aisément leur concours à l'effort collectif. En attendant de la déposer sur le CTAN [1], ce qui la rendra largement disponible et facilement installable, la classe de la *Lettre* et sa documentation sont disponibles sur le dépôt `Git` de l'association, à l'adresse :

`https://framagit.org/gutenberg/classe-lettre-gut`

Nous encourageons vivement les amateurs de jolis packages d'aller lire le code de la classe de la *Lettre*. Puisse-t-il susciter des vocations !

Sur ce, nous espérons surtout que le contenu de la présente *Lettre* vous intéressera. Elle paraît bien tardivement, c'est un fait. Mais pour parvenir aux pages que vous lisez, il nous aura fallu rien moins que 219 commits sur le dépôt `Git` de ce numéro, et 66 sur celui de la classe de la *Lettre* ! Et depuis le début du mois de juillet, nous sommes conscients du retard pris et de l'urgence qu'il y a à sortir ce numéro ; chaque jour nous y travaillions, et les échanges sont allés bon train au sein de la rédaction. Le numéro sort finalement en ce 12 août, toujours en deux versions [2] ; nous espérons qu'il vous procurera une agréable lecture estivale et que vos réactions seront nombreuses. À bientôt pour la *Lettre* 45 !

Patrick BIDEAULT

P.-S. 1 : notre correspondant de Landerneau nous signale qu'un membre du CA [3], revanchard et peut-être frustré du peu d'écho rencontré au sein du CA par ses fréquentes et navrantes critiques, a jugé bon de se répandre en contre-vérités dans le *TUGboat*. Pour surprenante que soit cette parution dans un prestigieux organe, dont j'espère que le manque d'exigence rédactionnelle ayant présidé à la publication de cet article ne fut que temporaire, elle confirme la piètre opinion que j'ai de cet administrateur, dont le nom n'est, en ces circonstances, pas même digne d'être cité ici.

P.-S. 2 : le secrétaire adjoint de l'association, Maxime CHUPIN, a publié un article sur l'association dans deux revues de la communauté de la recherche mathématique française, la Gazette des Mathématiciens et Matapli [4]. Autrement plus constructif que celui cité dans le post-scriptum précédent, cet article [5] a été lu avec plaisir par une mathématicienne reconnue, qui a pris la plume pour nous dire tout le bien qu'elle pensait de l'ouvrage de Raymond SÉROUL, préfacé par Dominique FOATA, *Le Petit Livre de TEX*, publié en 1989, ré-édité en 1992, dans lequel notre association figure en bonne place. Nous sommes très heureux de la postérité de cet ouvrage, travaillons à la pérennité de l'association et espérons voir ses travaux actuels toujours appréciés dans une trentaine d'années !

COMPTE RENDU DES CONFÉRENCES DE LA JOURNÉE GUTENBERG

Comme cela avait été annoncé, la Journée GUTenberg a été l'occasion de suivre les conférences données par trois contributeurs « francophones non français » à TEX (et logiciels annexes) :

1. *Literate programming* avec `Org mode` (Fabrice NIESSEN)
2. Nouvelle gestion du format PDF par LATEX (Ulrike FISCHER)
3. Courte introduction aux packages TikZducks et TikZlings (SAMCARTER)

1. *Comprehensive TEX Archive Network* [anglais] : réseau complet d'archives TEX.
2. Il s'agit d'une version pour lecture sur écran, dont le fond des pages est très légèrement coloré, et d'une version sur fond blanc pour ceux qui préfèrent imprimer les fichiers avant de les lire. Mais d'autres éléments différencient les deux versions de la *Lettre* et sont proposés à votre sagacité.
3. Conseil d'Administration.
4. Maxime CHUPIN est rédacteur en chef adjoint de la revue Matapli.
5. Cet article sera mis en ligne en octobre sur le site de Matapli.

Ceux de nos lecteurs qui n'auraient pu y assister en trouveront ci-après un compte rendu assez détaillé (sauf pour la dernière conférence) et, s'ils souhaitent en savoir davantage, pourront visionner leurs enregistrements, par exemple à l'adresse suivante :

https://tubedu.org/video-channels/gutenberg/videos.

LITERATE PROGRAMMING AVEC ORG MODE

Fabrice Niessen, de Leuven en Belgique, est informaticien. Il a participé au stage LaTeX à Dunkerque d'abord en tant que stagiaire, ensuite à plusieurs reprises en tant qu'intervenant sur deux aspects d'Emacs : l'édition de documents LaTeX et le mode majeur Org.

C'est ce second aspect qu'il a développé cette fois-ci en nous donnant un aperçu de certaines des possibilités offertes par Org mode :

1. l'édition simple à l'aide d'une syntaxe de type Wiki ou Markdown ;
2. la programmation lettrée qui comprend :
   (a) l'export en des fichiers notamment .tex (avec éventuelle génération du .pdf correspondant), .html, .odt ;
   (b) l'extraction de blocs de code source et génération de code « entrelacé » ;
3. l'exécution de programmes dans n'importe quel langage au sein même du fichier Org mode.

Sa conférence a été articulée selon ces trois aspects.

