Goeland-croq...

CC BY NC

Geste de savoir

Pourquoi vous devriez jeter un coup d'œil à ConTeXt

20 janvier 2019

Table des matières

1. Introduction

Ce billet va parler d'une... disons variante de LaTeX, qui s'appelle ConTeXt. Cette variante est développée par des néerlandais (Hans Hagen et Taco Hoekwater principalement, puis d'autres qui se sont rajoutés). ConTeXt proposant une syntaxe et une philosophie quelque peu différente de LaTeX tout en permettant les mêmes choses (ou presque).

Avant d'aller plus loin, il va falloir parler un peu du moteur sous le capot de LaTeX, à savoir TeX. Ensuite je vous parlerai un peu plus de comment ConTeXt a été créé et fonctionne, de ses avantages et de ses défauts par rapport à LaTeX.

2. Le moteur de base : TeX

TeX a été développé par le mathématicien Donald Knuth pour composer des documents sous système informatique dans les années 80, quand tous les autres logiciels n'étaient au mieux que balbutiants au pire inexistants. L'idée était de transformer un texte ASCII en document présentable. Ce document pouvait être de n'importe quelle nature, d'un livre entier ou d'un article scientifique remplis de formules mathématiques plus ou moins obscures et impossibles à faire simplement sur ordinateur à l'époque.

Aujourd'hui les autres outils se sont beaucoup développés et peuvent tout à fait concurrencer TeX. Néanmoins on observe qu'il y a une réticence à passer à ces solutions parce que LaTeX s'est démocratisé et permet de faire beaucoup de choses de façon bien, donc formant un cycle qui s'auto-entretient : puisque tout le monde l'utilise, l'outil se développe par son enseignement aux novices et les professionnels développent de nouveaux outils simplifiant l'utilisation pour les autres (et je ne critique pas ça du tout, c'est juste pour le pourquoi TeX est toujours utilisé via LaTeX).

Seulement, TeX, de base, est très austère. Il fonctionne, grosso modo, par lecture des caractères les uns à la suite des autres et les envoie dans le fichier selon ce que sont ces caractères :

— une lettre ou un chiffre est simplement envoyé dans le fichier
— un caractère spécial déclenche une action, comme le dollar `$` qui fait passer en mode mathématique, l'underscore `_` pour écrire en indice, l'exposant ^ pour écrire en exposant par exemple
— le caractère `\` qui est un peu spécial car il indique que ce qui suit est une *séquence de contrôle*, autrement dit une commande, qui s'apparente aux fonctions des langages de programmation

Par défaut, TeX ne connaît qu'un faible nombre de séquences de contrôle, les *primitives* de TeX. À ses débuts, il n'en existait que 300 environ. TeX a ensuite été complété par d'autres primitives pour combler certains manques (création des autres moteurs TeX : -TEX, pdfTeX, XeTeX et le dernier LuaTeX). Chacun rajoutait une couche qui étend les possibilités, mais dans tous les cas on n'a toujours que des primitives, c'est-à-dire des fonctions aussi barbares que

```
\hbox to\hsize{\hfill Bonjour\hfill}
```

pour ne faire que centrer le mot "Bonjour" au centre de la ligne. Bien sûr, plus on veut des affichages complexes, plus il faudrait aligner de ces commandes en connaissant leur fonctionnement. Donc, tout ceci n'est pour l'instant pas très pratique.

Heureusement, TeX permet de créer de nouvelles séquences de contrôle, appelées *macros*, qui permettent d'automatiser certaines tâches.

```
\def\centrerTexte#1{\hbox to\hsize{\hfill #1 \hfill}}% création de
    la macro
\centrerTexte{bonjour}
```

Ainsi on peut automatiser de nombreux processus ; le nombre de macros qu'on définit peut être aussi grands que ce qu'on veut, sous limite de la mémoire de l'ordinateur mais sous une machine moderne en général il n'y a aucun souci.

3. Passer un niveau d'abstraction supérieur : les formats

On peut alors se dire qu'on peut utiliser un fichier rempli de définitions de macros, qu'on nommerait "liste-def.tex" par exemple et que chaque nouveau document commencerait par `\include liste-def`

Ça marcherait dans l'absolu. En revanche pour des raisons pratiques, TeX n'a pas été prévu pour être utilisé comme ça mais pour que ses utilisateurs puissent créer un *format*, c'est-à-dire un ensemble de macros compilé, qui peut être chargé plus rapidement que la lecture de ces fichiers et leur interprétation par TeX. Tant que le nombre de macros définis est faible ça n'apporte pas d'avantages, mais comme on peut être amené à définir des milliers de macros, ça peut vite être nécessaire.

