La Lettre GUTenberg

Numéro 53 – octobre 2024

Heureuses utilisatrices de Châteauneuf-de-Randon et joyeux utilisateurs du village global,

dans ce numéro, comme à notre habitude, nous présentons l'activité de votre association et tentons de rendre compte de l'activité TEXoïdale, qu'elle soit locale ou d'importation. Nous y relatons quelque conseil d'administration, nous proposons divers bilans d'étape des projets en cours, nous évoquons des avancées, venant de contrées lointaines, qui nous semblent dignes d'intérêt.

Il est temps d'évoquer ici un projet qui a ces derniers temps bien occupé certains membres de l'association, et d'étendre notre propos à un aspect rarement évoqué de notre activité.

Tout d'abord, le projet : c'est celui de la version papier de la FAQ. Non contents d'offrir à la communauté ce merveilleux site, sans cesse mis à jour[1] et dotée de fonctionnalités bien pratiques permettant de la consulter sur des écrans de dimensions variées tout en gardant le

1. Sans cesse mis à jour par Yannick Tanguy, ce Sisyphe des temps modernes, qui cent fois sur le métier remet son ouvrage, et qui est imité par d'autres contributeurs !

Avez-vous pensé à régler votre cotisation ?

Si vous avez oublié, ce n'est ni trop tard ni difficile :
https://www.gutenberg-asso.fr/?Adherer-en-ligne
Voir aussi page 70.

confort nécessaire à une navigation aussi fluide qu'élégante, l'équipe de la FAQ a imaginé d'en proposer une version au format A4. Qui reprend la totalité des questions fréquemment posées (il y en a plus de 1200 !) ainsi que leurs réponses ! Illustrées par des exemples !

L'exigence requise explique que l'ensemble n'est pas complètement finalisé, mais vous pouvez consulter la version en cours ici :

`https://www.gutenberg-asso.fr/IMG/pdf/faqlatexgutenberg.pdf.pdf`

Il est aisé d'imaginer la difficulté d'une entreprise traitant d'un si grand nombre de briques logicielles, mais là n'est pas mon propos : je viens de constater que j'ai rarement parlé du plaisir qu'il y a à œuvrer au sein de l'association et au service de la communauté. Cela s'explique sans doute par la difficulté que j'ai à évoquer ce sujet ; décrire les fonctionnalités d'un package est plus aisé. Il m'est bien sûr arrivé d'évoquer les fructueux échanges TEXniques au sein de l'association… sans guère parler des sourires qui vont avec. Mais ce projet de version imprimée de la FAQ m'en donne l'occasion. Il est hébergé sur le serveur de l'association via `git`, un logiciel décentralisé de gestion de versions. À chaque fois qu'un contributeur enregistre un changement, il décrit celui-ci. Voici quelques-unes de ces descriptions :

> Les sous-questions des questions auto-pas-traitées comme questions
>
> Ne pas sous-entendre qu'il faut réactiver systématiquement
>
> Obligé presque d'utiliser la commande de sectionnement idoine
>
> Annulation des tentatives de ne pas utiliser "dawn" en HTML
>
> Essai très essayant relatif au sauvetage de sphinx-design
>
> Il manquait encore un % chez Philéas Fogg et Passepartout
>
> Encore du peaufinage pourtant il fait beau dehors (mais frisquet)
>
> Méthode plus directe pour imposer lexer latex pour make latex
>
> Revert "Sous-rubriques des questions sous forme interrogative reformulées"

… et la difficulté de la chose se ressent parfois dans l'écriture elle-même :

> Correction d'une erreur matteant le bazat dans les signets (sic)

Je précise que la plupart de ces messages viennent de Jean-François Burnol, un utilisateur que je n'ai jamais rencontré[2]. Mais je le remercie vivement, tant pour l'important travail qu'il fournit que pour le plaisir que j'ai eu à lire ses messages. Quel bonheur que de travailler au sein de l'association !

Nous espérons que vous apprécierez ce numéro, qu'il vous sera utile et que vous aurez du plaisir à le lire… d'autant que nous avons la joie de publier deux articles, l'un de Daniel Flipo consacré à l'indexation, l'autre de Maxime Chupin consacré aux mécanismes de composition des paragraphes… et ces deux textes sont passionnants !

Enfin, retrouvons-nous en présentiel, le 16 novembre prochain à Paris, pour la Journée 2024 ! Son programme est en page ci-contre.

Bonnes compilations !

Patrick Bideault

2. M. Burnol interviendra lors de la journée GUTenberg ; il y présentera son package xint.

JOURNÉE GUTENBERG 2024, LE 16 NOVEMBRE, À PARIS

Nous sommes très heureux de vous annoncer que la journée GUTenberg 2024 aura lieu, en présentiel, le samedi 16 novembre

à l'École normale supérieure (ÉNS)
45 rue d'Ulm - 75005 Paris
en salle Henri Cartan[3].

dont la localisation se trouve sur OpenStreetMap :

https://www.openstreetmap.org/#map=19/48.84171/2.34449

Nous prévoyons d'enregistrer les conférences et de les retransmettre en ligne, en direct, comme l'an dernier.

Journée

Le programme de la journée est le suivant :

09h30-10h00 Accueil et café
10h00-11h00 Exposé sur le système KerTEX par son auteur Thierry Laronde
11h00-12h00 Exposé sur les fontes variables par Jacques André
12h00-14h00 Repas de groupe proposé au restaurant Mauzac (à régler individuellement)
14h00-15h00 Exposé sur le package xint par son auteur Jean-François Burnol
15h00-16h30 Assemblée générale ordinaire
16h30-17h00 Assemblée générale extraordinaire
17h00-17h30 et plus Moment convivial

Même si la participation à la journée est gratuite, il est impératif de vous inscrire grâce au formulaire suivant, et ce, *avant le 3 novembre* :

https://framaforms.org/inscription-a-la-journee-gutenberg-2024-du-16-novembre-1725267106

Attention ! Seules les personnes inscrites pourront rentrer dans les bâtiments de l'ÉNS.

Ce formulaire vous permet aussi d'autoriser ou non la publication de votre inscription à la journée, ainsi que de vous inscrire pour le déjeuner afin que l'association prenne pour vous une réservation au restaurant Mauzac (à régler personnellement cependant).

Nous espérons pouvoir tenir la journée en « hybride », avec une retransmission sur l'usuelle instance BBB, pour permettre une participation la plus large possible. Nous reviendrons vers vous dès que nous nous serons assurés de la possibilité de pouvoir le faire, et nous publierons le lien de connexion quelques jours avant la journée.

Assemblée générale ordinaire

Nous proposons l'ordre du jour suivant pour l'assemblée générale ordinaire :

— Bilan moral, discussion et vote
— Rapport financier 2024 et vote du quitus
— Proposition de budget 2025 et vote
— Proposition de cotisation 2025 et vote
— Sièges à pourvoir au CA : candidatures, discussion et vote
— Le futur de GUTenberg : orientation générale pour les années à venir
— Questions diverses et discussion libre

3. Henri Cartan est un mathématicien français. Voir https://fr.wikipedia.org/wiki/Henri_Cartan.

Comme les statuts l'imposent, il nous faut renouveler le CA de moitié. Les mandats d'une partie de ses membres seront donc exceptionnellement écourtés pour que la moitié arrivent à échéance dès 2024 et que le système puisse fonctionner régulièrement à partir de 2026. En outre, afin de faciliter ce renouvellement par moitié tous les deux ans, les mêmes statuts prévoient que les membres du CA remplaçant des membres démissionnaires ne soient élus que pour la fin du mandat des membres qu'ils remplacent. Cette disposition concerne cette année Bastien Dumont et Yannick Tanguy, élus en 2023 sur les postes occupés jusque-là par Éric Guichard et Christian Hinque, dont le mandat devait se terminer en 2024. La prise en compte de cette disposition nous conduit à modifier ce que nous avions annoncé précédemment sur les listes de diffusion. D'autres membres écourteront leur mandat actuel et sont prêts à se représenter pour un nouveau mandat de quatre ans. Plus d'informations sur ces élections vous seront communiquées dans la convocation à l'AG, qui vous sera adressée dans quelques semaines.

Ces élections sont aussi l'occasion de faire entrer de nouvelles personnes dans le CA pour créer de nouvelles dynamiques. L'association, et en particulier le CA, manque de bras pour les nombreuses activités de l'association : comme indiqué dans les statuts, n'hésitez pas à soumettre votre candidature au secrétariat[4] (au plus tard une semaine avant la tenue de l'assemblée générale) et à la présenter aux adhérents, à l'adresse adherents@gutenberg-asso.fr. N'hésitez pas non plus à envoyer vos questions au secrétariat afin que le bureau puisse préparer les réponses avant l'assemblée générale ordinaire.

Assemblée générale extraordinaire

Suite à l'obtention du rescrit fiscal attestant de la reconnaissance d'intérêt général de l'association, nous devons modifier quelques articles des statuts de celle-ci.

Nous proposons l'ordre du jour suivant pour l'assemblée générale extraordinaire :

— Discussion et vote sur la modification des statuts

Vote

Les deux assemblées générales seront ouvertes à tous, mais seuls les membres à jour de cotisation pourront voter.

Pour permettre la participation la plus large, le vote se fera par le système de vote électronique Bélénios[5]. Les votes par procuration ne seront pas possibles mais les scrutins seront ouverts au moins 48 heures avant le début de l'AG et il sera possible de voter en présentiel sans passer par Bélénios.

Les informations sur la Journée seront mises à jour sur la page dédiée du site web de l'association :

https://www.gutenberg-asso.fr/Journee-GUTenberg-2024

Nous espérons que nous serons nombreux à nous retrouver enfin pour échanger non virtuellement autour de nos logiciels favoris.

Enfin, pour que l'association continue d'exister, elle a besoin d'adhérents. C'est pour cela que nous renouvelons ce récurrent message, dont l'importance est évidente : si vous trouvez l'association utile, adhérez ! Ça se passe ici :

https://www.gutenberg-asso.fr/Adherer-a-l-association

4. secretariat@gutenberg-asso.fr
5. https://www.belenios.org/

EXPOSÉS MENSUELS SUR (LA)TEX ET AUTRES LOGICIELS

Désormais, les exposés en visioconférence GUTenberg sont des rendez-vous mensuels ancrés dans la vie de l'association. Il est donc naturel d'en rendre compte dans la *Lettre*.

Exposés récents

Depuis la *Lettre* 52, l'association a organisé rien moins que quatre exposés.

Fonctionnement de la nouvelle version de la FAQ LaTeX GUTenberg du point de vue du contributeur

Le jeudi 2 mai 2024, à 20 heures, Bastien Dumont et Yannick Tanguy ont exposé le fonctionnement de la nouvelle version de la FAQ.

Ils nous ont présenté les différentes manières de contribuer à la FAQ :

— depuis l'interface en ligne

— ou via `git`, ce qui impose d'installer les sources de la FAQ et de les compiler sur son ordinateur… ce qui se révèle assez puissant !

Vous trouverez plus de détails sur la page du site de l'association dédiée à cet exposé :

https://www.gutenberg-asso.fr/2-mai-2024-Expose-sur-le-fonctionnement-de-la-nouvelle-version-de-la-FAQ-LaTeX

Par ailleurs, nous avons enregistré l'exposé et l'avons mis en ligne sur nos deux canaux de vidéo à la demande :

https://www.youtube.com/watch?v=mh9q4vTu4Ns

https://tubedu.org/w/s4xHNvESqCm8dDwrsDh2dn

Le package overarrows

Le jeudi 6 juin 2024, à 20 heures, Julien Labbé a présenté son package overarrows.

Des informations supplémentaires sont proposées sur la page du site de l'association dédiée à cet exposé :

https://www.gutenberg-asso.fr/6-juin-2024-Expose-sur-le-package-overarrows

Vous pourrez notamment y trouver son support de présentation.

Nous avons enregistré l'exposé et l'avons mis en ligne sur nos deux canaux de vidéo à la demande :

https://youtu.be/bbmnmZC3Cng

https://tubedu.org/w/uHs5wWvTx8MtuMw7fvxDPi

LuaLaTeX, introduction à l'utilisation de quelques *callbacks*

Le jeudi 12 septembre 2024, nous avons écouté Maxime Chupin présenter une introduction à l'utilisation de *callbacks* avec LuaLaTeX.

Plus d'informations sont disponibles sur la page du site de l'association dédiée à cet exposé :

https://www.gutenberg-asso.fr/12-septembre-2024-LuaLaTeX-introduction-a-l-utilisation-de-quelques-callbacks

Nous avons, là encore, enregistré l'exposé et l'avons mis en ligne sur nos deux canaux de vidéo à la demande :

https://youtu.be/4VFhPh6g8ZI

https://tubedu.org/w/j4fQ7NVUNkWE8f9K3wK3h7

`LaTeX-Workshop` transforme Visual Studio Code en un véritable IDE pour LaTeX

Le jeudi 3 octobre 2024, c'est avec intérêt que nous avons écouté Jérôme Lelong présenter le module `LaTeX-Workshop` dont il est mainteneur, intitulé « `LaTeX-Workshop`. Il transforme `Visual Studio Code` en un véritable IDE pour LaTeX ».

Le support de la présentation de cet exposé est disponible sur la page du site de l'association dédiée à cet exposé :

https://www.gutenberg-asso.fr/3-octobre-2024-LaTeX-Workshop-transforme-Visual-Studio-Code-en-un-veritable-IDE

Cet exposé a bien entendu été enregistré. Il est disponible sur nos deux canaux de vidéo à la demande :

sur Youtube : https://www.youtube.com/watch?v=mE_MiYSEjvg ;

sur Tubedu.org : https://tubedu.org/w/3yR59LToCjqB47GkPPd4uP.

Les exposés : un projet participatif

Pour rappel, l'association GUTenberg organise des exposés mensuels sur des sujets bien entendu connexes à (LA)TEX. Il pourra y être question :

- de la gestion de tel ou tel matériel (tableaux, formules de mathématiques, flottants, bibliographie, etc.) ;
- d'utilisation de packages ou de classes ;
- de création de packages ou de classes ;
- de programmation, notamment en LaTeX3 ;
- de typographie ;
- de création de dessins ;
- de polices de caractères ;
- d'éditeurs de texte ;
- de formats autres que LaTeX ;
- de logiciels de gestion de version (par exemple `Git`) ;
- etc.

et ce, sur des thématiques souvent généralistes, appartenant aux sciences humaines comme aux sciences dures. Ces exposés, en visioconférence, sont *a priori* d'une durée d'une heure au plus, questions comprises.

Lorsque nous avons annoncé l'organisation de ces exposés, nous avons bien entendu lancé un appel aux orateurs volontaires (si possible n'étant pas administrateurs de l'association afin de laisser la place à d'autres). Toutes suggestions (exposés par vous-mêmes ou par d'autres personnes, sujets, etc.) sont les bienvenues et sont à adresser au secrétariat de l'association (`secretariat@gutenberg-asso.fr`).

L'ensemble des informations (programmes passé et futur, lien de connexion, etc.) est accessible sur la partie dédiée de notre site internet :

https://www.gutenberg-asso.fr/-Exposes-mensuels-

Agenda numérique

Pour rester facilement informé et ne manquer ainsi aucun exposé, nous avons mis en place un *framagenda* pour les événements GUTenberg accessible à l'adresse :

https://framagenda.org/apps/calendar/p/DnkaimkPTyBDsQ7z/dayGridMonth/2023-09-01

qui fournit (en cliquant sur les paramètres de l'agenda) un lien `webcal` pour la synchronisation avec vos agendas personnels :

webcal://framagenda.org/remote.php/dav/public-calendars/DnkaimkPTyBDsQ7z/?export

Maxime Chupin

COMPTE RENDU DE LA RÉUNION DU CONSEIL D'ADMINISTRATION DU VENDREDI 3 MAI 2024

À trois heures de l'après-midi, en ce vendredi 3 mai 2024, la séance du conseil d'administration est ouverte en visioconférence.

Sont présents : Patrick Bideault (président de l'association), Denis Bitouzé (secrétaire), Maxime Chupin (secrétaire adjoint), François Druel (trésorier), Bastien Dumont (administrateur), Arthur Rosendahl (administrateur), Flora Vern (administratrice).

Sont excusé(e)s : Céline Chevalier (vice présidente), Yvon Henel (trésorier adjoint), Jean-Michel Hufflen (administrateur chargé des Cahiers GUTenberg), Yannick Tanguy (administrateur).

Le quorum étant atteint, le conseil d'administration est ouvert. Patrick Bideault assure le rôle de secrétaire de séance.

Ordre du jour

— Comptabilité et finances ;
— *Cahiers* ;
— Journée 2024 ;
— package CTAN GUTenberg et archives de l'association ;
— conférence TUG 2026 ;
— adhésion à l'April ;
— questions diverses.

Comptabilité de l'association

François Druel dresse un état temporaire des finances de l'association. Le bilan 2024 sera présenté lors de l'AG 2024, en fin d'année.

Comptes courants

L'association dispose de trois comptes courants, à savoir un à la Banque populaire, un à la Société générale et le compte Paypal, dont le fonctionnement s'approche d'un compte courant.

Patrick Bideault propose de fermer le compte de la Banque populaire. Que fait-on des fonds ? Où les investit-on sur la Société générale ? Le CA décide de s'en servir pour augmenter le solde du compte courant hébergé par la Société générale à 5000 euros et d'investir le reste sur le compte titres de l'association.

Finances de l'association

Elles sont globalement à l'équilibre, mais les adhésions sont en baisse. L'équilibre est essentiellement dû au fait que le budget des actions extérieures (typiquement la bourse pour la conférence TUG) n'a pas été utilisé. Nos besoins en serveur étant susceptibles d'augmenter, la situation n'est pas brillante : l'association aurait-elle de quoi financer un serveur plus puissant ?

Fonds de l'association

Le conseil évoque l'utilisation des fonds de l'association. Il faut les utiliser de manière efficace.

Patrick trouve que les fonds ne doivent pas servir à financer le quotidien de l'association mais être utilisés pour des actions exceptionnelles, comme la proposition de bourse pour le TUG : elle a eu du succès l'an dernier et a été utile à Victor Sannier. Elle a modestement contribué à la publication de son package ruscap. Malheureusement, cette bourse n'a pas été attribuée en 2024.

Plusieurs membres pensent que la bourse pour des TUG en Europe est tout à fait justifiée.

Cette bourse, si elle avait été attribuée aurait été prise sur nos fonds. Il faut en avoir conscience.

La bourse peut être prise sur les fruits du placement et donc cela peut relever de la gestion de l'association et ainsi, peut ne pas faire l'objet d'une décision en CA ou en AG.

En revanche, toute utilisation des fonds doit passer par une telle décision collégiale.

Logiciel de comptabilité de l'association

François montre l'utilisation du logiciel `Paheco` qui lui semble plus adapté que `Dolibarr` pour la gestion de la comptabilité de l'association.

Arthur Rosendahl nous dit que Paheco signifie « combinaison », en maori. Le CA salue les compétences linguistiques d'Arthur.

`Paheco` ne semble pas permettre de générer des factures, ce qui est un écueil.

Le conseil discute de la possibilité de migrer sur `Paheco`.

Interface de paiement en ligne

François montre l'utilisation du service de paiement par carte bancaire Stripe qui est un peu moins cher et permet le paiement sur notre site par carte bancaire (malgré le problème d'interface qu'il faut régler).

Publication des coordonnées bancaires de l'association

Patrick propose l'utilisation du récent package epcqrcode dans la *Lettre*, pour permettre aux lecteurs de cotiser en scannant un QR code. Ça permet d'éviter de mettre notre IBAN en clair.

Obtention de la qualification d'intérêt général

François évoque la rédaction de la demande de rescrit fiscal dont il s'occupe. Il prévoit de la déposer en juin-juillet[6].

Les *Cahiers*

Le CA fait le point et constate la non-parution du nouveau numéro des *Cahiers*. Celui-ci est en cours de finition. Patrick fait remarquer qu'`OJS`, le logiciel de gestion de contenu utilisé pour publier la *Lettre* en ligne, est un outil efficace pour de relecture collégiale et pourrait aisément être mis à la disposition du comité de rédaction des *Cahiers*.

6. Cette démarche a été effectuée. Elle est couronnée de succès. Voir page 4.

Par ailleurs, le dernier numéro des *Cahiers* n'est toujours pas en ligne alors que le suivant est presque prêt : le CA demande à Jean-Michel Hufflen les fichiers du *Cahier* 58 pour les mettre en ligne. Il aimerait pouvoir lui venir en aide, par exemple pour la fabrication et l'envoi du *Cahier*.

Ensuite, le CA s'intéresse à la publication de la classe des *Cahiers* : elle doit être publiée sur le CTAN et mise à disposition de l'association sur sa forge logicielle.

Ainsi, la chaîne de production (sur OJS) et la classe (sur la forge) seraient sur le serveur, ceci pour un meilleur fonctionnement, plus simple, collaboratif et amélioré.

La *Lettre*

Le CA se réjouit de la récente parution de la *Lettre* 52 et remarque que sa classe, intitulée letgut, doit aussi migrer sur la forge de l'association.

Journée 2024

Elle aura lieu en novembre. Le CA discute d'un lieu, d'une date et des différents sujets qui pourraient être abordés. Maxime Chupin va s'occuper tant du programme que des modalités pratiques d'organisation.

Dépôt usergroup CTAN de l'association

Le conseil regrette la situation actuelle, qui voit ce dépôt dans les seules mains d'un non-membre de l'association. Il se propose de reprendre contact avec lui et de le prier d'ôter les références à son site personnel, car il n'a pas de lien avec l'association.

Le CA envisage de demander au CTAN la rétrocession du dépôt de la FAQ à l'asso.

Par ailleurs, le conseil désire récupérer ses archives, toujours en possession de son ancien président, qui depuis a quitté l'association. François Druel et Patrick Bideault sont toujours volontaires pour aller les chercher à Lyon. Une *n*–ième prise de contact est envisagée.

TUG 2026

Le CA envisage l'organisation de la conférence TUG en 2026. Elle se propose de prendre officiellement contact avec le TUG, de poser des questions, de parler du projet avec eux. Elle évoque l'expérience de Didier Verna qui a participé à l'organisation de conférences du même type, quoi que dédiées à d'autres logiciels.

Questions diverses

— Relancer les ex-adhérents est évoqué : il s'agit d'en établir la liste et de rédiger un message à leur adresser. Yannick propose de les questionner au moyen d'un sondage.

— L'idée d'organiser en juin un nouveau rendez-vous informel (un « glou », selon son ancienne appellation) est évoquée.

Exposés mensuels

Le conseil échange au sujet des prochains exposés. Ce rendez-vous s'impose peu à peu au sein de la communauté francophone ; c'est un succès qui ne demande qu'à grandir.

Relations avec d'autres associations et salons

— François s'occupe de l'adhésion à l'April[7].

— Patrick soumet au conseil son idée d'ateliers mensuels à Paris, par exemple lors du premier samedi du libre[8], réunion mensuelle libriste, organisée par l'association Parinux[9] autour d'une install partie.

