

Une courte (?) introduction à ConT_EXt Mark IV

Une courte (?) introduction à ConT_EXt Mark IV

Version 1.6 [2 janvier 2021]

© 2020-2021, Joaquín Ataz-López

Titre original: Una introducción (no demasiado breve) a ConT_EXt Mark IV

Traduction française: A bon ami qui souhaite rester anonyme.

L’auteur du présent texte, ainsi que ses traducteurs anglais et français, autorisent sa libre distribution et utilisation, ce qui inclue le droit de copier et de redistribuer ce document sur support numérique à condition que l’auteur soit cité, et que le document ne soit inclus ni dans un paquet, ni dans une suite logicielle, ni dans une documentation dont les conditions d’utilisation ou de distribution ne prévoient pas la le droit de copie et de redistribution libre par ses destinataires. La traduction du document est également autorisée, à condition que la paternité du texte original soit indiquée, que son statut de traduction soit indiquée, et que le texte traduit soit distribué sous la licence FDL de la Fondation pour le Logiciel Libre (Free Software Foundation), une licence Creative Commons qui autorise la copie et la redistribution, ou autre licence similaire.

Nonobstant ce qui précède, la publication et la commercialisation de ce document, ou sa traduction, nécessitera l’autorisation écrite expresse de l’auteur.

Historique des versions :

- 18 août 2020 : Version 1.0 (Uniquement en espagnol) : Document original.
- 23 août 2020 : Version 1.1 (Uniquement en espagnol) : Correction de petites erreurs de frappe et de malentendus de l’auteur.
- 3 septembre 2020 : Version 1.15 (Uniquement en espagnol) : Autres corrections de petites erreurs de frappe et de malentendus.
- 5 septembre 2020 : Version 1.16 (Uniquement en espagnol) : Autres corrections de petites erreurs de frappe et de malentendus ainsi que des petites modifications qui améliorent la clarté du texte (je crois).
- 6 septembre 2020 : Version 1.17 (Uniquement en espagnol) : C’est incroyable le nombre de petites erreurs que je trouve. Si je veux m’arrêter, je dois arrêter de relire le document.
- 21 octobre 2020 : Version 1.5 (Uniquement en espagnol) : Introduction de suggestions et correction des erreurs signalées par les utilisateurs de la liste de diffusion [ntg-context](#).
- 2 janvier 2021 : Version 1.6 : Corrections suggérées après une nouvelle lecture du document, à l’occasion de sa traduction en anglais

Table des matières

Préface	4
I Qu'est ce que ConT_EXt ? Comment travailler avec ?	13
1 ConT_EXt : une vue d'ensemble	14
1.1 Qu'est-ce que ConT _E Xt ?	14
1.2 La composition typographique de document	15
1.3 Les langages de balisage	17
1.4 T _E X et ses dérivés	19
1.5 ConT _E Xt	22
2 Notre premier fichier source	32
2.1 Préparation de l'expérience outils nécessaires	32
2.2 L'expérience elle-même	35
2.3 La structure de notre fichier d'exemple	40
2.4 Quelques détails supplémentaires sur la façon d'exécuter « context »	40
2.5 Traitement des erreurs	42
II Global aspects of the document	46
III Particular issues	47
Appendices	48

Préface*

Chère lectrice, Cher lecteur, voici un document concernant ConT_EXt un système de composition de document dérivé de T_EX, qui, lui même également, est un système de composition créé entre 1977 et 1982 par DONALD E. KNUTH à l'Université de Stanford.

ConT_EXt a été conçu pour la création de documents de très haute qualité typographique – destinés à être soit imprimés sur papier soit affichés sur écran informatique. Il ne s'agit pas d'un traitement de texte, ni d'un éditeur de texte, mais, comme indiqué précédemment, d'un *système*, ou encore d'une *suite d'outils*, destinés à la composition de documents, c'est-à-dire à la mise en page et à la visualisation des différents éléments du document sur la page de papier ou à l'écran. En résumé, ConT_EXt vise à fournir tous les outils nécessaires pour donner aux documents la meilleure apparence possible. L'idée est de pouvoir générer des documents qui, en plus d'être bien écrits, sont également « beaux ». A cet égard, nous pouvons mentionner ici ce que DONALD E. KNUTH a écrit lors de la présentation de T_EX (le système sur lequel est basé ConT_EXt) :

Si vous voulez simplement produire un document passablement bon – quelque chose d'acceptable et d'essentiellement lisible mais pas vraiment beau – un système plus simple suffira généralement. Avec T_EX l'objectif est de produire la meilleure qualité ; cela nécessite de porter plus d'attention aux détails, mais finalement vous

* Cette préface a commencé avec l'intention d'être une traduction/adaptation à ConT_EXt de la préface de « The T_EXBook », le document qui explique *tout ce que vous devez savoir sur le T_EX*. En fin de compte, j'ai dû m'en écarter ; cependant, j'en ai conservé quelques éléments qui, je l'espère, pour ceux qui le connaissent, feront écho.

ne trouverez pas cela beaucoup plus difficile, et vous pourrez être particulièrement fier du produit fini.

