目录

[1] 这里选择“左右对齐”（以及下文的“上下对齐”）的称呼方式，是为了避免如“水平对齐”、“垂直对齐”这种称呼方式可能引起的误解，但是“水平居中”和“垂直居中”却不存在这种问题，因此这两个称呼照旧不变。

1 tabular 环境的必选参数：对齐方式

1.1 左右对齐[2]

必须在 tabular 环境的必选参数中指明每一列的对齐方式：l（左对齐）、c（水平居中）、r(右对齐)。LaTeX 的原生命令中还设置了一个 “p{}” 参数，用于设置列宽，同时默认对齐方式为左对齐和上对齐，将在下文具体介绍。

A 左对齐	B 水平居中	C 右对齐
D	E	F

可以发现，有几个【代表每一列对齐方式的字母】（下面称为 “对齐字母”），就说明有几列，如果没有合并列的操作，环境中每行上的 “&” 符号的数量就在对齐字母数量的基础上减一，在每一行的最后使用 “\\” 符号表示该行结束。

如果连续几个列都具有相同的对齐方式，可以用 “*{num}{cols}” 的形式设置必选参数。

AAA	BBB	CCC	DDD	EEE
F	G	H	I	J

1.2 上下对齐

代表每一列左右对齐方式字母的 “lcr” 可以替换成代表每一列上下对齐方式的字母：“p{}”、“m{}”、“b{}”（后两者由 array 宏包提供），分别代表上对齐、垂直居中、下对齐（吴康隆 P52–53 称为 “竖直居上”、“竖直居中”、“竖直居下”），在这三个字母后面的大括号中需要指定列宽，但是同时文字自动左对齐，无法再选择左右对齐方式（但是可以使用 array 宏包提供的 “>{decl}” 命令和 “<{decl}” 命令进行设置，见下文），效果如下所示：

[2]这里选择 “左右对齐”（以及下文的 “上下对齐”）的称呼方式，是为了避免如 “水平对齐”、“垂直对齐” 这种称呼方式可能引起的误解，但是 “水平居中” 和 “垂直居中” 却不存在这种问题，因此这两个称呼照旧不变。

						Texte	texte	texte
						texte	texte	texte
			Texte	texte	texte	texte	texte	texte
			texte	texte	texte	texte	texte	texte
Texte	texte	texte	texte	texte	texte	texte...		
texte	texte	texte	texte	texte	texte			
texte	texte	texte	texte...					
texte	texte	texte						
texte...								

但是经过检验可以发现，所谓的“上对齐”、“下对齐”，并不是直觉上的靠近表格上端和靠近表格下端，反而“上对齐”的文本靠近表格下端，“下对齐”的文本靠近表格上端，这其实反映出一个很关键的概念性问题：从“对齐”的本义出发，同一列上的文本的左右对齐方式应该相同（这也是 tabular 环境的必选参数中“lcr”设置的效果），同一行上的文本的上下对齐方式应该相同。上述命令的打印效果看似违反直觉，其实是由于同一行上的三列设置了不同的上下对齐方式导致的[3]。在 LaTeX 中，表格的对齐方式只能按照列来统一定义，因此可以为不同的列设置不同的左右对齐方式，却不能为不同的行设置不同的上下对齐方式（或许有相应的宏包可以达到这一效果）。

如果为不同列设置相同的上下对齐方式，这样就能发挥上下对齐真正的作用，不过要注意为每一列的文本设置不同的上下长度，以体现上下对齐的效果：

上对齐：

[3]但是上述命令的打印效果仍然可以反映出同一行上的三种不同的上下对齐方式对应的对齐基线被 LaTeX 放置在了同一条水平线上。

Texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte...	Texte texte texte texte texte texte texte...	Texte texte...

下对齐：

Texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte...	Texte texte texte texte texte texte texte...	Texte texte...

