‹ JI3-2011 ›

3ème édition des Journées Informatique des CPGE

Agadir du 12 au 14 Mai 2011

Utilisation avancée de LaTeX

Dossier proposé par*
Sadik Boujaida

*Ce dossier sera disponible prochainement en une version corrigée et augmentée d'autres articles dans un hors série de la revue "Les cahiers de prépas"

Table des matières

Quoi de neuf dans le monde TEX/LATEX ?

Extensions personnelles

Gestion des polices

Conception d'un fichier style pour séries d'exercices

Quoi de neuf dans le monde TeX/LaTeX ?

Introduction

LaTeX3 *est une cellule de réflexion sur l'avenir de LaTeX, qui est au même temps responsable du maintien de la version courante*

L'écosystème du système de préparation de documents TeX et son rejeton LaTeX a connu ces derniers temps une dynamique intense et ce à plusieurs niveaux. Certes le projet LaTeX3 tarde à se concrétiser, mais on assiste de plus en plus à l'apparition de technologies souvent initiées par une seule personne, mais qui finissent par s'imposer en standard s'attachant les services de communautés solidaires.

§1. L'extension graphique PGF/Ti*k*Z

`pstricks` *est l'extension graphique de LaTeX qui est peut être la plus utilisée. Elle offre une syntaxe relativement simple pour la description d'éléments graphiques. à la compilation, elle se charge de traduire le code utilisateur en du code Postscript qu'elle insère dans le fichier DVI. Mais elle souffre d'une grande limitation dûe à sa concéption même : l'impossibilité de directement compiler le code avec* `pdflatex`

PGF est une extension, crée à l'origine par Till Tantau, dont le propos est de donner des capacités graphiques plus évoluées à l'environnement TeX/LaTeX. C'est devenu un système complexe avec des capacités très variées et peuvant être étendues par le biais de bibliothèques spéciales. On peut l'utiliser pour créer des figures géométriques des plus élémentaires au plus complexes, des graphes de fonctions (soit en se basant sur les aptitudes (limitées) du moteur TeX lui même soit en utilisant un programme externe pour la génération des données (`gnuplot`)), des diagrammes sophistiqués, des calendriers, des schémas en 3D . . . , ou encore par l'intermédiaire de l'option `overlay` profiter de son excellent rendu pour l'ajout d'élément décoratifs à ses documents. Le manuel de PGF/Ti*k*Z (malheureusement uniquement en anglais ou en allemand, langue natale de l'auteur), fait dans sa dernière version plus de 760 pages. Mais il est écrit de façon très pédagogiques avec notamment un premier chapitre fait de tutoriels qui permettent de cerner l'étendue des compétences du système.

En apparence comparable au système historique `pstricks`, PGF diffère de ce dernier en plusieurs point dont les plus notables sont :

1. PGF est lui même écrit en langage TeX, alors que `pstricks` est un système relativement indépendant qui se contente de traduire le code utilisateur (`pstricks`) en code Postscript et d'insérer le code résultant dans le fi-

chier DVI, le rendu étant assuré par l'afficheur DVI ou par le pilote `dvips` qui convertira ce dernier en un fichier .ps.

2. En conséquence de ce premier point PGF est plus indépendant du pilote de production et peut par exemple être compilé par le programme `pdftex` (ou `pdflatex`), chose qu'on ne peut faire avec du code `pstricks` (du moins sans l'intervention d'autres extensions qui marchent plus au moins bien).

3. `pstricks` reste plus puissant dans certain domaine du fait qu'il donne accès à Postscrip. L'une des fonctionnalités disponible pour `pstricks` et qui ne peut (peut être ne pourra jamais) être accessible à PGF est la capacité d'agir sur le rendu des polices elles même (détourage, ombrage ...) , mais aussi sa capacité de tracer de façon autonome des graphes de fonctions complexes (qui sont exigentes en calculs).

4. En contre partie PGF bénéficie de sa proximité du moteur TeX, et peut revendiquer d'être plus facilement portable. À moyen terme, sa parfaite intégration avec `pdftex` lui sera énormément favorable.

Postscript *est un véritable langage de programmation spécialisé dans la description de documents graphiques vectoriels avec des capacités de calcul très évoluées. Il est notamment géré en natif par les systèmes d'impression professionnels.*

PGF *est très verbeux, dans le sens où il faut plusieurs ligne de code pour décrire un élément graphique simple.* TikZ *grâce à sa syntaxe simplifiée permet d'y remédier.*

Ti*k*Z est une surcouche (frontend) de PGF qui permet à l'utilisateur d'accéder aux fonctionnalité de ce dernier de façon plus simple (à la manière du rapport qui existe entre TeX et LaTeX).

Il est à noter qu'une extension plus célèbre utilise aussi PGF : BEAMER, l'extension pour préparation de présentations. d'ailleurs ils ont aussi en commun le même créateur.

La communauté autour de PGF/Ti*k*Z va en s'agrandissant. Une mention spéciale doit être faite pour le site web www.texample.net qui offre une galerie de compositions graphiques utilisant PGF/Ti*k*Z de divers auteurs de par le monde. Un must.

§2. BEAMER l'extension qui effaça toutes les autres [1]

Développé initialement (en 2003) pour les besoins personnels de son auteur (TILL TANTAU), l'extension BEAMER à connu un succés fulgurant au point de rendre obsolète les travaux du même genre qui l'ont précédé (tel prosper, qui était pourtant très prometteur). Elle permet de créer des présentations de haute qualité tout en conservant une complète compatibilité avec la majorité des extensions LaTeX.

En outre elle gère plusieurs thèmes de présentation (vous changer complètement l'apparence de votre document, en changeant une ligne dans le code), et il est possible de se créer son propre thème moyennant quelques connaissances de base.