Lorsque nous préparons un document avec ConT_EXt, nous indiquons exactement comment celui-ci doit être transformé en pages (ou en écrans) avec une qualité typographique et une précision de composition comparable à celle que l'on peut obtenir grâce aux meilleurs imprimeurs du monde. Pour ce faire, une fois que nous avons appris le système, nous n'avons guère besoin de plus de travail que ce qui est normalement nécessaire pour taper le document dans n'importe quel traitement de texte ou éditeur de texte. En fait, une fois que nous avons acquis une certaine aisance avec ConT_EXt, au total notre travail est probablement moindre si nous gardons à l'esprit que les principaux détails de formatage du document sont décrits globalement dans ConT_EXt, et que nous travaillons avec des fichiers texte qui sont – une fois que nous nous y sommes habitués – une façon beaucoup plus naturelle de traiter la création et l'édition de documents ; d'autant plus que ces types de fichiers sont beaucoup plus légers et plus faciles à traiter que les lourds fichiers binaires des traitements de texte.

Documentation

Il existe une documentation considérable sur ConT_EXt, presque exclusivement en anglais. Par exemple, ce que l'on considère comme étant la distribution *officielle* de ConT_EXt – appelée « ConT_EXt Standalone » ¹ – contient une documentation de quelques 180 fichiers PDF (la majorité en anglais, mais d'autres en néerlandais et en allemand) avec notamment des manuels, des exemples et des articles techniques ; et sur le [site web Pragma ADE](#) (la société qui a donné naissance à ConT_EXt) il y a (le jour où j'ai fait le décompte en mai 2020) 224 documents librement téléchargeables, dont la plupart sont distribués avec la « ConT_EXt Standalone » mais également quelques autres. Cependant, cette énorme documentation n'est pas particulièrement utile durant la phase d'apprentissage de ConT_EXt, car, en général, ces documents ne s'adressent pas à un lecteur désireux d'apprendre mais novice, qui ne connaît rien du système. Sur les 56 fichiers PDF que « ConT_EXt Standalone » appelle « manuels », un seul suppose que le lecteur ne connaît rien sur ConT_EXt. Il s'agit du document intitulé « [ConT_EXt Mark IV, an Excursion](#) » ou en français « ConT_EXt Mark IV, une escapade ». Mais ce document, comme son nom l'indique, se limite à présenter le système et à expliquer comment faire certaines

¹ Au moment de la première version de ce texte, ceci était vrai ; mais au printemps 2020, le Wiki ConT_EXt a été mis à jour et nous devons supposer qu'à partir de ce moment la distribution « officielle » de ConT_EXt est devenue LMTX. Cependant, pour ceux qui entrent dans le monde de ConT_EXt pour la première fois, je recommande quand même d'utiliser « ConT_EXt Standalone » puisque c'est une distribution plus stable. L'annexe ?? explique comment installer l'une ou l'autre des distributions.

choses qui peuvent être faites avec ConT_EXt. Ce serait une bonne introduction s'il était suivi d'un manuel de référence un peu plus structuré et systématique. Mais un tel manuel n'existe pas et l'écart entre le document « [ConT_EXt Mark IV, an Excursion](#) » et le reste de la documentation est trop important.

En 2001, un [manuel de référence](#) a été rédigé ; mais malgré son titre, d'une part il n'a pas été conçu pour être un manuel complet (la tâche étant titanesque), et d'autre part il était (est) destiné à la version précédente de ConT_EXt (appelé Mark II) et intègre donc des éléments obsolètes, ce qui perturbe l'apprentissage.

