Introduction à Plain TeX

2017-04-28__17 :30

Introduction à Plain TeX

Prérequis

Connaître LaTeX.

TeX vs LaTeX

LaTeX est bien différent de TeX.

Pour être compilable par LaTeX, un fichier doit contenir au minimum l'instruction de classe de document un groupe document :

```
\documentclass{article}
\begin{document}
Hello, world !
\end{document}
```

En ce qui concerne TeX, le fichier est plus simple :

```
Hello, world !
\bye
```

La seule commande nécessaire est `\bye` qui clôt lance la construction du fichier final au format DVI.

Nombreuses sont les habitudes d'un utilisateur de LaTeX qui ne fonctionnent pas avec TeX :

— Les listes `\begin{enumerate}...\end{enumerate}`
— Le chapitrage `\section{...}`, `\subsection{...}`
— Le tailles de police `\Huge`, `\large`, `\small`

Avec TeX, il faut tout gérer à la main : créer des compteurs, charger des fontes, etc.

En clair pour le travail de rédaction usuel, inutile de s'intéresser aux particularités de TeX, les macros LaTeX sont infiniment plus pratiques et dans toute la suite, nous compilerons les exemples avec LaTeX dans le cadre suivant

```
\documentclass{article}
\begin{document}
...
\end{document}
```

Pourquoi faudrait-il alors se plonger dans l'étude de la version originale ? Parce que TeX propose un véritable langage de programmation avec des commandes permettant de travailler avec des variables, de construire des structures conditionnelles des boucles et aussi des procédures.

Registres

Assignation de valeur et affichage

Pour stocker des valeurs, TeX utilise une notion proche de celle de variable typée nommée registre. Nous nous intéressons ici à deux types de registres

— `count` pour des nombres entiers
— `dimen` pour des nombres réels avec dimension

Pour son fonctionnement, TeX utilise évidemment de nombreux registres. Par exemple pour définir l'espacement entre les lignes, la taille des marges ou encore les numéros de page. Il fait donc être prudent quand on choisit de modifier un registre. Le plus simple est d'en créer soi-même avec des noms sans équivoque

```
\newcount\annee          % défnit le compteur \annee
\annee=2017              % assigne 2017 au compteur
\newdimen\valpi          % définit la dimension \valpi
\valpi=3.14pt            % assigne 3.14pt à la dimension

En \number\annee, $\pi$ s'approche de \the\valpi.
```

En 2017, π s'approche de 3.14pt.

On voit que grâce aux commandes `\number` et `\the`, on a pu afficher le contenu des registres, avec l'obligatoire unité de dimension pour le registre `\valpi`.

— `\number<nombre>` affiche l'équivalent décimal de `<nombre>` en supprimant les éventuels zéros inutiles.
— `\the<quantité>` affiche la valeur de `<quantité>`

Que se passe-t-il si on intervertit les commandes d'affichage dans l'exemple précédent ?

```
\newcount\annee          % défnit le compteur \annee
\annee=2017              % assigne 2017 au compteur
\newdimen\valpi          % définit la dimension \valpi
\valpi=3.14pt            % assigne 3.14pt à la dimension

En \the\annee, $\pi$ s'approche de \number\valpi.
```

En 2017, π s'approche de 205783.

Aucun problème pour le compteur `\annee`, tout se passe bien. En revanche, en ce qui concerne la dimension `\valpi`, deux choses :

— la dimension a disparu puisque `\number` affiche un entier sans unité
— la valeur algébrique réelle de `\valpi` est affichée sous la forme du nombre entier le plus proche de $3.14 \times 65536 = 205783.04$.

On comprend ainsi que dans les registres de type `\dimen`, TeX stocke en réalité un entier dont la valeur est l'équivalent en *scaled point* (unité TeX notée sp qui représente 1/65536 pt).