Syntaxe

Directives où l'on a vu que tout fichier Org mode commence par des directives qui sont l'équivalent du préambule d'un fichier .tex et qui permettent par exemple de spécifier l'intitulé et l'auteur du document mais aussi, pour l'export, la numérotation ou pas des sections, la présence ou pas d'une table des matières.

Certaines directives, spécifiques à l'export LaTeX, permettent de spécifier par exemple la classe (autre que article, par défaut) souhaitée.

Cycle des vues de la structure où l'on a appris qu'un document Org mode peut être structuré en l'équivalent des sections, sous-sections, etc., introduites au moyen des caractères (à placer en début de ligne) *, **, etc. (ou Alt+↵). Une fois cela fait, Org mode permet d'aisément (⇧+⇆) boucler entre 3 vues de la structure :

1. celle de (très) loin, où ne sont affichées que les rubriques de plus haut niveau ;
2. celle de loin, où ne sont affichées que les rubriques, mais toutes les rubriques, quel que soit leur niveau ;
3. celle « normale » où est affichée l'intégralité du contenu du document.

Mise en forme où l'on a vu comment, au moyen de simples caractères tels que *, /, =, ~, etc. (pouvant être masqués), on pouvait mettre du texte en gras, en italique, en *verbatim*, en « code » (en vue d'un export, par exemple \lstinline plutôt que \verb), etc.

Listes où l'on a appris combien il était simple de créer les équivalents des listes itemize, enumerate, description et d'en déplacer les « items » :

— soit horizontalement pour en modifier la profondeur ;

— soit verticalement pour en modifier l'ordre ;

et ce, accompagnés ou pas de leurs descendants.

Tableaux dont on a découvert à quel point ils étaient simples à créer, à modifier (par exemple pour déplacer des lignes ou des colonnes), à remplir (en créant des suites de nombres, en insérant des formules permettant des calculs, etc.), à clarifier (en alignant les colonnes après édition), et qu'ils offraient certaines des fonctionnalités pratiques des tableurs. On a pu aussi constater que tout peut être dynamique dans le contenu d'un fichier `Org mode`, notamment par la possibilité de chaîner les valeurs entre différents tableaux.

Programmation lettrée

La « programmation lettrée » (*Literate programming*), approche de la programmation préconisée par Donald E. Knuth, permet « […][d']obtenir, à partir d'un fichier source, deux représentations : l'une utilisée par un compilateur ou un exécutable, le code "entrelacé", et l'autre lue comme une documentation formatée, qui est dite "tissée" à partir de la source lettrée. » [1]

L'idée de la programmation lettrée est donc la suivante : plutôt que de documenter un fichier de code informatique au moyen de commentaires, créer un fichier :

— pouvant être exporté en une documentation exprimée dans le langage naturel d'un humain ;

— entrelacé de morceaux de codes qui, *in fine*, en seront extraits pour être regroupés dans un ou plusieurs fichiers de code (seulement), exploitables par l'ordinateur.

Ainsi que l'a noté Fabrice, sont donc à distinguer ici deux concepts :

***weave* :** *exporter* le fichier `Org mode` en entier comme documentation « tissée », formatée pour l'homme, généralement en LaTeX ou en HTML [6] ;

***tangle* :** *extraire* les blocs de code source et générer le code « entrelacé », formaté pour la machine, pour compilation ou exécution ultérieure ;

avec possibilité de changer l'ordre des blocs de code source.

Export

Fabrice a commencé par montrer comment exporter tout ou partie d'un document `Org mode` contenant les éléments vus précédemment :

— en un fichier LaTeX, éventuellement « publié » c'est-à-dire dont la compilation et éventuellement l'affichage du PDF [7] résultant sont lancés en sous-main ;

— en un fichier HTML, par défaut dépourvu de CSS [8] mais pouvant aisément s'en voir adjoindre, ce dont il a fait la démonstration au travers de deux thèmes HTML qu'il a conçus :

1. Bigblow ;
2. ReadTheOrg.

Extraction

Pour illustrer le « *tangling* », Fabrice a pris l'exemple d'un fichier `Org mode` contenant, en plus d'un laïus de *documentation*, un bloc de code LaTeX, en l'occurrence celui d'un quasi ECM [9]. Au moyen d'une directive passée à ce bloc de code indiquant qu'il devait être extrait et spécifiant dans quel fichier, une commande `Org mode` d'export a en effet généré ledit

6. *HyperText Markup Language* [anglais] : langage de balises pour l'hypertexte.
7. *Portable Document Format* [anglais] : format de document portable.
8. *Cascading Style Sheets* [anglais] : feuilles de style en cascade.
9. Exemple Complet Minimal.

fichier contenant l'intégralité du bloc de code. Il a ensuite montré qu'il était possible de scinder ce bloc en plusieurs sous-blocs, permettant ainsi de raffiner la *documentation* en insérant des explications entre chacun d'eux. Ceux qui connaissent les fichiers `.dtx` de LaTeX voient certainement la similarité des méthodes.