1. toutes les autres extensions de son genre

§3. XeLaTeX, ou comment accéder aux polices du système et aux caractères Unicode

Qu'est ce que l'Unicode ? *Classiquement les caractères sont encodés en 7 bits, ce qui permet de gérer au maximum 128 caractère et couvre les besoins de la langue anglaise. Cet encodage fut étendu à 8bit pour inclure les caractères latins et ensuite de créer des tables de code (les tables ASCII) pour chaque langue ou ensemble de langues similaires. Les chose se complexifient sachant que chaque plateforme à ses propres tables de code plus ou moins standards. Unicode à été introduit pour répondre à ce problème. Il est encodé en 16 bits (65536 caractères organisés en plages, soit tous les caractères pour toutes les langues existantes) TeX ne peut gérer un flux de caractères 16bits en natif (normal, il fût crée avant qu'on commence à parler de l'Unicode). d'où l'intérêt de XeTeX ou* LuaTeX

XeTeX est peut être la réponse la plus originale donnée à l'une des fonctionnalités les plus demandées et les plus attendues de TeX : le support en natif des caractères Unicode et la gestion des polices de caractères des autres langues que celle gérées par défaut. La gestion des caractères Unicode permettra d'écrire directement le source TeX dans sa langue natale, la gestion des polices Unicode permettra d'effectuer le rendu sans bidouille en utilisant le système sous-jascent TeX. Plusieurs projets initié dans ce sens sont devenus moribonds (`Omega`), XeTeX à eu le mérite d'être fonctionnel depuis ses premières versions.

XeTeX est un moteur (au même titre que TeX lui même) capable de réagir avec la plupart des extensions TeX classiques. Il produit uniquement des fichiers PDF. Outre le fait de gérer exclusivement l'Unicode, il permet (miraculeusement) d'utiliser toutes les polices du système. Au fils des versions une extension à été développée pour simplifier l'accès à ces polices en les appelant par leurs noms (comme on le ferait dans MS Word), il s'agit de `fontspec` et de son corollaire pour la gestion des polices mathématiques `mathspec`.

Pour compiler du code qui utilise ces fonctionnalités, il faut le faire avec la commande `xetex` pour du code PlainTeX et la commande `xelatex` pour du code LaTeX. Certains éditeurs ont des profils de compilation pour xeLaTeX.

§4. Le projet qui va réconcilier tout le monde : *Lua*TeX

`Lua` *est un langage de programmation complet et très léger. Il est de type interprété (par opposition à compilé) et peut être embarqué dans d'autres logiciels. Il est très utilisé dans certains domaines tel les jeux vidéos et l'informatique embarquée*

*Lua*TeX est LA réponse radicale au problème auquel propose une solution XeTeX : internationaliser TeX. La comparaison ne s'arrête pa là, puisque à travers la même extension (`fontspec`) *Lua*TeX est capable de faire appel à des polices du système de façon transparente. *Lua*TeX, par contre, embarque un langage de programmation au sens général du terme : le langage interprété `Lua`. Il est appelé à moyen terme à remplacer le moteur `pdftex`, d'ailleurs le développement de ce dernier s'est arrêté, pour reporter toutes les nouvelles fonctionnalités au bénéfice de son remplaçant. En d'autres termes c'est l'avenir du monde TeX/LaTeX, puisque les compilateurs classiques qui produisent du DVI ont déjà montré leurs limites.

Lua apporte donc des fonctionnalités supplémentaires au moteur TeX classique. Par exemple, du code Lua peut être directement exécuté à partir d'un document LaTeX à l'aide de la commande `\directlua`.

Par exemple : le code `$\pi = \directlua{tex.sprint(math.pi)}$` va produire sur votre document de sortie π = 3.1415926535898. On peut alors rêver d'un moteur de calcul formel (ou scientifique) totalement écrit en Lua et intégré dans l'environnement à la manière de ce que fait Scientific Works avec Maple par exemple.

§5. Les distributions et les éditeurs LaTeX

§6. Et encore !

`flowframe` Une extension capable de partager une page en plusieurs frames. Bien pratiques pour créer des brochures ou tout un magazine par exemple. La revue “Cahiers des prépas” utilise massivement flowframe.

`pgfplots` Une extension reposant sur PGF et spécialisée dans le tracé de graphe de fonctions. Elle cache les détails de la syntaxe relative à gnuplot pour les fonctions complexes.

`papertex` Une classe LaTeX permettant de créer un journal ou un magazine. Des commandes sont prévues pour gérer les différentes parties du journal (première page, table des matières), pour étaler les articles sur plusieurs colonnes avec des entête courant le long de la page ... Un exemple complet est fourni avec la documentation officielle.

Extensions personnelles

§1. Compteurs et macros de mesures LaTeX

Les compteurs

`\newcounter`
`\setcounter`
`\stepcounter`
`\addtocounter`
`\value`

Un compteur est derrière chaque objet numéroté automatiquement par LaTeX. Ce qui inclue les numéros de chapitres, de sections, des sous-sections, des items d'une énumération … En outre il est possible de créer ses propres compteurs.

On crée un compteur (appelé ici `MonCompt`) par la séquence

```
\newcounter{MonCompt}
```

Dés lors (les valeurs sont données à titre indicatif)

`\setcounter{MonCompt}{1}`	donne la valeur `1` au compteur `MonCompt`
`\stepcounter{MonCompt}`	incrémente le compteur `MonCompt` d'une unité
`\addtocounter{MonCompt}{3}`	ajoute la valeur `3` au compteur `MonCompt` (on peut ajouter une valeur négative)
`\value{MonCompt}`	retourne la valeur du compteur `MonCompt`, cette commande n'est pas faite pour imprimer la valeur du compteur, mais par exemple pour faire des tests.
`\theMonCompt`	contient la valeur du compteur `MonCompt`
`\arabic{MonCompt}`	imprime une chaine de caractère qui est la représentation entière de `MonCompt`
`\alph{MonCompt}`	imprime une lettre en minuscule (a,b,c, ...) qui correspond au compteur `MonCompt` dans l'ordre alphabétique

`\Alph{MonCompt}`	la même chose mais en lettre majuscules
`\roman{MonCompt}`	représentation du compteur en chiffre romain minuscule
`\Roman{MonCompt}`	chiffres romains majuscules

Il est possible de définir d'autres représentations (par exemple 1er, 2ème, ainsi de suite) mais cela dépasse le cadre de cet article.

