LaTeX, un peu, beaucoup

Pascal Dupont

Mots clés : LaTeX, tableaux

7. Tableaux (encore)

Dans la rubrique précédente, j'ai indiqué comment composer des tableaux, tout en mentionnant quelques défauts et limitations. Indiquons maintenant des solutions à certains d'entre eux.

Gestion des filets doubles

Un des reproches qui peuvent être faits, notamment, aux tableaux proposés par LaTeX, est que lorsque des filets doubles se croisent, c'est de manière assez inesthétique.

Exemple

Nom	Note
Angèle	10,7
Cynthie	09,4
Domitien	04,0
Fernande	14,5
Rodolphe	13,2
Zita	14,0

Exemple

Il n'y a aucune raison, en effet, dans un cas comme celui-ci, que les filets verticaux soient interrompus.

Pour résoudre ces problèmes, il suffit de faire appel au module *hhline* et d'utiliser la commande `\hhline` qu'il propose comme substitut aux commandes standards `\hline` et `\cline{`*m*`-`*n*`}`. Nous obtiendrons alors

Exemple

Nom	Note
Angèle	10,7
Cynthie	09,4
Domitien	04,0
Fernande	14,5
Rodolphe	13,2
Zita	14,0

ou

Nom	Note
Angèle	10,7
Cynthie	09,4
Domitien	04,0
Fernande	14,5
Rodolphe	13,2
Zita	14,0

Exemple

selon que nous codons :

```
\begin{tabular}{||l||r||}
\hhline{#=#=#}
\textbf{Nom}&\textbf{Note}\\
\hhline{#=#=#}
Angèle&10,7\\
      ⋮
Zita&14,0\\
\hhline{#=#=#}
\end{tabular}
```

ou :

```
\begin{tabular}{||l||r||}
\hhline{|t:=:t:=:t|}
\textbf{Nom}&\textbf{Note}\\
\hhline{|:=::=:|}
Angèle&10,7\\
      ⋮
Zita&14,0\\
\hhline{|b:=:b:=:b|}
\end{tabular}.
```

Donnons quelques explications. L'argument de `\hhline` doit comprendre autant de symboles « = », « - » ou « ~ » qu'il y a de colonnes dans le tableau ; ces symboles produisent respectivement un filet horizontal double, un filet horizontal simple et pas de filet dans la colonne corespondante. Ce qui se trouve entre ces symboles commande ce qui se passe aux intersections. L'exemple suivant illustre les 16 cas possibles lors de l'intersection d'un filet double horizontal avec un filet double vertical. À partir de là, le lecteur devinera sans trop de peine ce qu'il convient de faire si l'un des filets est simple.

Exemple

x	x	x
x	x	x
x	x	x

Exemple

Le codage de cet exemple est :

```
\begin{tabular}{||c||c||c||}
\hhline{::=:|=|:=||}
x&x&x\\
\hhline{:b:=:b|=|b:=|b|}
```

```
x&x&x\\
\hhline{:t:=:t|=|t:=|t|}
x&x&x\\
\hhline{:tb:=|=|tb:=|tb|}
\end{tabular}.
```

Pour chacune des intersections, deux symboles sont utilisés, « : » ou « | », pour indiquer s'il faut ou non un filet du côté gauche et du côté droit ; et entre ces deux symboles, soit rien, soit une ou deux des lettres « `b` » ou « `t` », selon que doivent être présents ou non les filets inférieur ou supérieur.

Jusque là, c'est logique ; mais cela le devient un peu moins si on sait que les deux-points peuvent parfois être omis et que « `#` » est synonyme de « `|tb|` ». Enfin, dans l'argument de `\hhline`, il est permis d'utiliser des expressions de la forme `*{n}{`*arguments*`}` pour indiquer une répétition, comme dans le descriptif de formatage d'un tableau.

Il va de soi que l'usage irréfléchi de ces nombreuses possibilités conduit à des tableaux laids à faire peur, comme celui qui précède, dont l'unique objectif était d'être exhaustif.

Alignement sur la virgule décimale

Lorsque des tableaux sont utilisés pour présenter, p. ex., des résultats comptables, il est utile de pouvoir aligner les chiffres de même rang des nombres. Avec les environnements *tabular* ou *array*, un tel résultat ne peut être obtenu qu'au prix de contorsions peu élégantes. Le module *dcolumn* est né pour résoudre ce problème. Il définit un nouveau symbole à placer dans le descriptif de formatage du tableau : `D{`*sepin*`}{`*sepout*`}{`*g*`,`*d*`}` ; *sepin* y représente le symbole qui sera utilisé comme séparateur décimal dans le fichier source et *sepout* le symbole qui sera utilisé comme séparateur décimal dans le document compilé ; dans l'usage francophone, l'un et l'autre seront normalement la virgule (et il n'est pas indispensable de la placer entre accolades, puisqu'il s'agit d'un argument constitué d'un seul symbole) ; *g* et *d* sont respectivement le nombre (maximal) de chiffres prévus à gauche et à droite du séparateur décimal ; si nous donnons une valeur trop petite, les nombres les plus longs déborderont dans la colonne voisine ; si l'un ou l'autre de ces nombres est négatif, la largeur s'adaptera automatiquement au maximum nécessaire ; enfin, indiquer `{`*d*`}` (un seul nombre, sans virgule) équivaut à `{-1,`*d*`}`.