Il se propose de s'y rendre le lendemain. François et Yannick l'y accompagneront.

L'ordre du jour étant épuisé, et nul ne demandant plus la parole, la réunion est levée à cinq heures de l'après-midi passées de cinquante-trois minutes.

Patrick Bideault

COMPTE RENDU DE LA RÉUNION DU CONSEIL D'ADMINISTRATION DU DIMANCHE 8 SEPTEMBRE 2024

À trois heures de l'après-midi, en ce dimanche 8 septembre 2024, la séance du conseil d'administration est ouverte en visioconférence.

Sont présents : Patrick Bideault (président de l'association), Denis Bitouzé (secrétaire), Bastien Dumont (administrateur), Jean-Michel Hufflen (administrateur chargé des Cahiers GUTenberg), Arthur Rosendahl (administrateur), Yannick Tanguy (administrateur).

Sont excusé(e)s : Céline Chevalier (vice présidente), Maxime Chupin (secrétaire adjoint), François Druel (trésorier), Yvon Henel (trésorier adjoint), Flora Vern (administratrice).

Le quorum étant atteint, le conseil d'administration est ouvert. Patrick Bideault, Bastien Dumont et Yannick Tanguy assurent collectivement le rôle de secrétaires de séance.

Campagne d'adhésion

Afin d'enrayer la baisse du nombre d'adhérents, Yannick Tanguy propose de diffuser des sondages à destination des anciens adhérents. Il est également question d'interroger des utilisateurs de LaTeX qui n'ont jamais adhéré à l'association par l'intermédiaire de différents sites et listes de diffusion.

Renouvellement du CA

Les statuts imposant un renouvellement du CA par moitié tous les deux ans, six membres du CA doivent présenter leur démission ou se présenter pour un nouveau mandat de quatre ans, alors même qu'aucun mandat n'est actuellement échu. Les personnes qui démissionnent ou qui remettront leur mandat en jeu feront connaître leur intention par mail aux adhérents avant la journée GUTenberg du 16 novembre 2024. Un appel à candidatures pour le CA doit être diffusé prochainement.

Une discussion s'engage les statuts actuels, qui imposent le renouvellement par moitié du CA tous les deux ans même si moins de la moitié des mandats sont échus, ce qui implique cette année que des membres du CA se portent volontaires pour écourter leur mandat. Cependant,

7. https://april.org/
8. https://samedis-du-libre.org/
9. https://parinux.org/

la situation actuelle a été provoquée par le fait que, en 2022, tous les membres du CA élus en 2020 sur la liste « GUTenberg-Continuons » ont choisi de remettre leur mandat en jeu. Il s'agit donc d'un cas exceptionnel qui ne devrait plus se reproduire. De plus, les statuts prévoient bien que les membres du CA élus pour remplacer des administrateurs démissionnaires ne sont élus que pour la durée restante du mandat de ceux-ci. Cette disposition, qui concerne actuellement Bastien Dumont et Yannick Tanguy élus en 2023 sur deux mandats entamés en 2020, garantira à partir de 2026 que la moitié des mandats soient échus tous les deux ans.

Denis Bitouzé soulève cependant la question de la fréquence des élections, aujourd'hui tous les deux ans. Il trouve que c'est lourd à organiser, en particulier à cause du vote électronique. Cependant, à la suite de Jean-Michel Hufflen, la plupart des participants à la réunion conviennent que ce renouvellement par moitié a l'avantage d'assurer aux mieux le tuilage entre les anciens et les nouveaux membres du CA.

Renouvellement des activités de l'association

En réaction à une préconisation de François Druel[10], une discussion s'engage sur le renouvellement des activités de GUTenberg, que l'on pourrait estimer nécessaire pour entretenir ou relancer le dynamisme de l'association. Denis Bitouzé souligne que l'association fait déjà beaucoup de choses et que, en l'état actuel des forces, lancer de nouveaux projets pourrait conduire à abandonner certaines des activités existantes, ce qui n'est pas souhaitable. Néanmoins, Yannick Tanguy indique plusieurs possibilités :

— Avancer sur les traductions, notamment le manuel de la classe memoir. Denis Bitouzé signale que cette entreprise permettrait d'impliquer de nouvelles personnes dans les projets de l'association, mais qu'il faudrait peut-être pour ce faire utiliser un outil de collaboration plus accessible que `Git` et plus fiable qu'`Overleaf`. Il faudrait se renseigner auprès d'autres associations qui ont mené à bien des projets similaires ;
— Afin d'améliorer l'accessibilité des documents en couleur aux personnes daltoniennes, créer une extension complémentaire de colorblind qui modifierait les couleurs du document pour permettre à son auteur de le visualiser tel que ces personnes le verrait. En effet, l'extension colorblind permet actuellement de générer un jeu de couleurs adapté au daltonisme, mais pas de visualiser la manière dont un document en couleurs existant pourrait être perçu ;
— Développer la présence à des rencontres ou des ateliers comme ceux de Parinux[11]. Patrick Bideault précise cependant que le stand de l'association aux samedis du Libre[12], bien qu'il ait prouvé son utilité pour aider des utilisateurs et pour contribuer à la FAQ, pourrait encore attirer davantage de participants. Denis Bitouzé suggère de communiquer à ce propos pour attirer des personnes qui utilisent des systèmes d'exploitation autres que Linux, et qui ne forment donc pas le public habituel des samedis du Libre.

Patrick Bideault signale également que l'association suscite parfois des travaux : par exemple, Daniel Flipo a pris l'initiative de créer un complément mathématique à la fonte Luciole à la suite de l'article paru à son sujet dans la *Lettre* 52[13].

Dans la même logique, les rencontres au café (anciennement dites « glou ») relancées par Patrick, bien qu'ouvertes à tous, attirent le plus souvent des utilisateurs plus chevronnés qui y trouvent l'occasion de discuter de leurs projets respectifs.

10. préconisation *in absentia*, on l'aura compris à la lecture du préambule.
11. https://parinux.org/
12. https://samedis-du-libre.org/
13. Voir la brève « Une fonte maths pour Luciole », page 67.

Organisation de la journée GUTenberg du 16 novembre 2024

Il est décidé de participer aux frais de déplacement et d'hébergement d'un orateur de la journée qui en a fait la demande. Le repas de midi aura lieu *a priori* dans le même restaurant et selon les mêmes modalités qu'en 2023.

Cotisations

Étant donné que le nombre d'adhérents a tendance à baisser et que le montant de la cotisation n'a pas évolué depuis de nombreuses années, François Druel avait au préalable informé les administrateurs de sa proposition de discuter en AG d'une augmentation, éventuellement en donnant la possibilité de mensualiser le paiement grâce à un prélèvement automatique. Patrick Bideault préconise cependant d'attendre que l'administration fiscale ait approuvé la demande de déclarer l'association GUTenberg d'intérêt général[14], ce qui permettrait aux membres de l'association de compenser cette augmentation par un crédit d'impôt. Yannick Tanguy propose également de créer une cotisation de soutien.

Cahiers GUTenberg

Jean-Michel Hufflen informe le CA que tous les articles du prochain numéro des *Cahiers* GUTenberg devraient être dans leur état définitif avant l'AG. Le numéro comptera entre 115 et 120 pages en couleurs. La dernière AG a décidé que le financement se ferait par souscription, mais il reste à définir le prix et la méthode. Il est convenu de faire le point durant la première semaine d'octobre, puis de lancer une souscription courant jusqu'à l'AG. Denis Bitouzé suggère de publier un échantillon de chaque article sur le site afin de faire de la publicité lors de la période de souscription.

Actuellement, les DOI des *Cahiers* GUTenberg ne sont pas valides. Maxime Chupin tente de régler ce problème, mais ce n'est pas simple. Néanmoins, les articles sont toujours accessibles sur le site des *Cahiers*[15]. Par ailleurs, Jean-Michel Hufflen signale que, dans l'idéal, il faudrait aussi pouvoir rediriger les adresses des anciens numéros des *Cahiers* (jusqu'au 57) vers la page où il est possible de les consulter actuellement.

Jean-Michel Hufflen donne son accord pour mettre les sources des articles à disposition sur le serveur de l'association.

Lettre

Patrick Bideault propose que l'on publie le numéro courant fin septembre, quitte à publier un numéro exceptionnel début novembre contenant uniquement les bilans moral et financier, qui ne peuvent être rédigés qu'après la clôture des comptes de l'association fixée au 31 octobre.

Questions diverses

Version PDF de la FAQ

Denis Bitouzé fait le point sur la version PDF de la FAQ. Le moteur `Sphinx` utilisé pour générer le site internet de la FAQ est en effet capable de produire en plus un fichier TEX. Jean-François Burnol a considérablement amélioré la conversion vers TEX grâce à son inlassable travail en collaboration avec Denis Bitouzé et, pour la mise en forme des logos, Bastien Dumont.

14. Voir le point « Obtention de la qualification d'intérêt général » page suivante.
15. `https://publications.gutenberg-asso.fr/cahiers`

L'outil est désormais pleinement fonctionnel et le rendu convaincant, si ce n'est que les intitulés des questions correspondent à des niveaux de structuration variés, ce qui impose de trouver des stratégies typographiques pour faciliter leur identification.

Un fichier sera prochainement présenté au CA. Il est envisagé de le publier sur le CTAN.

Obtention de la qualification d'intérêt général

Patrick Bideault informe que la demande de qualification d'intérêt général déposée par François Druel a été reçue le 31 juillet. Elle ouvrirait notamment aux membres la possibilité de demander un crédit d'impôt sur les cotisations. Il préconise de valoriser l'excellent dossier sur l'association préparé par François Druel en le rendant public[16].

Exposés mensuels

Dans l'immédiat, on ne manque pas d'orateurs ni de sujets pour les exposés GUTenberg, mais il devient plus difficile d'en trouver de nouveaux. Le CA sera heureux de recevoir des propositions de présentations de tous niveaux sur tous sujets relatifs à (LA)TEX.

Denis Bitouzé suggère de proposer aux francophones qui participent à une conférence du TUG d'adapter leur présentation dans le cadre des exposés. Jean-Michel Hufflen prévient que les exposés adaptés du TUG 2024 seraient d'un niveau avancé, mais, outre l'avantage de rendre ces présentations accessibles en français, il relève que le cadre des exposés permettrait aux orateurs de réaliser des démonstrations en direct, ce qui n'était pas possible lors du TUG.

Relations avec d'autres associations et salons

GUTenberg est désormais membre de l'April[17], ce qui permet de donner de la visibilité à l'association et, surtout, de contribuer à la promotion et la défense des logiciels libres, dont font partie TEX et la plus grande partie des programmes, formats et extensions développés autour. Il serait envisageable d'adhérer à Parinux[18], puisque cette association accueille généreusement GUTenberg lors de ses samedis du Libre[19].

Parinux a proposé à GUTenberg de participer à son forum les 7-8 décembre.

Migration de la liste `gut@ens.fr` vers Discourse

Denis Bitouzé rappelle le projet de transformer la liste `gut@ens.fr`, hébergée par l'École normale supérieure, en un forum `Discourse` hébergé sur le serveur de l'association. Il serait toujours possible de suivre les échanges et d'y participer par courriel. Denis est en contact avec l'un des mainteneurs de `Lilypond`, qui pourrait partager son expérience d'une telle migration.

Serveur et miroir du CTAN

Au jour de l'AG, même en opérant des nettoyages très réguliers, la partition allouée au serveur est utilisé à au moins 80 %, ce qui oblige à faire preuve d'une vigilance constante pour éviter qu'elle ne se remplisse entièrement. Or, les finances de l'association ne permettent pas de souscrire à un abonnement qui offrirait davantage d'espace disque. Considérant que la priorité doit être donnée aux services que l'association GUTenberg est la seule à fournir et à

16. Rappelons que ce rescrit fiscal a été obtenu. Voir page 4.
17. https://april.org/
18. https://parinux.org/
19. https://samedis-du-libre.org/

ses projets propres, et étant donné le projet de créer un forum, il est décidé de supprimer le miroir du CTAN hébergé par GUTenberg, ce qui permet de libérer 57 Go[20].

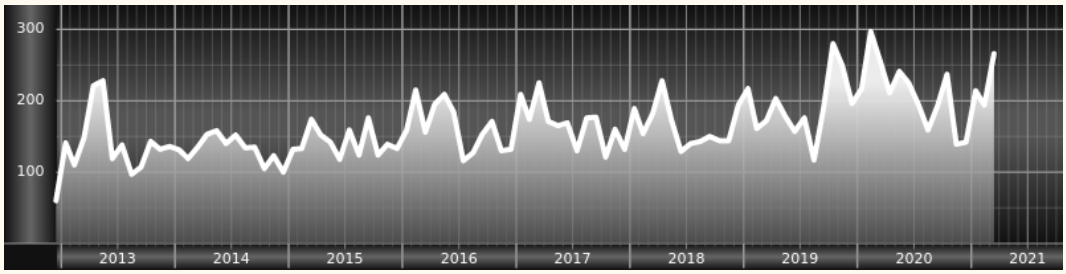
Activités diverses

Pendant la réunion du CA, Patrick Bideault supprime des pourriels du serveur de listes de diffusion et édite la *Lettre*. Yannick édite la FAQ. Malgré tout, ils participent assidûment aux discussions !

Plus rien n'étant à l'ordre du jour et personne ne demandant plus la parole, la réunion est levée à cinq heures de l'après-midi passées de cinq minutes.

Bastien Dumont

ET MAINTENANT, UNE BONNE *VIEILLE* VEILLE TEXNOLOGIQUE !



Chers adhérents, nous veillons TEXnologiquement pour vous !

En effet, la présente rubrique est dédiée aux nouveautés apparues sur le CTAN que vous auriez pu manquer. Elle témoigne de la vitalité de la communauté TEX. Nous y listerons la grande majorité des packages ou classes récemment apparus ainsi que parfois, parmi ceux « simplement » mis à jour, certains qui méritent à notre sens d'être signalés. Nous ne nous interdirons pas, le cas échéant, d'en mentionner de plus anciens, soit parce qu'ils nous semblent injustement méconnus, soit parce qu'ils sont les fruits de contributeurs francophones. Au sujet de la francophonie, nous signalons au moyen du logo les travaux de francophones.

Enfin, nous avons à cœur d'illustrer ces pages par des exemples. La plupart sont dûs aux auteurs des packages eux-mêmes : nous les avons trouvés dans leurs documentations et nous en publions le code en regard du résultat. Mais ce code est parfois trop long pour être publié en ces pages, auquel cas seul le résultat est utilisé ; il est néanmoins facile à trouver dans la documentation du package en question.

Nouveautés

Pour la rubrique du présent numéro, nous listons la plupart des nouveautés, classées par ordre chronologique, apparues depuis la précédente *Lettre*[21] et jusqu'au mois de septembre 2024 inclus. Nous espérons n'oublier aucun nouveau package. Si c'était le cas, merci de nous le faire savoir.

20. Maxime Chupin a effectué cette suppression dans les jours qui ont suivi la réunion du CA et a obtenu en compensation la création (à venir) d'un nouveau miroir hébergé par l'Université Paris Dauphine.

21. La précédente *Lettre* était numérotée 52 ; elle est parue le 30 avril 2024, mais nous en avions figé le contenu au début du mois d'avril. Les packages d'avril sont donc listés dans la présente livraison et non dans la précédente.

Avril 2024

coloredtheorem : utilise le package tcolorbox pour des environnements dédiés aux théorèmes, axiomes et autres assertions.

didactic : propose des environnements dédiés aux travaux pédagogiques. Il ajoute également diverses fonctionnalités aux classes beamer et memoir.

jsonparse : permet d'utiliser facilement des données JSON au sein d'un document LaTeX.

epcqrcode : dérive du package qrcode et crée des QR codes EPC, qui permet d'initier facilement des virements dans l'espace unique de paiement en euros (SEPA). Cette extension vérifie la validité du numéro international de compte bancaire (IBAN) qui est saisi.

l3kernel-dev : est la version de test du package l3kernel qui fournit l'interface de programmation LaTeX3 comme un ensemble de packages qui peuvent être utilisés avec LaTeX 2_ε.

l3backend-dev : est la version de test du package l3backend qui fournit l'interface pour les fonctions LaTeX3 dites de *backend*, c'est-à-dire permettant d'interagir avec les différents moteurs.

lscapeenhanced : améliore le package lscape, utilisé pour composer les pages à l'italienne. C'est un package de Markus Kohm[22], auquel on doit les nombreux packages KOMA.

exercisesheets : propose un environnement de composition de feuilles d'exercices et solutions. Cette extension peut fonctionner seule, pour des documents PDF classiques, ou avec beamer. Elle dispose d'une documentation de bonne qualité, en anglais, mais nous aurions aimé la voir illustrée d'exemples.

ruscap : est une fonte *rustica* produite avec METAFONT par Victor Sannier. Celui-ci a présenté son travail lors de la conférence TUG2023 : vous pouvez retrouver la vidéo de sa présentation ainsi que son article pour le *TUGboat* à partir de la page du TUG2023[23].

Exemple 1

code

```
\font\ruscap=ruscap10 at 14pt
{\ruscap SALVETE OMNES}
```

résultat

SALVETE OMNES

rpgicons : fournit des icônes destinées aux jeux de rôle. L'auteur du package fournit une documentation remarquable, en anglais. Les icônes sont obtenues très simplement :

Exemple 2

code

```
\ability{charisma}
\saving{intelligence}
\spellschool[positive]{abjuration}
\die{sixside seven}{2}
```

résultat

Il est évidemment possible d'obtenir des images plus élaborées :

22. https://www.ctan.org/author/kohm
23. https://tug.org/tug2023/

Exemple 3

code

```
\begin{tikzpicture}[scale=4]
  \definecolor{fillcolor}{HTML}{e76f51}
  \path[scale=.333, fill=fillcolor] (0, 0) circle[radius=.45
    cm];
  \node[proficiency, white, line cap=round, line join=round,
    thick, scale={4*.225}] at (0, 0) {};
\end{tikzpicture}
```

résultat

Mai 2024

mathgreeks : donne accès à un large éventail de caractères grecs pour la composition des mathématiques pris à des fontes diverses, *sans* que l'on ait à charger aucune de ces fontes. C'est très pratique !

cyrillic-modern : est un ensemble de fontes cyrilliques.

synthslant : permet d'incliner des caractères à volonté, que ce soit dans un sens ou dans l'autre : de quoi fabriquer des italiques *droits* ! Cette extension est dotée d'une documentation en anglais très détaillée.

typog : est créé par l'auteur du package précédent. Il propose un large éventail d'utiles commandes micro-typographiques, pour contrôler par exemple les coupures de mots, les corrections italiques ou encore l'espacement entre les mots. Il est doté d'une documentation conséquente (en anglais) et d'un fichier d'exemples très éclairant.

ifis-macros : propose deux macros, `\ifisint` et `\ifisdim`, permettant respectivement de tester en plain TEX si une chaîne de caractères est une représentation valide d'un entier ou d'une dimension.

ximera : est une classe permettant de composer simultanément des fichiers PDF et des documents interactifs à consulter en ligne. On trouvera de nombreux exemples d'utilisation à l'adresse `https://ximera.osu.edu/testing/examples`.

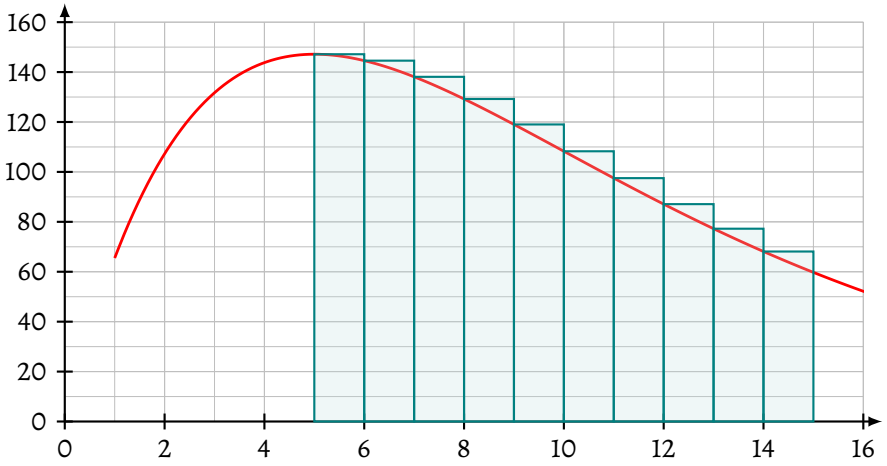
tkz-grapheur : est destiné aux professeurs de mathématiques, notamment en lycée. Il permet de dessiner des courbes, de composer facilement intégrales, tangentes et intersections. Les francophones apprécieront ses commandes à la syntaxe explicite : `\DefinirCourbe`, `\RecupererCoordonnees`, `\TrouverIntersections` etc.

Exemple 4

code

```
\begin{GraphiqueTikz}
  [x=0.66cm,y=0.033cm,Xmin=0,Xmax=16,Xgrille=2,Xgrilles=1,
  Ymin=0,Ymax=160,Ygrille=20,Ygrilles=10]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Couleur=red,Nom=cf,Debut=1,Fin=20,Trace]<f>{
    80*x*exp(-0.2*x)}
  \RepresenterMethodeIntegrale[Couleur=teal]<f>{5}{15}
\end{GraphiqueTikz}
```

résultat (suite)



lete-sans-math : charge la police mathématique `LeteSansMath.otf` — anciennement `LatoMath.otf` — destinée à accompagner les fontes Lato. Ce package est l'ancien package lato-math, qui a changé de nom en même temps que ladite fonte. Ce changement de nom a été demandé par les développeurs polonais de ces fontes. On appréciera la réponse fournie : elle remplace le mot polonais « lato », qui signifie « été », par… « l'été », en français !

colorblind : incorpore dans le processus d'écriture, pour reprendre la présentation par l'auteur, les procédures qui assurent la lisibilité du document par une personne daltonienne (achromatope).

rigidnotation : permet de composer avec rigueur les éléments de notation couramment utilisés dans la recherche en robotique pour les vecteurs et les matrices qui sont affublés de nombreux indices, etc. Il est à noter que ce package est publié en même temps qu'un article proposant une norme à ce sujet ([1]).

skillicons : permet d'insérer facilement dans vos documents des icônes représentant des logiciels ou des langages informatiques. L'extension en propose beaucoup ! Ce package n'a pour l'instant pas été intégré dans la TEX Live pour des problèmes de droits sur les icônes elles-mêmes : il n'est donc disponible que sur le CTAN et dans la distribution MiKTEX. Nous le regrettons car ce sympathique package se révèle très utile.

semesterplannerlua : compose des agendas en s'appuyant, comme son nom l'indique, sur LuaTEX.

catppuccinpalette : propose quatre palettes de couleurs, tirées de `catppuccin`, une initiative en ligne consacrée aux gammes de couleur : `https://catppuccin.com/`.

mpkiviat : est un package MetaPost qui permet de dessiner des diagrammes de Kiviat (ou diagrammes en radar, en étoile ou en toile d'araignées). Cela nous rappelle qu'Alain Matthes avait déjà proposé un package tkz-kiviat pour faire cela, mais il l'a depuis retiré de la publication.