如果想要将文本设置成真正的靠近表格上端或者靠近表格下端，可以通过 tabularray 宏包提供的 tblr 环境中的 “h{}” 参数和 “f{}” 参数进行设置（“h” 和 “f” 分别来自 “head” 和 “foot” 的缩写），但是这个环境同时也提供 lrcpmb 参数。

lcr 参数：

A1 左对齐	B1 水平居中	C1 右对齐
A2	B2	C2
A3	B3	C3

hpmbf 参数：

靠近表格 上端	上对齐 上对齐	垂直居中 垂直居中 垂直居中	下对齐 下对齐	靠近表格 下端

如果只看上面这张表格，“h{}” 参数和 “b{}” 参数，以及 “f{}” 参数和 “p{}” 参数的效果似乎是一样的，下面以 “h{}” 参数和 “b{}” 参数为例，展示这两者在效果上的不同，最左边的一列设置了 “h{}” 参数，而右边的两列设置了 “b{}” 参数。

Texte texte texte texte texte...		Texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte...
	Texte texte...	

1.3 小数点对齐

还有一种特殊的对齐方式是 **“小数点对齐”**[4]，即同一列上的所有小数通过小数点对齐，如果存在整数，则整数的后一位和同一列上的其他小数的小数点对齐。这种对齐方式由 dcolumn 宏包（“d” 应当是来自 “decimal” 的缩写）提供的 “\D{}{}{}” 命令搭配 array 宏包提供的 “\newcolumntypenamedefinition” 命令来完成。“\D{}{}{}” 命令的第一个参数表示输入的符号，第二个参数表示输出的符号，第三个参数是用来控制水平表线的；如果填入一个正数，表示每个小数点后的位数最好控制在这个数字之内，如果填入一个负数；表示小数点后的位数不受限制；如果填入 0，表示最好不要输入小数；如果填入一个小数，则整数部分表示每个小数点前的位数最好控制在这个数字之内，小数部分表示每个小数点后的位数最好控制在这个数字之内，下面展示几个例子。

定义输入 “,” 符号，输出 “.” 符号，并且小数点后位数最好在 3 位以内：

[4]根据下文的例子，可以发现，这里的 “小数点” 其实应该作广义的理解，即任何提前定义的作为对齐标准的符号。

1.234	2.345	3.456
4.567	5.678	6.789

定义输入 “:” 符号，输出 “,” 符号，并且小数点后位数不受限制：

1,234444444	2,345	3,456
4,567	5,678888	6,789999

定义输入 “,” 符号，输出 “,” 符号，并且小数前位数最好在 3 位以内，小数点后位数最好在 2 位以内：

100,2	22,45	3,6
4,56	500,6	6,78

定义输入 “:” 符号，输出 “,” 符号，并且小数点后位数不受限制，但是实际输入了 “,” 符号，编译不会报错，但是会无法根据小数点对齐：

1,234444444	2,345	3,456
4,567	5,678888	6,789999

定义输入 “:” 符号，输出 “:” 符号，并且小数点后位数最好在 2 位以内，但是实际出现了 3 位以上的小数，编译不会报错，也可以根据小数点对齐，但是水平表格线只会延伸到每一列都是 2 位小数的情况下的长度，即定义的最大值：

1.23	2.345	3.456
4.567	5.68	6.78999

2 tabular 环境的可选参数

2.1 表格不带上下水平表线时

可选参数 “c” 表示表格中央和文本所在行对齐（默认值）。

A	B	C
D	E	F

可选参数 “t” 表示**表格第一行**和文本所在行对齐。

A	B	C
D	E	F

A	B	C

可选参数 “b” 表示**表格最后一行**和文本所在行对齐。

D	E	F

2.2 表格带上下水平表线时

可选参数 “c” 表示表格中央和文本所在行对齐（默认值）。

A	B	C
D	E	F

可选参数 “t” 表示**表格上端**和文本所在行对齐。

A	B	C
D	E	F

A	B	C

可选参数 “b” 表示**表格下端**和文本所在行对齐。

D	E	F

可以发现，当表格带上下水平表线时，在设置可选参数为 “t” 或者 “b” 时，不再是表格的**第一行**或者**最后一行**和文本所在行对齐，如果仍要求保持这一对齐效果，可以将设置上下水平表线的命令由 “\hline” 分别替换为 “\firsthline” 和 “\lasthline”。