Un format se fait en créant sa liste de macros qu'on termine par la commande `\dump` et en chargeant TeX avec l'option `--ini` signifiant qu'on souhaite créer un format, du nom du fichier qu'on lui fait ingurgiter. Mettons qu'on crée le format **ZesteTex** ; on a un fichier de définition de macros, `zestetex.tex` et on invoque alors `tex --ini ZesteTex`. TeX compile le fichier en un fichier de format ZesteTex.fmt, plus rapidement chargé. TeX crée également un exécutable zestetex.exe sous Windows ou équivalent sous autre système, qui va charger par défaut automatiquement notre format lorsqu'on l'utilisera.

Le tout premier format à avoir été créé a été développé par Knuth, le créateur de TeX, qui l'a appelé plain TeX. Quelques macros bien utiles ont été faites, mais ça reste bien austère encore.

```
Ceci est un fichier plain \TeX.

\bye
```

Les macros fournies avec plain TeX sont vraiment le minimum vital pour s'en sortir. L'utiliser nécessite assez rapidement de connaître assez bien TeX pour faire ce qu'on veut, à moins d'avoir des demandes vraiment basiques. plain TeX propose une solution pour numéroter automatiquement les pages par exemple, pour changer la police d'écriture, pour mettre en place un en-tête et un pied de page, mais si on veut des systèmes permettant de faire des choses comme des chapitres numérotés avec des tables des matières complètes et ce automatiquement... débrouillez-vous.

4. LaTeX est venu, a vaincu

Assez rapidement, un autre format est apparu, LaTeX. Les versions de LaTeX se sont ensuite succédées pour avoir aujourd'hui la version LaTeX2e (et LaTeX 3 est en développement et s'annoncerait comme une révolution complète dans la syntaxe).

LaTeX a été conçu pour faciliter grandement l'utilisation de TeX par sa couche d'abstraction. Pour l'utilisateur lambda, inutile d'aligner les primitives pour faire ce qu'on veut, il existe une macro LaTeX et sinon, un utilisateur aguerri saura la créer. Donc, LaTeX a prévu le coup en permettant de charger des macros.

Ces macros ne peuvent pas être intégrés directement au format parce que chacun doit être libre de les rajouter s'ils en a besoin et de toute façon ça voudrait dire tout recompiler à tout ajout de macros, bref on imagine bien le merdier que ça peut devenir.

Donc, LaTeX permet de aux utilisateurs de définir et charger des packages et des formats de style prédéfinis (article, book, letter et ainsi de suite). Ces fichiers de style correspondent à des définitions de style d'écriture, de tailles de papier entre autres. Ces styles classiques répondent à la plupart des besoins, mais un utilisateur aguerri peut donc se risquer à définir sa propre feuille de style.

Donc la logique de LaTeX est avant tout de permettre de charger des fichiers de style ou de macros préconçus permettant de faire une bonne partie du boulot sans avoir à mettre les mains dans le cambouis.

LaTeX a également développé une syntaxe propre, à base de

```
\documentclass{article}
\begin{environment}
        blabla
\end{environment}
\renewcommand\<commande à redéfinir pour ses besoins>{nouvelle
    définition}
```

on utilise le système des environnements et on redéfinit les commandes préexistantes pour ses besoins (typiquement traduire un titre affiché automatiquement par l'appel d'une macro).

5. ConTeXt entre en scène

On l'aura compris, rien n'empêche quiconque de développer un autre format TeX, sauf peut-être son aversion pour le masochisme. Donc LaTeX domine parce que c'est déjà au point et que ça fait le travail demandé. Seulement, ce fonctionnement modulaire où chaque auteur rajoute sa couche, la syntaxe LaTeX et le manque de contrôle sur l'apparence du document (LaTeX préfère gérer tout ça pour vous automatiquement), ça n'a pas plus à tout le monde.

La légende raconte ainsi que les créateurs de ConTeXt vouent une haine éternelle à LaTeX (enfin, c'est l'explication qui est généralement proposée pour expliquer l'apparition de ConTeXt). Ils ont donc créé leur concurrent, qu'ils voulaient complet mais dans un style somme toute assez différent.

```
\starttext
\setupbodyfont[14pt]
\setuphead[section][color=red]
\section{Hello}
World!
\stoptext
```

Le fonctionnement des `\begin{env}` - `\end{env}` est remplacé par `\startenv`-`\stopenv`. Ça, c'est la partie émergée de l'iceberg parce que ça, dans l'absolu, tout le monde s'en fiche. Par contre, ce qu'on peut noter, c'est qu'il n'y a pas d'équivalent de `\documentclass` à charger et que peu de packages (modules dans le monde de ConTeXt) sont nécessaires.