Pour information 1 pt = 0.005363 μm puisque 1 pt = 0.35146 mm

Arithmétique sur les registres

Si les registres sont particulièrement intéressants, c'est parcequ'on peut faire des calculs mathématiques sur leurs contenus. Les calculs possibles se réduisent aux quatre opérations de base mais c'est déjà très bien

```
\newcount\cntA
\newdimen\dimA

\cntA=3                     \dimA=-\number\cntA pt
cntA : \number\cntA   \ --- dimA : \the\dimA

\advance \cntA by -2        \advance \dimA by -2.01pt
cntA-2 : \number\cntA \ --- dimA-2.01 : \the\dimA

\cntA=-3\cntA               \divide \dimA by -2
-3cntA : \number\cntA \ --- dimA/(-2) : \the\dimA
```

cntA : 3 — dimA : -3.0pt
cntA-2 : 1 — dimA-2.01 : -5.01pt
-3*cntA : -3 — dimA/(-2) : 2.50499pt

On peut constater que les calculs sont exacts à 10^{-5} près.

Application : calcul de discriminant

```
\newdimen\discr
\def\discriminant#1#2#3{
  \discr=#2pt\relax
  \multiply \discr by #2
  \dimen0=#1pt\relax
  \multiply \dimen0 by 4
  \dimen0=#3\dimen0
  \advance \discr by -\dimen0
  }
\makeatletter
\let\sanspt\strip@pt
\makeatother

\discriminant{5}{-10}{2}
\discriminant{0.25}{3}{0.1}
Pour $5x^2-10x+2,\ \Delta=\sanspt\discr$
et pour $0.25x^2+3x+0.1,\ \Delta=\sanspt\discr$
```

Pour $5x^2 - 10x + 2,\ \Delta = 60$ et pour $5x^2 - 10x + 2,\ \Delta = 8.9$

La définition de la commande `\discriminant` fonctionne ainsi : on commence par initialiser la dimension `\discr` avec le contenu du paramètre `#2`, c'est à dire le b de ax^2+bx+c. Après la ligne `\multiply \discr by #2`, le registre `\discr` contient alors b^2. Ensuite on initialise une dimension prédéfinie par TeX pour les usages temporaires : `\dimen0`. Elle va prendre la valeur de a, puis successivement $4a$ et $4ac$. Il ne reste plus qu'à affecter la valeur b^2-4ac à `\discr`, c'est ce que fait la dernière ligne `\advance \discr by -\dimen0`.

À ce stage, le travail est presque terminé mais on ne peut pas afficher directement le discriminant avec la commande `\the\discr` car elle fait apparaître l'unité de longueur pt : $\Delta = 60pt$. Pour éviter cela, on va utiliser une commande LaTeX conçue spécialement pour afficher une dimension sans son unité : `\strip@pt`. Problème : il s'agit d'une commande non utilisable telle quelle car elle contient le caractère spécial `@`. Avec `\makeatletter`, nous nous plaçons dans un contexte où l'usage de `@` est autorisé, cela permet de construire une copie de `\strip@pt` qui, elle, sera autorisée dans un contexte normal puisque sans caractère spécial : `\let\sanspt\strip@pt`. Il est alors temps de redonner au carctère `@` son statut particulier avec `\makeatother`.

Portée des registres

```
\annee=2017 \the\annee { \annee=1977 -- \the\annee } -- \the\annee
```

2017 – 1977 – 2017

Boucle répétitive

Exercice : Affichons les 10 premiers termes de la suite de Fibonacci : $F_1 = F_2 = 3$, $F_{n+2} = F_{n+1} + F_n$ pour tout entier n non nul.

Pour cela, nous avons besoin d'une structure répétitive. Nous y reviendrons plus tard mais le principe de base est simple : `\loop ... \if ... \repeat`

Voici une manière de s'y prendre. Dans ce qui suit, à chaque itération, on recalcule `F←Fm+Fmm` après que `Fm←F` et `Fmm←Fm`.

```
\def\fibo#1{
\newcount\F  \newcount\Fm  \newcount\Fmm
\F=1 \Fm=1
\newcount\n
\n=1
$F_0=1$ \par
\loop\ifnum\n<#1
  $F_{\number\n}=\number\F$ \par
  \Fmm=\Fm
  \Fm =\F
  \F=\Fm     \advance \F by \Fmm
  \advance \n by 1
\repeat
}

\fibo{10}
```

$F_0 = 1$
$F_1 = 1$

$F_2 = 2$
$F_3 = 3$
$F_4 = 5$
$F_5 = 8$
$F_6 = 13$
$F_7 = 21$
$F_8 = 34$
$F_9 = 55$