Les bases de LaTeX

Rédiger un algorithme

Serge Piau

20 avril 2018

Table des matières

Résumé

1 Présentation

LaTeX est un langage et un système de composition de documents créé par Leslie Lamport en 1983. Il s'agit d'une collection de macro-commandes destinées à faciliter l'utilisation du « processeur de texte » TeX de Donald Knuth. Le nom est l'abréviation de Lamport TeX. On écrit souvent LaTeX, le logiciel permettant les mises en forme correspondant au logo.

Du fait de sa relative simplicité, il est devenu la méthode privilégiée d'écriture de documents scientifiques employant TeX. Il est particulièrement utilisé dans les domaines techniques et scientifiques pour la production de documents de taille moyenne ou importante (thèse ou livre, par exemple). Néanmoins, il peut être aussi employé pour générer des documents de types variés (par exemple, des lettres, ou des transparents).

LaTeX exige du rédacteur de (ou au moins pousse le rédacteur à) se concentrer sur la structure logique de son document, son contenu, tandis que la mise en page du document (césure des mots, alinéas) est laissée au logiciel lors d'une compilation ultérieure. LaTeX sépare en deux phases la forme du contenu. Avec les logiciels de type WYSIWYG (What You See Is What You Get, ce que vous voyez est ce que vous obtenez) tels que LibreOffice Writer ou Microsoft Word, la structure est codée par les styles, la forme étant automatiquement et immédiatement visible à l'écran.

La rédaction d'un document LaTeX se fait la plupart du temps à travers un éditeur de texte, puis le document rédigé est traité (compilé) avec LaTeX afin d'obtenir sa version mise en forme au format de données PDF pour impression.

LaTeX requiert un apprentissage initial plus important que celui qui est nécessaire pour les logiciels de type WYSIWYG, du moins pour la mise en page de petits documents simples. Mais une fois cette phase d'apprentissage (dont la complexité s'apparente à celle de l'apprentissage du langage HTML) accomplie, le fait de se concentrer sur le contenu et de laisser à LaTeX le soin de présenter le document devient très appréciable : la qualité du document produit est élevée (formules mathématiques, respect des règles typographiques, polices modernes), la gestion des références bibliographiques (BibTeX), les numérotations et tables des matières sont cohérentes sans qu'on ait à s'en soucier. Par ailleurs, LaTeX laisse à l'utilisateur la possibilité de l'adapter à ses besoins spécifiques en créant ou modifiant des macro-commandes.

`https://fr.wikipedia.org/wiki/LaTeX`

2 Structures de base d'un document

Structure d'un document. Un document LaTeX commence toujours par un préambule (cf. fig. 1) commençant par la commande : `\documentclass[]{}`. Le corps du texte est encadré par les commandes `\begin{document}` et `\end{document}`. Le préambule joue le même rôle qu'un fichier "css" en HTML : il contient les caractéristiques typographiques du document. Le corps du texte contient la totalité du texte que vous souhaitez afficher.

Caractères spéciaux. Comme dans tous les langages de programmation LaTeX possède des caractères spéciaux dont l'utilisation est réservé au langage TeX, en voici la liste : `&,#,$,%;{,},_,^,~,\`. Toutes le commandes LaTeX commence par le caractère "\" qui est aussi appelé le caractère d'échappement car il modifie le comportement de tous les caractères qui le suive.

Insérer un commentaire. LaTeX est un langage de programmation, il prévoit donc la possibilité d'insérer des commentaires dans le code source qui ne seront pas visible dans le document final.

```
\documentclass[10pt,a4paper,french]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[frenchb]{babel}
⋮
\begin{document}
```

```
\begin{document}
\maketitle %affiche le titre
\tableofcontents % affiche la table des matière
\newpage % saute une page
⋮
\end{document}
```

FIGURE 1 – Préambule et corps du texte d'un document LaTeX

ainsi, tout ligne commençant par le caractère % sera un commentaire. Il s'en suit que si vous voulez afficher le symbole % dans votre document, il faut le faire précéder du caractère d'échappement "\" : le code "5\%" produit l'affichage "5%".

Définition des blocs. Comme dans tous les langages de programmations, le code est structuré sous forme de bloc. En LaTeX il y a deux types de bloc :

— ceux qui sont encadrée par les caractères "{" et "}" ;
— ceux qui sont encadrée par `\begin{}` et `\end{}`

Gestion des espaces. Ce qu'il est important de comprendre, quand on rédige un document avec LaTeX, c'est que l'on ne doit pas s'occuper de la typographie. LaTeX considère ainsi que c'est à lui de gérer les espaces séparant les mots et qu'un passage à la ligne ne signifie pas forcément un changement de paragraphe.

```
Voici un jolie      texte plein
d'espaces
inutiles.

Un autre paragraphe obtenu avec deux
    sauts de ligne.
```

Voici un jolie texte plein d'espaces inutiles.
Un autre paragraphe obtenu avec deux sauts de ligne.