En 2013, ce manuel [a été partiellement mis à jour](#) mais beaucoup de ses sections n'ont pas été réécrites et il contient des informations relatives à la fois à ConT_EXt Mark II et ConT_EXt Mark IV (la version actuelle), sans toujours préciser clairement quelles informations se rapportent à chacune des versions. C'est peut-être la raison pour laquelle ce manuel ne se trouve pas parmi les documents inclus dans « ConT_EXt Standalone ». Pourtant, malgré ces défauts, et une fois lu « [ConT_EXt Mark IV, an Excursion](#) », le manuel reste le meilleur document pour continuer à apprendre ConT_EXt. Autres informations également toujours très utiles pour démarrer avec ConT_EXt, celles contenues le sur [ConT_EXt Garden wiki](#), qui, au moment où nous écrivons ces lignes, est plein remaniement et présente une structure beaucoup plus claire, bien qu'elles mélangent également des explications qui ne fonctionnent que dans Mark II avec d'autres pour Mark IV ou pour les deux versions. Ce manque de différenciation se retrouve également dans la liste officielle des commandes « [ConT_EXt Commands](#) »¹ qui comprend, pour chacune d'elles, l'ensemble des options de configuration possibles mais ne précise pas quelles commandes ne fonctionnent que dans l'une ou l'autre des versions.

Fondamentalement, cette introduction a été rédigée en s'inspirant des quatre sources d'information énumérées préalablement : « [ConT_EXt Mark IV, an Excursion](#) », « [ConT_EXt Reference Manual 2013](#) », le contenu de [ConT_EXt Garden wiki](#), et la liste officielle des commandes « [ConT_EXt Commands](#) », en plus, bien sûr, de mes propres essais et tribulations. Ainsi, cette introduction résulte d'un effort d'investigation, et, durant un temps, j'ai été tenté de l'appeler « Ce que je sais à propos de ConT_EXt Mark IV » ou « Ce que j'ai appris sur ConT_EXt Mark IV ». Finalement, aussi vraie que soit leur teneur, ces titres ont été écartés car j'ai pensé qu'ils risquaient de dissuader certains de s'investir dans ConT_EXt. Il est certain, malgré que la documentation ait selon moi certaines lacunes, que ConT_EXt est un outil vraiment utile et polyvalent, pour lequel l'effort d'apprentissage vaut sans aucun doute la

¹ Pour la liste, voir section ??.

peine. Grâce à ConT_EXt, nous pouvons manipuler et configurer des documents texte pour réaliser des choses que ceux qui ne connaissent pas le système ne peuvent tout simplement pas imaginer.

Je ne peux pas empêcher – parce que je suis comme je suis – que mes regrets concernant les déficiences d'information resurgissent de temps en temps dans ce document. Je ne veux pas qu'il y ait de malentendu : je suis immensément reconnaissant aux créateurs de ConT_EXt d'avoir conçu un outil aussi puissant et de l'avoir mis à la disposition du public. C'est simplement que je ne peux m'empêcher de penser que cet outil serait beaucoup plus populaire si sa documentation était améliorée : il faut investir beaucoup de temps pour s'approprier ConT_EXt, non pas tant en raison de sa difficulté intrinsèque (qui existe, mais qui n'est pas supérieure à celle d'autres outils spécialisés similaires, c'est même plutôt le contraire), mais en raison du manque d'informations claires, complètes et systématiques, qui différencient les deux versions de ConT_EXt, expliquent ce qui fonctionne dans chacune d'elles et, surtout, précisent à quoi sert chaque commande, argument et option.

Il est vrai que de ce type d'information exigerait un fort investissement en temps. Mais comme de nombreuses commandes partagent des options avec des noms similaires, on pourrait peut-être rédiger une sorte de *glossaire* des options, ce qui permettrait également de détecter certaines incohérences produites lorsque deux options du même nom font des choses différentes, ou lorsque des noms d'options différents sont utilisés dans des commandes différentes pour faire la même chose.

Quant au lecteur qui s'intéresse à ConT_EXt pour la première fois, que mes plaintes ne le dissuadent pas, car s'il est vrai que le manque d'informations, claires complètes et systématiques, augmente le temps nécessaire à l'apprentissage, du moins pour les sujets traités dans cette introduction, j'ai déjà investi ce temps, de sorte que le lecteur n'aura pas à le refaire. Et déjà avec ce que vous aurez l'occasion d'apprendre dans cette introduction, vous pourrez produire des documents avec de puissants utilitaires insoupçonnés.

Incertitudes

Etant donné que ce qui est expliqué dans ce document provient essentiellement de mes propres conclusions, il est probable que, bien qu'ayant personnellement vérifié une grande partie de ce qui est exposé, certaines déclarations ou opinions soient incorrectes ou non orthodoxes. Bien évidemment, j'apprécierai toute correction, toute nuance ou encore précisions que vous accepterez de me faire parvenir à joaquin@ataz.org. Pour limiter les occasions d'erreur, j'ai essayé de ne pas aborder les sujets sur lesquels je n'ai pas trouvé d'informations ou que je n'ai pas pu (ou voulu) vérifier personnellement ; parce que parfois le résultat de mon test n'était pas concluant, d'autres fois parce que je n'ai pas toujours tout essayé : le nombre de commandes et d'options de ConT_EXt est impressionnant, et si je devais tout essayer, je n'aurais jamais réussi à finaliser cette introduction. Néanmoins, à certaines occasions je n'ai pas pu éviter de faire certaines *conjectures*, c'est à dire