通过 “\firsthline” 命令和 “\lasthline” 命令设置上下表线后，可选参数 “t” 表示**表格第一行**与文本所在行对齐。

A	B	C
D	E	F

通过 “\firsthline” 命令和 “\lasthline” 命令设置上下表线后，可选参数 “b” 表示**表格最后一行**和文本所在行对齐。

A	B	C
D	E	F

3 表格中的几个核心概念

3.1 单元格边距

我们首先要定义 **“单元格边距”** 的概念。为了便于说明，我们先绘制下面这一张表格，其实就是为上文中的第一张表格绘制了竖直表线，以及在每一个表格内容的通过 “\fbox{text}” 命令周围绘制了一个边框，我们将这个

边框称为 “文本框”[5]。

A 左对齐	B 水平居中	C 右对齐
D	E	F

可以发现，在每个文本框的左右两端和列的左右两端之间，各自存在一段距离,这段距离就定义为 “单元格边距”,控制单元格边距的是 “\tabcolsep” 命令或者 “\arraycolsep” 命令[6]，前者用于 tabular 环境，后者用于 array 环境，后者通常用来绘制数学中的表格，比如矩阵，此处不展开。这两者的默认值都是 6pt，它的长度就是一个一行一列的表格当中的单元格的左右边距，也是每一列上水平长度最长的一行的文本框的左右边距。可以发现，在为某一列设置了 “左对齐”、“水平居中对齐” 和 “右对齐” 的对齐方式后，该列的列宽就可以自动确定：它就等于该列上水平长度最长的一行的文本框长度加上其左右边距的长度。

可以通过 “\setlength{cmd}{length}” 命令或者 “\renewcommand{cmd}{def}” 命令来设置单元格边距，下面展示将 tabular 环境中的单元格边距设置为 12pt 之后的效果。

单元格边距默认值为 6pt：

AAA	BBB	CCC
D	E	F

单元格边距设置为 12pt：

AAA	BBB	CCC
D	E	F

3.2 左对齐、水平居中对齐和右对齐

在定义了 “单元格边距” 的概念后，我们再来定义 “左对齐”、“水平居中对齐” 和 “右对齐” 的概念。事实上，从 “对齐” 的本义出发，只有在文本的行数或者列数多于一行/列，并且这些不同行/列的文本的长度不同的时候，

[5]这其实是 LaTeX 中定义的 “箱子”（box）。

[6]“colsep” 显然是来自 “column separation” 的缩写，但是这个名称容易让人误以为是列与列之间的距离，更何况，“列与列之间的距离” 本身也是一个模糊的说法，因此，本文尝试用 “单元格边距” 来定义这两个命令控制的对象。

“对齐” 这一概念才有意义，以上面这两张表格为例，为了呈现竖直方向上的左右对齐的效果，我们不得不在竖直方向上设置两行长度不同的文本（水平方向上的上下对齐的情况同理）。在这一认识的基础上，我们就可以进一步去定义 “左对齐”、“水平居中对齐” 和 “右对齐” 的概念。

再次观察上面这张表格，我们可以发现，在设置了左对齐的一列上，第一行和第二行的单元格左边距是相等的，都等于默认值 6pt；在设置了水平居中对齐的第二列上，第一行和第二行的单元格各自的左右边距是相等的，其中，第一行的单元格的左右边距等于默认值 6pt；在设置了右对齐的第三列上，第一行和第二行的单元格右边距是相等的，都等于默认值 6pt。据此，我们就可以清楚地定义 “左对齐”、“水平居中对齐” 和 “右对齐”：“左对齐” 就是指在竖直方向的同一列上的多行内容，其文本框的左边距都等于默认值 6pt，“右对齐” 就是指在竖直方向的同一列上的多行内容，其文本框的右边距都等于默认值 6pt，“水平居中对齐” 就是指在竖直方向的同一列上的多行内容，其文本框各自的左右边距都相等，并且其中水平长度最长的文本的文本框的左右边框都等于默认值 6pt。

3.3 行距

水平方向上的上下对齐方式的定义可以参照左右对齐方式的定义进行理解，但是在 LaTeX 的配置中，竖直方向上并不存在类似于水平方向上的 “单元格边距” 的概念，即单元格的上下两端各自和行的上下两端之间的距离（可以比照 “单元格左右边距” 称呼为 “单元格上下边距”）并没有一个命令来控制，并且这两段距离其实是不相等的。LaTeX 只有一个 “\arraystretch” 命令用来控制**行距**，即行的上下两端之间的距离，其默认值为 1，下面展示普通的表格和将该值设置为 2 的时候的表格。

