|
      |_____|
      |environment|
Here is |_____| Tabu package !
      |made with |
      |_____|
      |_____|
And the next paragraph follows . . .

```

Note: `\firsthline`, `\lasthline` and `\hline` all have optional parameters to specify the length. Like: `\firsthline[5mm]`, `\lasthline[5mm]` and `\hline[5mm]`

tabulinestyle command:

```

\tabulinestyle {<line style specification>}

```

Some line specification forms of `\tabulinestyle` command:

- 3pt rule color on 4pt dash color off 5pt gap color
- rule color on 4pt dash color off 5pt gap color
- on 4pt dash color off 6pt gap color
- 3pt rule color
- on 4pt dash color

- off 5pt
- 3pt

Try these given forms to specify **tabulinestyle** command and check the output.

An example:

Here, we are going to provide an example:

```
\tabulinestyle{on 4pt blue off 6pt green}
\begin{tabu}{|X|X|}\tabucline-
First example&This is an example of tabulinestyle\\\tabucline-
\end{tabu}
```

It will produce output like:

First example	This is an example of tabulinestyle
---------------	-------------------------------------

Note: `\tabulinestyle` modifies the whole table outline. But it does not modify the horizontal lines(row separator) provided by `\hline`, `\firsthline` and `\lasthline`.

Some precautions for using specification with `\tabulinestyle` command:

Your color names can contain spaces but:

- If the first character in the line specification is not a letter, then it is taken as a dimension: the thickness of the line. Otherwise, the default thickness is used ie. `\arrayrulewidth`.
- Your color names must not contain any series of characters that match one the patterns:

on?

off?

newtabulinestyle command:

- `\newtabulinestyle {<style=line spec., style=line spec., ...>}` *babel*

This command defines a line style to be used in the first optional argument of `\tabucline` (horizontal lines) or the optional argument of `|` (vertical lines) or with `\tabulinestyle` (locally-global style). Style names and color names are babel-protected.

tabucline command:

<code>\tabucline [style or spec.]{start-end}</code>

- `\tabucline` is an attempt to give a versatile command to make horizontal lines.
- `\tabucline` is pretty good with vertical lines even if the thickness of the line grows up,
- `\tabucline` takes care of `\extrarowheight`
- `\tabucline` can make horizontal dashed lines, with a pgf/TikZ syntax:
`\tabucline [<width>on <dash>off<gap>]<first column>-<last column>`
- alternatively, you can give `\tabucline` a `\hbox` to make a leader with it:
The `<spec.>` must then begin with `\hbox`, `\box` or `\copy`
- finally you can give `\tabucline` a color name, after the line specification.

We have illustrated the example of using `\tabucline` before. Now, we are going to illustrate one more example of it, which is little complicated than the example we illustrated before.

*****	*****
Define the newline style	Use the line style
<code>\newtabulinestyle {myline=0.4pt on 2.5pt off 1pt red}</code>	<code>\tabucline [myline]{-}</code>
And this one is done by defining leader or a box to make a leader with it directly in the argument of <code>\tabucline</code>	
<code>\tabucline [\hbox {\$\scriptstyle \star \$}]{1-2}</code>	
*****	*****

This one is done by:

```

\begin{tabu}{|p{4cm}|p{7cm}|}
\tabucline [\hbox {$\scriptstyle \star$}]{1-2}
Define the newline style&Use the line style\\\tabucline-
{\bfseries{\textbackslash}newtabulinesyle
\{myline=0.4pt on 2.5pt off 1pt red\}}
& {\bfseries{\textbackslash}tabucline [myline]\{-\}}
\newtabulinesyle {myline=0.4pt on 2.5pt off 1pt red}
\\\tabucline [myline]\{-}
\multicolumn{2}{|p{11cm}|}{And this one is done by defining
leader or a box to make a leader with it directly in the argument
of {\bfseries{\textbackslash}tabucline \textbackslash}tabucline
[\textbackslash\hbox {\{$\scriptstyle \star$\}}{1-2}}}\\\\tabucline
[\hbox {$\scriptstyle \star$}]{1-2}
\end{tabu}

```

You can create more cool table borders by exploring cool L^AT_EX symbols.

Automatic horizontal lines:

everyrow command:

```
\everyrow {code}
```

- `\everyrow` can be used to insert horizontal lines automatically (with the specifications you mention in the code section.) Remember the code specification is not an optional parameter to `\everyrow` command.
- `\everyrow` can be used in `longtabu` as well.

Alternate row colors in tables:

taburowcolors command:

```
\taburowcolors [first line]<number>{first .. last}
```

\taburowcolors sets the alternate colors to be used on every row of the tabular. The command can be used before a tabu environment or inside it, at the end of a row.

The optional parameter [**first line**] tells the first row from which background colors are starting this optional parameter has no effect when **\taburowcolors** is used at the end of a row: background are starting immediately in this case.

<number> is the number of colors in the color series. If not specified, it defaults to 2 (for alternate rows color).

Finally **<first>** and **<last>** are the first and the last colors in the colorseries.

Examples:

If you use the following code:

```
\taburowcolors [1]{white!30..Aquamarine}  
\taburulecolor|white|{Blue}  
\arrayrulewidth=1pt \doublerulesep =1.5 pt  
\everyrow{\hline \hline }  
\begin{tabu} {X[ - 1 ]X}  
This is&just a test \\  
and I think &it will \\  
look&good \\  
\end{tabu}
```

It will produce output like:

This is	just a test
and I think	it will
look	good

Here, we did not mention any number. so, by default its 2. So, row colors will be alternated in odd and even rows.

You can use as many color shades as you want using the optional parameter "number":

```
\taburowcolors 5{green!25..yellow!50}
\begin{tabu}{X[-1]X}
Test&row number=1\\
Test&row number=2\\
Test&row number=3\\
Test&row number=4\\
Test&row number=5\\
\end{tabu}
```

This will produce output like:

Test	row number=1
Test	row number=2
Test	row number=3
Test	row number=4
Test	row number=5

Going back to the standard style:

tabureset command: To go back to "standard" parameters, tabu provides the command `\tabureset` which basically does:

```
\tabulinesep = Opt  \extrarowsep = Opt  \extratabsurround = Opt
\tabulinestyle {}   \everyrow {}         \taburulecolor ||{}
                    and
                    \taburowcolors {}
```

5.1.7 Modifying the font and the alignment in one row:

rowfont command:

```
\rowfont [alignment]{font specification}
```

Inside a `tabu` environment, you can modify the font for each cell in a row. `\rowfont` has priority over column font specification, exactly like `\rowcolor` (package `colortbl`) has priority over `\columncolor`.

The alignment of each cell in one row can also be changed to:

For normal settings		For ragged2e settings	
Alignment	Meaning	Alignment	Meaning
l	left	L	left
c	center	C	center
r	right	R	Right
j	justify	J	justify

5.1.8 Saving and restoring a `tabu`:

Savetabu command:

```
\savetabu {<user-name>}
```

The command `\savetabu` can be used at the end of any line of a `tabu` environment to save the parameters of a `tabu` environment. The saving is always global. This allows to easily make tabulars which share exactly the same shape throughout your document. This can also be used as a kind of tabbing environment which is able to remember the tabs positions...

If the `<user-name>` has been used before, an info is displayed in the `.log` file and the previous settings are overwritten.

With the `\tracingtabu >0`, informations about the saved parameters are reported in the `.log` file.

Recalling saved parameters are done with `\usetabu` (complete recovery) or `\preamble` (partial recovery of the preamble only).

usetabu command:

```
\usetabu {<user-name>}
```

`\usetabu` is the complement of `\savetabu`: it can be put alone in the tabu preamble instead of the usual columns specifications to restore any previous settings saved with `\savetabu`

The `<user-name>` must exist otherwise, you get an error.

The `<user-name>` must exist otherwise, you get an error.

`\usetabu` is a help to make several tabulars of exactly the same shape, same target, same preamble. The only parameter that can be changed is the optional vertical position parameter for the whole tabular.

`\usetabu` does not work with `longtabu` .

`\usetabu` locally restores:

- the preamble. ¹
- the vertical position `[c]`, `[b]` or `[t]`, unless another position is specified.
- the target width of the tabu in points: the saved target width does not contain any control sequence: it is fixed and stored in points.
- the width of tabu X columns: those widths are not calculated any more – even in the case of negative coefficients – and X columns are directly transformed into **p**, **m** or **b** columns of the same widths as the ones that where calculated at the time of `\savetabu`
- `\tabcolsep` (or `\arraycolsep` in math mode) `\extrarowheight`, `\extrarowdepth`, `\arraystretch` and `\extratabsurround`
- `\arrayrulewidth`, `\doublerulesep` and the parameters for `\everyrow` `\taburulecolor`, `\tabulinestyle`, and `\taburowcolors`
- `\minrowclearance` (package `colortbl`)

`\abovetabulinesep` and `\belowtabulinesep` are not restored, because they are related to the content of the tabular rather than to its shape.

¹The complete `\halign` preamble is restored.

Examples:

```
\tabcolsep=12pt \extrarowsep=1mm
\tabulinestyle{on 1pt ForestGreen }
\begin{tabu} to .7 \linewidth {|XXX|X[c]|}\savetabu{mytabu}
\tabucline -
This & i s & tabu & package \\\tabucline -
\end{tabu}
```

This will produce output like:

This	i s	tabu	package
------	-----	------	---------

Now, notice the code carefully. We have saved this style (we can only save the linestyles, row colors etc...) for further use with the name "mytabu". So that, even after resetting the tabu environment to its standard parameters we could use this style as many times as we want.

Now, you can use the following code to check it:

```
\tabureset
\begin{tabu}{\usetabu{mytabu}} \tabucline-
\multicolumn 3{| c |}{This is tabu} & package \\\tabucline -
\tabuphantomline
\end{tabu}
```

It will produce output like:

This is tabu	package
--------------	---------

\tabcolsep, rule colors etc. are not restored from **\savetabu**: the only tabu preamble is restored.

preamble command:

```
\preamble {<user-name>}
```

`\preamble` can also be used after `\savetabu`. This is a variant of `\usetabu` that locally restores:

- the `tabu` (or `longtabu`) preamble.
- the vertical position `[c]`, `[b]` or `[t]` (or `[c]`, `[l]` or `[r]` for `longtabu`), unless another position is specified.
- the `tabu` / `longtabu` target width, unless another target is specified

Any other tabular parameter is not restored.

Put `\preamble {<user-name>}` alone inside the `tabu` (or `longtabu`) preamble in place of the usual columns specifications.

`\preamble` works exactly as if you defined a custom environment for `tabu`.

`\preamble` works with `longtabu` .

5.1.9 Some other features:

Printing numbers inside `tabu` with `numprint` and `siunitx`:

tabudecimal command: `Tabu` provides a facility to print numbers inside columns. This facility is not implemented to replace `siunitx` `S` and `s` columns or `numprint` `n` and `N` columns or other packages that provide alignment such as `warpcol`, `dcolumn` or `rccol`. It just make easy to apply a macro you get already on each number in a column of a `tabu`.

`\tabudecimal` has been developped mainly because it makes possible to align numbers inside `tabu` `X` columns.

<code>\tabudecimal {<user-macro>}</code>
--

`\tabudecimal` can be used in the preamble of a `tabu` before a column specification. The `<user-macro>` is a macro with one parameter that has to be defined before.

Example with `\numprint`:


```

\def\usermacro#1{\numprint[\officialeuro]{\zap@space #1 \@empty}}
\nprounddigits{2} \npprintnull \npthousandsep{\,}
\npunitseparator{~}
\tabulinestyle{1 pt GreenYellow}
\begin{tabu}{|X|X|}\tabucline-
\rowfont{\bfseries} January & February
\\ \tabucline[ 1 pt on2pt GreenYellow]-
12.324 & 745.32 \\
21.13 & 0 \\
213.3245 & 12.342 \\
2143.12 & 324.325 \\ \tabucline-
\end{tabu}

```

It will produce output like:

January	February
12.324	745.32
21.13	0
213.3245	12.342
2143.12	324.325

Note: For using this code, you have to use **numprint** package.

Paragraph indentation: `tabu` takes care of paragraph indentation when it is used with X columns and its default target, no matter if it has been loaded or not with the `linegoal` option.

delarrayshortcuts: When you enclose your tabular with math delimiters using `delarray` shortcuts, `@b` tries to reach its target for the whole: the tabular and the delimiter(s).

5.2 longtabu environment:

```

\begin{longtabu}[l | c | r]{tabular preamble}
\begin{longtabu} to <dimen> [l | c | r]{tabular preamble}
\begin{longtabu} spread <dimen> [l | c | r]{tabular preamble}

```

Here, we mention three preamble structure. First one is normal. Second and third options support stretching of the table width (Like `tabu` to `<dimen>` and `tabu` spread `<dimen>`)

`longtabu` is just like `tabu` but page breaks are allowed between rows of the table. `longtabu` is based on the `longtable` package which must be loaded, and all features of the `longtable` environment works inside `longtabu`: `\endhead`, `\endfirsthead`, `\endfoot`, `\endlastfoot` and `\caption`.

`longtabu` enhances the `longtable` environment with the possibility to use `X` columns and line specifications for horizontal and vertical rules. `longtabu` is thus much easier than `ltxtable`.

A sample code structure is given as example:

```
\begin{longtabu} to \linewidth {1X[2]1X1}

\rowfont\bfseries H1 & H2 & H3 & H4 & H5 \\ \hline
\endhead

\\ \hline
\multicolumn{8}{r}{There is more to come} \\
\endfoot

\\ \hline
\endlastfoot

% Content ...
```

`longtabu` almost provide all the features except the following commands and features:

<code>\tabucline</code>	It is yet to be implemented with <code>longtabu</code> . <code>\tabucline</code> does not care about page breaks presently. Use <code>\hline</code> instead.
<code>\usetabu</code>	It is not available with <code>longtabu</code> . But <code>\savetabu</code> and <code>\preamble</code> work.
mathematical mode	<code>longtabu</code> is not designed to work in math mode
delarray shortcut	a delimiter cannot be spanned over pages...
<code>\tabuphantomline</code>	Useless inside <code>longtabu</code>

Since, we are now concentrating in creating tables which can support pagebreaks we will discuss about table creation with **longtable** and **supertabular** package.

6 longtable package and table creation:

The longtable package defines a new environment, longtable, which has most of the features of the tabular environment, but produces tables which may be broken by TEX's standard page-breaking algorithm. It also shares some features with the table environment. In particular it uses the same counter, table, and has a similar **\caption** command. Also, the standard **\listoftables** command lists tables produced by either the table or longtable environments.

6.1 Useful commands and feature:

- **\endhead** : At the start of the table one may specify lines which are to appear at the top of every page (under the headline, but before the other lines of the table). The lines are entered as normal, but the last **** command is replaced by a **\endhead** command.
- **\endfirsthead** : If the first page should have a different heading, then this should be entered in the same way, and terminated with the **\endfirsthead** command. The LTchunksize should be at least as large as the number of rows in the heading.
- **\endfoot & \endlastfoot** : These commands are used in the same way (at the start of the table) to specify rows (or an **\hline**) to appear at the bottom of each page. In certain situations, you may want to place lines which logically belong in the table body at the end of the firsthead, or the beginning of the lastfoot. This helps to control which lines appear on the first and last page of the table.
- **\caption** : The **\caption{...}** is essentially equivalent to