Voici un exemple et le code correspondant.

Exemple

1,7	1,7	1,7	1,7
,3	,3	,3	,3
2,718	2,718	2,718	2,718
5	5	5	5
2015	2015	2015	2015

Exemple

```
$\begin{array}
{|D,,1|D,,{-1}|D,,{2,3}|D,,{-1,-1}|}
1,7&1,7&1,7&1,7\\
,3&,3&,3&,3\\
2,718&2,718&2,718&2,718\\
5&5&5&5\\
2015&2015&2015&2015
\end{array}$
```

Le lecteur aura observé au passage, avec ravissement, que le petit blanc désagréable qui s'intercale malicieusement après la virgule d'un nombre écrit en mode mathématique ne se manifeste pas ici !

Grâce à la possibilité de déclarer séparateur décimal n'importe quel symbole, et de le remplacer dans le document compilé par n'importe quel autre, ce nouveau descripteur de colonne peut bien sûr être détourné de sa fonction première pour servir à d'autres usages...

Le module *array*

Le module *array* redéfinit complètement les environnements *tabular* et *array* du LaTeX de base, pour y résoudre quelques problèmes plus ou moins mineurs et, surtout, pour en enrichir considérablement les possibilités.

• Il y a d'abord de nouveaux descripteurs de colonne, `m{`*dim*`}` et `b{`*dim*`}`, qui fonctionnent comme `p{`*dim*`}`, mais ont comme ligne de référence, pour l'alignement vertical, la ligne du milieu et la ligne du bas de la cellule. Voici, comme exemple, un tableau qui, contrairement à ce qui pourrait être imaginé, ne comporte qu'une seule ligne !

Exemple

			b b	
		m m	b b	
+	p p	m m	b b	+
	p p	m m		
	p p			

Exemple

```
\begin{tabular}{|c|p{6mm}|m{8mm}|b{6mm}|c|}
\hline
```

```
+&
p p p p p p&
m m m m m m&
b b b b b b&
+\\\hline
\end{tabular}
```

• Il y a ensuite de nouveaux symboles pour placer du matériel entre les colonnes. C'est d'abord `!{texte}`, qui a le même rôle que `@{texte}`, mais qui, contrairement à celui-ci, ne supprime pas l'espace entre les colonnes. Cet effet est bien visible dans l'exemple suivant.

Exemple

o o+o + o o

Exemple

codé comme suit :

```
\begin{tabular}{cc@{+}c !{+}cc}
o&o&o&o&o
\end{tabular}
```

Toujours à placer entre les colonnes du tableau, il y a les très intéressantes (et puissantes !) commandes `>{arg}` et `<{arg}`, qui vont placer leur argument *arg*, pour la première, au début de toutes les cellules de la colonne qui suit, et pour la seconde, à la fin de toutes les cellules de la colonne qui précède. Cet argument peut être du texte, mais aussi des commandes (et toute combinaison des deux). Ceci permet notamment d'effectuer en une fois un changement de police (changement de taille ou de couleur, caractères gras, ...) dans toutes les cellules d'une colonne, ou de mettre toute une colonne d'un *tabular* en mode mathématique (et même, au besoin, en *displaystyle*). L'exemple suivant illustre ces différentes possibilités.

Exemple

moyenne arithmétique : $\frac{a+b}{2}$

moyenne géométrique : $\sqrt{ab}$

moyenne harmonique : $\frac{2ab}{a+b}$

moyenne quadratique : $\sqrt{\frac{a^2+b^2}{2}}$

Exemple

Il a été codé comme suit :

```
{\def\arraystretch{2.25}
\begin{tabular}
{>{\bf moyenne }l<{ :}
 >{$\displaystyle}l<$}
arithmétique & \frac{a+b}2\\
géométrique  & \sqrt{ab}\\
harmonique   & \frac{2ab}{a+b}\\
quadratique  & \sqrt{\frac{a^2+b^2}2}
\end{tabular}}
```

Remarquons bien que les deux-points appartiennent à la première colonne et non à l'espace intercolonne : ils ne sont pas alignés.

Par ailleurs, notons que si la combinaison `>$c<$` détermine une colonne en mode mathématique, centrée, lorsqu'elle est utilisée dans un *tabular*, elle commande au contraire une colonne en mode texte, centrée, lorsqu'elle est utilisée dans un *array*, puisque le `$` agit comme un « interrupteur », faisant passer du mode texte au mode mathématique et vice versa.

• Enfin, le module *array* prévoit la possibilité de définir soi-même de nouveaux descripteurs de colonne, ou plus exactement des abréviations pour des combinaisons de descripteurs fréquemment utilisées. La commande à utiliser est

`\newcolumntype{x}[n]{dev}`,

où *x* est le symbole choisi pour désigner le nouveau type de colonne (attention : il doit consister en un caractère et un seul !), *dev* est le « développement », c'est-à-dire la chaine de caractère par laquelle *x* sera remplacée, et *n* (facultatif) est le nombre d'arguments, entre 0 et 9, la valeur par défaut étant 0.