行距默认值为 1：

A 左对齐	B 水平居中	C 右对齐
D	E	F

行距设置为 2：

A 左对齐	B 水平居中	C 右对齐
D	E	F

4 tabular 环境的其他参数

4.1 竖直表线和水平表线

在此基础上，可以增加其他参数来丰富表格形式：在必选参数中使用“|” 符号来增加竖直表线，“|” 符号可以无限叠加，使用 “\hline” 命令增加水平表线，“\hline” 命令也可以无限叠加。

A	B	C
D	E	F

可以使用 “\cline{x-y}” 命令绘制从第 x 列到第 y 列的局部水平表线。

A	B	C
B	E	F

可以使用 makecell 宏包提供的 “\Xhline{}” 命令和 “\Xcline{}{}” 命令调整水平表线的宽度。

A	B	C
D	E	F
G	H	I

4.2 设置表格宽度

一共有四种方式来设置表格宽度，设置的对象又可以分为两类，一类是整表宽度，另一类是列宽。

第一种方式是使用 LaTeX 的原生命令中为 tabular 环境的必选参数设置的 “p{}” 参数，为每一列设置列宽。

设置了 “p{}” 参数后，将无法直接设置左右对齐方式，解决的办法将在下文具体介绍。另外，这种设置方式只能逐列进行，好处是可以为每列设置

不同的列宽，缺点是无法一次性为整表宽度进行设置。

A	B	C
D	E	F

第二种方式是使用 tabular* 环境，它比 tabular 环境多了一个控制整表宽度的必选参数，但是如果只设置这一参数，最后打印出的效果和没有这一参数的 tabular 环境打印出的表格是一样的。

通过 tabular 环境设置的普通表格：

A	B	C
D	E	F

通过 tabular* 环境设置的表格：

A	B	C
D	E	F

可以发现，即使设置了整表宽度为页宽，表格宽度也没有什么变化。解决的办法是，需要在此基础上，在控制对齐方式的必选参数的开头加上一条“@{\extracolsep\fill}”命令。

A	B	C
D	E	F

可以发现，新加的“@{\extracolsep\fill}”命令的效果是让表格从第二列开始，在每列单元格内容的左侧加上（或者减去）相同的宽度（调整的宽度是系统计算后自动填充的，这就是其中“\fill”命令的作用），使得最后整表的宽度达到设置的值[7]。

有意思的一点是，也可以将“@{\extracolsep\fill}”命令中的参数替换成一个具体的宽度，比如“@{\extracolsep{3cm}}”，在这种情况下，tabular* 环境的第一个控制整表宽度的必选参数中设置什么值就不重要了，最后的整表宽度以“@{\extracolsep{}}”命令中设置的宽度和原表格的宽度共同决定（经检验，即使此时将 tabular* 环境的第一个必选参数设置为一个负值，

[7]实际上，从第几列开始调整宽度，是由该命令所在的位置决定的，如果将该命令放置在代表第二列对齐方式的字母的右侧，则会从第三列开始，在每列单元格的的左侧调整宽度，但是该命令放置的位置一旦不对，经常会导致奇怪的效果，在实际使用中，通常将其置于必选参数的开头。

也仍然能够编译成功），换句话说，此时设置的就不再是整表宽度，而是部分的列宽。

这种设置方式虽然可以设置整表宽度，但是其只能从第二列或者之后的列开始调整宽度，并且只能在单元格内容的左侧调整宽度，打印出来的表格不太美观。和第一种方式相比，这种方式也无法为每列设置不同的宽度。另外，这一种设置方式虽然可以调整部分的列宽，却无法相应地调整对齐方式，这会导致某些内容无法在 “新列” 的范围内达到有效的对齐效果，以上面这张表格为例：虽然设置了第二列的对齐方式为水平居中对齐，但是第二列的内容并没有真的位于第二列的中间位置，而是仍然处于第二列偏右的位置，这并不是因为命令出现了故障，而是因为第二列的内容确实在 **“旧列”** 中水平居中了，只是由于在此基础上又在单元格左侧增加了一段距离，使得最后看来，在 **“新列”** 的范围内，该内容并没有水平居中对齐。