```
\multicolumn{n}{c}{\parbox{LTcapwidth}{...}}
```

where n is the number of columns of the table. You may set the width of the caption with a command such as **\setlength{LTcapwidth}{2in}** in the preamble of your document. The default is 4in. **\caption** also writes the

information to produce an entry in the list of tables. As with the `\caption` command in the `gure` and `table` environments, an optional argument species the text to appear in the list of tables if this is different from the text to appear in the caption. Thus the caption for table 1 was specified as `\caption[An optional table caption (used in the list of tables)]{A long table\label{long}}`.

You may wish the caption on later pages to be different to that on the first page. In this case put the `\caption` command in the first heading, and put a subsidiary caption in a `\caption[]` command in the main heading. If the optional argument to `\caption` is empty, no entry is made in the list of tables. Alternatively, if you do not want the table number to be printed each time, use the `\caption*` command.

The captions are set based on the code for the article class. If you have redefined the standard `\@makecaption` command to produce a different format for the captions, you may need to make similar changes to the `longtable` version, `\LT@makecaption`.

A more convenient method of customising captions is given by the `caption(2)` package, which provides commands for customising captions, and arranges that the captions in standard environments, and many environments provided by packages (including `longtable`) are modified in a compatible manner.

You may use the `\label` command so that you can cross reference longtables with `\ref`. Note however, that the `\label` command should not be used in a heading that may appear more than once. Place it either in the `rsthead`, or in the body of the table. It should not be the `rst` command in any entry.

Now, we already have mentioned `LTchunksize`. Now, we are going to discuss about it.

6.2 Chunk size:

- **LTchunksize** : In order to TEX multi-page tables, it is necessary to break up the table into smaller chunks, so that TEX does not have to keep everything in memory at one time. By default `longtable` uses 20 rows per chunk, but this can be set by the user, with e.g., `\setcounter{LTchunksize}{10}`. These chunks do not affect page breaking, thus if you are using a TEX with

a lot of memory, you can set `LTchunksize` to be several pages of the table. TEX will run faster with a large `LTchunksize`. However, if necessary, `longtable` can work with `LTchunksize` set to 1, in which case the memory taken up is negligible. Note that if you use the commands for setting the table head or foot, the `LTchunksize` must be at least as large as the number of rows in each of the head or foot sections.

6.3 Multicolumn entries:

The `\multicolumn` command may be used in `longtable` in exactly the same way as for `tabular`. So you may want to skip this section, which is rather technical, however coping with `\multicolumn` is one of the main problems for an environment such as `longtable`. The main effect that a user will see is that certain combinations of `\multicolumn` entries will result in a document needing more runs of LATEX before the various ‘chunks’ of a table align.

For more information, check the documentation for **longtable** package. Now, we are going to discuss about **supertabular** package.

7 supertabular package and table creation:

The package `supertabular` offers a new environment, the **supertabular** environment. As the name indicates it is an extension of the normal `tabular` environment. With the original `tabular` environment a `tabular` must always fit on one page. If the `tabular` becomes too large the text overwrites the page’s bottom margin and you get an `Overfull vbox` message.

The `supertabular` environment uses the `tabular` environment internally, but it evaluates the used space every time it gets a `\\` command. If the `tabular` reaches the `textheight`, it automatically inserts an optional `tabletail`, an `\end{tabular}` command, starts a new page, a new **tabular** environment and inserts the optional `tablehead` on the new page continuing the `tabular`.

7.1 Useful features, commands & environments:

<code>\tablefirsthead</code>	The command <code>\tablefirsthead</code> takes one argument, it defines the contents of the first occurrence of the tabular head. The use of this command is optional. Don't forget to close the head by a <code>\\</code> .
<code>\tablehead</code>	The command <code>\tablehead</code> takes one argument, it defines the contents of all subsequent occurrences of the tabular head. Don't forget to close the head by a <code>\\</code> .
<code>\tabletail</code>	The command <code>\tabletail</code> takes one argument, it defines something which should be inserted before each <code>\end{tabular}</code> , except the last.
<code>\tablelasttail</code>	The command <code>\tablelasttail</code> takes one argument, it defines something which should be inserted before the last <code>\end{tabular}</code> . The use of this command is optional.
<code>\topcaption</code> <code>\bottomcaption</code> <code>\tablecaption</code>	These commands all take the same arguments as L ^A T _E X's standard <code>\caption</code> command. They provide a caption for the super-table, either at the top or at the bottom of the table. When <code>\tablecaption</code> is used the caption will be placed at the default location, which is at the top.
<code>supertabular</code> <code>supertabular*</code>	The environments <code>supertabular</code> and <code>supertabular*</code> can be used much like the standard L ^A T _E X environments <code>tabular</code> and <code>tabular*</code> .
<code>mpsupertabular</code> <code>mpsupertabular*</code>	The environments <code>mpsupertabular</code> and <code>mpsupertabular*</code> work like the <code>supertabular</code> and <code>supertabular*</code> environments but put each page into a minipage first. Thus it is possible to have footnotes inside a <code>mpsupertabular</code> . The <code>footnotetext</code> is printed at the end of each page.

\shrinkheight The allowed maximum height of a part of the supertabular on a page can be adjusted using the command **\shrinkheight**. It takes one argument, the length with which to shrink (positive value) or grow (negative value) the allowed height.

7.2 Weakpoints of supertabular:

- When the material of a normal entry (not a p-arg) becomes larger than the estimated **\ST@lineht**, overfull **\vboxes** will be produced at all.
- When the last p-arg on a page gets more than 4 lines (probably even more than 3 lines) it will result in an overfull **\vbox**. Also some combinations of **\baselinestretch** **\arraystretch** and a large font may lead to one line too much.
- if accidentally the last line of the tabular produces a newpage, on the next page the tabletail will be written immediately after the tablehead. Depending on the contents this may result in an error message regarding misplaced **\noalign**. A quick but not very elegant solution: shrink the allowed height of the table with the command **\shrinkheight{...pt}** after the first **** of the supertabular.
- The mpsupertabular environment sometimes has problems with pagesbreaks when footnotes appear in the lower part of the tabular.

7.3 Example:

```
\begin{center}
\tablefirsthead{%
\hline
\multicolumn{1}{|c|}{\hspace*{0.1cm} Number} &
\multicolumn{1}{c}{Number$^2$} &
Number$^4$ &
\multicolumn{1}{c|}{Number!} \\\
\hline}
\tablehead{%
\hline
```

continued to next page

```

\multicolumn{4}{|l|}{\small\sl continued from previous page}\\
\hline
\multicolumn{1}{|c|}{\hspace*{0.1cm} Number} &
\multicolumn{1}{c}{Number$^2$} &
Number$^4$ &
\multicolumn{1}{c|}{Number!} \\
\hline
\tabletail{%
\hline
\multicolumn{4}{|r|}{\small\sl continued on next page}\\
\hline}
\tablelasttail{\hline}
\bottomcaption{This table is split across pages}
\begin{supertabular}{|r@{\hspace{6.5mm}}|r@{\hspace{5.5mm}}|r|r|}
1 & 1 & 1 & 1 \\
2 & 4 & 16 & 2 \\
3 & 9 & 81 & 6 \\
4 & 16 & 256 & 24 \\
5 & 25 & 625 & 120 \\
6 & 36 & 1296 & 720 \\
7 & 49 & 2401 & 5040 \\
8 & 64 & 4096 & 40320 \\
9 & 81 & 6561 & 362880 \\
10 & 100 & 10000 & 3628800 \\
11 & 121 & 14641 & 39916800 \\
12 & 144 & 20736 & 479001600 \\
13 & 169 & 28561 & 6.22702080E+9 \\
14 & 196 & 38416 & 8.71782912E+10 \\
15 & 225 & 50625 & 1.30767437E+12 \\
16 & 256 & 65536 & 2.09227899E+13 \\
17 & 289 & 83521 & 3.55687428E+14 \\
18 & 324 & 104976 & 6.40237370E+15 \\
19 & 361 & 130321 & 1.21645100E+17 \\
20 & 400 & 160000 & 2.43290200E+18 \\
\end{supertabular}
\end{center}

```

Copy and paste this code at the middle of a page so the table can spill across pages and check the output.

Now, we will discuss some packages which will help you to create table in a professional manner.

8 ctable package and table creation:

The ctable package lets you easily typeset captioned table and figure floats with optional footnotes.

Both caption and footnotes will normally be forced within the width of the table. If the width of the table is specified, then tabularx will be used to typeset it, and one or more X column specifiers should be specified. Otherwise tabular will be used.

This package defines the commands `\ctable`, `\tnote` and `\tmark`, and four `\tabularnewline` generating commands. The latter generate reasonable amounts of whitespace around horizontal rules and are also useful for tabulars outside this package.

Since the **ctable** package imports the **array** and **booktabs** packages, all commands from those packages are available as well.

Note that, in line with the comments that Simon Fear made describing his **booktabs** package, vertical rules for column separation can be produced with `\ctable`, but no provisions are made to have them make contact with horizontal rules.

8.1 Usages:

Unlike other table creating packages, ctables does not provide an environment named `ctable`.i.e to create a table using **ctable** package, you dont have to start with `\begin{ctable}` and end with `\end{ctable}`. It rather functions like a command.

Here, to create a table you have to use the `\ctable` command.

`\ctable` is called with 4 parameters, of which the first is optional:

```
\ctable[options] % key=value,...  
{coldefs} % for \begin{tabular}  
{foottable} % zero or more \tnote commands  
{table rows} % rows for the table
```

In the options(optional field) you can mention more than one specifications, which would be separated by comma.

If you don't want to specify all the options manually, **ctable** package also provides a command to set all the parameters to their default values.

The command is `\setupctable`.

`\setupctable` can set the defaults for all options except (of course) caption, cap, and label.

Actually, the initial option defaults are set by calling `\setupctable` as follows:

```
\setupctable{  
captionskip=0pt, framerule=0pt, nostar,  
center, framesep=0pt, pos=tbp,  
continued=(continued), maxwidth=0pt, super,  
doinside={}, mincapwidth=0pt, table,  
framebg=1 1 1, nonotespar, topcap,  
framefg=0 0 0, nosideways, width=0pt  
}
```

8.2 Options:

Currently these are the following options that you can specify in the optional parameter of **ctable** command

bgopacity=... :	Sets the opacity of the table's background color, where 1 is 100% opaque (the default), and 0 is completely transparent. One application is with watermarking: most watermarking packages print their watermark on the background. ctable's background color, which is opaque by default, may make the watermark (partially) invisible. You can avoid this by setting the bgopacity option to a value lower than 1. Note that this works only in PDF mode, a warning is issued otherwise.
botcap :	put the caption at the bottom of the float.
caption=... :	table caption; the braces are needed only if your caption contains a comma or an equals sign.
cap=...:	for a short caption to go to the <code>\listoftables</code> . Without the cap option, the full caption will go into the <code>\listoftables</code> . If cap is given an empty value, and you have loaded the caption package, no entry in the <code>\listoftables</code> will be made. This may be useful, for example, with the continued option.
captionskip=... :	moves the caption relative to the table; the default is <code>0ex</code> , which puts captions at their default LATEX positions. For the standard LATEX classes this means that a top caption's baseline at <code>1ex</code> above the top rule position of the table and a bottom caption's baseline at <code>4ex</code> below the bottom rule position. These dimensions may be different for other classes or when other packages are included. The memoir class and the caption package, for example, typeset both captions differently. Keep in mind that when you use the caption package in the memoir class, memoir's caption commands are suspended and caption's commands must be used.
captionsleft :	This option is defined for <code>\setupctable</code> only, and it is effective only where the sideways option is used. After <code>\setupctable{captionsleft}</code> all tables typeset with the sideways option will have their captions at the left.

captionsright :	This option is defined for <code>\setuptable</code> only, and it is effective only where the <code>sideways</code> option is used. After <code>\setuptable{captionsright}</code> all tables typeset with the <code>sideways</code> option will have their captions at the right.
captionsinside:	This option is defined <code>captionsinside</code> for <code>\setuptable</code> only, it is the default, and it is effective only where the <code>sideways</code> option is used. After <code>\setuptable{captionsinside}</code> all tables typeset with the <code>sideways</code> option will have their captions at the left in one-sided documents. In twosided documents, captions will be on the left for odd-numbered pages and on the right for even-numbered pages. This is the default
center :	center the table in the available text width; this is the default. See also: <code>left</code> , <code>right</code> .
continued=... :	if used, the table will be numbered the same as the previous table. If used without an argument, the caption will be suffixed with ‘ (continued)’, if used with an argument, the suffix will be the argument.
doinside=... :	command to be run inside, just before the <code>tabular</code> or <code>tabularx</code> environment. You can use this, for example, for the adjustment of the font size with <code>\small</code> .
figure :	produce a figure float instead of a table float.
footerwidth=... :	Footnotes are typeset within the width of the table. When you use the <code>mincapwidth</code> option, presumably because the table is very narrow, footnotes are given the same width as the caption. With small footnotes this may not be what you want; this option can be used to give the footnotes their own width. Without an argument, they will be typeset within the width of the table.

framebg=r g b :	set the background color of the frame (the color inside the frame) to the given triplet of rgb-values. The values should be numbers between 0 and 1. The default is 1 1 1 (white).
framefg=r g b :	set the foreground color of the frame (the rule color) to the given triplet of rgb-values. The values should be numbers between 0 and 1. The default is 000 (black)
framerule=... :	draw a frame around the table with the given rule thickness. The default is 0 pt, so that no frame will be seen.
framesep=... :	set the distance between the frame and the table to the given dimension. The default is 0pt
label=... :	labels the float with <code>\label</code> .
left :	left align the table in the available text width. See also: center, right.
maxwidth=... :	like the width option, but any X column specifiers will be replaced with l if the resulting table width would thus stay within the specified maximum width. This is especially useful where the L ^A T _E X source is generated by a script.
mincapwidth=... :	sets the minimum width of the float. Without this option, the width is set to that of the tabular, and the caption and footnotes are typeset within that width. This may be a problem with very narrow tables; mincapwidth can then be used to give the float a minimum width. The tabular will be centered in it. If you don't want the footnotes to be affected see the footerwidth option.
nonotespar :	typeset footnotes in a table; this is the default. See also: notespar.
nosideways :	undo the sideways option. See also: sideways.

nostar :	use the un-starred versions of the table and figure environments; this is the default
nosuper :	in the footnote table, typeset footnote markers on the line, instead of superscripted.
notespar :	typeset footnotes in a paragraph instead of in a table
pos=... :	float position, default: tbp.
right :	right align the table in the available text width.
sidecap :	put the caption at the side of the float. Currently, this works only if you have loaded the memoir class, otherwise an error message is generated. The parameters for the caption, such as its vertical positioning, width and more, must be set with the appropriate memoir commands. See also: botcap, topcap.
sideways :	rotate table or figure by 90 degrees anticlockwise and put it on a separate page. With the twoside option for the standard L ^A T _E X document classes, rotation will be -90 on even pages, unless the options captionleft or captionsright are used. If you use this option, the pos option is not allowed. See also: nosideways, captionsinside.
star :	use the starred versions of the table and figure environments, which place the float over two columns when the twocolumn option or the <code>\twocolumn</code> command is active. See also: nostar.
super :	in the footnote table, typeset footnote markers as superscripts; this is the default. See also: nosuper
table :	produce a table float (this is the default). See also: figure.

topcap :	put the caption top of the float; this is the default. See also: botcap, sidecap.
width=... :	tabularx will be used to typeset the table at the specified width—one or more X column specifiers must be provided.

8.3 The width and maxwidth options:

L^AT_EX sources containing tables are generated automatically by a script, it is often not known in advance what the maximum size of an l column will be. A good solution for this is to use an **X specifier**, typesetting the table at the text width with the **tabularx** package. However, this will result in too much white space in cases where the column contains small texts only.

This problem can be solved by using the **maxwidth** option (which can be explored by using this package) instead of the width option. The X specifiers will then be replaced with l as long as the width of the resulting table stays with the specified maximum width (or you can use X specifier if you wish to do so).

8.4 Tables wider than the text width:

When you make a table wider than `\textwidth`, it will extend in the right margin. If it is a large table, occupying a whole page, you can use the geometry package and surround your ctable call with `\newgeometry{width=...,margin=...}` and `\restoregeometry`. However, both geometry commands imply `\clearpage`, so your table will appear on an otherwise empty page.

Alternatively, you can center the table on the paper, extending in both margins, by using the option :

`doinside=\hspace*{<dimen>}` with an appropriate negative `dimen >`.

8.5 Other commands:

<code>\tnote :</code>	<code>\tnote[label]{footnote text}</code> places ^{label} footnote text under the table. This command can only be used in <code>\cetable</code> 's third argument, i.e. the foottable argument described above. The label is optional, the default label is a single a. For more detailed control, you can also replace this command with something like <code>labeltext&footnotetext\NN</code> . The footnotes are placed under the table, without a rule. You therefore probably will want to use the <code>\LL</code> (last line) command if you use footnotes.
<code>\tmark :</code>	<code>\tmark[label]</code> this command places the superscripted label in the table. It is equivalent with <code> \${label} \$</code> . The label is optional, the default label is a single a.

The newline generating commands are a combination of `\tabularnewline` and zero or one of booktabs `\toprule`, `\midrule` or `\bottomrule`. These combinations have been made, and short names have been defined, because source texts for complex tables often become very crowded:

<code>\NN :</code>	Normal Newline, generates just a normal new line. An optional <code>dimen</code> parameter inserts extra vertical space under the line. Is an alias for <code>\tabularnewline</code>
<code>\FL :</code>	First Line, generates a new line and a thick rule with some extra space under it. An optional <code>dimen</code> parameter sets the line width; the default is 0.08em. Is an alias for <code>\toprule</code>
<code>\ML :</code>	Middle Line, generates a new line and a thin rule with some extra space over and under it. An optional <code>dimen</code> parameter sets the line width; the default is 0.05em. Is an alias for <code>\tabularnewline\midrule</code>

\LL : Last Line: generates a new line and a thick rule with some extra space over it. An optional dimension parameter sets the line width; the default is 0.08em is an alias for **\tabularnewline\bottomrule**. These macros can be used outside **\ctable** constructs.

Finally, for completeness, here are some of booktabs' commands that may be useful:

\toprule : **\toprule[<wd>]** where <wd> is the optional thickness of the rule.

\midrule : **\midrule[<wd>]**.

\bottomrule : **\bottomrule[<wd>]**.

\cmidrule : **\cmidrule[<wd>](<trim>){a-b}** where <trim> can be r, l, or rl and the rule is drawn over columns a through b.

\morecmidrules : **\morecmidrules** must be used to separate two successive cmidrules.

\addlinespace : **\addlinespace[<wd>]** inserts extra space between rows.

\specialrule : **\specialrule{<wd>}{<abovespace>}{<belowspace>}**.

8.6 Examples:

Well, we are going to give an example of creating a table using **\ctable**