Les compteurs prédéfinis de LaTeX

Compteur	sa signification
`chapter`	numéro de chapitre
`section`	de section
`subsection`	numéro de sous-section
`subsubsection`	d'une sous sous-section
`paragraph`	de paragraphe
`subparagraph`	de sous-paragraphe
`page`	de la page courante
`equation`	d'un équation mathématique utilisant les environnements `equation`, `eqnarray`,…
`figure`	d'une figure (environnement `figure`)
`table`	d'une table (environnement `table`)
`footnote`	d'une note de bas de page
`enumi`	du premier niveau dans une énumération (environnement `enumerate`)
`enmii`	du deuxième niveau dans une énumération
`enumiii`	troisième niveau dans une énumération
`enumiv`	quatrième niveau dans une énumération
`day, mounth, year`	compteurs qui contiennent respectivement le jour, le mois et l'année courants

Ajoutons à cela les compteurs

`secnumdepth` compteur qui règle la profondeur de numérotations de sections. `\setcounter{secnumdepth}{1}` par exemple fera que seule les sections seront numérotées pas les sous-sections et leurs subdivisions. La valeur par défaut de ce compteur est 2. (les section et les sous-sections sont numérotées pas les subsubsection, paragraph, …)

`tocdepth` compteur réglant la profondeur de prise en considération des sections dans les tables de matières. Avec une valeur 1 les sous-sections et leurs subdivisions ne seront pas intégrées dans la tables des matières.

Astuces

☛ Pour changer le style de numérotation d'un compteur `compt` il suffit de redéfinir la commande `\thecompt` associée. Par exemple :

- `\renewcommand{\theenumii}{\roman{enumii}}` fera que les énumérations de deuxième niveau seront notée i, ii, iii, …(au lieu de a, b, c, …)
- `\renewcommand{\thesection}{\Roman{section}}` numérotera les sections I, II, III, …(au lieu de 1, 2, 3, …)

`\labelenumi`
`\labelenumii`
`\labelenumiii`
`\labelenumiv`

☛ L'environnement `enumerate` définit les commande `\labelenumi`, `\labelenumii` …, dont la valeur est le style de numérotation selon le niveau de l'énumération . Par exemple `\labelenumii` est définie par

```
\newcommand{\labelenumii}{(\alph(enumii)}.
```

Il suffit de redéfinir ces commandes pour changer de style de numérotations (avec ces informations il n'est plus utile d'utiliser des écritures comme `\item[1.a.i]` qui font perdre l'automatisme des numérotations)

Les mesures

Les unités de mesures sous TeX

`1pt (point)` =
`1pc (pica)` =
`1in (inch)` =
`1ex` = la hauteur de la lettre x dans la police courante
`1em` = la largeur de la lettre M dans la police courante
`1mu` = 1/18 ex mesure utilisée dans le mode mathématique et utilisée pour régler les espaces entre les caractères

On déclare une nouvelles macro de mesure avec la commande

```
\newlength{\MaMesure}
```

Dès lors on dispose des commandes de manipulation de mesures (LaTeX)

`\setlength{\MaMesure}{2pc}`	donne la valeur `2pc` à la macro `\MaMesure`
`\addtolength{\MaMesure}{3pc}`	ajoute la mesure `3pc` à `\MaMesure`, la valeur ajoutée peut être négative ou une autre macro de mesure
`\settowidth{\MaMesure}{<Du texte>}`	donne à `\MaMesure` la largeur du texte `<Du texte>` dans la police courante, fonctionne aussi si `<Du texte>` est une boite TeX
`\settoheight{\MaMesure}{<Du texte>}`	la même chose mais avec la hauteur dur texte `<Du Texte>`
`\settodepth{\MaMesure}{<Du texte>}`	la profondeur du texte `<Du texte>` cette fois

`\newlength`
`\setlength`
`\addtolength`
`\settowidth`
`\settoheight`
`\settodepth`

Les colles

Chaque macro de mesure peut recevoir non seulement une valeur fixe, mais une valeur avec des espaces élastiques. La commande

```
\setlength{\MaMesure}{2ex plus .1ex minus .2ex}
```

signifie que la valeur normale de `\MaMesure` et de `2ex` mais que si besoin elle peut être augmenté de `0.1ex` ou diminuée de `0.2ex`.

Exemples de mesures prédéfinies de LaTeX

`\paperheight,\paperheight`	la hauteur et la largeur du format de papier utilisé
`\textheight,\textwidth`	la hauteur et la largeur de la zone de texte dans le document
`\linewidth`	la longueur de la ligne courante, différent de `\textwidth` en général (dans un environnement `multicol` par exemple)
`\rightskip,\leftskip`	les marges gauches et droites du paragraphe courant
`\headheight,\headsep`	hauteur de la zone d'entête de la page et hauteur de la séparation de cette dernière de la zone de texte
`\footskip`	séparation de la zone de texte avec la zone de pied de page
`\fboxrule, \fboxsep`	l'épaisseur du trait, et espace séparant ce dernier du texte pour les commandes de mise boite LaTeX(`\fbox`,...)
`\columnsep`	en cas d'environnement de multicolonnage, mesure la séparation verticale des colonnes
`\arraycolsep`	mesure la hauteur de la séoaration des ligne dans un environnement mathématique `array`

Les mesures du mode mathématique

`\thinmuskip`	espace de séparation normal, valeur par défaut=`0mu`
`\medmuskip`	espace de séparation moyen, valeur par defaut=`1mu`
`\thickmuskip`	espace insérée des deux côté d'un opérateur binaire, valeur par défaut=`2mu`
`\arraycolsep`	espace séparant les lignes dans un environnement `array`, fournie par l'extension `amsmath`
`\matsurround`	espace des deux cotés d'une formule mathématique en mode inline

`\abovedisplayskip` `\belowdisplayskip`	espaces verticales en haut et en bas d'une formule en mode `displaystyle` (`$$...$$`), valeur par défaut = `12pt plus 3pt minus 9pt`
`\abovedisplayshortskip` `\belowdisplayshortskip`	espaces verticales en haut et en bas d'une formule en mode `displaystyle` quand les lignes de texte normal entourant la formule sont trop courte. Valeur par défaut = `0pt plus 3pt`

Gestion des polices

§1. Introduction

Les technos policières ;-)
Plusieurs technologies de gestion et de rendu des polices de caractères coexistent. Les techniques bitmaps sont obsolètes et les polices disponibles de nos jours sont exclusivement vectorielles. Une technologie s'est érigé en standard : OpenType. Développée en commun par les grandes compagnies d'informatiques, elle offre deux familles de polices : TrueType (Microsoft) et OpenType (Adobe). Les polices de la première familles portent en général l'extension `.ttf`*, ceux de la deuxième l'extension* `.otf`*. À côté survivent toujours les polices Postscript Type 1 qui sont encore très utilisées dans les milieux professionnels et par les logiciels de graphisme, et aussi . . . par TeX/LaTeX.*

La conception d'un document aussi facilement lisible que plaisant visuellement dépend avant tout du choix des familles de polices à utiliser. Les trois familles habituelles de polices (sérif, sans sérif et monospace), doivent être sélectionnée de façon harmonieuse. À cela s'ajoute pour TeX les polices propres au mode mathématique qui sont elles gérées de façon relativement indépendante.