Exemple 5

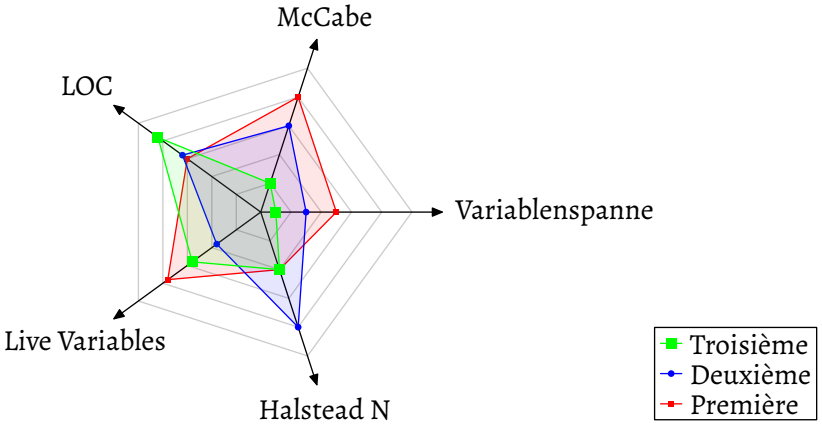
code

```
\begin{mplibcode}
input mpkiviat;
beginfig(1);
set_lattice_grid(100,500);
set_kiviat_unit(0.4cm);
set_axis("McCabe","LOC","Live Variables","Halstead N","
    Variablenspanne");
```

code (suite)

```
draw_axis;
filldraw_line(400,300,380,200,250)(red);
set_line_mark_type("circle");
filldraw_line(300,320,180,400,150)(blue);
set_line_mark_type("square");
set_line_mark_scale(2);
filldraw_line(100,420,280,200,50)(green);
draw_legends.lrt("Première", "Deuxième", "Troisième");
endfig;
\end{mplibcode}
```

résultat



nxuthesis : est une classe destinée aux thèses rédigées au sein de l'université du Ningxia, dont on n'ignore pas qu'elle est située à Yinchuan, en Chine.

rub-kunstgeschichte : est une classe assez simple, destinée aux documents de l'Institut d'histoire de l'art de l'université de la Ruhr à Bochum, en Allemagne.

Juin 2024

asy-overview : est une présentation en anglais du langage `Asymptote`.

ensps-colorscheme : donne accès à la palette de couleurs de l'École normale supérieure de Paris-Saclay.

xint-regression : est un package utilisant `xint` pour calculer les régressions classiques : linéaire, quadratique, cubique, puissante, exponentielle, logarithmique et hyperbolique. Les sources de ce package sont hébergées au sein de la forge logicielle des biens communs numériques de l'éducation nationale française.

Exemple 6

code

```
\def\LLX{83,71,64,69,69,64,68,59,81,91,57,65,58,62}%
\def\LLY{183,168,171,178,176,172,165,158,183,182,163,175,164,
    175}%
Les paramètres de l'équation cubique $ax^3+bx^2+cx+d$ la plus
proche de ces valeurs sont :
\xintcubreg[round=5/3/2/1]{\LLX}{\LLY}%
$a \approx \cubrega$ ; $b \approx \cubregb$ ;
$c \approx \cubregc$ et $d \approx \cubregd$.
```

résultat (suite)

Les paramètres de l'équation cubique $ax^3 + bx^2 + cx + d$ la plus proche de ces valeurs sont : $a \approx 0.0001$; $b \approx -0.039$; $c \approx 4.7$ et $d \approx 3.2$.

standardsectioning : est un package qui permet de faciliter la transition des classes standards vers les classes de KOMA-Script, souvent source d'erreurs concernant les commandes de sectionnement (notamment avec le package titlesec).

linearregression : permet d'obtenir un tableau donnant les caractéristiques des différentes droites de régression possibles (y en x ; x en y ; régression symétrique) à partir des coordonnées des points d'un nuage. Les coordonnées sont lues dans un fichier contenant deux valeurs par ligne, séparées par une espace ; un format fréquent pour ce type de fichiers. Une commande permet de représenter graphiquement le nuage de points et les droites de régression. De l'aveu de l'auteur, la mise en forme est assez sommaire : par exemple, les graphiques n'utilisent pas de couleurs. En revanche, la documentation de ce package est bien détaillée.

spelatex : facilite la lecture des personnes souffrant de troubles. Au stade actuel de son développement, ce package lit le source LaTeX, convertit le texte et les formules en fichiers audio et dote le PDF d'hyperliens vers ces fichiers, de sorte qu'on peut simplement écouter votre document en quelques clics. Pour le moment, le paquet est paramétré pour fournir une prononciation correcte en anglais et en néerlandais, mais l'auteur annonce que d'autres langues seront supportées à l'avenir.

tiet-question-paper : est une classe destinée aux sujets d'examen de l'institut Thapar d'ingénierie et de technologie[24], qui est sis à Patiala, dans le Pendjab, en Inde.

latexscreenshooter : est un programme Java et un package LaTeX permettant d'insérer automatiquement des captures d'écran de pages web, en compilant avec l'option `--shell-escape`. Ce système fonctionne avec différentes distributions Linux ainsi que sous Windows.

Juillet 2024

gelasiomath : est le complément mathématique des fontes gelasio.

tango : est une classe de documents pour le système de composition LaTeX à l'usage des professeurs de mathématiques (grosso modo, de bac–3 à bac+3). Elle est conçue pour composer divers types de documents, du petit polycopié au livre complet. Les formats de sortie sont A4paper, Letter, A5paper, BigTablet, Tablet, SmallTablet, eReader ou Smartphone. De nombreux environnements et des thèmes de couleurs sont proposés. Cette classe permet la composition de magnifiques documents. Encore un fabuleux travail de Michel Bovani. Nous espérons un article sur le sujet dans une prochaine *Lettre*.

passopt : permet une gestion fine des options des classes et packages.

moremath : est un package facilitant la composition des mathématiques. Il fournit en particulier de nombreuses commandes pour les opérateurs classiques (tels les cosinus, sinus, exponentielle, logarithme) permettant de gérer simplement les délimiteurs et leur ajustement autour de leurs arguments. Il redéfinit aussi quelques opérateurs pour en faciliter l'ajustement de leur composition (indice, exposant, graisse, etc.). Enfin, il fournit quelques commandes pour la composition des vecteurs et des matrices facilitant la gestion des alignements et des ajustements de leurs éléments et délimiteurs.

luwiantype : permet d'écrire en louvite hiéroglyphique. Ça nous change du syriaque.

wrapstuff-doc-en : est une traduction en anglais du manuel du package wrapstuff.

24. https://en.wikipedia.org/wiki/Thapar_Institute_of_Engineering_and_Technology

bonum-otf : fournit des commandes facilitant l'utilisation des fontes Bonum, y compris pour composer des mathématiques.

doibanner : permet de composer les DOI à l'intérieur d'une cartouche.

pgfplotsthemebeamer : permet de composer avec `pgfplots` des graphiques qui reprennent automatiquement les couleurs du thème `beamer` utilisé. On doit ce package à l'excellente `samcarter`[25].

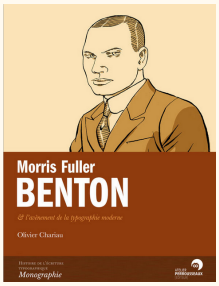
telprint : permet de composer correctement les numéros de téléphone allemands.

hypcap : permet d'harmoniser le comportement d'hyperref avec les flottants.

Août 2024

bib2qr : permet de créer le QR-code associé à une référence bibliographique en s'appuyant sur biblatex et qrcode.

polyomino : permet de créer des polyominos, ce qui réjouira les amateurs de pavages.



mfb-oldstyle : est une fonte OTF avec empattement. Elle fut originellement créée par Morris Fuller Benton et distribuée par l'*American Type Founders* en 1909 sous le nom de *Century Oldstyle*. La famille contient les fontes romaine, italique et grasse. Malheureusement, cette fonte ne contient pas les glyphes mathématiques…

Au sujet de ce célèbre créateur de caractères, rappelons qu'Olivier Chariau lui a consacré en 2019 un ouvrage illustré, de fort belle qualité, intitulé *Morris Fuller Benton & l'avènement de la typographie moderne*. Il est édité par l'atelier Perrousseaux[26] et sa couverture est reproduite ci-contre.

fillwith : fournit des commandes pour composer automatiquement des lignes (pleines ou en pointillés) sur les zones qui doivent être remplies à la main. Cela est très utile pour les examens ou les questionnaires.

hebdomon : est une classe de document pour des rapports ou des documentations. Une de ses particularités est de n'avoir modifié aucune macro standard : par exemple la classe fournit `\Chapter`, `\Section`, etc., tandis `\chapter` et `\section` peuvent aussi être utilisés pour produire le résultat classique de LaTeX. Cette classe fournit des environnements de type boîtes, des environnements de *listing* et bien d'autres choses encore.

luamml : utilise les *callbacks* de LuaTeX pour convertir les mathématiques LuaTeX en une sortie `MathML`.

edmaths : permet de composer des rapports ou des thèses pour la *School of Mathematics* de l'université d'Édinbourg, qui à notre humble avis s'occupe de mathématiques.

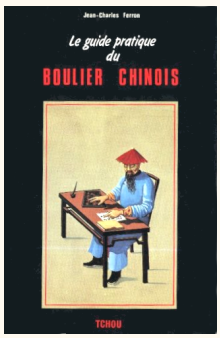
domaincoloring : de Herbert Voß s'ajoute aux `PStricks`, collection de macros permettant d'utiliser le langage `PostScript` depuis (La)TeX. Cette fois, il s'agit de représenter le comportement (N.-B. : un peu de jargon mathématique suit !) d'une fonction complexe d'une variable complexe en coloriant un point du plan d'origine en accord avec la valeur de son image. La documentation, très complète, fourmille d'exemples et renvoie à l'article que consacre le `Wikipedia` anglophone aux colorations de régions. Cette extension nécessite LuaLaTeX.

interlinear : fournit des commandes pour composer du texte entre les lignes principales. Il fait partie d'une ensemble plus large d'outils destinés à faciliter la production des documents de linguistique.

25. https://www.ctan.org/author/samcarter
26. On trouvera cet ouvrage consacré à Morris Fuller Benton sur la page suivante (hélas émaillée de coquilles) : https://www.adverbum.fr/GB/atelier-perrousseaux/olivier-chariau/moris-fuller-benton_oby5il2uwv8j.html

enverb : permet de composer des textes verbatim et d'afficher leur résultat. Plus précisément, ce package fournit des outils pour collecter le contenu d'un environnement verbatim afin de l'utiliser plus tard (un *listing*, un résultat, les deux, etc.).

xreview : permet d'annoter un document (suppressions, ajouts, remplacements, commentaires et remarques) et d'obtenir, grâce à un système d'interrupteurs, une version de travail, dans laquelle les couleurs peuvent être personnalisées, ou une version propre « prête à l'impression ». La documentation contient de nombreux exemples qui sont d'ailleurs repris dans le fichier `README` présent sur la page dédiée à xreview sur `GitHub`[27].



suanpan-l3 : permet de dessiner un boulier chinois (*suanpan*) grâce au module `l3draw` du package l3experimental. Sa documentation, bilingue chinois-anglais, est accompagnée de nombreux exemples. On peut colorer les boules et d'autres éléments du boulier. On pourra, si on le désire, se reporter au petit opuscule (59 pages) *Le guide pratique du boulier chinois* de Jean-Charles Ferron paru en 1987 aux éditions Sand & Tchou (voir ci-contre). Je profite de l'occasion pour rappeler les deux versions de l'extension qui permet de dessiner un boulier japonais : pst-soroban et pgf-soroban.

beamertheme-edmaths : est un thème beamer pour le département de mathématiques de l'université d'Édinbourg, en Écosse, laquelle fut fondée en 1583. Voir le package edmaths en page précédente.

Septembre 2024

À la fin du mois de septembre, le CTAN a entrepris une importante migration de serveur, qui a rendu indisponible la liste des nouveaux packages. Nous avons donc établi la liste ci-dessous en fonction des courriels que nous avons reçus ; peut-être quelques nouveautés sont-elles passées entre les mailles de notre filet. Nous vérifierons cela dès que nous aurons à nouveau accès aux informations habituelles.

csthm : propose des environnements de type « théorème » adaptés aux documents traitant d'informatique. On peut redéfinir la couleur d'accentuation et, à l'aide d'une option, activer le support de cleveref.

framedsyntax : fournit un environnement pour décrire du code TEX dans une boîte de mise en valeur, notamment pour la rédaction de documentation.

scrhack : est un ensemble de macros issues des classes KOMA-Script destinées à mieux gérer quelques compatibilités tierces. Il est désormais indépendant des classes en question.

quran-es : est une extension du package quran qui ajoute le support de trois traductions espagnoles du Coran.

aiplans : est un tutoriel pour représenter des cartes servant à une sous-discipline de domaine de l'intelligence artificielle.

euromoney : est un package de notre prolifique Cédric Pierquet permettant de dessiner des pièces et des billets en euros. Il fournit même une interface en français.

Exemple 7

code

```
\PiecesEuro%
[Empilage=V,HauteurRef=3cm,HauteurAuto,Style=full]{4*2+3*1+4*
    0.2+1*0.1+2*0.01}
```

27. https://github.com/LorenzoPeri17/xreview-latex

résultat (suite)

keytheorems : est un package expérimental implémentant une interface clé/valeur basée sur LaTeX3, sur laquelle reposent la plupart des fonctionnalités du package thmtools.

cascadiamono-otf : est une fonte à chasse fixe, ou plutôt la configuration fontspec pour utiliser la fonte CascadiaCode sans les ligatures. Encore une œuvre de Cédric Pierquet !

Exemple 8

code

```
\setmonofont{CascadiaMono}
\begin{lstlisting}[language=python]
def fibonacci(n):
  if(n <= 1):
      return n
  else:
      return (fibonacci(n-1) + fibonacci(n-2))
\end{lstlisting}
```

résultat

```
def fibonacci(n):
  if(n <= 1):
      return n
  else:
      return (fibonacci(n-1) + fibonacci(n-2))
```

zugferd : permet de composer des factures (notamment en faisant faire les calculs à LaTeX). Nous devons ce package à Marei Peischl, dont nous avons publié l'article consacré à expl3 dans la *Lettre* 52.

ezedits : est un package permettant d'annoter et de marquer facilement les modifications lors de l'écriture, par exemple à plusieurs, d'un document.

beamertheme-gotham : est un thème beamer minimaliste et élégant. Il fournit un grand nombre d'options de personnalisation sous la forme très pratique de clés/valeurs.

randintlist : permet à tout moteur TeX d'accomplir ce que luarandom réservait au seul LuaTeX. Cédric Pierquet nous offre une documentation bilingue et avec deux commandes — disposant d'un nom anglais et d'un nom français — un système de clés pour obtenir une liste d'entiers, tirés au hasard, avec ou sans remise, ordonnés ou non. Liste dont on peut extraire un élément.

commalists-tools : est un ensemble de macros permettant de manipuler des listes de valeurs séparées par des virgules. Cela permet de trier, de tester et d'effectuer des opérations sur les valeurs d'une liste. Encore un outil bien pratique proposé par Cédric Pierquet !

Comment soi-même veiller technologiquement ?

Pour être tenu informé en « temps réel » des nouveautés et mises à jour du CTAN, on peut par exemple consulter régulièrement la page https://www.ctan.org/ctan-ann ou, mieux, s'abonner aux flux ou à la liste CTAN-ann qui y sont mentionnés. Il est alors toutefois à noter que, si les nouveautés sont effectivement toutes signalées, les mises à jour ne le sont en revanche que si leurs auteurs ont estimé que l'annonce se justifiait.

Patrick Bideault, Denis Bitouzé, Maxime Chupin, Bastien Dumont & Yvon Henel

Références

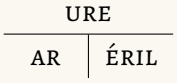
[1] Philippe NADEAU. *A Standard Rigid Transformation Notation Convention for Robotics Research*. 2024. arXiv : 2405.07351 [cs.RO].

[2] Olivier CHARIAU. *Morris Fuller Benton & l'avènement de la typographie moderne*. Gap, France : Atelier Perrousseaux, 2019. 119 p. ISBN : 978-2-36765-013-5.

[3] Jean-Charles FERRON. *Le guide pratique du boulier chinois*. France : Sand & Tchou, 28 juill. 1987. 59 p. ISBN : 978-2710703839.

MAIS À PRÉSENT, JOUONS UN PEU !

☞ **Rébus (*devise de François Arago*)**



Solution dans la prochaine Lettre

Yvon Henel

ORDRE LEXICOGRAPHIQUE EN FRANÇAIS

ALGORITHME UCA, TRI D'INDEX

Résumé

Doit-on classer *côte* avant ou après *coté* ?
Existe-t-il un algorithme informatique permettant de classer les mots et locutions françaises dans le même ordre que les dictionnaires classiques ?
Quel programme choisir pour créer un index en LaTeX ?
C'est à ces questions qu'essaie de répondre ce court mémoire.

Ordre lexicographique en français

L'ordre de classement des mots et locutions dans un dictionnaire français semble naturel : on triera par exemple sans hésitation[28] *acompte* < *Açores* < *à-côté* < *à-coup* < *a priori*.

L'ordre alphabétique est connu dans ses grandes lignes mais les détails le sont moins. Rappelons quelques règles de base :

— les caractères accentués (*à*, *é*, *ô*) et les diacritiques (*ç*) se classent avec leur lettre de base (*a*, *e*, *c* respectivement), en fait la lettre de base précède immédiatement sa variante accentuée, ainsi *cote* < *coté* et *côte* < *côté* ;

— l'ordre des accents en français est identique quelle que soit la lettre de base, nous le donnons uniquement pour le *e*[29] : *e* < *é* < *è* < *ê* < *ë* ;

— les ligatures *æ* et *œ* sont traitées comme deux lettres *ae* et *oe* :
exact < *ex æquo* < *exagération*
coefficient < *cœur* < *coexistence*

— la minuscule précède la majuscule (*a* < *A*) ;

— enfin les tirets (et avec eux les espaces, apostrophes, etc.) sont ignorés.

Une chose est claire : un tri binaire comparant les codes des caractères est totalement exclu quel que soit le codage envisagé (iso-latin-9, unicode, etc.). Rappelons en effet que le codage ASCII, base de tous les autres, place toutes les capitales avant les minuscules, on aurait donc *Zoo* avant *abat*… De même, les codes des lettres de base sont inférieurs à 127 tandis ceux des caractères accentués sont supérieurs à 128, ce qui conduirait à placer *coton* entre *cote* et *coté* !

Certains choix sont moins évidents, notamment le classement en présence d'accents multiples ; ainsi, les dictionnaires classiques (Larousse, Littré, Robert,…) ont toujours classé :
cote < *côte* < *coté* < *côté*
élève < *élevé*
gène < *gêne* < *gêné*
pèche < *pêche* < *péché* < *pêché*
relève < *relevé*

Les raisons linguistiques ayant conduit à ces choix sont disparates et pas toujours écrites, mais il semble y avoir un consensus implicite : on ne note pas de divergences entre Larousse, Littré et Robert par exemple. Une règle simple semble concerner les substantifs homographes[30] : le féminin précède le masculin, ainsi (la) *relève* doit être classée avant (le)

28. Dans tout ce document le signe < s'entend au sens de la relation d'ordre lexicographique, a < b signifiant *a* vient avant *b*.
29. Car c'est la lettre qui admet le plus de variantes accentuées en français.
30. Mots identiques aux accents et diacritiques près, exemple *cote*, *côte*, *coté*, *côté*.

relevé, (la) *côte* avant (le) *côté*[31].

Trouver un algorithme aboutissant automatiquement au classement traditionnel des mots et locutions françaises n'a pas été simple. Le problème est maintenant résolu grâce à l'algorithme UCA et aux contributions d'une équipe canadienne autour d'Alain LaBonté.

De nos jours, les recherches dans les bases de données imposent que s'applique un ordre unique connu de tous les utilisateurs. Un classement reposant sur un algorithme est un gage de stabilité et de reproductibilité. Mais le classement algorithmique se doit d'être compatible avec le classement historique des dictionnaires classiques de la langue concernée.

Notons que l'ordre lexicographique est très dépendant de la langue : un francophone s'attend à trouver *œdème* sous la lettre *O* tandis qu'un Suédois cherchera le mot *öde* (désert) à la lettre *Ö* après le *Z*. Les Allemands ont deux modes de classement, DIN 5007-1 (cas général : *ä* traité comme *a*) et DIN 5007-2 (listes de noms propres : *ä* traité comme *ae*).

Une dernière précision : il est d'usage de classer les locutions étrangères comme celles de la langue principale du document. L'index d'un document en français placera donc *föhn* entre *fœtus* et *foi* et le mot suédois *öde* à la lettre *O*.

Principes de l'algorithme UCA[32]

Un tri à trois niveaux

Vouloir classer ensemble les homographes amène à proposer un premier niveau de tri où tous les accents sont supprimés. L'usage étant de ne pas tenir compte (dans un premier temps) de la casse, on négligera également celle-ci au premier niveau.

À l'issue de cette première phase, les homographes sont classés ex æquo : ils pourront être départagés lors d'une seconde phase de tri. Une troisième phase permettra de départager majuscules et minuscules : *à* et *À* sont classés ex æquo aux niveaux un et deux.

L'algorithme UCA reprend ce principe. Nous le décrivons uniquement pour les langues latines[33]. L'idée de base est d'associer à chaque chaîne de caractères une clé de tri[34] qui pourra ensuite être traitée numériquement.

À chaque caractère sont associés trois « poids » (primaire, secondaire, tertiaire) qui déterminent l'ordre de classement aux niveaux 1, 2 et 3 respectivement.

Le poids primaire est pris en compte au premier niveau de tri (sans accents ni diacritiques). Ainsi un *a* ou un *à* auront le même poids primaire, plus faible qu'un *b*. La minuscule et la majuscule correspondante ont toujours le même poids primaire ce qui assure de les classer ex æquo au premier niveau.

La liste de ces « poids » est donnée dans une table appelée DUCET.

La prise en compte des poids se fait dans le sens de la lecture, donc de gauche à droite en français, et la comparaison s'arrête à la première différence trouvée.

31. NDLR : Il s'agit d'une régularité apparente plutôt que d'une règle. En effet, nous voyons parmi les exemples ci-dessus que (le) *gène* précède (la) *gêne* : dans le cas de *côte* et *coté*, il semblerait que ce principe de priorité du féminin l'emporte sur l'ordre de précédence des accents, tandis que pour *gène* et *gêne*, ce serait l'inverse. Nous verrons dans la section suivante qu'un ensemble de règles fondées exclusivement sur la graphie permet de mieux prédire l'ordre traditionnel des dictionnaires.