ConTeXt est extrêmement monolithique. Les packages classiques de LaTeX pour incorporer des figures, pour gérer les langues, l'encodage, les couleurs, les liens hypertextes et bien d'autres sont déjà inclus dans ConTeXt. C'était voulu par ses créateurs, qui ont créé un mastodonte pour vous permettre de tout faire sans avoir besoin de charger de module.

La logique de ConTeXt est également que chaque utilisateur spécifie les paramètres des fonctions qu'il utilise pour leur donner de nouvelles valeurs, plus adaptées à ses besoins que la valeur par défaut. La syntaxe typique pour cela est `\setupsomething[key1=value1, key2=value2]`.

ConTeXt tend à vous forcer la main à redéfinir les styles par défaut via ces commandes ; la philosophie est de remettre l'utilisateur maître de son rendu.

Par exemple, `\setuppagenumbering[left=- ,location={right, footer}]` si on souhaite placer le numéro de page en bas à droite, précédé de "- ". Citons encore

```
\setuppapersize[A3]% A4 est la valeur par défaut
\setuplayout[margin=1.7cm, margindistance=0.3cm, topspace=2cm]
```

Pour continuer l'exemple de la taille de papier, si on le souhaite, on peut créer sa propre dimension de papier

```
\starttext

\definepapersize[nouveau][width=108cm, height=98cm]%parce qu'il
    faut voir les choses en grand
\setuppapersize[nouveau]

Hello World !

\page

\setuppapersize[A4]

Hello world en format A4

\stoptext
```

La syntaxe de ConTeXt repose beaucoup également sur les arguments optionnels (entre crochets). Ces macros peuvent accepter un, deux, trois, parfois quatre voire plus d'arguments optionnels ; la documentation officielle explique le fonctionnement de ces macros en fonction des arguments qui leur sont fournis.

Conséquence directe du fonctionnement monolithique de ConTeXt : toutes les macros de première nécessité disons, comme la gestion des en-têtes et pieds-de-page, du placement des figures, de la gestion des marges ; tout ce qui a trait aux styles de paragraphes, de pages, ont été développées par les mêmes personnes, avec les mêmes syntaxes donc. Une fois le premier pas effectué, on doit comprendre assez bien la suite puisque les macros ont la même syntaxe, le même fonctionnement.

PS : les maths sont bien sûr toujours possibles

```
\startformula

\sum_{n=0}^{+\infty} \sqrt{2\pi n}\left({n\over e}\right)^n
\stopformula
```

pour afficher

$$\sum_{n=0}^{+\infty} \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

6. Pourquoi vous devriez passer à ConTeXt

Parce que la philosophie est différente. ConTeXt vous met directement dans les mains tous les outils pour personnaliser le rendu de vos documents (à vos risques et périls). La façon de procéder et tous les paramètres à fournir sont heureusement expliqués dans des manuels que vous trouvez en ligne sur le site contextgarden.net . ConTeXt a été conçu pour vous remettre au centre de la composition de vos documents et pour vous laisser soigner leur apparence.

Vous n'avez pas à passer du temps à chercher des packages pour faire ce que vous voulez car le côté mastodonte de ConTeXt vous permet de le faire directement. Vous devez juste paramétrer ConTeXt pour lui faire faire ce que vous voulez (la documentation ConTeXt contient toutes les informations pour, rassemblées sur un site unique). Utilisez directement les commandes de personnalisation `\setupsomething` et `\definesomething` à votre disposition.

ConTeXt fournit également de nombreux outils pour créer vos propres macros selon la logique ConTeXt lien ; les modules sont peut-être moins nombreux mais il est tout aussi possible d'en créer de nouveaux et de les utiliser via `\usemodule[module]`. A priori la plupart de ce qui est nécessaire existe mais c'est sûrement que personne ne s'est encore confronté à des besoins pour l'instant imprévus.

Enfin, les développeurs de ConTeXt sont à l'origine du développement de LuaTeX. Leur constat est que TeX permet de nombreuses choses déjà, mais pas toujours de manière simple. LuaTeX permet d'exécuter du code Lua qui envoit ses résultats à TeX. Donc, il y a un module ConTeXt de calcul permettant d'afficher le résultat d'un calcul complexe par exemple (inconcevable en TeX). Ceci n'est qu'un exemple stupide pour illustrer les possibilités derrière.

Autre exemple : les citations. Vous êtes comme moi et vous avez des centaines de références Bibtex dans votre fichier (parce que vous avez conçu votre liste bibliographique depuis Zotero et vous l'avez exporté en format Bibtex). Vous voulez citer une référence, par exemple "Medical Physiology, Boron et Boulpaep" mais vous ne savez plus le nom de la référence Bibtex généré par Zotero. Vous pouvez invoquer `\cite[match(author:Boron)]` et ConTeXt utilisera du code Lua pour effectuer la recherche.