La gestion de polices et leurs utilisation sous TeX était un vrai casse-tête (une science à part entière). Heureusement de nos jours les distributions TeX/LaTeX cachent cette complexité à l'utilisateur en installant et en configurant au mieux la partie responsable de leurs gestions. Elle permettent en général l'accès à telle ou telle police par le biais d'un jeu d'extensions. Il faut juste savoir que le moteur TeX en tant que tel n'a aucune connaissance des polices, il se contente de créer des boites et de déléguer ensuite à un autre programme le rôle de les remplir. L'insertion des polices incombe à un programme appelé METAFONT (crée aussi par Donald Knuth) qui utilisait à l'origine TeX uniquement des polices bitmap (toujours disponible, et sont même les polices par défaut dans une installation minimale). METAFONT néanmoins gère aussi les polices vectorielles de type Postscript. De base TeX/LaTeX ne supporte pas les polices OpenType et TrueType. Mais ces dernières sont accessibles à travers quelques évolutions modernes du moteur TeX (xetex et luatex).

§2. Les commandes de sélection de polices

`\textrm`, `\textsf`, `\texttt`
`\textsl`, `\textit`,
`\textbf`, `\textsc`

Les commandes LaTeX pour le changemment de corps de polices sont organisées en trois grandes familles :

1. Les commandes `\textxx` qui prennent un argument, la commande agissant sur l'argument seulement.

2. Les commandes à effet permanent qui agissent sur tout un bloc de texte délimité par un groupe LaTeX. Elles même se divisent en plusieurs catégories
   - les commandes de changement de familles (sérif, sans sérif, monospace) ; `\rmfamily`, `\sffamily`, `\ttfamily`
   - les commandes de changement de graisse (normal, gras) ; `\mdseries`, `\bfseries`
   - les commandes de changement de dessin (droit, penché, italique, petite capitale (ou smallcaps)) `\upshape`, `\slshape`, `\itshape`, `\scshape`

   Ces commandes ne permettent en aucun cas de changer de police elle même. Les polices doivent normalement être fixées une fois[1] pour toute dans le préambule à l'aide des commandes adéquates ou en utilisant une extension LaTeX. (ce deuxième choix est le plus judicieux en général).
3. Les commandes de changement de taille, qui comme les commande de la deuxième catégories sont à effet permanent.

   `\tiny`, `\scriptsize`, `\footnotsize`, `\small`, `\normalsize`, `\large`, `\Large`, `\LARGE`, `\huge`, `\Huge`

   Il est à noter que ces commandes n'agissent pas seulement sur la taille des polices mais règlent aussi l'interligne en conséquence.

Les commandes de la deuxième et troisième catégories ont des effets cumulables, contrairement aux commandes similaires `\bf`,`\it`,`\sc`,`\sl`, qui sont des commandes TeX (et non LaTeX), ces dernières sont à proscrire dans un document LaTeX.

Changer localement de famille de police

`\fontfamily`,`\fontshape`, `\fontseries`, `\fontsize`, `\selectfont` `\usefont`

On peut charger localement une police et fixer ses propriétés en utilisant les commandes (toutes ces commandes sont écrites à l'intention des développeurs et doivent être utilisé avec parcimonie par l'utilisateur final) :

`\fontencoding` permet de fixer l'encodage à utilser, OT1 est la valeur qu'il faut lui donner pour les caractères latins (inclus les caractères accentués)

`\fontfamily` permet de sélectionner la police elle-même à condition de connaître son nom interne (`ptm` pour Times, `ppl` pour Palatino, `phv` pour Helvetica …)

`\fontshape` les valeurs que prend cette commande sont `up`, `sl`, `it`, `sc`, l'effet est le même que la commande `\xxshape` correspondante ;

`\fontseries` prend les valeurs `md` et bf, l'effet et le même que `\mdseries` ou `bfseries` ;

`\fontsize`$\{t\}\{i\}$ pour fixer la taille de la police séléctionnée, t pour la taille de la police, i pour l'interligne. Indispensable si vous voulez utiliser une taille plus grande que `\Huge` ou plus petite que `\tiny` ;

`\selectfont` une combinaison des commandes précédentes doit être normalement suivie de la commande `\selectfont` pour rendre la sélection de la police ou de ses propriétés effectives. Ceci est inutile si vous faîtes suivre ces commandes, d'un `\bfseries`, `\itshape`, …

Par exemple la ligne

```
\fontfamily{pag}\fontsize{40}{48}\selectfont
```

sélectionne la police Avant Garde avec une taille de 40 points et un interligne de 48 points.

1. ce qui ne veut pas dire que changer de police en milieu de document n'est pas possible, mais …

`\usefont` s'utilise comme suit

```
\usefont{<encoding>}{<famille>}{<serie>}{<shape>}
```

qui est un raccourci de la suite de commandes

```
\fontencoding{<encoding>}\fontfamily{<famille>}
\fontseries{<series>}\fontshape{<shape>}\selectfont
```

Noter que `\usefont` exécute `\selectfont` après la sélection de la police qui devient donc immédiatement active.

Toutes ces commandes ont un effet permanent dont la portée peut être limitée par un groupe LaTeX.

Changer globalement une police

`\familydefault`
`\rmdefault`
`\sfdefault`
`\ttdefault`

On peut changer globalement une famille de polices en redéfinissant (avec `\renewcommand`) les macros suivantes

`\rmdefault` permet de changer la police sérif utilisée par défaut dans le document. La valeur à donner la macro est soit le nom interne d'une police soit une autre macro de changement global d'une police ;

`\sfdefault` la même chose pour la police sans sérif.