32. L'algorithme UCA est décrit en détail dans [1].

33. UCA est suffisamment flexible pour traiter toutes les langues alphabétiques ou non, arabe, chinois, langues indiennes comprises.

34. Les utilisateurs avisés de `makeindex` ont recours à une telle clé lorsqu'ils veulent classer correctement un mot comme *été* : ils codent `\index{ete@été}` pour associer la clé `ete` au mot *été*.

Prenons un exemple concret : comparons les chaînes de caractères *Cote*, *côte*, *coté* et *côté*. La table DUCET donne les « poids » suivants[35] (en base 10) :

	Primaire	Secondaire	Tertiaire
C	7617	32	8
c	7617	32	2
o	7972	32	2
ô	7972	39	2
t	8157	32	2
e	7665	32	2
é	7665	37	2

On constate que les poids primaires sont identiques pour *C* et *c*, ainsi que pour les voyelles avec ou sans accent (*o* et *ô*, *e* et *é*). Les poids des *c*, *e*, *o* et *t* sont bien en ordre croissant. Au niveau primaire, les quatre chaînes ont la même clé de tri (7617,7972,8157,7665) : elles seront donc classées ex æquo.

En revanche, un mot comme *coopérer* serait classé avant elles au niveau primaire (au bout de seulement trois comparaisons), le second *o* ayant un poids inférieur au *t*.

Autre exemple. Les deux premières lettres des mots *ça* et *car* ont le même poids primaire : c'est la longueur qui fait la différence dès le premier niveau, *ça* < *car*, parce que l'absence de troisième caractère dans *ça* est interprétée comme un caractère vide de poids 0.

Au niveau secondaire, le mot *Cote* sera évidemment classé en premier, sa clé de tri étant (32,32,32,32). Comparons ensuite *côte* et *coté* : en lisant de gauche à droite, le *o* a un poids inférieur à celui de *ô*, donc *coté* devrait être classé avant *côte* ; malheureusement ce classement contredirait l'usage… Une équipe canadienne, autour d'Alain LaBonté, a proposé d'inverser l'ordre de lecture pour classer les accents en français (donc uniquement au niveau deux). De fait, le résultat est alors conforme au classement traditionnel des dictionnaires français… pas seulement sur cet exemple ! Alain Rey lui-même a donné acte à Alain Labonté que sa méthode produisait un classement en tout point conforme à l'usage en français.

Vérifions-le sur notre exemple. Si on prend en compte les accents de droite à gauche, c.-à-d. à partir de la fin du mot en remontant :

— dès la première comparaison, les *é* finaux de *coté* et *côté* (poids 37) conduisent à les placer après *côte* (poids 32) ;
— il reste à comparer *côté* et *coté* : la troisième comparaison impose *coté* < *côté* (32 contre 39).

Le classement final est donc *Cote* < *côte* < *coté* < *côté*.

Dans la table DUCET, le poids secondaire des accents est propre à l'accent et indépendant de la lettre qui le porte. Il conduit au classement suivant qui convient parfaitement au français : *e* < *é* < *è* < *ê* < *ë* (poids respectifs 32, 36, 37, 39, 43).

Le consortium Unicode, garant de l'algorithme UCA, a officiellement entériné la possibilité d'inverser l'ordre de lecture pour chaque niveau. À ce jour, cette possibilité n'est utilisée qu'au niveau deux et uniquement pour le français.

35. En réalité, les lettres accentuées sont prises en compte comme deux caractères successifs (ou plus), la lettre de base suivie de son (ou ses) accents(s). Ainsi dans la table DUCET, l'accent aigu a pour poids (0,39,2), le grave (0,37,2). Sachant que les poids nuls sont ignorés à chaque niveau de tri, la présentation simplifiée que nous donnons ici est correcte, mais uniquement pour les langues n'ayant aucun caractère qui porte plusieurs accents. C'est le cas du français, mais pas du vietnamien par exemple.

Le troisième niveau permet par exemple de comparer *Cote* et *cote*. Les capitales ont un poids tertiaire de 8 contre 2 pour les minuscules, ce qui place *Cote* après *cote*, c'est effectivement l'ordre adopté en français. Une option permet d'inverser cet ordre pour placer les majuscules avant les minuscules, comme c'est l'usage en allemand par exemple.

Un quatrième niveau (optionnel)

Il reste à préciser comment classer les chaînes de caractères comportant des espaces, traits d'union, apostrophes, etc.

Dans l'algorithme UCA de base à trois niveaux, ces caractères sont traités de la même façon que les lettres, en fonction de leurs poids dans la table DUCET :

	code	Primaire	Secondaire	Tertiaire
`<espace >`	U+0020	521	32	2
`<tiret ->`	U+002D	525	32	2
`<apost. '>`	U+0027	785	32	2
`<apost. ’>`	U+2019	787	32	2

Leur poids primaire est nettement inférieur à celui des autres caractères, ainsi en anglais le tri UCA à trois niveaux classera *stand* < *stand by* < *stand for* < *stand up* < *standard*, l'espace ayant un poids primaire inférieur à celui du *a*. Le résultat est satisfaisant dans ce cas mais l'est moins si on considère les locutions *death*, *de-escalate* et *de facto*, qui seront classées *de facto* < *de-escalate* < *death*, l'ordre attendu étant plutôt *death* < *de-escalate* < *de facto*.

L'algorithme UCA propose plusieurs options permettant d'ignorer les caractères non alphabétiques (espaces, tirets et apostrophes ainsi que d'autres) aux trois premiers niveaux de tri. Un quatrième niveau de tri est alors nécessaire pour départager *tire-bouchon* et *tirebouchon* par exemple.

La table DUCET donne pour chaque caractère, en plus des trois poids mentionnés ci-dessus, une quatrième valeur binaire dite de « type » : soit « type lettre », soit « type autre ». Lors des trois premiers niveaux de tri, les caractères de type « lettre » sont toujours pris en compte, tandis que ceux de type « autre », aussi appelés « à poids variable » (*Variable weight*), peuvent être ignorés : en activant l'une des options `[Alternate Shifted]` ou `[Alternate Shift-Trimmed]`, les caractères de type « autre » voient leurs poids annulés aux niveaux un à trois[36] et sont crédités au niveau quatre d'un poids égal à leur ancien poids primaire de base. Ainsi, si l'une des options `[Alternate Shifted]` ou `[Alternate Shift-Trimmed]` est activée, le tiret U+002D est pris en compte avec les poids (0, 0, 0, 525), l'espace avec (0, 0, 0, 521), etc. L'option `[Alternate Shift-Trimmed]` attribue, au niveau quatre, un poids *maximal* aux caractères de type « lettre » (0xFFFF en hexadécimal soit 65535), tandis que `[Alternate Shifted]` leur attribue un poids *minimal* (inférieur à celui de tous les caractères de type « autre »).

En pratique, l'option `[Alternate Shift-Trimmed]` classe *au quatrième niveau*[37] les caractères de type « lettre » *avant* (*coop* < *co-op*, *LaFayette* < *La Fayette*), tandis que l'option `[Alternate Shifted]` les place *après* les espaces, tirets, apostrophes et autres (*co-op* < *coop*, *La Fayette* < *LaFayette*).

Exemples :

— Le tri de base donne :
  — *ça et là* < *cæsium* ;
  — *l'âme* < *lame* < *lamé* ;

36. Rappel : les poids nuls sont toujours supprimés de la clé de tri.
37. Il est bien rare qu'il faille aller jusque-là pour départager deux chaînes de caractères !

— *tire-clou* < *tire-d'aile* < *tire-dent* < *tirebouchon* < *tirefond*.

— Les options `[Alternate Shifted]` et `[Alternate Shift-Trimmed]` classent toutes deux :

— *cæsium* < *ça et là* < *cafard* ;

— *lame* < *l'âme* < *lamé*[38] ;

— *tirebouchon* < *tire-clou* < *tire-d'aile* < *tire-dent* < *tirefond*.

Les options `[Alternate Shift-Trimmed]` ou, à défaut, `[Alternate Shifted]`, sont recommandées pour le français.

Adaptations de l'algorithme UCA aux *locales*

Nous avons vu que les poids attribués par la table DUCET conviennent au traitement du français. Pour d'autres langues, il peut être nécessaire d'adapter cette table en modifiant les poids relatifs de certains caractères. Cette opération s'appelle *tailoring* en anglais, voir [1] section 8, ou [2] section 12.6.3.

La syntaxe à utiliser est assez simple : pour définir un ordre relatif pour les poids primaires on utilise le signe <, << pour les poids secondaires et <<< pour les poids tertiaires. Un locuteur d'une langue scandinave désirant classer dans l'ordre Æ, Ø et Å après *Z* imposera la règle `tailoring("&Z<Æ<Ø<Å")`. En espagnol, pour classer *llano* après *luz*, il suffit d'ajouter la règle `tailoring("&l<ll<<<lL<<<LL")`.

Mise en œuvre informatique : exemple de lua-uca

L'algorithme UCA a été codé d'abord en Java et plus récemment en Lua par Michal Hoftich, lua-uca [3]. Le tri pour le français fonctionne bien depuis la version 0.1d[39] sauf pour les locutions et mots composés (« ça et là », « tire-bouchon »), qui nécessitent le recours au quatrième niveau de tri de l'algorithme UCA. En effet, la version Lua de Michal Hoftich se limite aux trois premiers niveaux.

Le document [4] propose une liste de locutions francophones et indique l'ordre à obtenir. C'est un excellent test de torture pour les programmes de tri, les exemples qui suivent sont basés sur cette liste.

Voici un fichier `sort-list-fr.lua` à compiler avec LuaTEX, qui utilise l'algorithme de lua-uca pour trier la liste de mots contenus dans la table `t`. Cette liste est construite à partir de la liste canadienne (locutions et mots composés ou étrangers exclus), avec quelques ajouts personnels.

Exemple 9

```
#!/usr/bin/env texlua

local t = {"CÔTÉ", "cote", "Côté", "COTÉ", "côte", "COTE",
           "côté", "Coté", "coté", "Cote", "CÔTE", "Côte",
           "lésé", "péché", "bohème", "gêné", "pêche",
           "cæsium", "pêcher", "révèle", "pécher", "révélé",
           "Bohême", "relève", "PÉCHÉ", "maçon", "relevé",
           "Élève", "gêne", "élevé", "MÂCON", "gène",
```

38. Avec lecture des accents à rebours.

39. NDLR : Cette version étant récente, vous devrez probablement mettre à jour manuellement le package lua-uca pour en bénéficier (notamment si vous souhaitez tester l'exemple ci-dessous), en utilisant par exemple `tlmgr`.

```
            "Bohémien", "caennais", "lèse", "coexistence",
            "cœur", "coefficient", "cafard", "CŒUR", "CÆSIUM"}

kpse.set_program_name "luatex"
local ducet = require "lua-uca.lua-uca-ducet"
local collator = require "lua-uca.lua-uca-collator"
local languages = require "lua-uca.lua-uca-languages"

local collator_obj = collator.new(ducet)
-- load French rules
collator_obj = languages.fr_backward_accents(collator_obj)

table.sort(t, function(a,b)
  return collator_obj:compare_strings(a,b)
end)

for _, v in ipairs(t) do
  print(v .. ",")
end
```

L'exécution donne ceci :

```
$ luatex "sort-list-fr.lua"

bohème, Bohême, Bohémien, caennais, cæsium, CÆSIUM,
cafard, coefficient, cœur, CŒUR, coexistence, cote,
Cote, COTE, côte, Côte, CÔTE, coté, Coté, COTÉ,
côté, Côté, CÔTÉ, Élève, élevé, gène, gêne, gêné,
lèse, lésé, MÂCON, maçon, pêche, péché, PÉCHÉ,
pécher, pêcher, relève, relevé, révèle, révélé,
```

Le résultat est un sans-faute.

Un second test porte sur le classement des locutions et mots composés. Le quatrième niveau de tri n'étant pas implémenté dans lua-uca, le résultat n'est pas fameux. On reprend le même script avec la table suivante :

Exemple 10

```
local t = { "vice-président", "Ça", "vice versa", "C.A.F.",
            "tire-dent", "L'Haÿ-les-Roses", "tire-clou",
            "caennais", "co-op", "lame", "Mc Arthur","colon",
            "tirefond", "l'âme", "Canon", "McArthur", "lamé",
            "Mc Mahon", "tire-d'aile", "çà et là", "tirebouchon",
            "MacArthur", "lésé", "maçon", "cæsium", "coop", }
```

Avec cette nouvelle table la compilation donne :

```
$ luatex "sort-list-fr.lua"

C.A.F., Ça, çà et là, caennais, cæsium, Canon,
co-op, colon, coop, l'âme, L'Haÿ-les-Roses, lame, lamé, lésé,
```

```
MacArthur, maçon, Mc Arthur, Mc Mahon, McArthur,
tire-clou, tire-d'aile, tire-dent, tirebouchon, tirefond,
vice versa, vice-président,
```

Le bon ordre serait :

```
$ luatex "sort-list-fr.lua"
```

```
caennais, cæsium, çà et là, C.A.F., Canon, colon,
coop, co-op, lame, l'âme, lamé, lésé, L'Haÿ-les-Roses,
MacArthur, maçon, McArthur, Mc Arthur, Mc Mahon,
tirebouchon, tire-clou, tire-d'aile, tire-dent, tirefond
vice-président, vice versa,
```

En effet, les espaces, apostrophes, tirets et points devraient être ignorés aux trois premiers niveaux, le quatrième niveau départageant *coop* et *co-op* ainsi que *McArthur* et *Mc Arthur*.

Comparaison des programmes de création d'index pour LaTeX

Le cas le plus fréquent où se manifeste le besoin de trier une liste de mots est celui de la création d'un index.

makeindex est le programme le plus ancien, il est limité au tri des caractères ASCII, ce qui le disqualifie pour le français, à moins de créer des clés ASCII pour les mots accentués comme ceci : `\index{ete@été}`. Noter que le classement correct des mots *relève* et *relevé* suppose la création des clés suivantes :

- `\index{releve1@relève}`
- et `\index{releve2@relevé}`

Cela implique que l'auteur connaisse parfaitement les règles de tri en français.

xindy disponible depuis une vingtaine d'année est adapté aux caractères accentués, mais il a été conçu avant le développement d'Unicode essentiellement pour des codages 8-bits (le *é* est codé en LATIN1 ou en LATIN9 sur un seul octet, alors qu'il est codé sur deux octets en UTF-8). Aujourd'hui, les sources LaTeX devraient *tous* être codés en UTF-8 (codage par défaut pour pdfLaTeX et obligatoire pour LuaLaTeX et XeLaTeX), si bien que le recours au programme `xindy` n'est plus pertinent. De plus, `xindy` ne respecte pas complètement l'ordre lexicographique des dictionnaires français : il classe par exemple *relève* (féminin) après *relevé* (masculin).

xindex[5] est écrit en Lua et adapté au codage UTF-8 ; depuis la version 0.60 (mai 2024), le tri est fait par lua-uca (option par défaut) et donne des résultats satisfaisants en français, sauf bien sûr pour les mots composés comme indiqué ci-dessus.

upmendex [6] est un autre programme adapté au codage UTF-8, il est écrit en C, utilise la bibliothèque UCA d'origine et non le portage en Lua de Michal Hoftig. C'est le programme de tri d'index qui donne actuellement les meilleurs résultats en français.

Voyons maintenant comment utiliser `xindex` et `upmendex` sur un exemple compact mais exigeant en termes de tri. Le source LaTeX est le suivant :

Exemple 11

```
\documentclass[french]{article}
\usepackage{babel}
% Pour xindex
```

```
\usepackage{xindex}\makeindex
% Pour upmendex
%\usepackage{makeidx}\makeindex

\newcommand*{\IND}[1]{\index{#1}#1\par}
\setlength{\parindent}{0pt}
\begin{document}
\thispagestyle{empty}

\IND{CÔTÉ} \IND{cote} \IND{çà et là} \IND{Côté} \IND{Bohémien}
\IND{L'Haÿ-les-Roses} \IND{côte} \IND{COTE} \IND{tire-clou}
\IND{côté} \IND{Coté} \IND{C.A.F.} \IND{coté} \IND{l'âme}
\IND{Cote} \IND{MacArthur} \IND{élevé} \IND{CÆSIUM} \IND{lamé}
\IND{CÔTE} \IND{Mc Arthur} \IND{tirebouchon} \IND{Côte}
\IND{lésé} \IND{MÂCON} \IND{co-op} \IND{péché} \IND{tirefond}
\IND{bohème} \IND{lame} \IND{McArthur} \IND{gêné} \IND{pêche}
\IND{cæsium} \IND{tire-dent} \IND{Mc Mahon} \IND{pêcher}
\IND{révèle} \IND{cafard} \IND{relevé} \IND{Bohême} \IND{pécher}
\IND{relève} \IND{PÉCHÉ} \IND{vice versa} \IND{maçon}
\IND{coop} \IND{Élève} \IND{gêne} \IND{CŒUR} \IND{vice-président}
\IND{gène} \IND{COTÉ} \IND{caennais} \IND{lèse} \IND{coexistence}
\IND{cœur} \IND{coefficient} \IND{tire-d'aile \IND{révélé}

\newpage
\IND{coté} \IND{cœur}  \IND{tire-d'aile} \IND{péché}
\newpage
\IND{coté} \IND{vice versa} \IND{coefficient}

\printindex
\end{document}
```

Utilisation de `xindex`

À l'issue d'une première compilation créant le fichier `.idx`, on exécute la commande :
`xindex -l fr -i -c let-gut test-french`
qui crée le fichier `.ind`.

L'option `-l fr` (*indispensable*) impose la langue française pour le tri.

L'option `-i` (`ignore-spaces`) améliore le classement de *ça et là* qui est correctement classé entre *CÆSIUM* et *cafard* ; sans cette option, il serait classé en deuxième position juste après *C.A.F.*. Cette option n'a aucune influence sur le classement des mots composés comme *tire-bouchon* ou *l'âme*.

L'option `-c let-gut` charge le fichier de configuration `xindex-letgut.lua` qui influe sur la présentation de l'index à la manière d'un fichier `.ist` pour `makeindex` ou `upmendex`.

Une seconde compilation produit l'index de gauche, figure 1 page 34.

Utilisation de `upmendex`

`upmendex` est (presque) totalement compatible avec `makeindex`, les options sont les mêmes (à l'exception de `-g` qui concerne l'allemand sous `makeindex` et le japonais sous `upmendex`).

L'option `-s` permet de charger un fichier de style : noter que upmendex accepte plusieurs options `-s` successives, et que les fichiers de style conçus pour `makeindex` fonctionnent à l'identique sous upmendex.

Le choix de la langue de base pour le tri se fait dans un fichier de style `.ist`. En voici un exemple adapté au français, appelons-le `french.ist` :

Exemple 12

```
% Règles de tri pour le français :
icu_attributes "french-collation:on strength:tertiary alternate:
    shifted"
```

Pour produire un index d'aspect semblable au précédent on ajoute le fichier de configuration `myindex.ist` (syntaxe de `makeindex`) :

Exemple 13

```
% insertion avant la lettre
heading_prefix "{\\bfseries "
% ajout après la lettre
heading_suffix "\\hfil}\\nopagebreak\n"
% activation impression lettre
headings_flag  1

% n° de page
delim_0        ", "
delim_1        ", "
delim_2        ", "
```

La commande à exécuter pour produire le fichier `.ind` est la suivante :
`upmendex -s french.ist -s myindex.ist test-french`
Une description plus complète de upmendex est donnée dans *The LaTeX Companion*, vol. II, p. 364-370.

Une seconde compilation produit l'index de droite de la figure 1 page suivante.

Comparaison des deux index

`xindex` comme upmendex trient parfaitement les caractères accentués de notre langue, aucune inversion n'est à déplorer dans la liste canadienne.

La différence est perceptible sur les locutions et mots composés.

— L'option `-i` de `xindex` fonctionne bien pour les locutions ; les espaces étant ignorés, *ça et là* est bien classé entre *CÆSIUM* et *cafard* alors que sans cette option il serait en deuxième position juste derrière *C.A.F.* (le point est classé avant toute lettre). En revanche, le classement des mots composés (contenant une apostrophe ou un tiret) laisse à désirer : *l'âme* et *L'Haÿ-les-Roses* sont classés avant *lame*, de même *tire-clou* est avant *tirebouchon*, etc. Ceci résulte de la non prise en compte du quatrième niveau de tri dans l'implémentation lua-uca.

— upmendex donne un résultat parfaitement conforme aux attentes : *C.A.F.* est classé juste avant *cafard* et non plus en tête de liste, et les mots commençant par *tire* avec ou sans trait d'union sont dans le bon ordre.

Index

Index

Figure 1 – À gauche `xindex`, à droite `upmendex`

Conclusion

upmendex est actuellement le plus performant des programmes de création d'index, xindex vient juste derrière mais il est plus simple à utiliser et bien suffisant en pratique. Pour les départager, il a fallu recourir à une liste de mots vraiment sélective : qui a vraiment besoin de classer correctement *l'âme* ou *L'Haÿ-les-Roses* dans un index ? D'autre part, les traits d'union ont tendance à disparaître depuis la réforme de 1990 ; quant aux abréviations, on écrit plutôt *CAF* que *C.A.F.* pour *Caisse d'allocations familiales*…

La bonne nouvelle est que nous disposons *enfin* de *deux* programmes qui produisent des index conformes à l'ordre lexicographique traditionnel des dictionnaires français, les utilisateurs ont donc le choix : xindex ou upmendex.

Remerciements

Ayant toujours trouvé décevants les résultats obtenus pour le tri des index en français par makeindex et xindy, j'ai découvert il y a deux ans l'existence d'un nouveau programme de tri basé sur Unicode, lua-uca de Michal Hoftig.

Comme beaucoup, je n'avais pas d'idée précise sur le classement des mots *cote*, *côte*, *coté* et *côté* à part ce qui en était dit dans le *LaTeX Companion* à propos de xindy. Suite à mes questions posées sur la liste Typo, Jacques André (merci Jacques !) m'a mis en contact avec Alain LaBonté qui a eu la gentillesse de répondre patiemment à mes interrogations, je l'en remercie bien vivement.

Une fois connue la liste de référence établie par les Canadiens, il restait à la confronter aux résultats produits par lua-uca. Hélas, ils n'étaient pas brillants à l'époque, lua-uca ne proposant pas le tri à l'envers des accents, mais c'est maintenant corrigé depuis la version 0.1d. Ensuite, Herbert Voß a intégré cette version de lua-uca dans la version 0.60 de xindex. Merci donc à Michal et Herbert !

Daniel Flipo

Références

[1] *Unicode Collation Algorithm*. Document officiel sur le tri Unicode. URL : https://unicode.org/reports/tr10/.

[2] Patrick Andries. *Unicode 5.0 en pratique*. Dunod.

[3] Michal Hoftig. *Package lua-uca disponible sur CTAN*. URL : https://github.com/michal-h21/lua-uca/.

[4] *Norme canadienne CAN/CSA Z243.4.1-98*. Rechercher le fichier SGQRI004.pdf.