L'extraction se fait *a priori* dans l'ordre chronologique du document mais Fabrice a montré qu'il pouvait en être autrement et que les sous-blocs de code précédents pouvaient être nommés pour être extraits dans un ordre arbitraire. Cela suppose de créer un bloc de code supplémentaire :

— contenant (entre autres) les noms des blocs dans l'ordre souhaité, chacun entre balises spéciales permettant de faire comprendre à `Org mode` qu'ils sont à interpréter comme des références ;

— équipé d'une directive supplémentaire indiquant donc à `Org mode` de ne pas extraire verbatim les noms de code entre balises mais de les interpréter comme étant des références et de, à la place, injecter dans le fichier cible le contenu de chacun de ces blocs.

Cette directive a pour nom `noweb`, suggéré par Fabrice aux développeurs de `Org mode` en souvenir du package TeX éponyme qu'il utilisait du temps où il utilisait encore directement LaTeX.

Fabrice a en outre montré que si, dans le bloc de code contenant ces *références* (noms d'autres blocs entre balises spéciales), un texte (arbitraire) figure devant une telle référence, celui-ci est lors de l'extraction ajouté au début de chaque ligne du bloc *référencé*.

Exécution de programmes

Il s'agit ici de faire figurer dans le fichier `Org mode` des blocs de codes, de programmes informatiques dans à peu près n'importe quel langage et de demander l'exécution de ces programmes, leurs résultats pouvant être visibles non seulement dans le *buffer*[10] mais aussi également, si souhaité, dans les fichiers par exemple LaTeX ou HTML exportés. Il est en outre possible de chaîner les résultats des programmes de blocs différents, y compris s'ils sont programmés dans des langages différents ; cela peut notamment se faire dans un bloc de code LaTeX qui peut donc ainsi contenir des résultats de calculs effectués ailleurs dans le fichier `Org mode`, en pouvant choisir localement les valeurs initiales des variables impliquées !

Fabrice a terminé sa conférence par la démonstration d'un bloc de code `R` permettant la création d'un graphique, qu'il a pu afficher dans le *buffer* et qui, par exemple, pourrait possiblement l'être dans le fichier PDF après export en LaTeX.

Références

[1] *Programmation lettrée*. In : *Wikipédia*. 12 juin 2021. URL : https://fr.wikipedia.org/wiki/Programmation_lettr%C3%A9e.

NOUVELLE GESTION DU FORMAT PDF PAR LaTeX

Ulrike Fischer, de Mönchengladbach en Allemagne, est autrice de plusieurs packages LaTeX. Elle est aussi assez connue des dinosaures francophones LaTeX puisqu'elle répond très régulièrement à des questions sur le *newsgroup* `fr.comp.text.tex`. Mais elle répond en fait à des questions LaTeX un peu partout, à tel point qu'elle a été mise à l'honneur lors de la

10. « Tampon » temporaire contenant une copie de travail d'un fichier.

conférence du TUG [11] en 2015 à Darmstadt, au motif que tout le monde a au moins une fois dans sa vie reçu de l'aide d'Ulrike ! Depuis 2 ans et demi, elle est membre de la *LaTeX team*, ce qui témoigne de son niveau d'expertise. Elle qui a appris le français enfant alors qu'elle vivait en Suisse, a donné sa conférence dans un français tout à fait compréhensible.

Ulrike nous a donc présenté la nouvelle gestion du format PDF par LaTeX dont elle est experte puisqu'elle assure la maintenance et le développement des packages hyperref, bien connu de tous, et tagpdf qui a été créé pour expérimenter (avec pdfLaTeX et LuaLaTeX) certaines exigences des PDF « accessibles », notamment le *tagging* (marquage).

LaTeX permet de produire des fichiers de sortie dans de nombreux formats, mais le format PDF est actuellement le plus utilisé et le plus important. De façon surprenante, le noyau LaTeX contient très peu de code relatif à PDF et, pour la gestion de besoins basiques tels que les liens hypertextes, la couleur, l'inclusion de graphiques, la rotation des pages à l'écran, etc. il faut recourir à des packages.

Ulrike nous a exposé tous les problèmes que cela engendre :

— les packages en question peuvent entrer en conflit, notamment du point de vue de ce qu'ils font relativement à PDF ;

— un fichier PDF peut être obtenu via de nombreux biais, par exemple pdfLaTeX, LuaLaTeX et LaTeX + DVIPS [12] + `ps2pdf`. Tous ces moteurs et *backends* recourent à des commandes différentes pour faire les mêmes choses ; en outre, les packages qui manipulent le PDF doivent maintenir du code (des fichiers pilotes) propre à chaque moteur. Comme les moteurs et les packages évoluent, cela pose de sérieux problèmes de maintenance.

Cela a certes fonctionné jusqu'ici mais les exigences évoluent :

1. l'impression sur papier n'est plus l'objectif principal (écran, formulaires) ;
2. les normes PDF sont de plus en plus requises, par exemple la norme PDF/A pour l'archivage des documents, exigée par les services d'archives ainsi que par un nombre croissant d'universités ;
3. l'accessibilité devient un enjeu important ;
4. il est de nos jours fréquent que les utilisateurs souhaitent exporter de PDF vers d'autres formats, par exemple HTML ;
5. les métadonnées sont de moins en moins négligées.