`\ttdefault` pour la police monospace.

`\familydefault` la police principale (par défaut) du document

Par exemple la ligne

```
\renewcommand{\rmdefault}{put}
```

change la police sans sérif du document par la police Utopia.

```
\renewcommand{\familydefault}{\sfdefault}
```

fait que la police courante sans sérif est utilisée comme police principale.

Noter que chaque utilisation directe de ces commandes dans le corps du document sera simplement remplacé par le nom de la police correspondante. Voilà ce que produisent ces commandes dans ce document

`\rmdefault`	MinionPro-OsF
`\sfdefault`	iwona
`\ttdefault`	cmtt
`\familydefault`	MinionPro-OsF

§3. Les polices du mode mathématique

Elles sont gérés de façon indépendante de celles du texte normal. En général, il vaut mieux utiliser des extensions qui fixent aussi les polices pour le mode mathématiques, car leurs gestion est assez complexe.

On pourrait ici se contenter de citer quelque commandes qui permettent de changer les propriétés de la police du mode mathématique

`\mathnormal`	$abc\ XYZ\ \alpha\beta\gamma\ \Phi\Psi$
`\mathrm`	$\mathrm{abc\ XYZ\ \alpha\beta\gamma\ \Phi\Psi}$
`\mathsf`	$\mathsf{abc\ XYZ\ \alpha\beta\gamma\ \Phi\Psi}$
`\mathtt`	$\mathtt{abc\ XYZ\ \alpha\beta\gamma\ \Phi\Psi}$
`\mathit`	$\mathit{abc\ XYZ\ \alpha\beta\gamma\ \Phi\Psi}$
`\mathbf`	$\mathbf{abc\ XYZ\ \alpha\beta\gamma\ \Phi\Psi}$
`\mathcal`	$\mathcal{XYZ}\ \alpha\beta\gamma\ \Phi\Psi$

```
\mathnormal,\mathrm,\mathsf,
\mathtt,\mathit,\mathbf,\mathcal
```

Ces commandes doivent être insérés au sein même du mode mathématique. On peut noter que

- `\mathrm`,`\mathsf`,`\mathtt`,`\mathit`,`\mathbf` n'agissent que sur les lettres latines et ignorent les lettres grecques, mais aussi les symboles et les oprérateurs ;
- `\mathcal` par contre n'agit que sur les lettres latines majuscules.

À coté, il y'a la commande `\mathversion` qui doit elle être insérée en dehors du mode mathématique et son effet est permanent et peut être limité par un groupe LaTeX. Elle prend deux arguments `normal`, mode par défaut et `bold` pour mettre les textes mathématiques en gras

```
\mathversion{normal}
\mathversion{bold}
```

```
\mathversion{bold}
$\Gamma(x)=\int_{0}^{+\infty}t^{x-1}e^{-t}dt$
```

$$\Gamma(x) = \int_0^{+\infty} t^{x-1}e^{-t}dt$$
`\mathversion{normal}`

$$\boldsymbol{\Gamma(x) = \int_0^{+\infty} t^{x-1}e^{-t}dt}$$
`\mathversion{bold}`

où on constate que dans la version avec `\mathversion{bold}`, tout (ou presque) est mis en gras sauf le symbole =.

Remarques

- La commande `\boldmath` est un raccourci de `\mathversion{bold}` ;
- Quand on change le style d'un titre de section qui contient du texte mathématique, il vaut mieux utiliser une graisse normale et de jouer sur la taille ou la famille de la police pour distinguer le titre du reste du texte. Il n'y a aucune solution totalement satisfaisante pour mettre des mathématiques en gras. Le problème n'incombe pas à TeX ou à METAFONT mais au fait qu'il n'y a pas de police assez complète pour offrir une version en gras pour tous les caractères utilisable en maths ;
- Pour mettre du texte mathématique en gras, il existe une extension : `bm` dont le seul rôle est de définir la commande `\bm` qui met son argument en gras.

Finalement signalons l'existence de paquets très complets de polices pour le mode mathématique, mais qui sont malheureusement tous commerciaux. Il existe néanmoins un projet libre dont l'objectif est de créer une police pour le texte mathématique et qui est distribuée au format Postscript Type 1, mais aussi OpenType : STIX.

§4. Les paquets de polices officiels

Au début fût Computer Modern

Historiquement TEX utilisait des polices bitmaps (la famille Computer Modern) conçues avec le système sous-jacent (METAFONT) par Donald Knuth lui même. L'utilisation de ces polices dans un fichier qui va être converti en PDF pose certains problèmes : Le rendu des polices dans Adobe Reader (mais pas dans d'autres visionneurs PDF) est mauvais et le PDF généré est non "cherchable", comprenez par là que la fonction de recherche du viewer PDF ne fonctionnera pas (essayez avec les fichiers des programmes officiels des CPGE).

Latin Modern, presque les mêmes en Postscript

METAFONT fût amélioré pour qu'il gère aussi les polices Postscript (vectorielles), et une version à ce format a été préparée à partir des polices CM originales (qui constituent l'ensemble le plus complet utilisable sous TEX à nos jours). Pour les activer il suffit de charger l'extension `lmodern`.

```
\usepackage{lmodern}
```

Noter que le paquet `cm-super` installable en option (assez lourd, il pèse plus que 64Mo) contient (entre autre) une copie exacte des polices Computer Modern en version Postscript et Truetype. Installer ce paquet fera que, sans besoin d'intervenir, les fichiers PDF générés par `pdflatex` utiliseront automatiquement la version Postscript de CM. L'autre intérêt d'avoir des version Postscript et TrueType de CM est évident quand on crée par exemple des graphiques dans un logiciel séparé.

PSNFSS2, le pack standard

ce pack de polices est connu sous le nom PSNFSS2

LaTeX2e à introduit un ensemble de spécifications sur la gestion des polices Postscript connu sous le nom de NFSS2. NFSS2 est distribué avec un ensemble de polices à ce format qui est installé par défaut sur toute distribution TEX/LaTeX et qui ne souffre pas de problèmes de licence . Il peut s'avérer suffisant pour couvrir la plupart des besoins, mais en conséquence la majorité des documents produit avec ces polices de base auront presque la même apparence. Pour celui qui cherche à pousser la personnalisation de ces documents les solutions non officielles sont nombreuses.