[5] Herbert Voß. *Package xindex disponible sur CTAN*. URL : https://gitlab.com/hvoss49/xindex.

[6] Takuji Tanaka. *Package upmendex disponible sur CTAN*. Doc : upmendex.man1.pdf. URL : https://github.com/t-tk/upmendex-package.

[7] Frank Mittelbach et Ulrike Fischer. *The LaTeX Companion*. 3e éd. Boston : Addison-Wesley Professional, avr. 2023. 1569 p. ISBN : 978-0-1381-6648-9.

[8] Frank Mittelbach et Michel Goossens. *LaTeX Companion*. Trad. par Jacques André et al. 2e éd. Paris : Pearson Education France, oct. 2005. 1120 p. ISBN : 978-2-7440-7182-9.

LUA(LA)TEX : COMMENT AVOIR ACCÈS À LA FABRICATION D'UN PARAGRAPHE

Comme nous l'a expliqué Didier Verna dans son exposé donné dans le cadre des exposés mensuels de GUTenberg[40] « *TEX est aujourd'hui encore un standard de-facto en matière de mise en forme typographique. Une part non négligeable de son succès est due à l'algorithme de justification de paragraphe dont il est équipé, le fameux "Knuth-Plass", conçu et développé entre 1977 et 1982, et que Donald Knuth lui-même a décrit comme "probablement l'algorithme le plus intéressant de TEX".* » Une des choses fabuleuses avec LuaTEX [1] est qu'il donne accès à quelques étapes de cet algorithme et permet d'en modifier le comportement. Pour un tour d'horizon, nous vous renvoyons au *Cahiers* dédié à LuaTEX [2], qui, malgré ses quelques années, reste une très bonne introduction.

Nous allons ici reprendre[41] en grande partie l'excellent article de Paul Isambert dans le *TUGboat* « LuaTEX : What it takes to make a paragraph » [3] en l'adaptant à LuaLATEX et à la situation en 2024. Cet article se situe entre le plagiat et la traduction donc…

Exécuter du code Lua

Avec le moteur LuaTEX, on a accès au langage Lua. Nous ne pouvons pas proposer ici une introduction à ce langage, et nous supposerons que les bases de programmation sont connues (la syntaxe n'est cependant pas trop difficile à comprendre).

La commande (primitive) du moteur LuaTEX[42] qui permet d'exécuter du code Lua dans un fichier source (LA)TEX est `\directlua`. L'argument de cette commande est le code Lua à exécuter. L'exemple classique d'utilisation de cette commande est le suivant :

Exemple 14

code

```
$\pi=\directlua{tex.print(math.pi)}$
```

résultat

π = 3.1415926535898

On peut ainsi très simplement afficher les valeurs de la suite de Fibonacci :

Exemple 15

code

```
\directlua{
function fibonacci (n)
    if n == 0 then
        return 0
    elseif n == 1 then
        return 1
    else
        return fibonacci(n-1) + fibonacci(n-2)
```

40. https://www.gutenberg-asso.fr/11-janvier-2024-Expose-sur-l-algorithme-de-Knuth-Plass.

41. Vous pouvez aussi visionner la vidéo d'un exposés mensuel GUTenberg que j'ai donnée le 12 septembre 2024 : https://www.gutenberg-asso.fr/12-septembre-2024-LuaLaTeX-introduction-a-l-utilisation-de-quelques-callbacks.

42. Et donc celle accessible peu importe le format : plain TEX, LATEX, etc.

code (suite)

```
    end
end
}

Le terme de rang $16$ de la suite de Fibonacci est :
\[u_{16}=\directlua{tex.print(fibonacci(16))}.\]
```

résultat

Le terme de rang 16 de la suite de Fibonacci est :

$$u_{16} = 987.$$

On voit au passage la fonction `print` Lua du module `tex` (et donc l'appel `tex.print`) qui permet de transmettre un affichage de Lua à TEX.

Comme il s'agit de code Lua dans un fichier source LATEX, on se doute bien qu'il peut y avoir des soucis dès qu'on utilise des caractères spéciaux de TEX dans le code Lua. Nous ne nous attarderons pas sur ces problèmes, cependant nous conseillons l'utilisation du package luacode [4] qui fournit des commandes et des environnements bien pratiques pour exécuter du code Lua en gérant peut-être de façon plus intuitive les divers développements et autres caractères spéciaux. De plus, luacode fournit deux commandes `\LuaCodeDebugOn` et `\LuaCodeDebugOff` qui permettent de faire afficher dans le fichier de log, le code Lua écrit dans le source TEX tel qu'il sera lu par l'interpréteur Lua. Ainsi, dans cet article, nous utiliserons principalement l'environnement `luacode` et sa variante étoilée pour les longs blocs de code Lua, et `\luadirect` et `\luaexec` pour les commandes analogues à `\directlua`. Sauf pour l'environnement `luacode*`, les arguments sont développables et donc on peut transférer l'effet d'une commande TEX à Lua.

Le tableau de la documentation de luacode fait un résumé des différences entre ces environnements et commandes :

	`\luadirect`	`\luaexec`	`luacode`	`luacode*`
Macros	Oui	Oui	Oui	Non
Simple contre-oblique	`\string\`	`\string\`	`\string\`	Simplement `\`
Double contre-oblique	`\string\\`	`\\`	`\\`	`\\`
Tilde	`\string~`	`~`	`~`	`~`
Dièse	`\string#`	`\#`	`\#` ou `#`	`#`
Pourcent	Pas de façon simple	`\%`	`\%` ou `%`	`%`
Commentaire TEX	Oui	Oui	Non	Non
Commentaire Lua	Non	Non	Oui	Oui

Exemple 16

code

```
\begin{luacode}
-- fonction Lua définie par récurrence pour le calcul
-- du terme de rang n de la suite de Fibonacci
function fibonacci(n)
    if (n==0) then
        return 0
```

code (suite)

```
    elseif (n==1) then
        return 1
    else
        return fibonacci(n-1)+fibonacci(n-2)
    end
end
\end{luacode}
\def\NN{16}
Le terme de rang $16$ de la suite de Fibonacci est :
\[u_{16}=\luaexec{% on affiche le résultat calculé par Lua
    tex.print(fibonacci(\NN))}.\]
```

résultat

Le terme de rang 16 de la suite de Fibonacci est :

$$u_{16} = 987.$$

Du source au paragraphe

Pour produire un paragraphe, de nombreuses étapes sont nécessaires : les caractères sont lus, interprétés, exécutés, les glyphes sont produits, les lignes sont mesurées, ajustées, etc. Avec LuaTEX, comme nous le disions, on peut avoir accès à toutes ces étapes, décortiquer ce qui se passe, et, dans une certaine mesure, en modifier le comportement. C'est ce que nous allons voir dans cet article.

Callbacks

Les *callbacks* sont des fonctions accessibles par l'utilisateur qui implémentent les opérations internes de TEX ; celles-ci peuvent être améliorées en les combinant avec d'autres fonctions *ad hoc*, mais on peut aussi les remplacer complètement. Tous les *callbacks* sont écrits en Lua.

Nous allons voir que, comme les fonctions enregistrées comme des *callbacks* interviennent dans une séquence d'actions qui constitue le traitement par le moteur TEX, beaucoup d'entre elles recevront des arguments (et doivent donc être définies pour les gérer) et, surtout, beaucoup sont censées renvoyer des valeurs. Les arguments passés, le cas échéant, et les valeurs retournées varient d'un *callback* à l'autre.

Alors qu'il existe de très nombreux *callbacks* pour énormément d'opération du moteur[43], nous ne nous intéresserons ici qu'aux principaux *callbacks* qui servent à la production d'un paragraphe :

`process_input_buffer` qui détermine comment TEX lit chaque ligne en entrée (source) ;

`hyphenate` qui détermine où et comment sont insérées les césures ;

`ligaturing` qui détermine où les ligatures doivent être produites ;

`kerning` qui détermine où les crénages sont insérés ;

`pre_linebreak_filter` qui permet de faire des opérations avant que le paragraphe soit construit ;

`linebreak_filter` qui construit le paragraphe ;

43. Pour trouver des fichiers, lire des fichiers, gérer l'affichage, et traiter des données : https://wiki.luatex.org/index.php/Callbacks.

post_linebreak_filter qui permet de faire des opérations après que le paragraphe est construit.

Si on veut connaître l'ensemble des *callbacks* fournis par LuaTEX, on peut utiliser la fonction Lua `callback.list()`. L'exemple suivant permet d'afficher la liste dans le terminal à la compilation du fichier avec LuaLATEX.

Exemple 17

```
\directlua{
    info = callback.list()
    for k, v in pairs(info) do
        print(k)
    end
}
```

Enregistrer un *callback*

Il y a deux types de *callbacks* : ceux qui étendent la fonctionnalité existante avec une nouvelle, et ceux qui (quand cela est possible) remplacent la fonctionnalité existante. Évidemment, lorsqu'on remplace la fonctionnalité existante, il est attendu que notre nouvelle définition fonctionne de façon similaire.

Pour enregistrer un des *callbacks* listés ci-dessus, on utilisera la syntaxe (Lua) suivante :

Exemple 18

```
id, error = callback.register(<string> callback_name, <function>
    func)
```

Si l'on souhaite rétablir le comportement par défaut, il suffira d'exécuter le code suivant :

Exemple 19

```
id, error = callback.register(<string> callback_name, nil)
```

L'argument à `nil` rétablira le comportement par défaut du *callback*.

Ces fonctions sont celles du moteur LuaTEX. L'utilisation du format LATEX peut poser quelques problèmes. David Carlisle et Joseph Wright ont développé le package ltluatex (présent dans le noyau LATEX) qui fournit des commandes TEX et des fonctions Lua adaptées au format. En particulier, on utilisera les fonctions, aux noms plus explicites :

Exemple 20

```
luatexbase.add_to_callback(<string> callback_name, <function>
    func, <string> description)
```

et

Exemple 21

```
luatexbase.remove_from_callback(<string> callback, <string>
    description)
```

Lire les lignes en entrée

Le *callback* `process_input_buffer` est exécuté lorsque TEX a besoin d'une ligne en entrée. Cette fonction Lua a donc un argument qui se trouve être la chaîne de caractères (le type Lua `string`) de la ligne et elle doit retourner une ligne, possiblement la même. Dans le fonctionnement par défaut de LuaTEX, rien ne se passe à cette étape et TEX lit la chaîne de caractères de la ligne du document source. À ce stade de la construction du paragraphe, la ligne n'a pas été traitée du tout : le contenu mis après un signe de commentaire est bien présent, les espaces multiples n'ont pas été réduites à des espaces uniques, etc. La chaîne de caractères contient donc exactement la ligne de texte du document source.

On peut se demander si avoir accès à ce *callback* est utile. Paul Isambert, dans son article, propose deux exemples d'utilisations : lire une entrée dans n'importe quel codage (UTF-8, ASCII, etc.) ; composer du texte verbatim. Nous allons étudier ici cette deuxième application.

Une des principales difficultés de l'écriture de texte verbatim avec TEX réside dans la gestion des *catcode*[44]. Tout devient encore plus compliqué lorsque l'on souhaite composer le texte verbatim et l'exécuter (on utilise souvent pour cela un fichier externe).

Avec le callback `process_input_buffer` tout devient plus simple. Les lignes du source peuvent être stockées et utilisées ensuite de différentes manières.

Nous allons voir qu'on peut définir un couple de commande `\myVerbatim` et `\Endverbatim`, la première enregistrant une nouvelle fonction Lua au *callback* qui stockera les lignes dans une table (Lua) jusqu'à ce que le *callback* soit réinitialisé lorsque la ligne `\Endverbatim` est rencontrée. La table ainsi construite pourra être utilisé soit pour afficher le code, soit pour l'exécuter.

Le code Lua permettant de stocker les lignes entre `\myVerbatim` et `\Endverbatim` est le suivant :

Exemple 22

```lua
local verb_table
local function store_lines (str)
    if str == "\\Endverbatim" then
        luatexbase.remove_from_callback("process_input_buffer",
            "Store lines of verbatim")
    else
        table.insert(verb_table, str)
    end
    return ""
end
```

Ce code est fait pour être exécuté avec l'environnement `luacode` et donc pour obtenir la contre-oblique de la macro TEX `\Endverbatim`, il nous faut utiliser `\string\`. On peut aussi utiliser l'environnement `luacode*` pour que le code Lua ne soit pas développé au sens de TEX.

44. La page « Que sont les catcodes ? » (https://faq.gutenberg-asso.fr/2_programmation/syntaxe/catcodes/que_sont_les_catcodes.html) nous explique ce que sont les *catcodes* : « plutôt que de définir des primitives pour des tâches aussi courantes que le passage en mode mathématique ou la mise en exposant ou en indice, Donald Knuth a préféré réserver certains caractères à ces tâches. Par exemple le dollar pour passer en mode mathématique, l'underscore pour passer en indice. D'autres caractères ont des significations particulières lorsqu'on écrit un document TEX : les accolades et crochets pour délimiter les arguments des commandes, ou simplement l'antislash pour appeler les commandes. Certains caractères sont différents des autres pour TEX. »

La fonction `store_lines` ajoute donc chaque ligne dans la table *globale* `verb_table` sauf si la ligne contient uniquement `\Endverbatim` (on pourrait aussi utiliser une *expression régulière* pour détecter `\Endverbatim` parmi du texte), auquel cas le *callback* est réinitialisé[45]. Très important : cette fonction retourne une chaîne de caractères vide car si elle ne retournait rien, LuaTEX traiterait cela comme si le *callback* n'avait pas été utilisé, et la ligne originale serait affichée.

Pour rendre cette fonction active dans le callback `process_input_buffer`, on va définir la fonction Lua `register_verbatim` suivante :

Exemple 23

```
function register_verbatim ()
    verb_table = {}
    luatexbase.add_to_callback("process_input_buffer",
        store_lines, "Store lines of verbatim")
end
```

Cette fonction réinitialise la table `verb_table` dans laquelle nous stockons les lignes de texte verbatim et appelle la fonction `callback.register` vue plus haut.

On va alors définir la macro `\myVerbatim` qui active le *callback* modifié :

Exemple 24

```
\newcommand\myVerbatim{\luadirect{register_verbatim()}}
```

Avec cette façon de faire, nous n'avons pas besoin de définir `\Endverbatim` car cette ligne sera « mangée » par notre *callback*, et donc jamais vue par TEX.

On va maintenant fabriquer une fonction Lua qui permet d'afficher les lignes stockées dans `verb_table` soit après traitement par (LA)TEX, soit verbatim.

Exemple 25

```
function print_lines (catcode)
    if catcode then
        tex.print(catcode, verb_table)
    else
        tex.print(verb_table)
    end
end
```

La fonction Lua `print` du module `tex`, fourni donc dans le moteur LuaTEX, accepte comme argument soit une chaîne de caractères soit une table de chaînes de caractères (ici notre `verb_table`). De plus, il est possible d'avoir un argument optionnel (un entier relatif) permettant de définir le régime de *catcode* à utiliser pour les caractères. Pour définir une table de *catcode* (le régime de *catcode*), on utilisera le code suivant :

45. La description `"Store lines of verbatim"` est celle qui a été utilisée pour enregistrer `store_lines` dans le *callback* `process_input_buffer` (voir plus bas). Ainsi, lorsque la fonction `store_lines` arrive à `\Endverbatim`, elle se supprime elle-même du *callback* `process_input_buffer` et n'est donc plus appelée quand la ligne suivante est lue.

Exemple 26

```
\newcatcodetable\verbcatcode
\newcommand\createcatcodes{
    \begingroup
    \catcode`\\ = 12
    \catcode`\{ = 12
    \catcode`\} = 12
    \catcode`\$ = 12
    \catcode`\& = 12
    \catcode`\# = 12
    \catcode`\^ = 12
    \catcode`\_ = 12
    \catcode`\% = 12
    \catcode`\  = 13
    \catcode`\^^M=13
    \savecatcodetable\verbcatcode
\endgroup}
\createcatcodes
```

Ici, on crée donc un contexte dans lequel tous les caractères spéciaux de TEX deviennent des caractères normaux. Les commandes `\newcatcodetable` et `\savecatcodetable` ont été utilisées pour créer cela. On déclare donc la table que l'on nomme `\verbcatcode` et on sauve les enregistrements de catcodes grâce à `\savecatcodetable`. Le passage par une commande `\createcatcodes` est une façon de gérer le fait que, dès lors qu'on utilise `\catcode`\\ =12` qui transforme donc la contre-oblique en caractère normal, on perd la possibilité d'écrire des commandes.

Ceci fait, il ne nous reste qu'à définir deux commandes : `\useverbatim` et `\printverbatim` pour respectivement produire le résultat du code et afficher le verbatim du code.

Exemple 27

```
\newcommand\useverbatim{%
    \luadirect{print_lines()}%
}
\newcommand\printverbatim{%
    \bgroup\parindent=0pt \ttfamily
    \luadirect{print_lines(luatexbase.catcodetables.verbcatcode)}
    \egroup
}
```

On remarquera que notre `\savecatcodetable\verbcatcode` a généré, du côté de Lua, la valeur (un entier) `luatexbase.catcodetables.verbcatcode`.

Une fois tout cela fait, on pourra utiliser notre machinerie comme l'illustre l'exemple suivant.

Exemple 28

code

```
\myVerbatim
\newcommand\myluatex{%
  Lua\kern-.01em\TeX
}%
```

code (suite)

```
\Endverbatim
\useverbatim % fabriquer vraiment la commande
Avec le code :\par
\printverbatim\par
on définit la commande permettant de générer \myluatex.
```

résultat

Avec le code :
```
\newcommand\myluatex{%
Lua\kern-.01em\TeX
}%
```
on définit la commande permettant de générer LuaTEX.

Tout cela ouvre la porte à de nombreuses améliorations. On peut imaginer, grâce à un paramètre ajouté aux commandes `\printverbatim` et `\useverbatim`, ne spécifier qu'une partie du code à afficher ou utiliser. On pourrait aussi, au moment de la composition du verbatim, entrecouper les lignes de verbatim de macros qui, elles, obéiraient à un régime classique de *catcode*.

Insérer des césures

Nous allons désormais nous intéresser aux nœuds que TEX a créés et concaténés dans une liste horizontale, c'est-à-dire, pour simplifier, la liste de mots et de blancs qui constituent le prochain paragraphe[46]. C'est à ce moment que la composition commence réellement. Le *callback* `hyphenate` reçoit une liste de nœuds (*nodes* en anglais) qui est le matériel de base à partir duquel est construit un paragraphe. Ce *callback* doit ajouter les points de césure, ce qu'il fait par défaut si aucune fonction n'a été enregistrée comme *callback*.

Dans ce *callback*, mais aussi dans d'autres, il est souvent intéressant de savoir quels nœuds sont passés en argument. Pour cela, voici une fonction Lua qui prend une liste de nœuds et qui affiche leur champs `id` (d'identification) dans le terminal et dans le fichier de journal `.log`[47] ou, si le nœud est un glyphe (`id=37`, mais il est préférable de récupérer la valeur avec la fonction `node.id`), affiche directement le caractère.

Exemple 29

```lua
local GLYF = node.id("glyph")
function show_nodes (head)
    local nodes = ""
    for item in node.traverse(head) do
        local i = item.id
        if i == GLYF then
            i = unicode.utf8.char(item.char)
        end
        nodes = nodes .. i .. " "
    end
    texio.write_nl(nodes)
```

46. Dans son article, Paul Isambert illustre avant l'utilisation du callback `token_filter`. Cependant, depuis l'écriture de cet article, ce callback a été enlevé de LuaTEX.

47. La correspondance entre nombre et type de nœud est spécifiée dans le chapitre 8 du manuel de référence de LuaTEX [1].

```
12  end
```

On va enregistrer cette fonction dans le callback `hyphenate` comme nous avons fait pour les callbacks précédemment. Ainsi, le code suivant :

Exemple 30

```
1  \luadirect{
2      luatexbase.add_to_callback("hyphenate",show_nodes,"Show nodes
       ")
3  }
4  Est-ce effectif?
5
6  \luadirect{
7      luatexbase.remove_from_callback("hyphenate","Show nodes")
8  }
```

produira dans le terminal la sortie suivante :

```
41 9 0 E s t - c e 12 e f f e c 7 t i f ?
```

Notre fonction montre donc que le callback `hyphenate` reçoit la liste :

41 9 0 E s t - c e 12 e f f e c 7 t i f ?

Le premier nœud, identifié par le nombre 41, est un nœud présent pour des raisons techniques qui ne nous intéressent pas ici. S'ensuit le nœud d'identification 9 qui est un *whatsit* (élément extraordinaire[48]), et son sous-type est 6 ce qui indique que c'est un *whatsit* `local_par` qui contient, parmi d'autres choses, la direction de composition du paragraphe (gauche-droite ici). Le troisième nœud est une liste horizontale (identificateur 0), c'est-à-dire une `hbox`. Son sous-type est 3 ce qui indique que c'est une boîte d'indentation, et si on demande sa largeur, la valeur sera celle de `\parindent` en *scaled point*.

Les nœuds représentant des caractères ont de nombreux champs parmi lesquels le champ `char` (un nombre) que notre fonction `show_nodes` utilise pour afficher quelque chose de plus parlant qu'un simple entier (grâce à la fonction `unicode.utf8.char()`). Les autres champs sont `width`, `height`, `depth` (des nombres aussi en *scaled points*) et `font` (encore un nombre car les fontes sont aussi représentées en interne par des nombres). Leurs sous-types nous intéresseront plus tard.

Enfin, les nœuds avec l'identificateur 12 sont des *glues*, qu'on appellera *ressorts* en français, c'est-à-dire l'espace entre deux mots et l'espace qui se trouve à la fin de ligne du paragraphe (qui n'est pas là si le dernier caractère est immédiatement suivi par un `\par` ou un signe de commentaire). Les ressorts sont, avec les boîtes et les dimensions, parmi les types d'objet de données dans TEX traditionnel qui ne sont pas des valeurs simples. Ils sont insérés lorsque TEX voit un espace dans le flux de texte, mais aussi par `\hskip` et `\vskip`. La structure qui représente les composants des ressorts est appelée `glue_spec`, et elle a les champs `width`, `stretch`, `stretch_order`, `shrink` et `shrink_order`.

Maintenant, que peut-on faire avec ce *callback*? Bien, tout d'abord, on peut insérer des points de césure dans notre liste de nœuds comme LuaTEX le fait par lui-même, si nous n'avions pas modifié le *callback*. La fonction `lang.hyphenate` permet de faire cela :

48. Voir le lexique français-anglais de la FAQ GUTenberg : https://faq.gutenberg-asso.fr/1_generalites/documentation/comment_traduire_ce_terme.html.

Exemple 31

```
\begin{luacode}
function myhyph (head, tail)
    lang.hyphenate(head)
    show_nodes(head)
end
\end{luacode}
\luadirect{
    luatexbase.add_to_callback("hyphenate", myhyph, "My Hyphenate
    ")
}
Est-ce effectif?

\luadirect{
    luatexbase.remove_from_callback("hyphenate", "My Hyphenate")
}
```

Le terminal nous montre alors la liste suivante :

41 9 0 E s t 7 c e 12 e f 7 f e c 7 t i f ?