第三种方式是通过间接设置 “单元格边距” 来设置列宽。设置的方式已经在上文介绍过，这里再次展示将单元格边距设置为 12pt 之后的表格。

AAA	BBB	CCC
D	E	F

这种方法和第二种方式相比，解决了调整的宽度只能从第二列或者之后的列开始，并且只能在单元格的左侧调整宽度的问题，打印出的表格看起来更美观些。但是其存在和第二种方式一样的问题：无法相应地调整对齐方式，因此打印出的表格的内容在 “新列” 的范围内无法达到有效的对齐效果。可以观察上面这张表格的左右两列，其各自设置了左对齐和右对齐，但是内容和列的边缘之间由于重新设置设置了单元格的左右边距，不再是默认值 6pt，也就不再符合 “左对齐” 和 “右对齐” 的定义（关于定义可以参看前文）。实际上，相应的内容在 **“旧列”** 的范围内确实是左对齐或者右对齐的，但是由于在此基础上又在单元格的两边各增加了一段距离，使得最后看来，在 **“新列”** 的范围内，该内容并没有左对齐或者右对齐。除此之外，和第二种方式相比，这种方式无法为整表宽度进行统一的设置，与第一种方式相比，这种方式无法为为每一列设置不同的宽度。

第四种方式是使用 tabularx 宏包提供的 tabularx 环境。这种方式和第三种方式一样，解决了第二种方式中增加的宽度不均匀分配的问题。并且，这种设置方式可以在 “新列” 的范围内设置对齐方式。由于 tabulax 环境中只能使用 “X” 表示对齐方式（左对齐且上对齐），如果需要额外设置不同的

对齐方式，可以参照下文的介绍。

AAA	BBB	CCC
D	E	F

第四种方式同时解决了第二种和第三种方式各自存在的问题，既可以为整表宽度进行统一的设置，并且可以在所有的列上均匀地分配宽度，因此是调整表格宽度的常用方式。和第一种方式相比，这种方式无法为每列设置不同的宽度。在实际操作过程中，可以根据需要选择第一种设置方式或者第四种设置方式。

5 其他操作

5.1 合并行和合并列

可以使用 multirow 宏包提供的 “\multicolumn{n}{cols}{text}” 命令设置跨列内容（类似于 Excel 中的合并同行的多列单元格）。

A	B	C
D		E

结合以上两条命令绘制的表线：
例一

A	B	<-C[8]
D	<-E[9]	

例二

A	B	C[10]->
D	E	

可以使用 multirow 宏包提供的 “\multirow{number of rows}{width or *}{text}” 命令设置跨行内容（类似于 Excel 中的合并同列的多行单元格）。

[9]C 左侧的竖直表线通过 tabular 环境的必选参数设置。
[9]E 左侧的竖直表线通过 “\multicolumn{n}{cols}{text}” 命令的第二个必选参数设置。
[10]C 右侧的竖直表线通过 tabular 环境的必选参数设置。

设置合并后的列宽：

A	B	C
D	E	F
G	H	
J	K	L

不设置合并后的列宽，使其自适应宽度：

A	B	C
D	E	F
G	H	
J	K	L

如果不在命令中删除合并前属于合并范围内的其他内容（红色的 F1），编译不会报错，但是会造成奇怪的效果：

A	B	C
D	E	F
G	H	F1
I	J	K

如果要进行设置跨行兼跨列的操作，则需要将 “\multirow{number of rows}{width or *}{text}” 命令放置在 “\multicolumn{n}{cols}{text}” 命令的内部，即先进行合并行的操作，再进行合并列的操作，如果嵌套的顺序反过来，编译会报错，下面分步演示这一操作。

原表格：

A	B	C
D	E	F

先合并行：

A	B	C
D		F

再合并列（第一步）：

A C	B

可以发现，上面这一步打印出的效果很奇怪，这是因为只对第一行进行了合并列的操作，第二行内部出现的一个竖直表线正是 tabular 环境的必选参数绘制的，还需要对第二行进行同样的合并列的操作，但是不输入合并列之后填入的内容，这样，得到的才是最终的正确结果。