Cela a conduit au projet d'envergure (sur 3 ou 4 ans) « *Tagged PDF* », mené par la *LaTeX team* en collaboration avec la *PDF Association* et Adobe. Ce projet, qui a pour ambition de régler les problèmes vus précédemment, nécessite *beaucoup* plus de code spécifique au PDF dans le noyau LaTeX ainsi que de meilleures interfaces.

Pour cela, l'équipe a développé du code s'attachant à résoudre certains des problèmes évoqués ci-dessus. Ulrike nous en a fait le tour du propriétaire que voici.

Objectif n° 1 : résolution des conflits

Les conflits visés ici sont ceux entre les commandes propres à chaque moteur ou *backend* alors qu'elles ont le même but. Par exemple pour le moteur pdfTeX, les commandes sources de conflits sont les primitives `\pdfinfo`, `\pdfcatalog`, `\pdfpageattr`, `\pdfpagesattr`, `\pdfpageresources`.

11. *TeX User Group* [anglais] : groupe (international) d'utilisateurs de TeX.
12. *DVI-to-PS (translator)* [anglais] : (convertisseur) DVI vers PS.

Pour résoudre ces conflits, la commande `\pdfmanagement_add:nnn` a été créée ; il s'agit d'une commande expl3 [13] qui est donc *a priori* destinée davantage aux auteurs de *packages* qu'aux auteurs de *documents* (quoi qu'une commande LaTeX à eux destinée pourrait être fournie ultérieurement). Cette commande doit désormais être utilisée à la place des commandes ci-dessus. Par exemple au lieu de :

```
\pdfinfo{/Title (Titre A)}
```

il est recommandé d'utiliser :

```
\pdfmanagement_add:nnn {Info} {Title} {Titre A}
```

En plus d'éviter les conflits, on bénéficie alors d'autres fonctionnalités, par exemple la prévention des doublons dans les métadonnées du fichier PDF.

Cela entraîne toutefois des dommages collatéraux : les auteurs de certains packages doivent adapter leur code, et il en est de même pour certains packages (ou classes) s'appuyant sur les premiers.

Mais les auteurs de *documents* sont aussi concernés : lorsque le code source de ces packages a été retravaillé, la LaTeX *team* a dû faire le choix, nécessairement unique, du codage d'entrée et c'est l'UTF-8 qui a été choisi. En effet, celui-ci présente de nombreux avantages sur les autres ; par exemple, si des caractères accentués figurent en argument de `\section`, le fichier `.toc` va les contenir inchangés tandis que, avec les autres codages d'entrée, il sera truffé de scories (telles que `\IeC {\'e}`). Or, dans le cadre d'un *tagged* PDF, les informations telles que les intitulés de sections doivent justement être écrites dans le PDF. Les auteurs de documents qui utiliseront ces packages retravaillés dans des fichiers sources `.tex` utilisant des codages d'entrée autres que l'UTF-8, par exemple `latin1`, s'exposent donc à plus ou moins long terme à des dysfonctionnements.

Du fait de ces nombreux changements et incompatibilités, la LaTeX *team* a mis en place un filet de sécurité : le nouveau code n'est pas encore dans le noyau LaTeX, mais dans le package temporaire pdfmanagement-testphase à charger et à activer explicitement comme suit :

```
\RequirePackage{pdfmanagement-testphase} % Chargement
\DeclareDocumentMetadata % Activation de la gestion du PDF
{
  ⟨options⟩
}
%
\documentclass{...}
```

c'est-à-dire notamment avant `\documentclass`. Si un *backend* non automatiquement détectable par LaTeX doit être utilisé, il doit être explicitement déclaré, par exemple ainsi :

13. Reconnaissable à son nom qui contient un ou plusieurs _ et un :. Une telle commande n'est utilisable que si elle est précédée de la bascule `\ExplSyntaxOn` qui modifie le code de catégorie des deux précédents caractères afin qu'ils soient considérés comme des « lettres ». On rétablit le régime habituel au moyen de la bascule `\ExplSyntaxOff`. Entre ces deux bascules (dont on notera la similitude avec `\makeatletter` et `\makeatother`), les espaces et changements de lignes sont ignorés en ce sens qu'ils ne provoquent pas d'espaces dans le document final, ces derniers étant obtenus au moyen de ~.

```
1 \DeclareDocumentMetadata{backend=dvipdfmx}
```

Objectif n° 2 : meilleur support des *backends*

Cela nécessite une meilleure abstraction des différents moteurs et *backends*. Par exemple, pour insérer de la couleur dans un PDF, de nombreuses primitives internes propres doivent être mises en jeu, notamment :

- `\pdfcolorstack` avec le moteur pdfLaTeX ;
- `\pdfextension colorstack` avec le moteur LuaLaTeX ;
- `\special{color push ...}` avec le *backend* DVIPS ;
- `\special{pdfcolorstack ...}` avec le moteur XeLaTeX.