L'accès à ces polices peut se faire de façons individuelles ou par l'intermédiaire d'extensions. Deux de ces extension offrent aussi un ensemble de polices relativement complet pour le mode mathématique :

`mathpazo` qui utilise la police Palatino. Palatino est utilisée par exemple par les énoncés du CNC mais à travers une extension qui est devenue obsolète : `palatino`;

`mathptmx` qui utilise la police Times.

Extension	Sérif	Sans Sérif	Monospace	Maths
par défaut	CM Roman	CM Sans Serif	CM Typewriter	CM Roman
mathpazo	Palatino			Palatino
mathptmx	Times			Times
helvet		Helvetica		
avant		Avant Garde		
courier			Courier	
chancery	*Zapf Chancery*			
bookman	Bookman	Avant Garde	Courier	
newcent	New Century Schoolbook	Avant Garde	Courier	
charter	Charter			

Table 3.1: Les extensions de NFSS2

Nom interne	Nom Postscript	Texte témoin
pag	Avant Garde	Théorème de Cauchy – Lipschitz
pbk	Bookman	Théorème de Cauchy – Lipschitz
bch	Charter	Théorème de Cauchy – Lipschitz
pcr	Courier	Théorème de Cauchy – Lipschitz
phv	Helvetica	Théorème de Cauchy – Lipschitz
pnc	New Century Schoolbook	Théorème de Cauchy – Lipschitz
ppl	Palatino	Théorème de Cauchy – Lipschitz
ptm	Times	Théorème de Cauchy – Lipschitz
pzc	Zapf Chancery	*Théorème de Cauchy – Lipschitz*
put	Utopia	Théorème de Cauchy – Lipschitz

Table 3.2: Les polices du pack PSNFSS2

§5. Autres paquets disponibles sur les ctan

§6. Les supers paquets avec polices pour le mode mathématique : cm-super, fourier, MinionPro et mtpro2

§7. L'alternative : fontspec et XeLaTeX

XeLaTeX comme mentionné auparavant donne accès aux polices installés dans le système d'exploitation et non seulement celles visibles par TeX. À travers son extension `fontspec`, l'utilisation de ces polices est simplifiée à l'extreme, et permet même d'utiliser les propriétés très pointues de certaines polices. Les commandes LaTeX usuelles qui agissent sur les polices (`\bfseries`,`\sffamily`,`\scshape`,…) restent valables.

Pour utiliser `fontspec`, il suffit de charger l'extension du même nom. Les extensions `fontenc` et `inputenc` ne devraient pas être chargées, mais plutôt l'extension`xunicode` qui fait le même travail et qui est propre à XeLaTeX.

La compilation devrait se faire ensuite par la commande `xelatex` (et non pas `latex` ou `pdflatex`). Si l'extension `fontspec` est chargée, toute tentative de compilation avec les commandes classiques se terminera par une erreur reclamant l'utilisation de `xelaltex`.

```
\usepackage{xunicode}
\usepackage{fontspec}
```

`fontspec` conformément aux usages LaTeX, offre des commandes pour fixer chacune des familles de polices une fois pour toute dans le préambule, mais aussi une commande qui permet de sélectionner localement une police et de lui attribuer des propriétés.

`\setmainfont`
`\setromanfont`
`\setsansfont`
`\setmonofont`

`\setmainfont`, `\setromanfont` `\setsansfont`, et `\setmonofont` permettent de fixer dans le même ordre la police principale du document, la police sérif, la police sans sérif et la police monospace (ce qui rappelle les commandes `\familydefault`, `\rmdefault`, `\sfdefault`, `\ttdefault`). Leurs arguments devrait être le nom d'une police tel qu'il apparait dans le gestionnaire de polices de votre système d'exploitation.

```
\setmainfont[Mapping=tex-Text,Numbers=OldStyle]{Times New Roman}
\setsansfont{Candara}
\setmonofont{Consolas}
```

Conception d'un fichier style pour séries d'exercices

§1. Les ingrédients

exercise

est une extension spécialisé dans la gestion de feuilles d'exercices. Elle offre des fonctionnalités assez complète et reste largement personnalisable. L'une des choses importantes qu'elle permet et la possibilité de saisir la solution d'un exercice immédiatement après son énoncé, mais de reporter son intégration dans le document en un lieu qui peut être décidé par l'utilisateur.

enumitem

La solution la plus complète et la plus polyvalente de personnalisation des styles de tous les environnements d'énumération LaTeX, avec la possibilité d'en créer de nouvelles. Permet entre autre de changer les labels des énumérations (plus besoin pour les rédacteurs de corrigés de sujets de concours d'utiliser dans leurs documents des monstruosités comme `\item [I.3.a)]` (contraire à l'esprit LaTeX).

titlesec

Une extension très puissante (mais au prix d'une prise en main un peu délicate) de personnalisation des titres de sectionnement sous LaTeX, parties, chapitres, sections, sous-sections … tous y passe et avec une large plage de manœuvre.

En outre `titlesec` est accompagné d'une autre extension, `titletoc`, qui permet de personnaliser les tables de matières.

En option Ti*k*Z

Ti*k*Z au travers de son option `overlay` permet d'insérer des éléments graphiques (y compris du texte) par leurs coordonnées absolues sur toute la surface de la page. Associé à une extension qui permet d'insérer des objets sur toutes les pages d'un document, on peut pousser la personnalisation de son document très loin. Un exemple de telles extensions : `eso-pic`.

§2. Utilisation de l'extension `exercise`

Chargement de l'extension

```
\usepackage[<options>]{exercise}
```

L'extension possède les options suivantes :

`noexercise` cache tous les énoncés (ne les intègre pas dans le fichier produit après compilation) ;

`noanswer` cache les solutions des exercices ;

`answerdelayed` sauvegarder temporarairement les solutions, en attentdant de les intègrer ailleurs dans le document avec la commande `\shipoutAnswer`. C'est peut être l'option la plus utile ;

`exercisedelayed` la même chose mais avec les énoncés cette fois. Peut être utile dans un cours par exemple. On intègre les énoncés par la commande `\shipoutExercise`

`lastexercise` s'il n'y a pas de référence donnée à une solution, alors elle se réfère au dernier énoncé.