合并列（第二步）：

A D	B

5.2 为同一列设置统一的格式

可以使用 array 宏包提供的 “>{decl}” 命令和 “<{decl}” 命令（“decl” 是 “declaration” 的缩写）为每一列的内容设置统一的格式，前者放在每一列对应的 lcrpmb 参数之前，后者放在 lcrpmb 参数之后，一般情况下设置前者就足够了，后者是在设置环境格式时需要的。这两条命令使得同一列上不仅可以设置对齐方式，而且可以设置除了对齐方式以外的其他格式。

A	B	*C*
D	E	*F*

也可以使用 array 宏包提供的 “\newcolumntype{name}{definition}” 命令为同一列新定义一个格式命令，不过要求命令形式是和 lcrpmb 一样的单字母形式。

A	B	C
D	E	F

前文提到，如果在 tabular 环境的必选参数中设置了同一列的上下对齐方式，就无法再在必选参数中设置该列的左右对齐方式，但是可以通过 array 宏包提供的 “>{decl}” 命令和 “<{decl}” 命令进行设置。

第一种设置方式：

Texte texte texte texte texte texte texte texte texte texte texte texte texte...	Texte texte texte texte texte texte texte...	Texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte...

第二种设置方式：

Texte texte texte texte texte texte texte texte texte texte texte texte texte...	Texte texte texte texte texte texte texte...	Texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte texte...

5.3 在同一列的相同位置插入符号

下面讨论 “@{intercolumn sign}” 命令，该命令放置在 tabular 环境的必选参数之中。如果要深刻理解这一命令的运作机制，可以先观察一下该命令不输入参数的形式，即 “@{}” 命令对打印效果的影响，以下展示 “@{}” 命令在必选参数的不同位置对打印效果的影响。

A	B	C
D	E	F

必选参数：{lll}，不带竖直表线

AB	C
DE	F

必选参数：{l@{}ll}，第一列和第二列之间的间距消失

A	B	C
D	E	F

必选参数：{|l|ll}，增加竖直表线

A	B C
D	E F

必选参数：{@{}|l|ll}，表格没有变化

A	B C
D	E F

必选参数：{|@{}l|ll}，第一条竖直表线和第一列之间的间距消失

A	B C
D	E F

必选参数：{|l@{}|ll}，第一列和第二条竖直表线之间的间距消失

A	B C
D	E F

必选参数：{|l|@{}ll}，第二条竖直表线和第二列之间的间距消失

A	B C
D	E F

必选参数：{|l@{}|@{}ll}，第一列和第二列之间的间距消失

通过观察以上一系列的必选参数和其对应的打印效果，我们可以发现，“@{}” 命令的作用是让相应的间距归零，“间距” 的情况有如下几种：第一，如果列与列之间不存在竖直表线，这段 “间距” 就是前一个单元格的右边距加上后一个单元格的左边距（以下称为 “列间距”），如前文所述，单元格边距的默认值为 6pt，因此，列间距的默认值就是 12pt，如果在两个代表对齐方式的字母之间插入 “@{}” 命令，就会导致相应的列间距从 12pt 变为 0；第二，这段 “间距” 也可以是边上的一列和表格竖直边线之间的距离（以下称为 “列边距”），这个距离其实就是表格两边的单元格的左边距或者右边距，也就是 6pt，如果在位于必选参数两边的 “|” 符号和其对应的两边的代表对齐方式的字母之间插入 “@{}” 命令，就会导致相应的列边距从 6pt 变

为 0；第三，如果列与列之间存在竖直表线，表线会从列与列的中间竖下，这段 “间距” 就是两列和中间的竖直表线之间的距离，也就是左边单元格的右边距或者右边单元格的左边距（以下称为 “列中距”），即 6pt，如果在代表对齐方式的字母和位于必选参数内部的 “|” 符号之间插入 “@{}” 命令，就会导致相应的列中距从 6pt 变为 0。

此时再来讨论 “@{intercolumn sign}” 命令。这一命令的效果可以分成前后两步：第一，执行 “@{}” 命令的效果，将相应的间距归零；第二，在该间距归零的位置上插入通过该命令参数设置的符号。比如，参数设置为 “-”，打印出来的效果就是在每行对应的同一位置输出 “-” 这一符号；如果参数设置为 “|”，打印出来的效果就是在每一行对应的相同位置输出一个 “|” 符号，这和在必选参数中设置一个单纯的 “|” 符号的效果显然是不同的。以下是普通的表格和设置了 “@{intercolumn sign}” 命令之后的表格的效果比较。