Heureusement pour les utilisateurs finaux, ils n'ont pas à se plonger dans ces arcanes et se contentent d'utiliser une « abstraction » (`\color`) qui s'occupe de ces détails. La LaTeX *team* souhaite offrir davantage de telles abstractions, les intégrer aux noyaux LaTeX et ainsi éviter aux auteurs de packages d'avoir à coder et maintenir des fichiers pilotes pour gérer toutes ces primitives propres aux moteurs et *backends*.

L'intégration dans le noyau LaTeX de telles abstractions a déjà commencé :

- en février 2020 : expl3 et l3backend ;
- au début 2021 : l3pdf et l3color ;
- en mars 2021 : pdfmanagement-testphase avec :
  - le l3backend-testphase fournissant davantage de commandes pour les *backends* ayant vocation à intégrer le noyau LaTeX ;
  - les modules :
    - l3pdfannot : annotations, liens hypertextes ;
    - l3pdfxforms : « *Xobjects* » (sortes d'images dans les fichiers PDF) ;
    - l3pdffile : incorporation de fichiers ;
    - l3pdffield : formulaires ;

    dont les commandes fonctionnent dans tous les modules.

La création de ces commandes abstraites n'a rien de banal car les commandes propres à chaque moteur ne diffèrent pas que par leur nom, mais aussi par leurs actions pas toutes identiques. Par exemple, la commande `\pdf_version_gset:n` qui permet de fixer la version du PDF à créer, ne fonctionne actuellement qu'avec pdfLaTeX, LuaLaTeX et XeLaTeX, et notamment pas avec DVIPS ou `dvisvgm`.

Exemples d'utilisation/d'utilité

Transparence

Le package transparent, qui permet d'insérer du texte en transparence (éventuellement à cheval sur plusieurs pages), ne fonctionne qu'avec pdfLaTeX et LuaLaTeX. Un nouveau package, transparent-ltx, le remplace si pdfmanagement-testphase est utilisé.

Ulrike nous a montré le code du premier qui n'est pas très long et qui est modérément complexe [14]. Pour placer des éléments dans `/ExtGState` [15], il fait usage des commandes primitives de pdfLaTeX et LuaLaTeX qui posent problème et ne doivent plus être utilisées.

14. NDLR Selon elle, en tout cas !
15. Dictionnaire PDF *external graphics state* chargé de placer des éléments graphiques sur la page.

Pour nous permettre de comparer, Ulrike nous a également montré le code du deuxième, significativement plus court (environ 25 lignes contre 150) et plus simple : il ne définit que deux commandes publiques mais s'appuie sur une « abstraction », en l'occurrence la commande `\opacity_select:n` définie dans le package expérimental l3opacity. Ce dernier est utilisable et fonctionne partiellement avec XeLaTeX et DVIPS mais ces moteurs vont nécessiter des adaptations pour assurer une prise en charge complète.

Couleurs

On peut désormais colorer son document sans devoir charger de package dédié. Parmi les commandes couramment utilisées analogues à celle du package xcolor, on trouve :

```
% Équivalents de \color (bascule)
\color_select:n {⟨(expression de) couleur⟩}
\color_select:nn {⟨modèle(s)⟩} {⟨valeur(s)⟩}
% Équivalent de \colorlet
\color_set:nn {⟨nom⟩} {⟨(expression de) couleur⟩}
\color_set:nnn {⟨nom⟩} {⟨modèle(s)⟩} {⟨valeur(s)⟩}
```

utilisables donc par exemple comme suit :

```
\documentclass{article}
\begin{document}
\ExplSyntaxOn
\color_select:n {red} Rouge. \par
\color_select:n {cyan!50!magenta!10!yellow} Mélange~ de~ couleurs. \par
\color_set:nnn {europablue} {RGB} {0,33,99}
\color_select:n {europablue} Bleu~ d'Europe.
\ExplSyntaxOff
\end{document}
```

On peut aussi créer de nouveaux modèles par la commande `\color_model_new:nnn`, par exemple une « séparation », ce qui permet de faire figurer dans le PDF les informations nécessaires à l'imprimeur pour le choix de l'encre à utiliser (il s'agit de ce qu'on appelle les couleurs d'accompagnement). Jusqu'ici, seul le package colorspace de Javier Bezos fournissait cette fonctionnalité ; désormais, cela est fourni d'emblée (du moins avec l'actuel package pdfmanagement-testphase) :

```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{uncompress}
\documentclass{article}
\begin{document}
Bla bla...

\ExplSyntaxOn
\color_model_new:nnn {BarTone} {Separation} {
  name = BarTone~ 555~ GN ,
  alternative-model = cmyk ,
  alternative-values = { 0.1, 0.2, 0.3, 0.4 }
}
\color_select:nn{BarTone}{1} BarTone \par
\color_select:nn{BarTone}{0.5} BarTone
\ExplSyntaxOff
```

```
\end{document}
```

et Ulrike nous a montré que les informations correspondantes se retrouvaient bien dans le fichier PDF [16] :

```
6 0 obj
[ /Separation /BarTone#20555#20GN /DeviceCMYK 5 0 R ]
endobj
```

et :

```
/color1 cs 1.0 scn /color1 CS 1.0 SCN
 0 -11.955 Td [(BarT)83(one)]TJ
/color1 cs 0.5 scn /color1 CS 0.5 SCN
 0 -11.955 Td [(BarT)83(one)]TJ 154.421 -543.96 Td [(1)]TJ
```