Par exemple, si nous avons inclus dans le document source (avant le tableau ; l'emplacement le plus normal est bien sûr dans le préambule) la commande `\newcolumntype{d}{>{$\displaystyle}l<$}`, le tableau des moyennes, ci-dessus, peut être codé de la manière plus légère que voici :

```
{\def\arraystretch{2.25}
\begin{tabular}
{>{\bf moyenne }l<{ :}d}
arithmétique & \frac{a+b}2\\
géométrique  & \sqrt{ab}\\
harmonique   & \frac{2ab}{a+b}\\
quadratique  & \sqrt{\frac{a^2+b^2}2}
\end{tabular}}
```

Voici un emploi un peu plus... subtil. Il s'agit de créer, dans un tableau, des colonnes de largeur prédéfinie pouvant éventuellement accueillir plusieurs lignes de texte, comme des colonnes créées par le descripteur `p{larg}`, mais avec le texte *centré*

et non justifié des deux côtés. À première vue, il semble que l'utilisation de `>{\centering}p{larg}` devrait faire l'affaire. Cependant, ce n'est pas le cas : cette solution fonctionne parfois, mais pas toujours ; en particulier, elle ne fonctionne pas lorsqu'une ligne du tableau est suivie d'un filet horizontal. La raison en est que le module *array* redéfinit la commande `\\` à l'ouverture d'un tableau, perturbant ainsi l'effet de la déclaration `\centering`. Une solution est d'utiliser, comme commande pour changer de ligne dans le tableau, `\tabularnewline` au lieu de `\\`. Mais cela alourdit un peu la frappe et rend le document source légèrement moins lisible. Une meilleure solution est de placer dans le préambule les deux lignes suivantes :

```
\newcommand{\SDC}[1]
     {\let\temp=\\#1\let\\=\temp}
\newcolumntype{P}[1]
     {>{\SDC\centering}p{#1}}
```

La première définit une macro `\SDC` (« *Sauver Double Controblique* ») à un argument qui va enregister la commande `\\` sous le nom `\prov`, puis exécuter ce qui est passé comme argument, et enfin restituer à `\\` sa valeur initiale. La seconde définit un nouveau type de colonne, désigné par la lettre `P`, avec un argument ; il s'agit d'une colonne de largeur fixe (donnée par l'argument), avec le texte centré.

Exemple

Titre
Voici un petit poème en prose, sympa mais pas bien malin.

Exemple

est ainsi obtenu grâce au codage suivant :

```
\begin{tabular}{|P{30mm}|}
\hline
Titre\\\hline
Voici un~petit~poème en~prose,
sympa~mais pas~bien~malin.
\\\hline
\end{tabular}
```

Certains utilisateurs de LaTeX chargent, par principe, un grand nombre de modules au début de chaque document, la liste en étant simplemnt recopiée d'un document à l'autre. Je ne suis pas de cette école-là, sans doute parce que ma formation initiale en informatique date d'une époque où les ressources, rares et chères, devaient n'être utilisées qu'avec parcimonie. Donc, si j'ai ma liste de modules favoris, je prends bien soin, dans chaque document, de « pourcenter » ceux qui ne seront pas utilisés. Tout ceci pour bien vous faire comprendre que si je charge le module *array* (presque) par principe, c'est que je le considère comme quasi-indispensable. Il améliore tellement les tableaux (de manière bien visible ou presque invisible) qu'il serait assez stupide de s'en passer.

Cellule barrée en diagonale

Le besoin apparait parfois de tracer une diagonale à travers une cellule. C'est par exemple le cas dans le triangle de Pascal que voici :

Exemple

n \ k	0	1	2	3	4	…
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
⋮	⋮					⋱

Exemple

Pour ce faire, il est nécessaire de charger le module *diagbox* et d'utiliser la commande `\diagbox{InfG}{SupD}`. Pour obtenir l'autre diagonale, c'est `\diagbox[NE]{SupG}{InfD}` qu'il convient d'utiliser.
La toute première cellule du tableau de l'exemple précédent contient donc `\diagbox{$n$}{$k$}`.

Ce qui précède se veut une introduction à l'utilisation de quelques modules utiles. Davantage de détails (certifiés, ceux-là — ☺) pourront être trouvés dans leur documentation officielle, disponible sur `http://www.ctan.org`, le plus simple étant d'introduire le nom du module dans le champ de recherche de la page d'accueil ; ceci renvoie à la page contenant toutes les informations sur le module, y compris, généralement, son mode d'emploi détaillé.

Il reste encore, naturellement, de nombreuses questions à aborder au sujet des tableaux. J'y reviendrai donc sans doute encore — peut-être en fonction des demandes de mes lecteurs.

Pascal Dupont est chargé de cours à HEC•ULg. ✉ `pascal.dupont@ulg.ac.be`.