普通的表格：

A	B	C
D	E	F

设置了 “@{intercolumn sign}” 命令之后的表格：

A-B|C
D-E|F

由于 “@{intercolumn sign}” 命令（包括 “@{}” 命令）会导致相应的间距归零，array 宏包提供了 “!{intercolumn sign}” 命令，只需将上述命令中的 “@” 符号替换为 “!” 符号即可，这一命令也能达到类似的效果，但是相应的间距不会归零（相反，在插入符号的情况下，间距反而会拉大），看起来更加美观。

普通的表格：

A	B	C
D	E	F

设置了 “!{intercolumn sign}” 命令之后的表格：

A - B | C
D - E | F

5.4 为表格中的单词设置自动换行

先提一个问题：为什么 LaTeX 的原生命令中，只为 tabular 环境设置了一种上下对齐方式，即 “上对齐”？这是因为 “p{}” 参数最初的功能**并不是设置上下对齐方式**，而是**设置列宽**，否则无法解释为什么它没有像左右对齐的三个命令一样不带有参数。以上面的这张表格为例，如果必选参数是用 lcr 来表达的，由于没有列宽的定义，表格不会自动换行，而会在同一行上打印出所有的内容，导致表格超出页面。而如果使用 “p{}” 参数为每一列设置列宽，就可以保证表格在固定的列宽内自动换行，不会超出页面。

现在再看单词的问题，上文所说的 “自动换行”，其实需要符合三个条件：第一，表格设置了列宽；第二，文本的长度超过了所在列的列宽；第三个条件最容易被忽视，那就是这段文本应该包含两个及两个以上的单词。自动换行是从第二个单词开始的，这包含两种情况：第一，文本另起一行开始第二个单词；第二，文本按照音节界限从第二个单词内部断开，第二个单词的前面一部分留在第一行，后面一部分另起一行开始，甚至单词过长，列宽过小的时候，可能需要将第二个单词分割成两行以上。因此，“自动换行” 的说法，其实是不包含第一个单词在内的。而表格并不像普通的页面，后者一般不可能在一行上连一个单词都容纳不下，表格的一个单元格完全有可能无法容纳下一个比较长的完整的单词，因此，如果要表格内的文本从第一个单词开始就可以根据列宽进行自动换行，需要进行另外的设置，下面这张表格展示了一个长单词、两个长单词在一个列宽短于该单词长度的表格中的表现情况，左边一列利用 “>{\hspace{0pt}}” 命令进行了另外的设置，达到了从第一个单词开始就自动换行的效果，右边一列没有进行另外的设置，因此可以看到第一个单词并没有自动换行，而是从第二个单词开始自动换行。

Super- con- scious- ness	Superconsciousness
	Superconsciousness Super- con- scious- ness

5.5 在单元格内手动设置换行

上文介绍的是文本在表格内自动换行，但有的时候我们希望能够控制换行的位置，这可以使用 makecell 宏包的 “\makecell” 命令和 “\thead{}” 命令来完成。

“\thead{}” 命令具有和 “\makecell{}” 命令相同的效果和参数，唯一的一点不同是其打印出来的字体大小要更小一些，通常用于表头（“thead” 来自于 “table head” 的缩写）。

在 “\makecell{}” 命令后面加上 “*” 符号，会使得换行后的单元格的竖直距离更大一些，可以比对下面两个表格的高度，右边的高度更大一些，但是 “\thead{}” 命令没有这一区别

A 右对齐 下对齐	BBB	CCC	D 右对齐 上对齐
E1 E2	F	G	H

A 右对齐 下对齐	BBB	CCC	D 右对齐 上对齐
E1 E2	F	G	H

5.6 分割表头

可以使用 diagbox 宏包的 “\diagbox” 命令分割表头。

分割成两部分：

右边 左边	A	B
C	D	E
F	G	H

分割成三部分：

中间 \ 右边 左边	A	B
C	D	E
F	G	H