Rotation des pages

Lorsqu'un élément du document est trop large pour tenir dans la largeur de l'empagement, une solution est de le faire pivoter. Dans ce cas, on souhaite que, lors de la visualisation à l'écran, la page contenant cet élément pivote elle aussi. Cela est simple à réaliser, sauf pour les éléments flottants (notamment les figures et les tables) car il est difficile de prévoir les pages sur lesquelles ils se trouveront *in fine* et qui devront (elles seules) être pivotées. Avec pdfLaTeX, c'est possible mais pas simple ; avec le nouveau code, cela fonctionne d'emblée grâce à la commande `\pdfmanagement_add` [17] :

```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{}
\documentclass[french]{article}
\usepackage[T1]{fontenc}
\usepackage{kantlipsum}
\usepackage{rotating}
\usepackage{babel}
\begin{document}
\kant[1]

\begin{table}[t]
  \centering
  \fbox{xxxxxx}
  \caption{Un tableau}
\end{table}

\kant[1-5]

\begin{sidewaystable}
  \ExplSyntaxOn
  \pdfmanagement_add:nnn {ThisPage} {Rotate} {90}
  \ExplSyntaxOff
  \begin{tabular}{l}
```

16. Celui-ci étant décompressé, on peut examiner au moins une partie de son contenu en l'ouvrant dans un éditeur de texte.

17. Pour voir la différence, compiler ce fichier avec la ligne 21 décommentée, puis commentée.

```
    \hline
    Un tableau trop large ne pouvant être visualisé qu'en format
     à l'italienne (plus connu sous le nom de mode paysage) \\
    \hline
  \end{tabular}
  \caption{Un autre tableau}
\end{sidewaystable}
\end{document}

\kant
```

Package hyperref

À l'occasion de son travail sur la nouvelle gestion du format PDF, Ulrike modifie également hyperref. Parmi les modifications qu'elle y a apporté, le remplacement de certains fichiers pilotes propres à certains moteurs (`hpdftex.def`, `hluatex.def`, `hxetex.def`, `hdvips.def`, `pdfmark.def`) par un fichier unique : `hgeneric-testphase.def`. Cela présente de nombreux avantages de maintenance puisque ça n'est plus qu'à un seul endroit que doivent être :

- implémentées les nouveautés ;
- corrigées les erreurs.

Par ailleurs, Ulrike en a profité pour procéder à un certain nombre de changements, par exemple :

- les couleurs des liens hypertextes, qui faisaient l'objet de fréquentes plaintes au motif qu'elles n'étaient pas « belles ». Maintenant, elles le sont davantage ;
- les propriétés des liens hypertextes peuvent être modifiées de façon plus fine. Par exemple, il est possible de masquer les liens de type URL [18] seulement ;
- les formulaires PDF, objets de nombreux bogues, dont le code a été revu et amélioré, à commencer par celui des cases à cocher. Il est désormais possible de les créer avec des *appearances*, c'est-à-dire par exemple des images indiquant leurs différents états ; nous en avons eu la démonstration avec des images d'animaux du package TikZlings (dont il est question ci-après).

Conteneur pour les propriétés des documents

Il est apparu important que les documents contiennent davantage de données sur leurs propres propriétés. Pour cela, ont été ajoutées les commandes suivantes utilisables directement dans LaTeX et par les auteurs de packages :

- `\AddToDocumentProperties` ;
- `\ShowDocumentProperties` ;
- `\GetDocumentProperties`.

Un exemple d'utilisation est le suivant [19].

```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{
  pdfversion = 2.0,
  pdfstandard = A-1b
}
```

18. *Uniform Resource Locator* [anglais] : localisateur uniforme de ressource.
19. Ce code corrige celui présenté par Ulrike, auquel il manquait les « labels » `geometry/` et `top-level/`.

```
\documentclass{article}
\usepackage{hyperref}
\hypersetup{
  pdfauthor = Ulrike Fischer,
  pdftitle = Conteneur pour les propriétés du document
}
% Enregistrement de données supplémentaires
\AddToDocumentProperties[geometry]{paper}{A4}
\AddToDocumentProperties{topic}{Demo}
\begin{document}
\ShowDocumentProperties
\GetDocumentProperties{geometry/paper}
\GetDocumentProperties{top-level/topic}
\end{document}
```

Prise en charge des normes

L'une des normes importantes est PDF/A, déjà mentionnée. Pour implémenter celle-ci, il est nécessaire d'incorporer un profil de couleurs. On trouve sur Internet des fragments de code permettant cela, mais ils sont tous dépendants d'un certain moteur. Le package pdfmanagement-testphase en offre une syntaxe unifiée sous deux formes :

— une d'assez bas niveau permettant un contrôle fin mais nécessitant de fournir des instructions PDF ;

— une de plus haut niveau :

```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{pdfstandard = A-2b}
\documentclass{article}
\usepackage{hyperxmp}
\usepackage{hyperref}
```

Réinsertion des liens

Si un fichier PDF est inclus dans un autre au moyen de la commande `\includegraphics`, ses liens hypertextes et plus généralement ses annotations sont perdus car supprimés du fichier principal. Le package pax de Heiko Oberdiek permet de résoudre le problème mais ne fonctionne qu'avec pdfLaTeX et nécessite une application Java externe s'appuyant sur une librairie désormais obsolète.