Dans la fonction Lua `myhyph`, nous n'avons pas besoin de retourner une liste puisque LuaTEX s'occupe de cela dans à l'intérieur de ce *callback*, comme c'est aussi le cas dans les *callbacks* `ligaturing` et `kerning`. Aussi, ces trois *callbacks* prennent deux arguments : `head` et `tail`, repectivement le premier et dernier nœud de la liste à traiter. Le dernier nœud peut en général être ignoré.

Maintenant, regardons ce que l'insertion de points de césure a produit. Comme attendu, des éléments ont été ajoutés avec l'identification 7 : ce sont des nœuds de césure avec trois champs `pre`, `post` et `replace` qui sont les équivalents des premier, deuxième et troisième arguments de la macro `\discretionary`. Le `pre` est la liste des nœuds à insérer avant une coupure de ligne, le `post` est la liste des nœuds à insérer après une coupure de ligne, et le `replace` est la liste de nœuds à insérer si ce point de césure n'a pas été choisi. Ici, nous avons trois nœuds avec l'identificateur 7 :

entre `Est` et `ce` : le caractère initial est un tiret, et ainsi, le nœud devient un point de césure potentiel. Le nœud a donc un champ `replace` contenant le caractère `-` à afficher lorsqu'il n'y pas de coupure à cet endroit, et classiquement, un champ `pre` contenant lui aussi `-` (à n'afficher que lorsque la ligne se coupe à cet endroit) et un champ `post` vide.

entre les deux `f` et entre le `c` et le `t` : Ici, il s'agit de point de césure très classique, sans champ `replace`, un champ `pre` contenant `-`, et un champ `post` vide.

Un dernier mot sur les césures. Avec TEX, il est possible d'ajouter des mots dans le dictionnaire d'exception avec la commande `\hyphenation`. Sa syntaxe est simple, il suffit d'écrire les mots en indiquant les points de césures par un tiret :

Exemple 32

```
\hyphenation{man-u-script man-u-scripts ap-pen-dix}
```

Cette primitive a été étendue avec LuaTEX et permet désormais d'utiliser l'équivalent de la structure `pre`, `post` et `replace` en insérant à la place du simple tiret une séquence de trois groupes : `{pre}{post}{replace}`. Les utilisateurs allemands (et sans doute d'autres

nombreuses nationalités) seront ravis de ne plus avoir à donner une attention particulière à leur *backen* dans leur document : une simple déclaration comme la suivante devrait suffir :

Exemple 33

```
\hyphenation{ba{k-}{}{c}ken}
```

Enfin, comme on le voit avec `Est-ce`, avec un trait d'union en premier et troisième argument, on configure très simplement les points de césure des mots composés.

Ligatures

Comme son nom[49] l'indique, le *callback* `ligaturing` doit s'occuper d'insérer les ligatures (et ceci se produit par défaut si aucune fonction additionnelle n'est enregistrée au *callback*). Comme précédemment, on va utiliser notre fonction `show_nodes` juste après la fonction Lua `node.ligaturing` lors d'un enregistement dans le *callback* `ligaturing`. Cela donne le code suivant :

Exemple 34

```
\begin{luacode}
function mylig (head, tail)
    node.ligaturing(head)
    show_nodes(head)
end
\end{luacode}
\luadirect{
    luatexbase.add_to_callback("ligaturing", mylig, "My
    Ligaturing")
}
Est-ce effectif?

\luadirect{
    luatexbase.remove_from_callback("ligaturing", "My Ligaturing")

}
```

Cependant, ce code, écrit initialement pour LuaTEX par Paul Isambert, ne produit pas le résultat escompté avec LuaLATEX. En effet, LuaLATEX charge désormais par défaut une fonte OTF, et le travail de ligature est alors laissé à la fonte. Le callback `ligaturing` devient alors une coquille vide qui n'a pas vraiment d'utilité. Pour retrouver le fonctionnement de LuaTEX de base, on pourra (mais ce n'est pas à conseillé) forcer l'utilisation d'une fonte T1 avec le chargement du package fontenc :

Exemple 35

```
\usepackage[T1]{fontenc}
```

Si l'on fait cela, alors, nous aurons le résultat suivant : 41 9 0 E s t 7 c e 12 e 7 e c 7 t i f ? □

49. Même en anglais, cela se laisse entendre.

Que s'est-il passé ? Pourquoi *effectif* a-t-il été raccourci de la sorte ? Simplement parce qu'il y a une interaction entre la création des césures et la création des ligatures. Si le point de césure dans *effectif* est choisi, le résultat sera alors `ef-fectif`. Si le point n'est pas choisi alors le résultat sera `e<ff>ectif` où `<ff>` représente la ligature. En d'autres termes, la présence de ligatures dépend des césures. Ici, le nœud de césure a son champ `pre` égal à `f-`, `f` dans le champ `post` et `<ff>` dans le champ `replace`.

Les nœuds de ligatures sont des nœuds de glyphe avec un `subtype` qui vaut 2, alors qu'un nœud de glyphe classique a un `subtype` de 1. Ainsi, ils ont un champ spécial, `components`, qui pointe vers une liste de nœud composée des glyphes individuels qui constituent la ligature. Les composants de `<ff>` sont donc `f` et `f`. Les ligatures peuvent ainsi être décomposées quand cela est nécessaire.

Comment LuaTEX (soit avec le fonctionnement par défaut du *callback* `ligaturing`, soit avec la fonction Lua `node.ligaturing`) fait-il pour savoir quelle séquence de nœuds de glyphes doit-être transformée en ligature ? L'information est codée dans la fonte : LuaTEX regarde dans la table `ligatures` associée (si elle existe) à chaque caractère, et si le caractère suivant est présent dans cette table, alors la ligature est créée. Par exemple, pour le `f` de *Computer Modern Roman*, la table `ligatures` a une entrée à l'index 105, ce qui correspond au `i`, et qui contient la ligature `<fi>`. Ainsi, LuaTEX ne traite finalement pas autre chose que des ligatures de deux glyphes. La ligature `<ffi>` est le résultat d'une ligature entre `<ff>` et `i`.

Cependant, les fontes, et particulièrement les fontes OTF, peuvent définir des ligatures avec plus de deux glyphes ; par exemple, l'entrée `3/4` peut produire quelque chose comme ¾ (un glyphe unique). On peut choisir, lorsqu'on crée la fonte à partir du fichier OTF, de définir une ligature *fantôme* `<3/>` et de configurer ¾ comme la ligature entre notre ligature fantôme `<3/>` et `4`. Avec ce mécanisme LuaTEX peut gérer cela automatiquement. Il est sans doute plus élégant et générateur de moins d'erreurs, de gérer ce type de ligatures à la main, c'est-à-dire d'enregistrer une fonction dans le *callback* `ligaturing` qui à partir d'une suite de nœud, crée la ligature. C'est cependant plus lent.

C'est aussi dans ce *callback* que des choses comme les substitutions contextuelles peuvent être orchestrées. Par exemple, les formes initiales ou finales des glyphes, qui sont différentes par exemple en arabe ou dans quelques fontes latines sophistiquées, peuvent être gérées ici. En théorie, c'est assez simple : il suffit d'analyser le contexte d'un nœud particulier, c'est-à-dire les nœuds l'entourant ; si ceux-ci correspondent à un certain contexte d'une certaine forme, alors il suffit de l'utiliser. Par exemple, avec la fonte Minion (disponible avec Adobe Reader), et l'option OTF `ss02` activée, les `r` et les `e` suivis d'une espace (un nœud *glue*) peuvent être remplacés par leurs variantes finales. Dans la pratique, les choses sont un peu plus compliquées, en particulier parce que l'on doit lire ces contextes de substitution depuis les fichiers de fontes.

Cependant, on peut tout de même réaliser un type simple de substitution contextuelle. Le code utilisé pour charger une fonte avec LuaTEX[50] utilise souvent la fonctionnalité `trep` (inspirée de XETEX) pour que l'accent grave ` et les guillemets simples ' soient remplacés par les guillemets anglais simples gauche et droit. Cependant, on peut vouloir permettre une utilisation encore plus facile et utiliser les guillemets doubles droits " partout et qu'ensuite, suivant le contexte, les bons guillemets soient substitués.

Voici un code simple permettant de faire cela : si " est trouvé, le code le remplace par ' si le nœud précédent est un glyphe et que son caractère n'est pas une parenthèse ouvrante, et par ` sinon[51]. C'est ce que proposait Paul Isambert dans son article initial. Ici, nous

50. Et pas avec LuaLATEX et fontspec.

51. Ici, on a utilisé `not (x==y)` au lieu de `(x~=y)`, qui aurait été plus simple, pour éviter le problème du développement du caractère~.

allons le transformer légèrement pour utiliser des guillemets français. Évidemment, notre implémentation est donnée pour l'exemple : elle n'est absolument pas satisfaisante, puisqu'il faudrait ajouter des espaces fines insécables, permettre l'insertion d'espaces dans le source, etc.

Pour obtenir le code d'un caractère, il faut regarder la table unicode et convertir l'index écrit en hexadécimal en base 10. Ainsi, le double guillemets droit se trouve à la place U+0022 ce qui correspond au code décimal 34. Les doubles chevrons se trouvent en place U+00AB et U+00BB qui correspondent respectivement au nombre décimaux 171 et 187.

Exemple 36

code

```
\begin{luacode}
local GLYF = node.id("glyph")
function guillemets(head, tail)
    for glyph in node.traverse_id(GLYF, head) do
        if glyph.char == 34 then
            if glyph.prev and glyph.prev.id == GLYF
                and not (glyph.prev.char == 40)
            then
                glyph.char = 187
            else
                glyph.char = 171
            end
        end
    end
    node.ligaturing(head)
end
\end{luacode}
\luadirect{
    luatexbase.add_to_callback("ligaturing", guillemets, "
    Guillemets")
}
Voici un "test" !

\luadirect{
    luatexbase.remove_from_callback("ligaturing","Guillemets")
}
```

résultat

Voici un «test»!

Cette substitution conditionnelle est un simple exercice, et tout cela suppose de ne pas avoir de cas pathologique, et impose une forme stricte. On peut tout de même s'amuser à améliorer un peu la chose. En effet, les doubles chevrons sont normalement ajoutés avec des espaces fines insécables en français. On va donc les ajouter. Pour ce faire, on va reprendre le code précédent et ajouter un nœud de type `glyph` avec la fonte courante et ayant pour caractère l'espace fine insécable, c'est-à-dire le caractère Unicode U+202F ce qui correspond à l'entier 160 en décimal. Pour cela, à chaque itération lors du parcours des nœuds de la liste, si on trouve un double guillemet droit, nous allons définir un nouveau nœud :

Exemple 37

```
local fine = node.new(GLYF)
fine.font=font.current()
fine.char = 160
```

Ensuite, si on change le caractère pour le double chevrons fermant, on ajoutera le nœud nouvellement créé avant le nœud du double chevrons, et si le caractère est le double chevrons ouvrant, on ajoutera l'espace après. Cela se fait avec les fonctions `insert_before` et `insert_after` de la bibliothèque Lua de LuaTEX node. L'exemple précédent devient alors le suivant.

Exemple 38

code

```
\begin{luacode}
local GLYF = node.id("glyph")
function guillemets(head, tail)
    for glyph in node.traverse_id(GLYF, head) do
        if glyph.char == 34 then
            local fine = node.new(GLYF)
            fine.font=font.current()
            fine.char = 160
            if glyph.prev and glyph.prev.id == GLYF
                and not (glyph.prev.char == 40)
            then
                glyph.char = 187
                head,fine = node.insert_before(head,glyph,fine)
            else
                glyph.char = 171
                head,fine = node.insert_after(head,glyph,fine)
            end
        end
    end
    node.ligaturing(head)
end
\end{luacode}
\luadirect{
    luatexbase.add_to_callback("ligaturing", guillemets, "
    Guillemets")
}
Voici un "test" !

\luadirect{
    luatexbase.remove_from_callback("ligaturing","Guillemets")
}
```

résultat

Voici un « test » !

Insertion de crénages

De façon analogue au *callback* `ligaturing`, le *callback* `kerning` est supposé insérer les crénages de la fonte, et encore une fois, cela se produit si aucune fonction n'est enregistré dans le *callback*. On va de nouveau utiliser notre fonction `show_nodes` pour observer ce qu'il se passe.

Exemple 39

```
\begin{luacode}
function mykern (head, tail)
    node.kerning(head)
    show_nodes(head)
end
\end{luacode}
\luadirect{
    luatexbase.add_to_callback("kerning", mykern, "My Kerning")
}

\fontencoding{T1}
Vous? Est-ce effectif?
```

Le code ci-dessus produira dans le terminal ceci : 41 9 0 V 13 o u s ? 12 E s t 7 c e 12 e 7 e c 7 t i f 13 ?

On constate donc que des nœuds identifiés avec le nombre 13 ont été ajoutés entre le `V` et le `o` et entre le `f` et le `?`. Ces nœuds sont des crénages : le *o* derrière le *V* est plus joli s'il est légèrement rapproché. Comparez « Vo » à « Vo » (le cas de « f? » est beaucoup moins criant). Ces crénages sont définis par les fontes, comme les ligatures, et c'est bien normal puisque les crénages, comme les ligatures, vont dépendre de l'aspect des glyphes.

Comme les ligatures et les substitutions contextuelles, il existe un positionnement contextuel. Les crénages sont codés dans la fonte comme les ligatures, c'est-à-dire que les glyphes ont une table `kerns` indexée avec les index des caractères et les dimensions comme valeurs (en *scaled point*). Ainsi, le crénage est automatique avec les paires de glyphes seulement, et les positionnements contextuels doivent être traités à la main. Par exemple, dans « A.V. », un crénage (négatif) devrait être inséré entre le premier point et le *V* ; cependant, cela doit arriver uniquement lorsque que le point est précédé par *A* ; si la première lettre est un *T* (c'est-à-dire « T.V. »), le crénage n'est sans doute pas nécessaire (ou alors un crénage différent devrait être utilisé).

Enfin, ce *callback* est le bon endroit pour gérer à la main quelques crénages particuliers. Par exemple, selon les règles typographiques du français, une espace fine doit être insérée avant certains signes de ponctuation, sauf quelques exceptions : si le point d'interrogation doit en effet être précédé d'une telle espace, la règle ne s'applique pas si ce signe de ponctuation est placé juste après une parenthèse ouvrante (?) ou un autre point d'interrogation. Cela peut être codé dans la fonte, mais il est plus simple de gérer ce type de comportement dans le *callback* `kerning`. Si on choisit de procéder ainsi, il faudra s'assurer que tous les nouveaux crénages introduits sont de sous-type (`subtype`) 1, parce que les crénages de sous-type 0 sont des crénages de fonte et peuvent être réinitialisés si l'élargissement des glyphes (*font expansion*) est activé.

Un dernier *callback* avant de construire le paragraphe

Le *callback* précédent s'applique peu importe si on on est en train de construire un paragraphe ou simplement une `\hbox`. Les *callbacks* que nous allons voir dans la suite sont utilisés uniquement dans le premier cas, c'est-à-dire (TEXniquement parlant) lorsqu'une commande verticale est rencontrée en mode horizontal. Le premier d'entre eux est le *callback* `pre_linebreak_filter`, et on peut simplement, comme précédemment, regarder ce qu'il se passe avec notre fonction `show_nodes`. Cette fois ci, il faut cependant retourner la liste head des nœuds dans notre *callback*.

Exemple 40

```
\begin{luacode}
function pre_filter (head, groupcode)
    show_nodes(head)
    return head
end
\end{luacode}
\luadirect{
    luatexbase.add_to_callback("pre_linebreak_filter", pre_filter,
     "My Pre")
}

\fontencoding{T1}
Vous? Est-ce effectif?
\luadirect{
    luatexbase.remove_from_callback("pre_linebreak_filter", "My
    Pre")
}
```

Ce code donnera en sortie dans le terminal et le fichier de log la liste suivante : 9 0 V 13 o u s ? 12 E s t 7 c e 12 e 7 e c 7 t i f 13 ? 14 12

Tout d'abord, on s'apperçoit que le premier nœud temporaire au début de la liste (avec l'identifiant 41) a disparu. Ensuite, un nouveau nœud a été inséré avec l'identifiant 14 : il s'agit d'une pénalité. Si on demande d'afficher son champ `penality`, la valeur retournée sera de 10 000. D'où vient cette pénalité infinie ? Les lecteurs et les lectrices savent peut-être que, lorsque TEX prépare un paragraphe, il supprime le dernier espace (c'est-à-dire le dernier nœud `glue`) de la liste horizontale et le remplace par une *glue* de valeur `\parfillskip` qu'il préfixe avec une pénalité infinie pour qu'aucune coupure de ligne ne se produise. C'est ce qui se passe ici, le dernier nœud est une `glue` avec identifiant 12, mais il n'est pas identique aux nœeuds 12 précédents, puisque son sous-type indique qu'il s'agit d'un nœud `\parfillskip`.

Rien de particulier n'est supposé se produire dans le *callback* `pre_linebreak_filter` et TEX ne fait rien par défaut. Ce *callback* est utilisé pour introduire des effets spéciaux avant que la liste soit coupée en lignes : comme on peut le voir dans le code précédent, ses arguments sont la liste de nœuds à traiter et une chaîne de caractères indiquant dans quel contexte le paragraphe est construit. Les valeurs possibles pour ce deuxième argument sont la chaîne vide (si on se trouve dans la liste verticale principale), `"vbox"`, `"vtop"`, et `"insert"`.

Enfin, la construction d'un paragraphe !

Enfin, nous pouvons construire le paragraphe. On utilise alors le *callback* `linebreak_filter` ; par défaut, le paragraphe est construit automatiquement (heureusement), mais, si on enregistre une fonction dans le *callback*, on peut choisir où couper les lignes nous-mêmes. Enfin, plus ou moins : comme précédemment, il y a une fonction `tex.linebreak` qui fait cela.

Le *callback* reçoit deux arguments : une liste de nœuds et un booléen. Ce dernier est `true` si on construit la partie d'un plus grand paragraphe qui se trouve avant un élément mathématique hors ligne ; sinon, il est à `false`. Étant donné une liste de nœuds en entrée, une fonction pour ce *callback* doit retourner une autre liste d'une nature totalement différente : elle doit être une liste de boîtes horizontales (les lignes de texte), des *glues* (le ressorts entre les lignes), des pénalités (par exemple pour les lignes veuves et orpheline), et possiblement des insertions[52] ou du matériel pour ajuster verticalement le contenu, etc. Comme nous l'avons mentionné, la fonction `tex.linebreak` fait tout cela ; cette fonction peut même prendre un argument optionnel, une table avec des paramètres TEX comme clés (par exemple `hsize`, `tolerance`, `widowpenalty`), pour que les paragraphes puissent être construits en tenant compte de valeurs particulières affectées à ces paramètres.

Comme exemple d'utilisation de ce *callback*, nous allons essayer de construire un paragraphe dont la première ligne est en petites capitales, comme cela se fait parfois pour le premier paragraphe d'un chapitre. On utilise LuaLATEX, et donc, on ne souhaite pas utiliser trop d'astuces sophistiquées, et on souhaite que TEX construise quand même le *meilleur* paragraphe possible (c'est-à-dire qu'on ne souhaite pas ajuster simplement, mais probablement disgracieusement, les espaces de la première ligne) : par exemple, on souhaite laisser la possibilité qu'il y ait une césure à la première ligne. Le code qui va suivre est un simple prototype, mais il donne un aperçu d'une approche possible. Premièrement, nous avons besoin d'une fonte et d'une commande pour déclarer que le prochain paragraphe doit être traité spécifiquement. Pour déclarer cette fonte, on utilisera la commande `\font`, et dans notre cas, on utilisera *AlegreyaSC-regular* de la fonte du numéro de cette *Lettre*.

Exemple 41

```
\font\firstlinefont={file:AlegreyaSC-Regular.otf}
```

Ensuite, nous définissons un environnement qui désactive les trois *callbacks* de traitement des nœuds avec `luatexbase.reset_callback("<callback>", false)`, car on souhaite garder la liste originale des nœuds avec seulement les remplacements qui se font avant le *callback* `pre_linebreak_filter`, et nous allons opérer les césures, les ligatures et le crénage à la main. En pratique, il serait préférable de déterminer et sauvegarder les fonctions additionnelles enregistrées dans ces *callbacks*, et utiliser celles-là, pour rendre notre code plus robuste et permettre aux utilisateurs et utilisatrices de l'utiliser avec leurs modifications du comportement par défaut. Cela pourrait être fait, par exemple, avec la commande `callback.find`, mais nous ne nous attarderons pas sur ce point ici. Nous enregistrons ensuite notre fonction `mypar` au *callback* `linebreak_filter` et à la fin de l'environnement, nous rétablissons le comportement par défaut.

Exemple 42

```
\newenvironment{firstparagraph}{\luadirect{
    luatexbase.reset_callback("hyphenate", true)
    luatexbase.reset_callback("ligaturing", true)
```

52. Par exemple des notes de bas de page.

```
    luatexbase.reset_callback("kerning", true)
    luatexbase.add_to_callback("linebreak_filter",mypar,"MyPar")
    }%
}%
{\luadirect{
    luatexbase.reset_callback("hyphenate", false)
    luatexbase.reset_callback("ligaturing", false)
    luatexbase.reset_callback("kerning", false)
    luatexbase.reset_callback("linebreak_filter", false)
    }
}
```

Notre fonction Lua `mypar` est la suivante :

Exemple 43

```
function mypar (head, is_display)
    local par, prevdepth, prevgraf = check_par(head)
    tex.nest[tex.nest.ptr].prevdepth = prevdepth
    tex.nest[tex.nest.ptr].prevgraf = prevgraf
    return par
end
```

Notre fonction va appeler une autre fonction Lua `check_par` (que nous allons détailler plus tard) qui retourne un paragraphe et de nouvelles valeurs pour `prevdepth` et `prevgraf`, que nous pourront affecter au niveau courant d'*emboîtement* (*nesting level*), donc à la liste de boîtes dans laquelle on se trouve.

Ces valeurs sont à retrouver dans la table Lua `tex.nest[tex.nest.ptr]`. Voici une description de l'utilité de ces deux valeurs :

`prevdepth` est la profondeur[53] de la dernière ligne dans une paragraphe construit ;

`prevgraf` est le nombre de lignes d'un paragraphe construit.

Avant d'expliquer en quoi consiste la fonction `check_par`, nous allons présenter ici la fonction auxiliaire qui est utilisée pour faire ce que nous avons empêche LuaTEX de faire : insérer les points de césure, les ligatures et les crénages, et ensuite fabriquer le paragraphe.