Les développeurs de LuaTeX l'ont conçu pour être opérationnel pour tout format TeX. Donc, tout ce qui est possible via Lua l'est aussi en LaTeX. ConTeXt est cependant à la pointe de l'utilisation des possibilités de Lua puisque ce sont les mêmes personnes qui le développent.

Dernière raison, vous ne faîtes pas comme tout le monde et ça donne une impression de vachement cool (oui cette raison est bidon mais je l'aime bien).

Les développeurs de ConTeXt répondent à cette question ici .

7. Pourquoi vous devriez rester à LaTeX

Parce qu'on ne tournera pas autour du pot, peu de gens l'utilise. Donc, pour travailler en groupe, ce n'est pas pratique si vous êtes le seul à jouer les alter-mondialistes. De la référence existe, mais principalement en anglais ; en néerlandais on doit aussi pouvoir en trouver, peut-être que ce sera utile pour nos amis belges bilingues mais pour les autres francophones je doute que ça aide (je ne veux nullement sous-estimer vos compétences en langue étrangère mais j'en serai très surpris !).

Parce que LaTeX bénéficie d'une richesse de package qui n'ont pas nécessairement leurs équivalents ConTeXt, notamment pour les domaines très spécifiques. Porter le package en ConTeXt est très certainement faisable mais guère pratique à faire et vous n'avez probablement pas de temps à perdre.

Et puis, peut-être que cette syntaxe ne vous plaira pas, trop de choses à gérer par vous-mêmes pour des bénéfices qui ne vous intéresse pas, ou toute autre raison personnelle.

Ce qui est clair, c'est que tant que la communauté ConTeXt sera faible par rapport à LaTeX, ce sera un cercle vertueux pour LaTeX et vicieux pour les alternatives comme ConTeXt. Prenez peut-être le temps d'y jeter un coup d'œil pour vous forger votre propre opinion ?

8. Comment faire le premier pas et installer ConTeXt

La procédure est expliquée sur le site contextgarden .

Vous pouvez y trouver une version ConTeXt stand-alone, qui ne fournit que ConTeXt. J'imagine qu'elle marche bien, pour ma part je garde TeX Live qui fournit tous les formats soumis au CTAN, donc ConTeXt inclus.

La subtilité, c'est que ConTeXt a recours à des scripts, `texexec` et `context`, pour fonctionner plutôt que le fonctionnement traditionnel d'appeler "(la)tex document". Ces scripts automatisent de nombreux processus comme les bibliographies en appelant les programmes requis et en effectuant le nombre de compilations nécessaires, tout ça pour vous, en une seule ligne de commande. Pratique !... mais, ces scripts nécessitent respectivement Perl et Ruby.

Il existe deux versions de ConTeXt, appelés "Mark II" et "Mark IV". Seule Mark IV est encore développée, Mark II est toujours fournie mais n'est plus mise à jour. Mark II était conçu pour tourner sous pdfTeX et XeTeX. `texexec` permet d'exécuter MarkII et d'envoyer soit à pdfTeX soit à XeTeX (vous spécifiez en mettant en commentaire –xetex en début de document).

`context` charge Mark IV, développée pour exploiter les possibilités de LuaTeX (et ne tournant que sous ce moteur). La syntaxe change un peu ; globalement ce qui était possible en Mark II reste compatible en Mark IV mais est dépréciée. Le site contextgarden présente les nouveautés et les nouvelles syntaxes à employer. Préférez peut-être commencer avec celle-ci puisque c'est elle qui sera maintenue à jour.

Supposons que vous avez installé Perl pour exécuter `texexec` ou Ruby pour `context`. Il faut ensuite s'assurer que les bases de fichiers soient bien en place et recensent tous les fichiers ConTeXt puis s'assurer que le format est bien compilé (pour une raison qui m'échappe, ce n'est pas toujours le cas lorsque vous installez TeX Live).

Chargez un terminal pour effectuer les commandes suivantes, selon ce que vous voulez paramétrer

— texexec :

```
mktexlsr
texexec --make --all
```

— context :

```
mtxrun --generate
context --make
```

Ces processus doivent prendre un peu de temps et vous remplir votre terminal d'informations (la compilation de ConTeXt se fait en chargeant de très nombreux fichiers). Si rien ne se passe, c'est peut-être que Ruby n'est pas correctement installé. Ou que c'est plus grave encore, je ne sais pas. Disons qu'on est content une fois que la compilation est faite et que ConTeXt tourne.

9. Conclusion

Ce qui est sûr, c'est que c'est assez différent de LaTeX. Peut-être que ça vous plaira ? Le site contextgarden contient une documentation complète pour installer et faire ce dont vous avez besoin pour créer vos documents.