À nouveau, Ulrike a profité de son travail sur la nouvelle gestion du format PDF pour également créer un nouveau package, newpax, qui règle ces problèmes. Celui-ci fonctionne avec tous les moteurs et s'appuie sur Lua qui est fourni avec toutes les distributions TeX. Pour tester cela, on pourra commencer par sauvegarder le code suivant dans un fichier `test-lien.tex` :

```
\documentclass{article}
\usepackage[paperheight=.5cm,paperwidth=4cm,margin=0cm]{geometry}
\pagestyle{empty}
\usepackage{hyperref}
\begin{document}
\href{https://www.gutenberg.eu.org/}{Lien vers Gutenberg}
\end{document}
```

le compiler avec le moteur de son choix puis inclure le PDF résultant dans un autre fichier, ainsi :

```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{uncompress}
\documentclass{article}
\usepackage[paperheight=1.25cm,paperwidth=5cm,margin=0cm]{geometry}
\pagestyle{empty}
\usepackage{hyperref}
\usepackage{newpax}
\ifluatex
% Chargement du code Lua
\directlua{require(newpax)}
% Écriture des fichiers `.newpax' pour newpax.sty
\directlua
{
  newpax.writenewpax(test-lien)
}
\fi
\begin{document}
\includegraphics{test-lien}

\newpaxsetup{addannots=false}

\includegraphics{test-lien}
\end{document}
```

On commencera par compiler ce dernier une première fois au moyen de LuaLaTeX *nécessairement*, puis ensuite avec le moteur de son choix.

Commentaires et perspectives

Pour conclure, Ulrike insiste sur le fait que ce projet a certes pour but initial les *tagged PDF* mais qu'il devrait également permettre d'améliorer significativement la gestion du format PDF et ainsi aider bien sûr les auteurs de packages, mais également les auteurs de documents.

Ce code modifie beaucoup de choses. Aussi la *LaTeX team* nous prie-t-elle de, même sur des documents « curieux », le tester en faisant précéder `\documentclass` des commandes :

```
\RequirePackage{pdfmanagement-testphase}
\DeclareDocumentMetadata{⟨options⟩}
```

et de faire remonter tout problème rencontré. Cela a bien sûr pour but de minimiser les bogues pour le jour où ce code sera intégré au noyau LaTeX. On pourrait croire que les commentaires ne sont attendus que de ceux qui, avec LaTeX, génèrent des fichiers PDF. Mais ce n'est en fait pas le cas car la *LaTeX team* ne veut pas non plus introduire de bogues dans les autres formats de sortie et flux de travail (SVG [20], TeX4ht, etc.).

Le code est loin d'être terminé : pour les formulaires par exemple, seules les cases à cocher ont été traitées et la prochaine étape est la gestion des champs de texte. C'est un processus d'autant plus long qu'il faut également modifier certains packages tiers et en contacter parfois les auteurs. Ce travail ne devrait pas être achevé avant un an ou deux.

20. *Scalable Vector Graphics* [anglais] : graphiques vectoriels extensibles.

Les personnes souhaitant en savoir davantage peuvent consulter la page https://www.latex-project.org/publications/indexbytopic/pdf/ et consulter les documentations disponibles localement au moyen de :

```
texdoc -l pdfmanagement
```

Pour pouvoir tester ce code nouveau, il est bien évidemment nécessaire de disposer d'une distribution TEX récente, TL [21] 2021 (à jour) ou MiKTeX.

COURTE INTRODUCTION AUX PACKAGES TikZducks ET TikZlings

Samcarter, qui nous vient aussi d'Allemagne où elle est physicienne des astroparticules, est certainement connue de ceux qui fréquentent le site de questions et réponses anglophone https://tex.stackexchange.com sur lequel elle répond fréquemment aux questions posées. Elle est également très présente sur le site https://topanswers.xyz/tex. Mais c'est certainement sur le site francophone similaire https://texnique.fr qu'on peut la remarquer le plus : elle y intervient fréquemment en apportant des réponses à la fois très sympathiques et très savantes. Elle est tellement incollable sur la classe beamer qu'elle en est devenue récemment la principale contributrice. Elle est l'autrice plusieurs packages et, lors de cette conférence, elle nous a fait une courte introduction à deux d'entre eux : TikZducks et TikZlings, fondés sur le package TikZ. Comme elle est en français moins à l'aise à l'oral qu'à l'écrit, elle avait pré-enregistré une vidéo que nous avons diffusée et à l'issue de laquelle elle a répondu en anglais aux questions qui lui ont été posées.