Exemple 44

```
local function do_par (head)
    lang.hyphenate(head)
    head = node.ligaturing(head)
    head = node.kerning(head)
    local p, i = tex.linebreak(head)
    return p, i.prevdepth, i.prevgraf
end
```

Il n'est pas nécessaire de réaffecter à `head` la sortie de `lang.hyphenate` car aucun point de césure ne peut être ajouté en premier nœud d'une liste (de toute manière, la sortie de `lang.hyphenate` est un booléen qui indique le succès ou l'échec de l'exécution de la fonction). En plus du paragraphe lui-même, `tex.linebreak` retourne une table avec les

53. Distance maximale sur laquelle le texte se prolonge sous la ligne de base (déterminée par exemple par la longueur du descendant de p s'il y en a un).

valeurs `prevdepth` et `prevgraf` (mais aussi les valeurs `looseness` et `demerits`, qui sont des valeurs utilisées pour la fabrication *optimale* du paragraphe).

On va aussi récupérer la fonte pour les petites capitales que l'on a définie plus tôt côté TEX. Pour cela, on utilisera le code suivant :

Exemple 45

```
local firstlinefont = font.id("firstlinefont")
```

qui nous fournit la représentation numérique de la fonte.

On peut maintenant s'attaquer à la fonction principale `check_par`. Le début du code est le suivant :

Exemple 46

```
local HLIST = node.id("hlist")
local GLYF = node.id("glyph")
local KERN = node.id("kern")
function check_par (head)
    local par = node.copy_list(head)
    par, prevdepth, prevgraf = do_par(par)
    local line = par
    while not (line.id == HLIST) do
        line = line.next
    end
```

Tout d'abord, on crée une copie de la liste de nœuds pour ne pas modifier l'originale, notamment ne pas lui ajouter des points de césure qui seront supprimés par la suite. Ensuite, on lance une première tentative de construction du paragraphe avec notre fonction `do_par`. On cherche alors la première ligne du paragraphe construit en parcourant la table `par` jusqu'à obtenir une boîte horizontale `hlist` (les premiers éléments de la liste du paragraphe peuvent être une *glue*, ou du matériel pour ajuster verticalement le paragraphe).

Exemple 47

```
    local again
    for item in node.traverse_id(GLYF, line.head)
        do if not (item.font == firstlinefont) then
            again = true
            for glyph in node.traverse_id(GLYF, head)
                do if not (glyph.font == firstlinefont) then
                    glyph.font = firstlinefont
                    break
                end
            end
            break
        end
    end
```

Ensuite, pour la première ligne (`line.head`), on va vérifier que tous les glyphes ont la bonne fonte (notre `\firstlinefont`) ; dès que l'on rencontre un glyphe qui n'est pas en petites capitales, on inspecte alors tous les glyphes de la liste originale (`head`) jusqu'à ce qu'on

recontre le premier qui n'est pas en petites capitales, et on modifie la fonte avec celle désirée. Le ou la lectrice peut sans doute voir où on va : nous allons reconstruire le paragraphe autant de fois que nécessaire, à chaque fois en passant un glyphe supplémentaire de la première ligne en petites capitales, jusqu'à ce que tous les glyphes de la première ligne soient en petites capitales. Cela explique pourquoi nous devons à chaque itération réinsérer les points de césures, les ligatures et les crénages : à chaque fois qu'une partie du texte passe en petites capitales, la largeur des glyphes concernés peut changer, si bien que la composition du paragraphe doit être calculée à nouveau.

Le booléen `again` indique qu'on a trouvé un glyphe en bas de casse que l'on a passé en petites capitales, et donc qu'il faut relancer l'algorithme. Quand c'est le cas, on retire le paragraphe de la mémoire de TEX et on relance la fonction avec la liste `head` modifiée ; c'est ce que fait le code suivant :

Exemple 48

```
    if again then
        node.flush_list(par)
        return check_par(head)
```

La suite du code traite le cas où il n'y a pas besoin de relancer la procédure de changement de fonte.

Exemple 49

```
    else
        local secondline = line.next
        while secondline
            and not (secondline.id == HLIST) do
            secondline = secondline.next
        end
        if secondline then
            local list = secondline.head
            for item in node.traverse_id(GLYF,list)
                do if item.font == firstlinefont then
                    item.font = font.current()
                else
                    break
                end
            end
```

Dans ce cas, tous les glyphes de la première ligne sont en petites capitales, mais il y a encore une chose à vérifier. Supposons que le dernier caractère ayant été mis en petites capitales est x. Par définition, x est à la fin de la première ligne avant que sa fonte ait été changée, mais est-ce le cas après le changement ? Pas nécessairement, il peut tout à fait être possible qu'avec les nouvelles dimensions de x, il faille[54] couper la ligne *avant*, et peut-être même pas juste avant, mais quelques glyphes avant. Ainsi, il est possible d'obtenir quelques petites capitales en deuxième ligne de paragraphe. Ces petites capitales doivent alors être changées de nouveau. Oui, mais comment ? On change leur fonte, et on construit le paragraphe de nouveau ? On ne peut pas faire cela au risque de se trouver bloqué dans une boucle inifine (des bas de casses dans la première ligne, des petites capitales dans le secondes, et encore et encore…). La solution que nous avons adoptée ici est de réaffecter la fonte originelle (avec

54. C'est l'optimisation de TEX qui détermine cela.

`font.current()`) à ces glyphes et de les laisser là.

La fin du code de la fonction permet de gérer encore quelques problèmes.

Exemple 50

```
            list = node.ligaturing(list)
            for kern in node.traverse_id(KERN, list)
                do if kern.subtype == 0 then
                    node.remove(list, kern)
                end
            end
            list = node.kerning(list)
            secondline.head = node.hpack(
                list, secondline.width, "exactly")
        end
        node.flush_list(head)
        return par, prevdepth, prevgraf
    end
end
```

En effet, si les premiers glyphes de la seconde ligne sont *f* et *i*, en petites capitales, ils ne forment sans doute pas de ligature, mais une fois la fonte courante rétablie ? Nous devons donc rétablir les ligatures. Et en ce qui concerne les crénages ? On doit supprimer tous les crénages de fonte (c'est-à-dire de sous-type 0) et reconstruire les crénages. On doit enfin ajuster la largeur de la deuxième ligne (qui a été modifié par le changement de fonte) pour qu'elle reprenne la largeur d'avant le changement de fonte, les *glues* vont alors être étirées ou compressées. Ce procédé n'est pas optimal, mais les cas où des petites capitales sont présentes en deuxième ligne sont très rares. Les dernières lignes de la fonction `check_par` suppriment la liste `head` originale et retournent le paragraphe et les deux paramètre `prevdepth` et `predgraf`.

Exemple 51

code

```
\begin{firstparagraph}
\ipsum<Lang=FR>[1]
\end{firstparagraph}

\ipsum<Lang=FR>[3]
```

résultat

SANS QU'ELLE DAIGNÂT LE DIRE À PERSONNE, UN ACCÈS DE FIÈVRE D'UN DE SES FILS LA mettait presque dans le même état que si l'enfant eût été mort. Un éclat de rire grossier, un haussement d'épaules, accompagné de quelque maxime triviale sur la folie des femmes, avaient constamment accueilli les confidences de ce genre de chagrins, que le besoin d'épanchement l'avait portée à faire à son mari, dans les premières années de leur mariage. Ces sortes de plaisanteries, quand surtout elles portaient sur les maladies de ses enfants, retournaient le poignard dans le cœur de Mme de Rênal. Voilà ce qu'elle trouva au lieu des flatteries empressées et mielleuses du couvent jésuitique où elle avait passé sa jeunesse. Son éducation fut faite par la douleur. Trop fière pour parler de ce genre de chagrins, même à son amie Mme Derville, elle se figura que tous les hommes étaient comme son mari, M. Valenod et le

résultat (suite)

sous-préfet Charcot de Maugiron. La grossièreté, et la plus brutale insensibilité à tout ce qui n'était pas intérêt d'argent, de préséance ou de croix ; la haine aveugle pour tout raisonnement qui les contrariait, lui parurent des choses naturelles à ce sexe, comme porter des bottes et un chapeau de feutre.
Mais la demoiselle du comptoir avait remarqué la charmante figure de ce jeune bourgeois de campagne, qui, arrêté à trois pas du poêle, et son petit paquet sous le bras, considérait le buste du roi, en beau plâtre blanc. Cette demoiselle, grande Franc-Comtoise, fort bien faite, et mise comme il le faut pour faire valoir un café, avait déjà dit deux fois, d'une petite voix qui cherchait à n'être entendue que de Julien : Monsieur! Monsieur! Julien rencontra de grands yeux bleus fort tendres, et vit que c'était à lui qu'on parlait.

Le lecteur ou la lectrice aura peut-être repéré quelques failles dans ce code. Une solution complète aurait largement dépassé les limites de cet article déjà assez long. Pour l'améliorer (ce que nous laissons en exercice), on peut essayer de proposer une solution qui ne repose pas sur l'hypothèse qu'aucune fonction n'est enregistrée dans les autres *callbacks*. On peut aussi mieux gérer l'introduction possible de petites capitales au début de la deuxième ligne (peut-être en reconstruisant le paragraphe à partir de cette ligne sans toucher à la première?). On peut aussi imaginer une solution qui change le caractère et non la fonte: il faut pour cela aller chercher le caractère en petite capitale dans la table Unicode (mais le test pour savoir si cela a déjà été fait est alors différent).

Traitement du paragraphe une fois construit

Le *callback* `post_linebreak_filter` est très calme après tout ce que nous venons de voir: par défaut, rien ne s'y passe. Il lui est donné comme premier argument ce que le *callback* `linebreak_filter` retourne, c'est-à-dire une liste de listes horizontales (les lignes du paragraphe), des pénalités, des *glues* et peut-être du matériel interligne. En second argument, il est possible de lui passer une chaîne de caractère à la manière du *callback* `pre_linebreak_filter`. Paul Isambert, dans son article [5], montre plusieurs utilisations de ce *callback*, en particulier pour souligner du matériel. Paul Isambert conseille aussi pour les lecteurs qui cherchent des défis, d'essayer d'adapter à LuaTEX le code de la section 5.9.6 de *TEX by Topic: A TEXnician's Reference*.

Le *callback* retourne un paragraphe, potentiellement le même que celui reçu. Le paragraphe est alors ajouté à la liste verticale englobante, et ce qui est suit est le travail du constructeur de page. Notre s'exploration s'arrête ici.

Conclusion

La plupart des opérations que nous avons vues ici ne sont pas nouvelle en TEX : LuaTEX donne simplement accès à ces opérations. Depuis le tout début, TEX lit des lignes et des *tokens* (lexèmes) et construit des listes de nœuds ; c'est son principal travail. Le contrôle du processus de composition est ce qui fait de TEX un si bon outil, peut-être meilleur que d'autres outils de composition ; LuaTEX amène le contrôle encore plus loin et permet la manipulation des plus petits atomes qui constituent la typographie numérique: les caractères et les glyphes, ainsi que quelques autres détails techniques. Paul Isambert conclut son article, source principale de cette adaptation, en disant que du point de vue de la liberté, de manière rétrospective, il peut trouver TEX82 et ses successeurs un peu dictatoriaux par comparaison avec LuaTEX.

Maxime Chupin

Références

[1] THE LUATEX TEAM. *The* luatex *package. The LuaTeX engine*. 9 déc. 2021. URL : https://ctan.org/pkg/luatex.

[2] *Cahiers GUTenberg : Introduction à LuaTEX* 2010.54-55 (2010). URL : http://www.numdam.org/issues/CG_2010___54-55/.

[3] Paul ISAMBERT. « LuaTEX: What it takes to make a paragraph ». In : *TUGboat* 32.1 (2011).

[4] Manuel PÉGOURIÉ-GONNARD. *The* luacode *package. Helper for executing lua code from within TEX*. Version 1.2a. 24 juin 2016. URL : https://ctan.org/pkg/luacode.

[5] Paul ISAMBERT. « Three things you can do with LuaTEX that would be extremely painful otherwise ». In : *TUGboat* 31.3 (2010).

[6] Victor EIJKHOUT. *TEX by Topic: A TEXnician's Reference*. Dante, 2014.

LA FONTE DE CE NUMÉRO : ALEGREYA

La super-famille[55] de fontes Alegreya est l'œuvre de Juan Pablo del Peral[56], pour la fonderie argentine Huerta Tipografica[57]. Cette fonderie distribue ces fontes sous la version 1.1 de la licence SIL Open Font[58] et en propose une version commerciale, dite *pro*, comprenant caractères grecs et cyrilliques ainsi que des caractères latins additionnels, permettant différents raffinements typographiques dont nous ne traiterons pas ici[59]. En l'état, Alegreya offre déjà de nombreuses fonctionnalités, qui permettent de composer un document satisfaisant ; cette *Lettre* en témoigne (nous espérons que vous partagez notre avis).

Grâce au très prolifique Bob Tennent[60, 61], Alegreya bénéficie d'un package support pour LaTeX, pdfLaTeX, XeLaTeX et LuaLaTeX, qui est très utile. Pour utiliser les fontes, il suffit d'ajouter au préambule de votre document l'appel suivant :

Exemple 52 : avec empattements

```
1 \usepackage{Alegreya}
```

Une vaste palette

Alegreya propose un grand choix de fontes que nous détaillons ici. Il est facile de les obtenir grâce au package dédié.

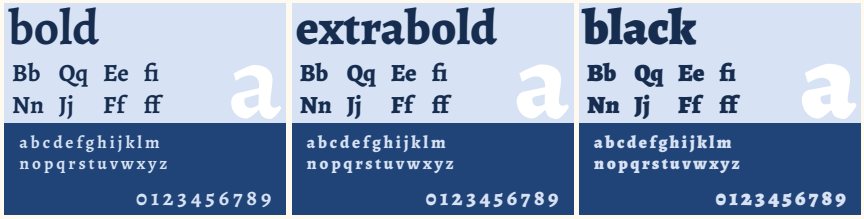
Les italiques sont intéressants — on appréciera notamment le magnifique *g* italique, que nous n'avons pas laissé passer (il figure en filigrane de la couverture de ce numéro) :

55. Voir, en anglais, https://en.wikipedia.org/wiki/Font_superfamily.
56. juan chez huertatipografica point com point ar
57. http://www.huertatipografica.com.ar
58. Le trésorier de l'association, François Druel, connaît bien cette licence et a prévu d'en parler dans un prochain article consacré à une autre fonte en bénéficiant.
59. Le logiciel fontforge nous permet de voir que de nombreux caractères grecs et cyrilliques sont présents dans la fonte à laquelle nos logiciels favoris donnent accès. C'est tant par manque de compétences que de temps que nous n'abordons pas dans cet article le cas des caractères grecs et, surtout, cyrilliques.
60. Bob Tennent a *packagé* de très nombreuses fontes pour (all)TEX. Voir https://ctan.org/author/tennent
61. Cet article est largement inspiré du README de la documentation du package Alegreya, dû à Bob Tennent. Merci à lui !

FIGURE 2 – Alegreya : quelques glyphes romains et italiques, avec de beaux empattements

Alegreya offre une large palette de graisses (nous détaillons plus loin la manière de les sélectionner) :

FIGURE 3 – Alegreya : différentes graisses… grasses



La fonte Alegreya (avec empattements, donc) est utilisée par défaut pour le texte courant. Mais sa version sans empattements est également disponible via la commande :

Exemple 53 : mise à disposition des caractères sans empattements

```
1 \usepackage{AlegreyaSans}
```

Les glyphes d'Alegreya Sans seront ensuite activés par la commande `\textsf` placée dans le corps du texte :

Exemple 54 : du texte avec et sans empattements

code

```
1 Glyphes avec et \textsf{sans empattements !}
```

résultat

Glyphes avec et sans empattements !

Utiliser Alegreya Sans par défaut pour le texte courant s'obtient par l'option `sfdefault`, comme suit :

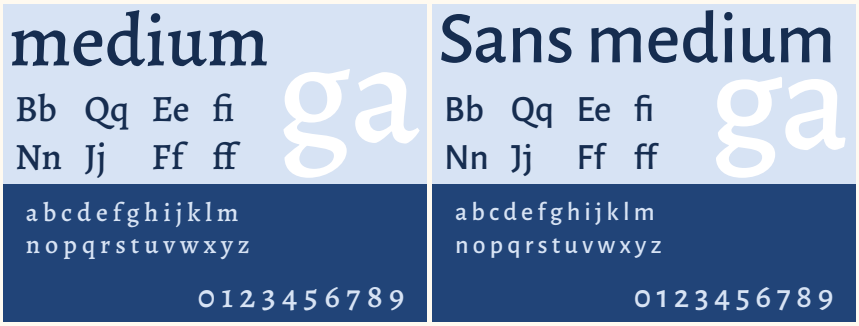
Exemple 55 : tout le document sans empattements

```
1 \usepackage[sfdefault]{AlegreyaSans}
```

FIGURE 4 – Alegreya : quelques glyphes sans empattements

Le package permet une sélection aisée des différentes graisses disponibles : l'option `black` passée à l'appel de chacun de ces packages choisit cette série de glyphes comme fonte grasse par défaut. De la même manière, utiliser l'option `medium` revient à choisir cette série pour le texte courant.

FIGURE 5 – Alegreya : les graisses medium



Les options `thin`, `light` et `extrabold` sont disponibles, mais seulement pour la famille Alegreya-Sans (donc sans empattements).

FIGURE 6 – Alegreya Sans : versions light et thin

En toute logique, les commandes `\Alegreya`, `\AlegreyaMedium`, `\AlegreyaExtraBold`, `\AlegreyaBlack` et leurs pendants `\AlegreyaSansThin`, `\AlegreyaSansLight`, `\AlegreyaSans`, `\AlegreyaSansMedium`, `\AlegreyaSansExtraBold` et `\AlegreyaSansBlack` sélectionnent les familles de caractères en question[62].

62. NDLR : que l'on veuille bien nous pardonner cette curieuse composition. Elle est due au fait que les noms

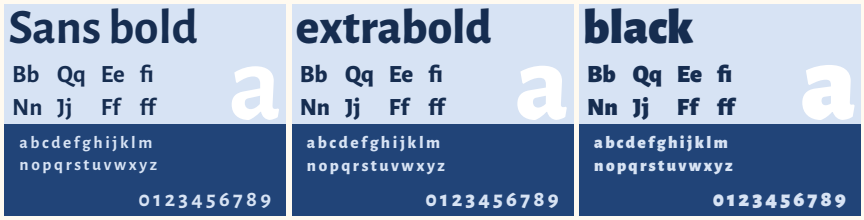


FIGURE 7 – Alegreya Sans: différentes graisses… grasses, mais sans empattements

Les utilisateurs de LuaLaTeX et XeLaTeX qui préféreraient les fontes de type 1 (T1) ou ne pas utiliser le package fontspec choisiront l'option `type1` :

Exemple 56 : sélection des fontes type 1

```
\usepackage[type1]{AlegreyaSans}
```

Enfin, les options `scaled=<nombre>` ou `scale=<nombre>`, passées à l'appel du package, sont utilisées pour jouer sur la hauteur d'x des caractères selon le facteur `<nombre>`. Voir le dernier exemple de cet article, en page suivante, qui montre l'utilisation de cette fonctionnalité… pour l'ensemble de ce numéro !

Chiffres et mathématiques

Par défault, les chiffres sont en capitales de chasse variable, mais on obtient les chiffres elzéviriens grâce à l'option `oldstyle` (ou `osf`) ; les chiffres tabulaires (à chasse fixe, ou monospace) sont obtenus par l'option `tabular` (ou `tf`).

Les commandes `\AlegreyaLF`, `\AlegreyaOsF`, `\AlegreyaTLF` et `\AlegreyaTOsF` sont des commutateurs : elles permettent d'utiliser d'autres chiffres que ceux sélectionnés dans le préambule par l'option passée au package.

Les exposants s'obtiennent par les commandes `\sufigures`, `\infigures`, et les indices par `\textsu{...}` ou `\textin{...}`.

La commande `\useosf` permute les chiffres par défaut en chiffres elzéviriens ; elle est prévue dans le cas de l'emploi d'un package de mathématiques qui utiliserait par défaut les chiffres en capitales.

Cette famille de fontes ne dispose malheureusement pas de fonte mathématique. On peut, comme nous l'avons fait pour la *Lettre* 52 et la fonte *Luciole*, utiliser une fonte mathématique (avec le package unicode-math) et le package mathastext pour remplacer les caractères alphabétiques de la fonte mathématique par la fonte du texte. Ci-dessous, nous avons choisi d'utiliser la fonte mathématique Garamond, mais il est sans doute possible de faire mieux :

Exemple 57 : préambule pour une fonte maths avec Alegreya

```
\usepackage{unicode-math}
\setmathfont{Garamond-Math.otf}[StylisticSet={7,9}]
\usepackage{Alegreya}
\usepackage[italic,LGRgreek]{mathastext}
```

Nous avons aussi utilisé le package metalogox pour ajuster les différents logos (La)TeX.

La figure 8 illustre ces choix.

de commande sont insécables.

FIGURE 8 – Illustration de la composition des mathématiques avec la fonte mathématique *Garamond* et le package mathastext pour remplacer les caractères courants.

Theorème 1 (des résidus). Soit f une fonction analytique dans une région G à l'exception des points isolés $a_1, a_2, \ldots, a_m$. Si γ est une courbe dans G fermée simple, rectifiable, positivement orientée qui ne passe par aucun des points a_k, alors

$$\sum_k \operatorname{Res}(f, a_k) = \frac{1}{2\pi \mathrm{i}} \int_\gamma f(z)\mathrm{d}z.$$

Absence de fonte monospace

Si les petites capitales sont bien offertes par les fontes Alegreya, ce n'est pas le cas des caractères à chasse fixe. Nous comprenons tout à fait que l'auteur ait jugé pouvoir s'en passer, vu que cette fonte nous semble plutôt destinée à l'édition de texte courant.

Mais pour la *Lettre*, qui fait abondamment appel aux fontes non proportionnelles pour ses exemples de code, nous avons été obligés d'adjoindre à Alegreya des caractères monospace (appelés par la commande `\texttt{}`) provenant d'une autre fonte. Nous avons choisi Noto Sans mono, dont nous avons parlé dans la *Lettre* 50, parue en juin 2023.

Nous avons appelé cette fonte ainsi :

Exemple 58 : préambule pour une fonte monospace avec Alegreya

```
\usepackage{noto-mono}
```

Nous espérons que notre choix de fonte monospace vous conviendra.

Si l'on reprend les fragments de code cités plus haut, on constate que le choix des fontes utilisées pour ce numéro de la *Lettre* est donc effectué via les appels suivants :

Exemple 59 : préambule pour les fontes de ce numéro

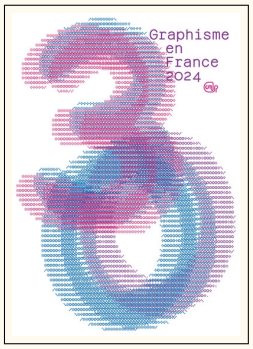
```
\usepackage{unicode-math}
\setmathfont{Garamond-Math.otf}[StylisticSet={7,9}]
\usepackage[scale=0.85]{noto-mono}
\usepackage{Alegreya}
\usepackage[italic,LGRgreek]{mathastext}
% ajustement des logos TeX
\usepackage{metalogox}
```

Malgré l'absence de fontes mathématique et monospace, nous espérons que vous serez sensibles à l'élégance d'Alegreya. Cette famille de fontes nous semble tout à fait adaptée à des documents littéraires.