L'environnement Exercise

```
\begin{Exercise}
\end{Exercise}
```

Une fois l'extension chargée, on dispose de l'environnement `Exercise` avec une série d'options de type `clé=valeur` dont la liste est :

`label={chaine}` Label pour référencer l'exercice, on pourra plus tard utiliser cette référence en utilisant la commande standard LaTeX `\ref`. (genre : "d'après l'exercice `\ref{thm:darboux}`" quand on à déjà labélisé un exercice avec l'entrée `label={thm:darboux}`

`title={chaine}` Titre de l'exercise, le résultat de l'usage de cette option sera la définition de la macro `\ExerciseTitle` avec comme valeur, la valeur passée à l'option `title`. Si l'option n'est utilisé la macro `\ExerciseTitle` aura comme valeur `{}`.

`difficulty={nombre}` indication de la difficulté de l'exercice, une ou plusieurs étoile

seront insérés à gauche de l'entête, elle sera disponible ensuite à travers la commande `\ExerciseDifficulty`;

`origin={chaine}` origine de l'exercice, un ouvrage ou un sujet de concours, elle sera disponible ensuite à travers la commande `\ExerciseOrigin`;

`name={chaine}` Pour changer la dénomination de l'exercice, genre question, problème, ...

`counter={compt}` compteur de l'exercice, le compteur compt devrait être déclarée auparavant avec par exemple `\newcounter{compt}`;

`number={numero}` permet de numéroter manuellement l'exercice, toute chaine de caractères est acceptée.

Par exemple

```
\begin{exercise}[name=Exo,title={Inégalité de Haddamard},
origin={E3A, 2oo1, Maths B, PSI}]
 .... <texte de l'énoncé> ....
\end{exercise}
```

va produire l'entête (style par défaut)

Exo 1 Inégalité de Haddamard (*E3A, 2001, PSI*)

Il est possible ensuite de définir de nouvels environnements se basant sur `Exercise`. Avec des noms et des compteurs différents. Cela permettra d'avoir dans une même feuille des exercices et des problèmes numérotés de façons différentes.

```
\newcounter{Probleme}
\newenvironment{Probleme}
{
    \begin{Exercise}[name={Problème},counter={Probleme}]
}
{
    \end{Exercise}
}
```

Le problème ici c'est qu'on perd la gestion des options de l'environnement `Exercise`. On pourrait alors ajouter un argument optionnel (vide par défaut) à notre environnement `Probleme`, qui sera ajouté à la liste des options de `Exercise`

```
\newcounter{Probleme}
\newenvironment{Probleme}[1][]
{
    \begin{Exercise}[name={Problème},counter={Probleme},#1]
}
{
    \end{Exercise}
}
```

L'extension permet de personnaliser presque tous les éléments de l'entête des exercices au travers de macros et de commandes de formatage. Dans ce qui suit une liste de celles

qui sont peut être les plus utiles. Noter que pour changer la définition d'une macro on n'a qu'a la redéfinir avec la commande `\renewcommand`

\ExerciseName
\AnswerName
\ExerciceHeaderTitle
\ExerciseHeaderOrigin
\ExerciseHeaderNB
\ExerciseHeader

`\ExerciseName` le nom utilisé pour l'entête des énoncés, par défaut **Exercise** (en anglais). En principe si `babel`, est chargé avec l'option `francais` (avant de charger `exercise`), alors les noms sont francisés. Sinon on pourra toujours les redéfinir. Ici par exemple il suffit de faire

```
\renewcommand{\ExerciseName}{Exo}
```

pour que les exercices soient appelés Exo 1, Exo 2, …

`\AnswerName` le nom utilisé pour les entêtes des solutions, par défaut **Answer of** ;

```
\renewcommand{\AnswerName}{Solution de}
```

`\ExerciseHeaderTitle` Formatage par défaut du titre de l'exercice s'il y'en a un. Voilà comment la commande est définie par défaut

```
\newcommand{\ExerciseHeaderTitle}{\quad---\quad\
ExerciseTitle}
```

`\ExerciseHeaderOrigin` Formatage par défaut de l'origine de l'exercice

```
\newcommand{\ExerciseHeaderOrigin}{%
\ ({\usefont{\encodingdefault}{\rmdefault}{m}{it}\
ExerciseOrigin})}
```

`\ExerciseHeaderNB` Style pour le compteur de l'exercise. Définition par défaut

```
\newcommand{\ExerciseHeaderNB}{\theExercise}
```

`\ExerciseHeader` C'est la commande qui définit le style de tout l'entête, et qu'on aura avantage à redéfinir. Sa définition par défaut est

```
\newcommand{\ExerciseHeader}{%
    \centerline{%
    \textbf{\large\ExerciseHeaderDifficulty%
    \ExerciseName\ %
    \ExerciseHeaderNB%
    \ExerciseHeaderTitle%
    \ExerciseHeaderOrigin}
    } %
    \medskip
}
```

`\AnswerHeader` la même chose que `\ExerciseHeader` mais pour les solutions. La définition par défaut

```
\newcommand{\AnswerHeader}{%
    \medskip
    \centerline{
        \textbf{\AnswerName\ \ExerciseName\ \
        ExerciseHeaderNB
        }
    \smallskip}
}
```

L'extension permet de faire bien d'autres choses. Le lecteur insatisfait pourra regarder le manuel officiel qui est plutôt bien fait (32 pages).

Insertion des solutions

Si l'option `answerdelayed` a été utilisé lors du chargement de l'extension avec

```
\usepackage{answerdelayed}{exercise}
```

alors les solutions des exercices ne sont intégrés dans le document final que par ordre explicite de l'utilisateur et à la place qu'il décide, en respectant bien sûr les numérotations des énoncés (si jamais le rédacteur néglige de rédiger une solution).

`\shipoutAnswer` Il suffit d'insérer la commande

```
\shipoutAnswer
```

§3. Styles de sectionnement avec `titlesec`

Il n'est pas facile de personnaliser les styles des titres de sections avec le système de base. `titlesec` est une extension très puissante, bien que la syntaxe qu'elle impose est difficile à digérer et que la documentation n'est pas un modèle de clarté.

Un titre se compose en général d'un label (un numéro de sectionnment, du texte comme Chapitre 1, …) et le texte du titre. Ses deux parties sont normalement traitées séparément (et ce même avec les commandes de base LaTeX), elles sont en général placées dans des paragraphes LaTeX séparés.