```
\ctable[
cap = The Skewing Angles,
caption = The Skewing Angles ( $\beta$ ) for
 $\mu(H)+X_2$  and  $\mu(H)+HX^{\sim}$ ,
label = nowidth,
```

continued to next page

```

pos = h
]{rlcc}{
\tnote{for the abstraction reaction,
$\fam0 Mu+HX \rightarrow MuH+X$.}
\tnote[b]{1 degree$ = \pi/180$ radians.}
\tnote[c]{this is a particularly long note, showing that
footnotes are set in raggedright mode as we don't like
hyphenation in table footnotes.}
}{ \FL
& & $\fam0 H(Mu)+F_2$ & $\fam0 H(Mu)+Cl_2$ \ML
&$\beta$(H) & $80.9^\circ$ \tmark[b] & $83.2^\circ$ \NN
&$\beta$(Mu) & $86.7^\circ$ & $87.7^\circ$ \LL
}

```

Note: Now, as you can see in the code that **ctable** package provides different commands for giving footnotes in the table as well as mark the footnote in the appropriate place. The first one is done by **\tnote** command and the second one is done by **\tmark** command. Now, In both cases, footnote marking start from 'a' (by default).

This will produce output like:

Table 1: The Skewing Angles (β) for $Mu(H) + X_2$ and $Mu(H) + HX$ ^a

	H(Mu) + F ₂	H(Mu) + Cl ₂
$\beta(H)$	80.9 ^{°b}	83.2 [°]
$\beta(Mu)$	86.7 [°]	87.7 [°]

^a for the abstraction reaction,
 $Mu + HX \rightarrow MuH + X$.

^b 1 degree = $\pi/180$ radians.

^c this is a particularly long note, showing
that footnotes are set in raggedright mode
as we don't like hyphenation in table
footnotes.

Now, as we mentioned before that the command `\cap` is used for creating a short caption. You can't see it in the table. But if you want a list of tables at the beginning of your document and use `\listoftables` to do so, then this short caption will be displayed in that section. But, while you are creating tables with the help of `tabular` or `tabu` environment then for same purpose, you have to wrap the `tabular` or `tabu` environment within `table` environment. Now if the caption provided to the table is short, then there is no problem in displaying it in the **List of Tables**. But if the caption is too long, then you have to explore the complete features provided by the `\caption` command. since, the command has an optional parameter '`short`'.

`\caption{short}{long}`

you have to mention the shorter caption in the short parameter field.

Also, if you don't want to specify the width of the table, you have to set the label as `nowidth`.

Now in the first example we have not specify the table width. In the next example, we will illustrate how to specify the width of a table created by the use of this package.

```
\ctable[
caption = Example with a specified width of 100mm,
label = width,
width = 100mm,
pos = ht,
left
]{c>{\raggedright}Xc>{\raggedright}X}{
\tnote{footnotes are placed under the table}
}{ \FL
\multicolumn{4}{c}{Example using tabularx} \ML
\multicolumn{2}{c}{Multicolumn entry!} & THREE & FOUR \NN
\cmidrule(r){1-2}\cmidrule(rl){3-3}\cmidrule(l){4-4}
one&
The width of this column depends on the width of the table.\tmark &
three&
Column four will act in the same way as column two,
with the same width. }
```

This will produce output like:

Table 2: Example with a specified width of 100mm

Example using tabularx			
Multicolumn entry!		THREE	FOUR
one	The width of this column depends on the width of the table. ^a	three	Column four will act in the same way as column two, with the same width.

^a footnotes are placed under the table

8.6.1 Use of different options in the optional field of `\ctable`:

center, left, right: These options align the float in the page; the default is center:

```
\ctable[
caption=Centered
]{c}{ }\FL Here the table is centered.\LL}
```

Table 3: Centered

Here the table is centered.

Now, as you can see, the table no. is 3. However, if you want to set the table no. to a different value you have to use:

`\addtocounter{table}{desired no.}`

Also, notice that for this table, we have not mentioned any alignment, since the default alignment is center.

Now, in the next example, we will use the table alignment **left**.

```
\ctable[
caption = Left,
left
]{c}{ }\FL Here, the table is in left side.\LL}
```

It will produce output like:

Table 4: Left

Here, the table is in left side.

And, we will use the **right** alignment now.

```
\ctable[
caption = Right,
right
]{c}{c}{\FL Now, the table is in the right side\LL}
```

It will produce output like:

Table 5: Right

Now, the table is in the right side

Positioning of table: You could specify the float position of the table using option `{pos}`. You could use all the float specifiers discussed previously to specify the float position. For instance if you use the float position of a table as `b`, then the table will be created at the bottom of a page.

```
\ctable[
caption = At the bottom,
pos=b
]{c}{c}{\FL Now, the table is at the bottom of the existing page\LL}
```

In our document, we will not typeset the output of this code. However, from this code, you could learn how to specify the floating position of a table.

super, nosuper: Footnote markers in `ctable` are typeset superscripted by default. Use the `nosuper` option to place them on the base line.

Now, in the first example of the corresponding topic, we will show that the footnote markers in `table` produces superscripted output by default.

```
\ctable{c}{
\tnote{First footnote}
\tnote[b]{Second footnote}
}{\FL Table's\emark\ first\emark[b]\ row\LL}
```

Now, look at the output of the code. It can be clearly seen that the footnotemarkers are supercripted in this table.

Table's ^a first ^b row
^a First footnote
^b Second footnote

Now, if we use **nosuper** option:

```
\ctable[nosuper]{c}{
\tnote[a.]{First footnote}
\tnote[b.]{Second footnote}
}{\FL Table's\emark\ first\emark[b]\ row\LL}
```

It will place the footnotemark in the base line:

Table's ^a first ^b row
<i>a.</i> First footnote
<i>b.</i> Second footnote

notespar, nonotespar: By default, footnotes in ctable are typeset in a table, one line per note. This corresponds with the **nonotespar** option. You can also typeset them in a paragraph, one after the other, by using the **notespar** option.

In the first example, we will show you the default output (i.e. the footnotes are typeset in a table):

```
\ctable{c}{
\tnote{First note}
```

continued to next page

```
\tnote[b]{Second note}
\tnote[c]{Third note}
}{\FL Table's\tmark\ first\tmark[b]\ row
with footnotes\tmark[c]\LL}
```

This will produce output like: In the next example, we will mention the **notespar**

Table's^a first^b row with footnotes^c

^a First note
^b Second note
^c Third note

option.