Patrick Bideault & Maxime Chupin

COMPTES RENDUS DE LECTURE

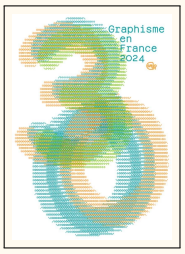
Graphisme en France a 30 ans !



Graphisme en France 2024. Paris, France : Centre national des Arts plastiques, juin 2024

Le numéro de *Graphisme en France* de cette année est paru en juin. Cette revue annuelle, gratuite, publiée par le Centre national des arts plastiques, fête son trentième anniversaire. Elle est disponible dans les fonds régionaux d'art contemporain, écoles et centres d'art, en ligne et sur demande à `graphisme@cnap.fr`.

Le nouveau numéro est un assez bel objet, de 25 x 18 cm. Il est très bien imprimé sur un beau papier non couché de Fedrigoni. Il est mis en page par Louise Garric, qui a choisi le caractère Pachinko d'Émilie Rigaud, de la fonderie `A is for fonts`[63].



Il contient trois longs articles :

— *Graphisme d'utilité publique — L'apparence et le sens*, d'Olivier Huz
— *Design graphique — Trente mille ans de culture visuelle, trente ans de recherche*, de Vivien Philizot
— *Design graphique et intelligence artificielle*, d'Étienne Mineur

Je ne partage pas forcément le choix d'en imprimer le texte en couleurs, mais je me réjouis que, depuis l'an dernier, cette excellente revue propose une version anglaise (voir sa couverture ci-contre, en vert). Je suis certain qu'elle contribuera au rayonnement de cette vibrante scène graphique française dont *Graphisme en France* se fait l'écho.

`https://www.cnap.fr/actualites/graphisme-en-france`

Et je rappelle combien le numéro de l'an dernier était passionnant. Il s'agissait d'une conversation entre Sara De Bondt, Catherine Guiral et Alice Twemlow.

Patrick Bideault

Enseigner la physique avec la *TEXnische Komödie*

Die TEXnische Komödie, 2023 & 2024. Heidelberg, Allemagne : Dante e.V.

Le groupe d'utilisateurs germanophones, dante, publie une excellente revue, en toute logique appelée *La TEXnique Comédie* (en allemand *Die TEXnische Komödie* ou DTK).

En 2023, l'auteur du package schulmathematik, Keno Wehr, y a publié une série d'articles consacrés à l'enseignement de la physique. Il consacrait chaque livraison à un sujet particulier, commençant par les grandeurs et unités, puis les schémas de connexions électriques, les diagrammes permettant d'analyser des valeurs mesurées… à l'aide des instruments de mesure que son package permet de représenter ! Je recommande ses verniers et dynamomètres : même un non-germanophone admirera la qualité de ces figures[64].

Je pensais que la série d'articles, constituant une sorte d'extension de la documentation du package en question, s'arrêterait là. J'ai eu l'heureuse surprise de découvrir, en début d'année, que l'auteur continuait en traitant de mécanique et d'astronomie, avec les packages fiziko, pst-pulley et pst-solarsystem. Et dans le numéro suivant, il abordait les champs électriques et magnétiques, avec les packages pst-electricfield et pst-magneticfield !

63. France Culture a consacré à cette créatrice de caractères un bref article orné d'une vidéo : `https://www.radiofrance.fr/franceculture/emilie-rigaud-typographe-du-futur-3022988`
64. C'est l'un des avantages de la TEX Live : on peut y admirer des documentations rédigées dans des langues inconnues de nous !

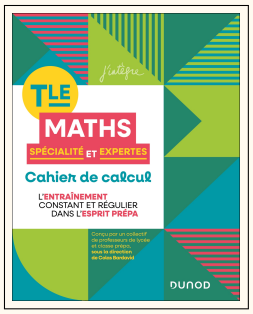
La dernière livraison le voit traiter de l'optique, tout d'abord avec le package pst-optic. Puis il montre qu'il est possible d'obtenir des résultats similaires avec MetaPost : il fournit les commandes pour dessiner les lentilles requises, etc.

Puis il en vient aux réfractions et réflexions, toujours avec pst-optic, et aux interférences, avec pgf-interference, avant les diffractions, magnifiquement illustrées à l'aide de pst-diffraction. Et il conclut par la présentation de tikz-optics et de pst-optexp : autant dire que cet auteur est agnostique en matière de langages de dessin !

Je serais très heureux de mettre cette excellente série d'articles à la disposition du lectorat francophone, mais je n'en ai malheureusement pas la possibilité de me lancer dans une pareille entreprise : accaparé comme je le suis par ma vie professionnelle, le temps me manque. Mais qui sait, peut-être quelque lecteur de la *Lettre* GUTenberg aurait-il envie de traduire ces articles ?

Patrick Bideault

Les cahiers de calcul pour le lycée



Colas BARDAVID, éd. *Cahiers de calcul en maths*. Malakoff : Dunod. ISBN : 9782100864195

Membre de l'association GUTenberg, investi dans sa vie, rien ne me plaît plus que les productions collectives ! Travaillant dans la recherche en mathématiques à l'université et enseignant de temps en temps, les *cahiers de calcul pour le lycée* m'ont particulièrement plu. Ces cahiers, destinés aux lycéens de classes de première et terminale, sont à mon avis très utiles pour la première année universitaire. Une trentaine de professeurs en classe préparatoire et en lycée a produit trois cahiers de calculs :

— le cahier de calcul - Première spécialité ;
— le cahier de calcul - Terminale spécialité ;
— le cahier de calcul - mathématiques expertes.

Les documents sont élégants et permettent encore une fois de constater à quel point LaTeX peut être utile.

Voici les conditions d'utilisation et de partage de ces documents :

— Les versions PDF des cahiers de calcul sont librement diffusables par les professeurs auprès de leurs élèves, sous format papier ou PDF ;
— Les noms des coordinateurs et participants au projet doivent être indiqués ;
— Les participants au projet peuvent les mettre à disposition sur leur site web ;
— Les sources LaTeX sont mises à disposition des membres de l'UPS, via la BEDOC et la forge `Gitea` de l'UPS ;
— Les sources LaTeX ne sont pas diffusables en dehors de la BEDOC et du dépôt sur `Gitea` ;
— Le contenu des cahiers de calcul est modifiable et réutilisable pour de spécifiques usages (à l'exclusion de toute utilisation commerciale).

Le tout peut se consulter ici :

`https://colasbd.github.io/cdc-lycee/`

… et ces cahiers ont même été édités par la maison Dunod[65] et sont vendus dans le commerce !

Maxime Chupin

65. On les trouve ici : `https://www.dunod.com/livres-colas-bardavid`

EN BREF

Forum du design graphique et de la typographie le 10 octobre

À l'occasion des trente ans de la revue *Graphisme en France* (voir page 63), le Centre national des arts plastiques et la bibliothèque Kandinsky du Centre Pompidou, à Paris, s'associent pour organiser une journée de rencontre et de discussion sur le design graphique et la typographie. Quatre tables rondes thématiques y sont organisées :

— Festivals, biennales, rencontres : quels enjeux pour les publics, la transmission, la diffusion ?

— Archives : les constituer, les organiser, les valoriser (expositions, professionnels, recherche etc.)

— Commandes exemplaires, processus de travail, comment réussir son projet de design graphique ?

— Les formidables développements de la typographie française : outil de recherche, médiation, diffusion

Des temps d'échanges avec le public auront lieu lors de celles-ci.

Plus d'informations ici :

`https://www.cnap.fr/forum-du-design-graphique-et-de-la-typographie`

Fonds d'écran GUTenberg, encore

Comme je l'avais écrit dans la *Lettre* 51, j'ai créé avec (LA)TEX des fonds d'écran pour ordinateur que j'ai déposés dans un dépôt `Gitlab` pour rendre le projet collaboratif.

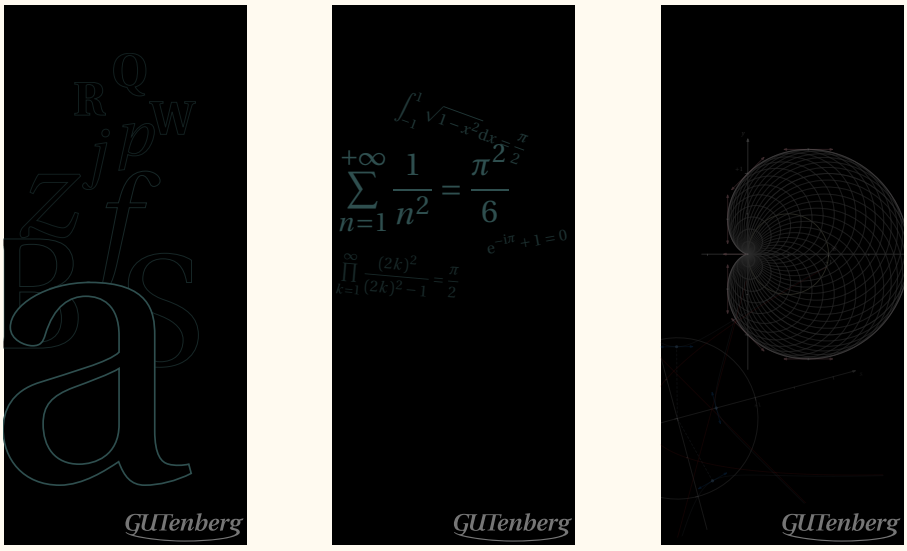
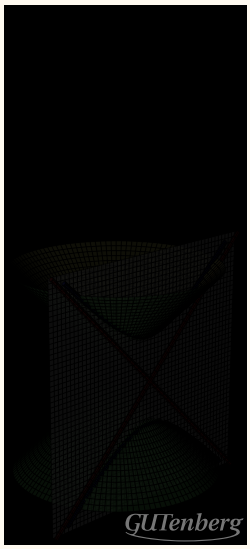
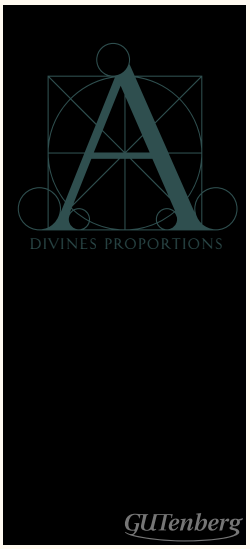


Figure 9 – Des fonds d'écran au format *smartphone*.

Malheureusement, cela n'a pas déclenché de contributions autres que les miennes… mais j'ai tout de même produit les versions pour *smartphone* ! Peut-être seront-elles utiles. Ces fonds d'écran, volontairement sombres, ont une résolution adaptée aux écrans en question. Ils sont reproduits en ci-dessus et page suivante.

FIGURE 10 – D'autres fonds d'écran au format *smartphone*.





Tant pour ordinateurs que pour téléphones, ces fonds sont publiés sur notre instance GitLab :

https://gitlab.gutenberg-asso.fr/gutenberg/fond-d-ecran

N'hésitez pas à rejoindre le projet et à proposer, vous aussi, des fonds d'écran réalisés avec (LA)TEX !

Par ailleurs, je rappelle que les dépôts de l'association ont vocation à héberger tout projet francophone lié à (all)TEX : si vous avez besoin d'une instance GitLab, merci de vous manifester auprès du secrétariat.

Maxime Chupin

Les vidéo des exposés du TUG 2024 sont en ligne

Les vidéos des exposés du TUG2024, qui s'est tenu du 19 au 21 juillet dans la ville de Prague en Tchéquie, sont désormais en ligne sur la chaîne YouTube du TUG :

https://www.youtube.com/@TeXUsersGroup

Vous pourrez notamment y visionner les exposés de Jean-Michel Hufflen et de Didier Verna, deux francophones présents à la conférence.

Bon visionnage.

Un service à tester

Connaissez-vous detexify ? Ce service existe depuis plusieurs années. Il permet de retrouver un symbole présent dans un package quelconque en le dessinant sur le web. C'est assez efficace.

http://detexify.kirelabs.org/

Un service analogue est désormais disponible pour les dessins : deTikZify. Il suffit d'y télécharger un dessin et le service vous renvoie le code TikZ correspondant.

https://huggingface.co/spaces/nllg/DeTikZify

Le service n'est pas parfait et propose plusieurs solutions pour chaque image que vous lui proposerez. Le tester en tant que générateur de code TikZ est amusant et j'ai découvert plusieurs fonctionnalités de TikZ en parcourant les solutions proposées.

Pour rappel, il est facile de tester du code en le compilant en ligne sur TeXLive.net :

https://texlive.net/run

Ce service est offert par David Carlisle, sur un serveur mis à disposition par Stefan Kottwitz (qui fait de même pour https://texnique.fr/) et financé par Dante, l'association germanophone.

Les derniers articles des *Cahiers* sont en ligne

Chacun des articles du numéro 58 (le dernier à être paru) est désormais accessible en ligne, sur le site de l'association :

https://publications.gutenberg-asso.fr/cahiers/article/view/35/172

Nous prévoyons de rendre les autres *Cahiers* accessibles sur ce même site, et que leurs adresses originelles pointent vers celui-ci.

Une fonte maths pour Luciole

Luciole

Avez-vous apprécié Luciole, la fonte utilisée dans la *Lettre* 52[66, 67] ?

Grâce à Daniel Flipo, elle va connaître un développement mathématique. Elle devrait donc devenir la première fonte pour les personnes malvoyantes à proposer un support mathématique !

L'association vient de signer une convention qui l'associe à ce beau projet auquel participent le Centre technique régional pour la déficience visuelle de Vaulx-en-Velin, dans le Rhône, et le studio typographies.fr[68]. Nous vous tiendrons informés de son développement et, surtout, de sa publication.

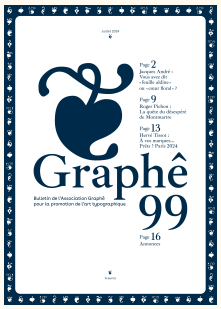
Par ailleurs, M. Fabreguettes, qui coordonne ce beau projet avec M^me^ Malet, répond à notre interrogation formulée dans une note de bas de page (numérotée 111 !), au sujet de la personne chargée du remplissage des tables Unicode (la « distribution ») de Luciole. Il s'agit de Laurent Bourcellier, qui est le dessinateur des caractères. Comme quoi, ce métier a de multiples facettes !

Des grappes de Ratdolt, suite

Paru dans le numéro précédent de la *Lettre*, l'article de Jacques André intitulé *Feuille aldine ou… grappe ?*[69] a eu l'heur d'intéresser *Graphê*[70], la prestigieuse revue de l'association de promotion de l'art typographique :

https://typo-graphe.com/

66. La *Lettre* 52 est accessible ici : https://doi.org/10.60028/lettre.vi52
67. Nous aurions tant aimé que la note précédente soit la cinquante-deuxième de la revue, mais nous avons lamentablement échoué ; d'où la présente note.
68. Studio dont le nom de domaine est facile à deviner !
69. Ce article est disponible ici : https://doi.org/10.60028/lettre.vi52.155.
70. Je précise être membre du comité de rédaction de *Graphê* ; j'y ai fait circuler la *Lettre*, et l'excellent article de Jacques André a éveillé l'intérêt des autres membres dudit comité. [PB]



Une version remaniée de cet article est parue cet été, dans le numéro 99 de la revue. Elle est intitulée *Vous avez dit « feuille aldine » ou « cœur floral » ?* Appréciez ci-contre la couverture de la revue, ornée de grappes de Ratdolt et mise en page par Laura Léotard-Chausson.

Par ailleurs, on nous a rapporté que le rouge manquait parfois dans l'article de Jacques André, notamment pour la figure 25, page 91. Nous avons mis longtemps à comprendre ce phénomène : il n'est pas dû aux fichiers PDF que nous produisons, mais bien aux visualiseurs utilisés ! Nous vous déconseillons donc de consulter la *Lettre* via un navigateur web. Le mieux est de la télécharger pour la lire grâce à un logiciel dont la seule fonction est la consultation de fichiers PDF.

Patrick Bideault

Références

[1] Jacques ANDRÉ. *Vous avez dit « feuille aldine » ou « cœur floral » ?* Paris, France : Graphê, numéro 99, Association Graphê pour la promotion de l'art typographique, juill. 2024.

PROCHAINES RENCONTRES

Voici les prochaines rencontres concernant nos logiciels préférés :

- les grapholinguistes du monde entier se retrouveront à Venise du 23 au 25 octobre, et leur conférence est par l'ACL[71] et l'Association typographique internationale[72]. Donald Knuth y prendra la parole. https://grafematik2024.sciencesconf.org ;
- et la prochaine journée GUTenberg aura lieu le samedi 16 novembre 2024 (voir notre annonce page 3).
- le TUG 2025 aura lieu dans le Kerala, en Inde, mi-juillet 2025. Il sera hébergé par la société TEXFolio[73]. La page https://tug.org/tug2025/ sera mise à jour petit à petit et reprendra toutes les informations utiles ;

Et tous les mois, en ligne

Chaque mois, retrouvons-nous en ligne pour nos exposés !

https://www.gutenberg-asso.fr/-Exposes-mensuels-

Mais aussi tous les mois, en présentiel, à Paris

Le premier samedi de chaque mois, en présentiel, à Paris, lors des samedis du libre !

https://samedis-du-libre.org/Premier-Samedi-du-Libre

71. ACL pour *Association for Computational Linguistics*, en français « Association pour la linguistique informatique » – https://fr.wikipedia.org/wiki/Association_for_Computational_Linguistics.
72. https://atypi.org/
73. https://texfolio.org

ACRONYMES

AG Assemblée Générale

ASCII *American Standard Code for Information Interchange* (code américain normalisé pour l'échange d'information)

BBB `BigBlueButton`

BBI *Best Bowling in Innings* (meilleur lanceur de la manche — cette notion n'entretient que des rapports éloignés avec nos logiciels préférés ; en revanche, elle est utile quand on s'intéresse au cricket)

BEDOC Base d'échange de documents scientifiques

CA Conseil d'Administration

CTAN *Comprehensive TEX Archive Network* (réseau complet d'archives TEX)

DIN Deutsches Institut für Normung (Institut allemand de normalisation)

DOI *Digital Object Identifier* (identifiant numérique d'objet)

DUCET Default Unicode Collation Element Table (Tableau Unicode d'agencement par défaut)

ÉNS École Normale Supérieure

EPC *European Payments Council* (Conseil européen des paiements)

FAQ *Frequently Asked Questions* (questions fréquemment posées, souvent librement traduit en « foire aux questions »)

IBAN *International Bank Account Number* (Numéro international de compte bancaire)

IDE *Integrated Development Environment* (environnement de développement)

JSON `JavaScript` *Object Notation* (Notation des objets `JavaScript`)

NDLR Note de la Rédaction

OTF *Open Type Format* (format Open Type)

PDF *Portable Document Format* (format de document portable)

SEPA *Single Euro Payments Area* (Espace unique de paiement en euros)

TikZ *TikZ ist* kein *Zeichenprogramm* (TikZ *n'*est *pas* un programme de dessin)

TUG *TEX User Group* (groupe international d'utilisateurs de TEX)

UCA *Unicode Collation Algorithm* (algorithme d'agencement Unicode)

UPS Union des Professeurs de classes préparatoires Scientifiques

UTF *Universal (Character Set) Transformation Format* (format de transformation universel, ou plus exactement *format de transformation du jeu universel de caractères codés*)

La rédaction tient à remercier Anne Bideault, Antoine Desrosiers et MFerrer pour les erreurs qu'ils ont remarquées dans la *Lettre* 52 ; ces erreurs ont été corrigées. Si vous en remarquez d'autres, merci de nous les signaler par courriel à `secretariat@gutenberg-asso.fr`.

La rédaction rappelle au lecteur que le code source de la présente revue est inclus *dans* le présent PDF. On y accède en fin d'article en cliquant sur les trombones.

Enfin, nous vous conseillons de télécharger la présente *Lettre* pour la lire grâce à un logiciel dont la seule fonction est la consultation de fichiers PDF : cela évitera d'éventuelles erreurs d'affichage.

Par leur motivation et leur engagement sans faille, ont contribué à cette *Lettre* : Yvon Henel, Yann Denichou, Daniel Flipo, Philippe Nadeau, Jasper Habicht, MFerrer, Céline Chevalier, Maxime Chupin, Patrick Bideault, Cédric Pierquet, Denis Bitouzé, François Druel et Bastien Dumont.

Association GUTenberg
15 rue des Halles – BP 74
75001 Paris
France
secretariat[at]gutenberg-asso[dot]fr

Site Internet : https://www.gutenberg-asso.fr/
***Cahiers* :** http://www.numdam.org/journals/CG/
Problèmes techniques :
la liste gut : https://www.gutenberg-asso.fr/-Listes-de-diffusion-
le site TEXnique de questions et réponses : https://texnique.fr/
la foire aux questions : https://faq.gutenberg-asso.fr/

Cette association est la vôtre : faites-nous part de vos idées, de vos envies, de vos préoccupations à l'adresse secretariat[at]gutenberg-asso[dot]fr.

ADHÉSION À L'ASSOCIATION

— Les adhésions sont à renouveler en début d'année pour l'année civile.
— Les administrations peuvent joindre un bon de commande revêtu de la signature de la personne responsable ; les étudiants doivent joindre un justificatif.

Tarifs 2024

Les membres de GUTenberg peuvent adhérer à l'association internationale, le TUG, et recevoir son bulletin *TUGboat* à un tarif préférentiel :

tarif normal : 65 € (au lieu de 85 $)
tarif étudiant : 40 € (au lieu de 55 $)

Type d'adhésion	Prix
Membre individuel	30 €
Membre individuel + adhésion TUG	95 €
Membre individuel étudiant/demandeur d'emploi	15 €
Membre individuel étudiant + adhésion TUG	55 €
Association d'étudiants	65 €
Organisme à but non lucratif	130 €
Organisme à but lucratif	229 €

Règlements

Les règlements peuvent s'effectuer par :

— **virement bancaire**[74] (IBAN : FR76 3000 3001 0900 0372 6086 280)

Veillez à bien *indiquer vos nom et prénom* dans les références du virement !

— Paypal[74] : https://www.gutenberg-asso.fr/?Adherer-en-ligne
— bulletin et chèque[74] : https://www.gutenberg-asso.fr/?Adherer-a-l-association

La* Lettre *GUTenberg
Bulletin irrégulomestriel de l'association GUTenberg
Directeur de la publication : Patrick Bideault
Comité de rédaction : Patrick Bideault, Denis Bitouzé, Céline Chevalier & Bastien Dumont
Adresse de la rédaction : Association GUTenberg, 15 rue des Halles – BP 74, 75001 Paris
ISSN : 2742-6149 (version numérique)

74. Nous vous remercions de **privilégier** le **virement bancaire**.