`\titleformat` La commande `\titleformat` offerte par l'extension prend 7 argument dont 2 sont optionnels et s'utilise de la manière suivante

```
\titleformat{<section>}[<style>]{<format>}{<label>}
            {<sep>}{<avant>}[<apres>]
```

où

`<section>` est une commande de sectionnement : `\chapter`, `\section`, `\subsection`, `\paragraph`, …

`<style>` est un style prédéfini par l'extension, la liste des styles supportés est donnée par le tableau 4.3 page 29

`<format>` description du style qui va s'appliquer au titre en entier, label et texte. Il peut être décrit par des commandes LaTeX standards plus les commandes ajoutées par `titlesec`

`<label>` description du style qui s'applique seulement au label du titre, normalement c'est ici qu'on doit aussi préciser le label lui même.

`<sep>` espace qui sépare le label du titre de son texte. L'espace est verticale pour le style `display` et c'est l'espace qui sépare le texte du titre du cadre pour le style `frame`.

`<avant>` description du style (avec des commandes) qui s'applique au texte du titre, la dernière commande aura pour argument ce texte. Si l'option `explicit` de

l'extension a été utilisée alors l'argument `#1` contient ce texte, si cet argument n'est pas utilisé alors le texte sera ignoré.

`<apres>` commandes à exécuter à la fin du titre, on peut insérer des espaces verticales avec la commande `\addvspace`, mais aussi du texte …

Une autre commande est disponible, `\titlespacing`, dont le rôle est de régler les espaces autour de l' unité de sectionnement

`\titlespacing`

```
\titlespacing{<section>}{<gauche>}{<espavant>}
{<espapres>}[<droite>]
```

avec

`<section>` une commande de sectionnement ;

`<gauche>` espace ajouté à gauche du titre. Par contre c'est la longueur du titre pour les styles `leftmargin`, `rightmargin` et `drop`, la longueur maximale du titre dans le style `wrap` et c'est l'indentation du texte dans le mode `runin`

`<espavant>` espace verticale avant le titre ;

`<espapres>` espace entre le titre et le texte qui le suit. Elle est verticale dans les modes `hang`, `bloc` et `display`, et horizontale dans les modes `runin`, `drop`, `wrap` et `xxxmargin` ;

`<droite>` optionnel, pour les modes `hang`, `block` et `display`, espace pour la marge droite du titre.

Les commandes suivante sont ajoutées par l'extension et peuvent être utilisées dans les différentes parties précédentes (là où elles ont un sens)

`\fillast` justifie le texte du titre sauf pour sa dernière ligne qui est centrée.

`\filleft` Justifie le texte du titre à droite ;

`\filright` justifie le texte du titre à gauche ;

`\filcenter` centre le texte du titre

`\titlerule[epaisseur]` tire un trait horizontal, il est possible de donner l'épaisseur du trait en option.

`\titlerule*[largeur]<motif>` permet non pas de tracer un trait simple mais de remplir la ligne avec des `<motif>`, l'argument optionnel `largeur` spécifiant la largeur de la boite dans laquelle est enfermé chaque motif. Par exemple `\titlerule*[5ex]{<|>}` tracera créera des boîtes de `5ex` de largeur dans lesquelles seront placés les motifs :

<|> <|> <|> <|> <|> <|> <|> <|> <|> <|> <|> <|> <|> <|> <|>

Style	**Description**
`hang`	Valeur par défaut, elle imite le style standard LaTeX
`block`	tout le titre dans un seul block
`display`	le label et le texte du titre dans des blocks séparés
`runin`	le texte normal continu le long du texte titre
`leftmargin`	le titre dans la marge de gauche
`rightmargin`	le titre dans la marge de droite
`wrap`	le texte normal enveloppe le texte du titre
`frame`	comme `display` mais un cadre entoure le titre, le label est inséré sur la ligne d'en haut du cadre

TABLE 4.1: Tableau des styles supportés par `titlesec`

Exemples d'utilisation de `titlesec`

```
\titleformat{\subsection}[wrap]
    {\large}
    {\S\arabic{subsection}}
    {1ex}
    {#1}
\titlespacing{\subsection}
    {10pc}
    {1ex minus .1ex}
    {1pc}
```

§2. Un titre pour voir l'effet des réglages

Du texte normal pour bien voir comment se comporte le réglage effectué avec l'extension `titlesec`. Texte assez long pour vraiment bien voir l'effet des réglages. `titlesec`, puissant mais casse pieds et sa documentation est vraiment mal foutues.

```
\titleformat{\subsection}[runin]
    {\large}
    {\S\arabic{subsection}}
    {1ex}
    {#1}
\titlespacing{\subsection}
    {0pt}
    {1ex minus .1ex}
    {1pc}
```

§3. Un titre pour voir l'effet des réglages Du texte normal pour bien voir comment se comporte le réglage effectué avec l'extension `titlesec`. Texte assez long pour vraiment bien voir l'effet des réglages. `titlesec`, puissant mais casse pieds et sa documentation est vraiment mal foutues

```
\titleformat{\subsection}[
leftmargin]
    {\large}
    {\S\arabic{subsection}}
    {1ex}
    {#1}
\titlespacing{\subsection}
    {6pc}
    {1ex minus .1ex}
    {1pc}
```

§4. Un titre pour voir l'effet des réglages

Du texte normal pour bien voir comment se comporte le réglage effectué avec l'extension `titlesec`. Texte assez long pour vraiment bien voir l'effet des réglages. `titlesec`, puissant mais casse pieds et sa documentation est vraiment mal foutues

```
\titleformat{\chapter}[frame]
{}
{\filright\sffamily\ Chapitre \arabic{subsection}\ }
{1ex}
{\large \filleft #1}
[\addvspace{-2ex}\hfill\scriptsize\itshape Stage \LaTeX]
```

Chapitre 5

Un titre pour voir l'effet des réglages

Stage LaTeX

Du texte normal pour bien voir comment se comporte le réglage effectué avec l'extension `titlesec`. Texte assez long pour vraiment bien voir l'effet des réglages. `titlesec`, puissant mais casse pieds et sa documentation est vraiment mal foutues

§4. Entête et pieds de pages