```
\ctable[notespar]{c}{
\tnote[a]{First note.}
\tnote[b]{Second note.}
\tnote[c]{Third note, this one is a
little longer and forces a
new line at the end.\\}
\tnote[d]{And here is e very long note:
\input{thuan}}
}{\FL Table's\tmark\ first\tmark[b]\ row
with footnotes\tmark[c]\LL}
```

This will produce output like: The continued option suffixes the caption with ‘(continued)’, and lowers the table number by one, so that it obtains the same number as the previous table. This option can be given an argument to replace the default suffix:

```
\ctable[
caption = Caption,
mincapwidth = 50mm,
]{c}{\FL Table's first row\LL}
```

Table's^a first^b row with footnotes^c

^a First note. ^b Second note. ^c Third note, this one is a little longer and forces a new line at the end.

^d And here is e very long note: Had our solar system included two suns, the problem would have involved three bodies (the two suns and each planet), and chaos would have been immediately obvious. Planets would have had erratic and unpredictable orbits, and creatures living on one of these planets would never have been able to perceive the slightest harmony. Nor would it have occurred to them that the universe might be ruled by laws and that it is up to man's intellect to discover them. Besides, it is not at all obvious that life and conscience could even emerge in such a chaotic system.

Table 6: Caption

Table's first row

This will produce the output like:

Now, if you want to continue the same table with the same caption, you have to adjust the table numberings and have to use the 'contoption:

```
\addtocounter{table}{-1}
\ctable[
caption = Caption,
mincapwidth = 50mm,
continued
]{c}{\FL Table's first row\LL}
```

This will produce output like:

Table 5: Caption
(continued)

Table's first row

Note: To make it reliable that the last table is the continuation of the table-6 we provide the same caption to it through **caption**. We set the table numbering as **Table:-6** and we use **continued** option. You have to do all these three things to make it reliable that the table you have just produced is the continuation of the previous one.

Now, if you use continued option you will get the word 'Continued' typeset beside your table caption inside a parentheses. If you want to replace that word '**Continued**' with the word *contd* within a parentheses:

```
\addtocounter{table}{-1}
\ctable[
caption = Caption,
mincapwidth = 50mm,
continued = \textit{(contd)}
]{c}{\FL Table's first row\LL}
```

It will produce output like:

Table 4: Caption (*contd*)

Table's first row

Notice that, if we use continued option you get the same word typeset beside the caption of the corresponding table within a parentheses. Here, you get the parentheses around the word by default. But if you want to use *contd* instead of the word 'continued', L^AT_EX does not provide you the parentheses around it by default. You have to insert it manually.

mincapwidth: ctable forces caption and footnotes to stay within the width of the table. Sometimes, however, tables are so narrow, that this is not really what

you want. In such cases, use the **mincapwidth** option to give caption and footnotes some extra room.

In the first example, we are not going to use mincapwidth to make you understand what will happen if the table itself is narrower than the caption and the footnote.

```
\ctable[
caption = a lengthy caption
]{c}{\tnote{this is a footnote}}
{\FL row1\tmark{LL}}
```

It will produce output like:

Table 5: a lengthy caption
<hr/>
row1 ^a
<hr/>
^a this is a foot- note

Now, to solve this problem, you have to use **mincapwidth** option.

```
\ctable[
mincapwidth=60mm,
caption = a lengthy caption
]{c}{\tnote{this is a footnote}}
{\FL row1\tmark{LL}}
```

Now, check the difference:

maxwidth: We have already discussed about it. But, to refresh your memory, we are going to discuss it again.

When L^AT_EX -sources containing tables are generated automatically by a script, it

Table 6: a lengthy caption

row1 ^a
^a this is a footnote

is often not known in advance what the maximum size of an l column will be. A good solution for this is to use an X specifier, typesetting the table at the text width with the **tabularx** package. However, this will result in too much white space in cases where the column contains small texts only. This problem can be solved by using the maxwidth option instead of the width option. The X specifiers will then be replaced with l as long as the width of the resulting table stays with the specified maximum width.

First example:

```
\ctable[framerule=.1pt, maxwidth=3cm
]{lX}{}{\FL 1 & first row\LL}
```

It will produce output like:

1	first row
---	-----------

Second example:

```
\ctable[framerule=.1pt, maxwidth=3cm
]{lX}{}{\FL 1 & test\LL}
```

It will produce output like:

1	test
---	------

Coloring of tables created by ctable: The following examples show the use of frames and backgrounds. Every table is typeset by ctable with a frame around it, but the frame is, by default, drawn with a zero width line, and is therefore invisible. You can make it visible by either changing the linewidth to a positive value or by giving it a background color, which will be used to fill the frame. Here is a simple table without a frame, followed by one with a red, 1pt thick frame:

```
\ctable[
caption = Frame,
]{c}{c}{\FL Table's first row\LL}
```

As you can see, it produces a simple table without frame.

Table 7: Frame

Table's first row

And now we are going to typeset a table with a red, 2pt thick frame:

```
\ctable[
caption = Frame,
framerule = 2pt,
framefg = .8 0 0
]{c}{c}{\FL Table's first row\LL}
```

The output of the code:

Table 8: Frame

Table's first row

Now, **framefg** is used to set the foreground color of a table. It uses **r g b** model(red,green,blue) of color. Here, we specify r,g and b as .8,0 and 0 respectively. And **framerule** draws a frame around the table with the given rule thickness. Here, we specify 2pt as the thickness of the framerule. Now, notice that , the frame fits closely to the first (**\FL**) and last (**\LL**) table lines. This can be a reason to either remove those lines, or to introduce some whitespace between the frame and the table with the **framesep** option.

```
\ctable[
caption = Frame,
framerule = 2pt,
framefg = .8 0 0,
framesep=10pt
]{c}{\FL Table's first row\LL}
```

It will produce output like:

Table 9: Frame

Table's first row

Here, in the code, we just change one thing, that is we change the **framesep** which is 0pt by default, to 10pt. So, the framerule is now separated from **\FL** and **\LL**.

And, finally, we are going to change the background color of the table

```
\ctable[
caption = Frame,
framebg = 1 1 0,
framesep=10pt
]{c}{\FL Table's first row\LL}
```

It will produce output like:

Table 10: Frame

Table's first row

The background color is changed by **framebg**. It also follows the **r g b** model of color. Here, we use the background color as yellow. But you can change it by changing the value of r,g and b.

You can change the both if you wish to do so. We are not going to illustrate any example for that.

captionskip: Now, the caption is by default on the top. If we create a table using **ctable** package and use the **caption** option. i.e for a caption in a table created by ctable, the option is set to **topcap** as by default. Now, the distance between a top caption and the table is 2ex, but it can be varied with captionskip.

First example:

```
\ctable[
caption = Caption,
]{c}{c}{\FL Table's first row\LL}
```

The output:

Table 11: Caption

Table's first row

As you can see, this is a table with caption(since we have not used the captionskip the distance between caption and the table is 2ex).

Now, if you want to change it to 1ex:

```
\ctable[
caption = Caption,
captionskip = 1ex,
]{c}{c}{\FL Table's first row\LL}
```

It will produce output like:

Table 12: Caption

Table's first row

We can do the same thing while we are using **botcap**

```
\ctable[
caption = Caption,
botcap
]{c}{c}{\FL Table's first row\LL}
```

If you use the given code, you will get the default output,i.e the space between the table and the bottom caption is 2ex.

Table's first row

Table 13: Caption

Now, if you want to change the space between caption and the table as 1ex, use the following code:

Table's first row

Table 14: Caption

```
\ctable[
caption = Caption,
captionskip = -1ex,
botcap
]{c}{\FL Table's first row\LL}
```

It will produce output like:

figure, botcap: By default, ctable generates a table float, but with the figure option, a figure float is generated instead. The caption stays on top, so if you are accustomed to have bottom caption for your figures, you will probably also need the **botcap** option.

```
\newcommand{\M}[1]{
\includegraphics[width=\hsize]{#1}
}
\newcolumnntype{H}[1]{>\hsize=#1\hsize}X}
\ctable[
caption = a figure,
figure, botcap,
width=.4\hsize,
]{H{.4}H{.45}}{\FL
\M{Tiger}& \M{Mouse}\LL
}
```

This will produce output like:



Figure 1: a figure

doinside: The argument of `doinside` is supposed to be a command to be run inside, just before the `tabular` or `tabularx` environment. You can use this, for example, for the adjustment of the font size with `\small`:

```
\ctable[
caption=Doinside,
doinside = \scriptsize]{l}{
}{\FL
This table has all rows \NN
set at script size \LL
}
```

It will produce output like:

Table 15: Doinside

This table has all rows set at script size

This topic ends here. Now, we are going to discuss about creating table with **booktabs** package.

9 booktabs package and table creation:

Many professionally typeset books and journals feature simple tables, which have appropriate spacing above and below lines, and almost never use vertical rules.

Many examples of LaTeX tables (including this Wikibook) showcase the use of vertical rules (using `|`), and double-rules (using `\|`), which are regarded as unnecessary and distracting in a professionally published form. The `booktabs` package is useful for easily providing this professionalism in LaTeX tables, and the documentation also provides guidelines on what constitutes a "good" table.

9.1 Useful commands:

<code>\toprule</code>	This produces a horizontal line at the top of the table.
<code>\midrule</code>	produces a horizontal line in the middle of the table.
<code>\bottomrule</code>	produces a horizontal line after you finish the table.
<code>\toprule[<wd>]</code>	<code>\toprule</code> with width specification (<code>wd</code> = a TeX dimension)
<code>\midrule[<wd>]</code>	<code>\midrule</code> with width specification
<code>\bottomrule[<wd>]</code>	<code>\bottomrule</code> with width specification
<code>\cmidrule</code>	<code>\cmidrule[<wd>](<trim>){a-b}</code> where <code><trim></code> can be <code>r</code> , <code>l</code> , or <code>rl</code> and the rule is drawn over columns <code>a</code> through <code>b</code> .
<code>\morecmidrules</code>	<code>\morecmidrules</code> must be used to separate two successive <code>\cmidrules</code> .
<code>\addlinespace</code>	<code>\addlinespace[<wd>]</code> inserts extra space between rows.
<code>\specialrule</code>	<code>\specialrule{<wd>}{<abovespace>}{<belowspace>}</code> produces a rule when all the three dimensions are defined.

Note: For providing spaces between double rules (Those rules can be typeset by `\toprule`, `\midrule` or `\bottomrule`) you can use ordinary L^AT_EX separator `\doublerulesep`. But to provide space between two successive `\cmidrules` you have to use `\morecmidrules`. Also note that, the space provided by `\morecmidrules` is less than the space provided by `\doublerulesep`.

9.2 Booktabs and longtables:

If you have both `booktabs` and `longtable` packages loaded, the `booktabs` rule commands can now all be used exactly as described above, within a `longtable`. There is an addition worth noting: within a `longtable`, you can use the optional left and right trimming commands, which normally only work for `\cmidrules`, with `\toprule`, `\midrule` and `\bottomrule` (and if you must, also with `\specialrule`). However, if you want your table to be right trimmed you can also use `@` as the last column specifier.

Note: within a `longtable`, `\hline` and `\hline\hline` both produce a double rule (to allow for page breaks occurring at that point). But the `booktabs` rules do not. `Longtable`'s automatic doubling of `\hline` is questionable, even according to the documentation within that package. But doubled `booktabs` rules make almost no sense at all. In the unfortunate event that a `booktabs` rule should occur at a page break, then you will have to make the necessary adjustments by hand. (In general, this will mean deleting the offending rule.)

9.3 Booktabs and ctable package:

If both packages are loaded in the preamble of your document then you can use the commands of `booktabs` package while you are creating a table with `ctable` package.

9.4 Booktabs and the colortbl package:

`Booktabs` is now compatible with the `colortbl` package. The `\arrayrulecolor` command will result in coloured rules if the `colortbl` package is loaded.

We are not giving any examples of creating a table with `booktabs` package here. Check the example we provided in the **Professional Tables** section.

10 Footnotes in tables:

The **tabular** environment does not handle footnotes properly. The **longtabular** fixes that.

Instead of using **longtabular** we recommend **tabu** which handles footnotes properly, both in normal and long tables.

10.1 Creating footnotes with footnotes command:

For instance, if you write the following code:

```
\tabureset
%this is used to reset all parameters of tabu
\everyrow{\hline}{
\begin{tabu}{|l|c|}
Command&Use\\
\textbackslash{}footnotesize&This specifies the font size
will be of footnotesize\\
\end{tabu}}
\footnote{This table is for size specification of font.}
```

This will produce output like: %this is used to reset all parameters of tabu

Command	Use	2
<code>\footnotesize</code>	This specifies the font size will be of footnotesize	

In the example, we use **tabu** environment to create the table. However, this footnote option works fine for **tabular** environment too. No additional package is needed for this.

10.2 Creating tablefootnote or tablenote using tablefootnotepackage:

You could alternatively load **tablefootnote** package in the preamble of your document and use the `\tablefootnote` command to get a **tablenote**. It is to be used in a table or `sidewaystable` environment.

²This table is for size specification of font.

Let's try tablenote with table environment. Since we have not discussed about **sidewaystable** environment yet.

```
\begin{table}[ht]
\begin{tabular}{|>\bfseries l|p{8cm}|}
\hline
Command&Use\\\hline
\&\&column separator\\\hline
\textbackslash{}\textbackslash{}&start new row
(additional space may be specified after \textbackslash{}
\textbackslash{} using square brackets, such as
\textbackslash{}\textbackslash{}[6pt])\\\hline
\textbackslash{}hline&horizontal line\\\hline
\textbackslash{}newline&start a new line within a cell
(in a paragraph column)
\\\hline
\textbackslash{}cline\{i-j\}&partial horizontal line beginning
in column i and ending in column j\\\hline
\end{tabular}
\tablefootnote{This is about table commands}
\end{table}
```

It will produce output like:

Command	Use
&	column separator
\\	start new row (additional space may be specified after \\ using square brackets, such as \\[6pt])
\hline	horizontal line
\newline	start a new line within a cell (in a paragraph column)
\cline{i-j}	partial horizontal line beginning in column i and ending in column j

3

Note: you have to use `\tablefootnote` command inside the table environment after like it is done in in the code. If you use the `\tablefootnote` command after the use of `\end{table}` command then you will not get the proper output.

10.3 Creating footnotes with `\footnotetext` command:

For using `\footnotetext` command no additional package loading is required.

So far, what you've seen that whenever we are trying to add a footnote, we get a footnote numbering at the outside of the created table. Now, Suppose you want to create a table which contains more than one footnotes. You also want the footnotemarks inside the particular cells. then what should you do?

Below here, we illustrate an example for such problems (Remember the table which contains data about different table specifications and their meanings? well, we are going to recreate that):

<code>l</code>	left-justified column
<code>c</code>	centered column
<code>r</code>	right-justified column
<code>p{'width'}</code>	paragraph column with text vertically aligned at the top
<code>m{'width'}</code>	paragraph column with text vertically aligned in the middle ⁴
<code>b{'width'}</code>	paragraph column with text vertically aligned at the bottom ⁵
<code> </code>	vertical line
<code> </code>	double vertical line

³This is about table commands

⁴This requires the use of array package

⁵This requires the use of array package too

This is done by using the following code:

```
\begin{table}[ht]
\begin{center}
\begin{tabular}{|l|p{8cm}|}\hline
l&left-justified column\\\hline
c&centered column\\\hline
r&right-justified column\\\hline
p\{'width'\}&\paragraph column with text vertically aligned
at the top\\\hline
m\{'width'\}&\paragraph column with text vertically aligned
in the middle\protect\footnotemark\\\hline
b\{'width'\}&\paragraph column with text vertically aligned
at the bottom \protect\footnotemark\\\hline
$|&$vertical line\\\hline
$|&$double vertical line\\\hline
\end{tabular}
\end{center}
\end{table}
\addtocounter{footnote}{-1}
\footnotetext{This requires the use of array package}
\addtocounter{footnote}{1}
\footnotetext{This requires the use of array package too}
```

To get the footnote numbering you have to use `\protect\footnotemark` command. Now, in the example the footnote numberings are 3 & 4. since we have previously used two footnotes. To get the proper numbering of the footnotes in the **footer** portion of the page, you have to adjust it with the `\addtocounter{footnote}{numbering}` command.

10.4 Creating footnotes in other table environments and using other packages:

- The **longtable** package provides the longtable environment as replacement for the combined table and tabular environments. Footnotes are real footnotes (not just tablenotes), are continuously numbered and hyperlinked (when using the hyperref package), and the hyperlinks really work. As drawback the appearance of the caption changes slightly (e. g. distance to the table, width of the caption), which can probably be changed back manually.

Furthermore, longtables are meant to break over more than one page. If that is not wished, it must be prevented by `\nopagebreak`-commands and by ending the longtable lines with `*` instead of `\\`. longtables do not float. (Therefore using the tablefootnote package and `\FloatBarrier` from the picins package before and after the table environment is similar - but tablefootnote does not change the table-caption!) sidewaysstable does not work with it.

- The **supertabular** package provides the **mpsupertabular** environment as replacement for the combined table and tabular environments. Footnotes are just tablenotes (with working hyperlinks when using the hyperref package), i. e. numbered a, b, c and placed below the table and not at the end of the page. Therefore there is no float problem (because the tablenotes numbering is not included in the continuous numbering of the footnotes). Placing the supertabular inside of a sidewaysstable breaks the hyperlinks to the tablenotes.(we will discuss it later).
- The **ctable** package has its very own notation for defining tables. It can create tablenotes and sideways-tables. The tablenotes are not automatically hyper- linked. The ctables float. Because the tablenotes numbering is not included in the continuous numbering of the footnotes there is no float problem.(we will discuss it later).
- The **footnote** package provides `\makesavenoteenv{table}`. After loading the package and using that command in the preamble, in tables `\footnote{...}` can be used. Using `\makesavenoteenv{tabular}` and `\makesavenoteenv{sidewaysstable}` is possible, but it neither solves the float problem, nor do the created hyperlinks work (i. e. they aim at wrong locations). The mdwtab from the same bundle is incompatible with other table-related packages (e. g. supertabular, array) and not 100% compatible with the tabular environment.(So, no point of discussing it in the details)
- The **tabularx** package does produce footnotes for sidewaysstables, but uses a, b, c instead of 1, 2, 3. The hyperlinks to the footnotes do not work. Because the footnotes numbering is not included in the continuous numbering of the other footnotes there is no float problem.
- Placing a tabular inside a minipage inside a table produces tablenotes. Therefore there is no float problem (because the footnotes are not continuously numbered). The hyperlinks to the table notes indeed work.(you have not learnt the use of tabular inside a minipage yet).

- The **threeparttable** package creates tablenotes again. Therefore there is no floatproblem (because the tablenotes are not continuously numbered with the footnotes). There are no hyperlinks to the table notes (at least not automatically). Using sidewaysstable (with table notes) works. Now, we are going to discuss it in details.

10.5 Use of threeparttable package in creating tablenotes:

This package facilitates tables with titles (captions) and notes. The title and notes are given a width equal to the body of the table (a tabular environment).

By itself, a **threeparttable** does not float, but you can put it in a **table** or a **table*** or some other floating environment. (This causes extra typing, but gives more flexibility.)

Inside a **threeparttable** there should be a caption, followed by a **tabular** environment (**tabular**, **tabular***, **tabularx** or the like), possibly followed by a series of itemized tablenotes. (The caption may also go after the tabular environment.) The code structure of creating footnotes using **threeparttable** package is like:

```
\begin{table}
\begin{threeparttable}[b]
\caption{...}
\begin{tabular}...% or {tabular*}
...42\tnote{1}&.... ...
\end{tabular}
\begin{tablenotes}
\item [1] the first note ...
\end{tablenotes}
\end{threeparttable}
\end{table}
```

As you can see, the use of **threeparttable** package provides you an environment named **threeparttable**. The footnotes are added by using the **tablenotes** environment. For getting proper numbering of the footnotes in the footer portion of a particular page, you need to itemize the footnotes like it is shown in the code structure.

The **threeparttable** environment takes an optional vertical-placement parameter, **[t]**, **[b]**, or **[c]**; the default is **[t]**.

There are several commands which should be redefined for customizing the behaviour of **threeparttable**, especially the table notes. Some options are provided for common variations of the table notes:

- **para**: Notes come one-after-another without line breaks
- **flushleft** No hanging indentation on notes
- online **\item tag** is printed normal size, not superscript
- **normal** restores default formatting

These options can be given to the **\usepackage** command or to each individual **tablenotes** environment. The **[normal]** option is intended to reverse the whole document options for a particular table; e.g.

Suppose, you want to use the **para** option. you can do:

```
\usepackage[para]{threeparttable}
```

i.e mentioning it in the preamble of the document. Or:

```
\begin{tablenotes}[normal,flushleft]
```

i.e mentioning it as an optional parameter with the **tablenotes** environment. These few options will not give you every format you might want, so you may find that you need to redefine one or more of the configuration commands.

Note that mixing options with redefinitions is unlikely to work smoothly: Please submit your redefinitions to be used as options in future versions! Configuration commands:

- **\TPTminimum:** command giving minimum caption width. Default 4em; change with `\def` or `\renewcommand`.
 - **\TPTrlap:** A command with one argument, to make notes go out of the column, into the column separation (for right-aligning).
 - **\TPTtagStyle:** command with one argument to set appearance of the tag (number) in `\tnote{tag}`. It defaults to nil. It could be `\textit`.
 - **\tnote:** You can redefine the `\note` command.
 - **\TPTnoteLabel:** Command with one argument to format the item label in the tablenotes list (`\makelabel`); default uses `\tnote`.
 - **\TPTnoteSettings:** A command to issue all the list-environment setup commands for the tablenotes.
 - **\tablenotes** or **\TPTdoTablenotes:** Yes, you can redefine the whole tablenotes environment. (`\tablenotes` processes optional parameters, then invokes `\TPTdoTablenotes`; the `[para]` option replaces `\TPTdoTablenotes`).
- The threeparttablex package creates tablenotes again. Therefore there is no float problem (because the tablenotes are not continuously numbered with the footnotes). With option referable the tablenotes are hyperlinked.

11 Sideways tables:

Tables can also be put on their side within a document using the **rotating** or the **rotfloat** package.

Then, what you need to do to create sideways table(or table in landscape mode) that you have to use **sidewaystable** environment before the use of different table creating environments specially **tabular** i.e you have to wrap the use of **tabular** environment with **sidewaystable** environment.

12 Floats,figures and captions:

12.1 Floats:

Floats are containers for things in a document that cannot be broken over a page. LaTeX by default recognizes "table" and "figure" floats, but you can define new ones of your own . Floats are there to deal with the problem of the object that

won't fit on the present page, and to help when you really don't want the object here just now.

Floats are not part of the normal stream of text, but separate entities, positioned in a part of the page to themselves (top, middle, bottom, left, right, or wherever the designer specifies). They always have a caption describing them and they are always numbered so they can be referred to from elsewhere in the text. \LaTeX automatically floats Tables and Figures, depending on how much space is left on the page at the point that they are processed. If there is not enough room on the current page, the float is moved to the top of the next page. This can be changed by moving the Table or Figure definition to an earlier or later point in the text, or by adjusting some of the parameters which control automatic floating.

12.1.1 Placement specifiers:

h	Place the float here, i.e., approximately at the same point it occurs in the source text (however, not exactly at the spot)
t	Position at the top of the page.
b	Position at the bottom of the page.
p	Put on a special page for floats only.
!	Override internal parameters \LaTeX uses for determining "good" float positions.
H	Places the float at precisely the location in the \LaTeX code. Requires the float package, e.g., <code>\usepackage{float}</code> . This is somewhat equivalent to h!.

12.1.2 creating a list of tables:

This is done by using `\listoftables` command. This adds a list of tables at the beginning of the document.

12.2 Tables:

To treat table as a float, you have to wrap the corresponding table environment (like `tabu`, `tabular` etc...) within `table` environment and to use the placement specifiers.

12.3 captions:

It is done by placing the `\caption{...}` command with **table** environment.

12.3.1 Examples:

```
\begin{table}[h!]  
  \begin{center}  
    \begin{tabular}{|l c r |}  
      \hline  
      1 & 2 & 3 \\  
      4 & 5 & 6 \\  
      7 & 8 & 9 \\  
      \hline  
    \end{tabular}  
  \end{center}  
  \caption{A simple table}  
\end{table}
```

It will produce output like: To add the caption above the table:

1	2	3
4	5	6
7	8	9

Table 16: A simple table

```
\begin{table}[h!]  
  \caption{A simple table}  
  \begin{center}  
    \begin{tabular}{|l c r |}  
      \hline  
      1 & 2 & 3 \\  
      4 & 5 & 6 \\  
      7 & 8 & 9 \\  
    \end{tabular}  
  \end{center}
```

continued to next page

```

\hline
\end{tabular}
\end{center}
\end{table}

```

It will produce output like:

Table 17: A simple table

1	2	3
4	5	6
7	8	9

12.4 Subfloats:

A useful extension is the subcaption package which uses subfloats within a single float. This gives the author the ability to have subfigures within figures, or subtables within table floats. Subfloats have their own caption, and an optional global caption. An example will best illustrate the usage of this package:

```

\begin{table}[\<placement specifier>]
  \begin{subtable}[\<placement specifier>]{\<width>}
    \centering
    ... table 1 ...
    \caption{\<sub caption>}
  \end{subtable}
  ~
  \begin{subtable}[\<placement specifier>]{\<width>}
    \centering
    ... table 2 ...
    \caption{\<sub caption>}
  \end{subtable}
\end{table}

```

13 References:

- <http://en.wikibooks.org/wiki/LaTeX>
- tex.stackexchange.com
- dcolumn package documentation by **David Carlisle**.
- tabulax package documentation by **David Carlisle**.
- tabulary package documentation by **David Carlisle**.
- tabu package documentation.
- longtable package documentation by **David Carlisle**.
- supertabular package documentation by **Johannes Braams and Theo Jurrrens**.
- ctable package documentation by **Wybo Dekker**.
- booktabs package documentation by **Simon Fear**.