Ce qui suit ne couvre que le début de chacune des deux parties de la conférence car celle-ci est avant tout visuelle [22] et le lecteur est invité à en visionner la vidéo, ne serait-ce que pour ne pas manquer les traits d'humour qui l'émaillent.

TikZducks

Ce package permet de dessiner des canards en caoutchouc, tel que celui de la figure 1 qui

Figure 1 – Un canard en caoutchouc

s'obtient on ne peut plus aisément :

```
\documentclass{standalone}
\usepackage{tikzducks}
\begin{document}
\begin{tikzpicture}
  \duck
\end{tikzpicture}
\end{document}
```

21. *TEX Live* [anglais].
22. Ne pas manquer les animations finales !

En chargeant la librairie Ti*k*Z `ducks`, on peut aussi en insérer en tant que *short picture* (`pic`) :

```
\documentclass{standalone}
\usepackage{tikz}
\usetikzlibrary{ducks}
\begin{document}
\begin{tikzpicture}
  \draw (0,0) pic {duck};
\end{tikzpicture}
\end{document}
```

Ces canards peuvent être personnalisés de multiples façons, par exemple en modifiant les couleurs des différents éléments qui les constituent [23].

code

```
\duck[
  body=blue,
  head=green,
  bill=red,
  eye=yellow
]
```

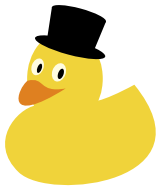
résultat

Ils peuvent également être ornés d'accessoires :

code

```
\duck[tophat]
```

résultat

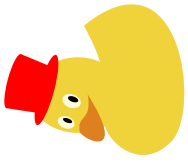


En plus des options spécifiques à ce package, la commande `\duck` admet toutes celles de Ti*k*Z.

code

```
\duck[tophat=red,rotate=90]
```

résultat



Ti*k*Zlings

Ce package, « collection de nombreux animaux adorables », fonctionne de façon analogue à Ti*k*Zducks : il suffit de faire figurer dans un environnement `tikzpicture` la commande portant le nom de l'animal souhaité [24]. Ces commandes acceptent les mêmes options que celles de `\duck` mais sont enrichies de paramètres supplémentaires :

23. Dans la suite, les environnements `tikzpicture` sont sous-entendus.
24. Parmi `\anteater`, `\bear`, `\bee`, `\cat`, `\chicken`, `\coati`, `\elephant`, `\hippo`, `\koala`, `\marmot`, `\mole`, `\mouse`, `\owl`, `\panda`, `\penguin`, `\pig`, `\rhino`, `\sheep`, `\sloth`, `\squirrel`, `\snowman`.

code

```
\penguin
\penguin[xshift=2cm,3D,eye=red]
\penguin[xshift=4cm,back]
```

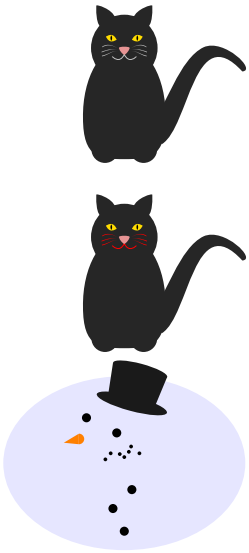
résultat

dont certaines propres à certains personnages [25] :

code

```
\cat[
  schroedinger
]
\cat[
  yshift=-2.5cm,
  whiskers=red
]
\snowman[
  yshift=-5cm,
  globalwarming,
  tophat
]
```

résultat



On appréciera l'humour de SAMCARTER et la qualité des solutions techniques qu'elle utilise pour ses personnages. Et on jettera un œil sur son nouveau package TikZbricks !

Denis BITOUZÉ

COMPTE RENDU DE L'ASSEMBLÉE GÉNÉRALE DE L'ASSOCIATION GUTENBERG

Le 17 avril 2021 a eu lieu l'assemblée générale de l'association (en visioconférence). Les discussions ont pris place sur la plate-forme BigBlueButton, de vive voix et sur le *chat*, et les votes annoncés à l'ordre du jour ont été menés sur l'outil de sondage de BigBlueButton. Les procurations étaient autorisées et ont été comptabilisées *a posteriori*.

Comme lors de la précédente rencontre entre adhérents, la discussion fut riche, foisonnante de débats et ouverte aux attentes de la communauté : le compte rendu ci-dessous tente de lui être le plus fidèle possible, en respectant l'ordre chronologique, tout en se permettant quelques libertés pour réorganiser *a posteriori* les idées échangées. L'auteur de ces lignes prie par avance les personnes citées de bien vouloir l'excuser en cas de mauvaise retranscription de leurs interventions et est bien entendu tout à fait disposée à en discuter [26] et à corriger ou amender publiquement des points dans la prochaine *Lettre* le cas échéant.

Ordre du jour

Après les présentations relatées précédemment, l'assemblée générale proprement dite est ouverte à 14 heures 45 avec une trentaine de personnes en ligne sur la centaine d'adhérents

25. Attention ! L'option `schroedinger` de la commande `\cat` rend l'état du chat imprévisible.
26. Les commentaires sont les bienvenus à l'adresse `secretariat@gutenberg.eu.org`.