Rédiger des mathématiques. LaTeX est conçu pour rédiger des mathématiques et la typographie des formules mathématiques suit des règles différentes de la typographie français. C'est pourquoi les formules mathématiques doivent être intégrer dans une environnement mathématiques. Il existe plusieurs environnement de ce type, le plus simple est d'encadrer la formule par des `$`. Ainsi le code LaTeX `$x\mapsto\sqrt{x^2+1}$` produit le résultat $x \mapsto \sqrt{x^2+1}$. Vous remarquerez que la typographie des lettres est elle aussi changée pour que la formule soit lisible. En règle générale, les commandes mathématiques ne sont utilisables que dans un environnement mathématique.

```
\begin{itemize}
\item "Être ou ne pas être",
\item Avec des accents ?
\item "telle est la question"
\item Et en couleur ? %comment
\item \(a^2<0\), jolie formule !
\item "$A$"
\item $a\notin\mathbb{R}$
\end{itemize}
```

— "Être ou ne pas être",
— Avec des accents ?
— "telle est la question"
— Et en couleur ?
— $a^2 < 0$, jolie formule !
— "A"
— $a \notin \mathbb{R}$

3 Écrire un algorithme

Syntaxe de base. Tous les algorithmes doivent être rédigés dans un bloc encadré par `\begin{algorithm}[H]` et `\end{algorithm}[H]`

On place la description des paramètres passé en entrée (input) dans un bloc `\KwIn{}`. On place la description de la sortie de l'algorithme dans un bloc `\KwOut{}`. Les variables sont déclarées dans un bloc `\KwVar{}`

Caractère marquant le fin d'une instruction. La commande "`\;`" permet de passer à la ligne et d'ajouter le ";" qui marque la fin d'instruction. De très nombreux langages de programmation (JAVA, C, C++, ...) utilise ce caractère et c'est une bonne chose de s'habituer tout de suite à son utilisation.

Titre d'un algorithme. La commande `\caption{titre du code}` permet de donner un titre à votre algorithme.

Initialiser des variables. Pour initialiser ou assigner une valeur à une variable, on utilise au choix :
- la commande "`\assign`" qui produit "$\longleftarrow$ "
- la commande "`=`" qui produit "="

Rédiger une structure "if then". Il y a plusieurs façon de rédiger une structure de la forme "if then else". Si vous ne voulez pas utiliser la condition "else", utilisez `\If{<condition>}{<instructions>}`, sinon, utilisez `\eIf{<condition>}{<instructions>}{<instructions>}`. Dans tous les cas, `<condition>` désigne une expression booléenne qui ne peut prendre que les valeurs vrai ou fausse, et `<instructions>` désigne une ou plusieurs expressions qui se terminent toutes par un ";".

```
\begin{algorithm}[H]
\KwIn{deux nombres entiers $a$, $b$}
\KwOut{}
\If{(a == b)}
  {print ("nombres égaux") \;}
print (a,b) \;
\caption{if ... then}
\end{algorithm}
```

Algorithm 1: if ... then

Input: deux nombres entiers a, b
Output:
if *(a == b)* **then**
  print ("nombres égaux") ;
print (a,b) ;

```
\begin{algorithm}[H]
\KwIn{deux nombres entiers $a$, $b$}
\KwOut{}
\eIf{(a == 2*b) or (b == 2*a)}
  {print (a + b) \;}
  {print ("rien") \;}
\caption{if ... then ... else}
\end{algorithm}
```

Algorithm 2: if ... then ... else

Input: deux nombres entiers a, b
Output:
if *(a == 2*b) or (b == 2*a)* **then**
  print (a + b) ;
else
  print ("rien") ;

```
\begin{algorithm}[H]
\uIf{if condition}
  {something if \;}
\uElseIf{elseif condition}
    {something elseif \;}
\Else{something else \;}
\caption{if then ... else if ... else
    }
\end{algorithm}
```

Algorithm 3: if then ... else if ... else

if *if condition* **then**
  something if ;
else if *elseif condition* **then**
  something elseif ;
else
  something else ;

```
\begin{algorithm}[H]
\KwIn{un jour de la semaine, dans une
     variable de type \type{string}
    nommée \code{jour}}
\KwOut{divers messages}
\Switch{jour}{
\Case{"lundi"}
  {print("Le lundi c'est nul") \;}
\Case{"mardi"}
  {print("On est mardi !") \;}
\Case{"mercredi"}
  {print("On est mercredi") \;}
\Case{"jeudi"}
  {print("Bientôt le week end") \;}
\Case{"vendredi"}
  {print("Enfin le week end") \;}
\Case{"samedi"}
  {print("C'est le week end") \;}
\Case{"dimanche"}
  {print("Fin du week end !") \;}
\Other{print("terme incorrecte") \;}
}
 \caption{écrire une condition
     multiple, "selon \ldots"}
\end{algorithm}
```

Algorithm 4: écrire une condition multiple, "selon . . . "

Input: un jour de la semaine, dans une variable de type `string` nommée `jour`
Output: divers messages
switch *jour* **do**
  case *"lundi"* **do**
    print("Le lundi c'est nul") ;
  case *"mardi"* **do**
    print("On est mardi !") ;
  case *"mercredi"* **do**
    print("On est mercredi") ;
  case *"jeudi"* **do**
    print("Bientôt le week end") ;
  case *"vendredi"* **do**
    print("Enfin le week end") ;
  case *"samedi"* **do**
    print("C'est le week end") ;
  case *"dimanche"* **do**
    print("Fin du week end !") ;
  otherwise do
    print("terme incorrecte") ;

Rédiger une structure de boucle. Pour rédiger une structure de boucle avec un "pour", on utilise `\For{i = ...}{<instructions>}`

```
\begin{algorithm}[H]
\KwVar{result : integer\;}
result := 0 \;
\For{i=1 to 25}
  {result := result+i\;}
print (result) \;
\caption{afficher la somme des 25
    premiers nb entier}
\end{algorithm}
```

Algorithm 5: afficher la somme des 25 premiers nb entier

Var :
  result : integer;
result := 0 ;
for *i=1 to 25* **do**
  result := result+i;
print (result) ;

Codes sources LaTeX