une déclaration que je considère comme probable mais dont je ne suis pas totalement sûr. Dans ce cas, en marge du paragraphe, l'image qui peut être vue à gauche de cette ligne indiquera visuellement la présence d'une conjecture¹. D'autres fois, je n'ai pas eu d'autre choix que d'admettre que je ne sais pas et que je n'ai même pas d'hypothèse raisonnable à ce sujet : dans ce deuxième cas, l'image utilisée est celle insérée ici en marge du paragraphe afin de représenter plus que de simples conjectures, l'ignorance. Mais n'ayant jamais été très doué pour les représentations graphiques, je ne suis pas certain que les images sélectionnées parviennent vraiment à transmettre ces nuances.



Public visé

Autre aspect, cette introduction a été écrite pour un lecteur qui ne connaît rien à $\text{T}_{\text{E}}\text{X}$ et $\text{ConT}_{\text{E}}\text{Xt}$, bien que j'espère qu'elle pourra également être utile à ceux qui s'approchent pour la première fois de $\text{T}_{\text{E}}\text{X}$ or $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ (le plus populaire des outils dérivés de $\text{T}_{\text{E}}\text{X}$). Je suis cependant conscient qu'en essayant de satisfaire des lecteurs aussi différents, je risque de n'en satisfaire aucun. Par conséquent, en cas de doute, j'ai toujours été clair sur le fait que le principal destinataire de ce document est le nouvel arrivant dans le monde de $\text{ConT}_{\text{E}}\text{Xt}$, le nouveau venu dans cet écosystème fascinant.

Être un néophyte $\text{ConT}_{\text{E}}\text{Xt}$ n'implique pas d'être également néophyte dans la manipulation des outils informatiques. Bien que cette introduction ne présuppose aucun niveau spécifique de compétence informatique chez son lecteur, elle suppose une certaine « aisance raisonnable » en informatique qui implique, par exemple, de connaître dans les grandes lignes la différence entre un éditeur de texte et un traitement de texte, de savoir comment créer, ouvrir et manipuler un fichier texte, de savoir comment installer un programme, de savoir comment ouvrir un terminal et y exécuter une commande... et un peu plus.

En lisant les parties de cette introduction déjà écrites au moment de la rédaction, je me rends compte que parfois, je m'emporte et me lance dans des problèmes informatiques qui ne sont pas nécessaires à l'apprentissage de $\text{ConT}_{\text{E}}\text{Xt}$ et peuvent effrayer le néophyte, tandis que d'autres fois, je suis occupé à expliquer des choses assez évidentes qui peuvent ennuyer le lecteur expérimenté. Je demande votre indulgence. Rationnellement, je sais qu'il est très difficile pour un total débutant en traitement de texte informatique de connaître l'existence même de $\text{ConT}_{\text{E}}\text{Xt}$ mais, d'un autre côté, dans mon environnement professionnel, je suis entouré de personnes qui se battent continuellement en utilisant les logiciels de traitement de texte, et elles s'en sortent raisonnablement bien, mais néanmoins, n'ayant jamais travaillé avec des fichiers texte, elles ignorent des aspects aussi élémentaires que, par exemple, ce qu'est l'encodage ou la différence entre un éditeur et un traitement de texte.

¹ Je n'ai pas dessiné l'image moi-même, elle provient d'internet (<https://es.dreamstime.com/>), où il est indiqué qu'elle est libre de droit.

Le fait que ce manuel ait été conçu pour des personnes qui ne connaissent rien à ConT_EXt ou à T_EX implique que j'ai dû y inclure des informations concernant non pas à ConT_EXt mais à T_EX, le système de base mais j'ai compris qu'il n'était pas nécessaire de surcharger le lecteur avec des informations qui ne l'intéressent guère, par exemple de savoir si une telle commande qui fonctionne *de facto* provient de ConT_EXt ou bien de T_EX. Par conséquent, ce n'est qu'en certaines occasions, lorsque je l'ai considéré utile, qu'il est précisé que certaines commandes appartiennent réellement à T_EX.

Structuration En ce qui concerne l'organisation de ce document, le contenu est présenté en trois parties :

- **La première partie**, qui comprend les quatre premiers Chapitres, donne une vue d'ensemble de ConT_EXt, en expliquant ce que c'est, comment l'utiliser, en montrant un premier exemple de transformation d'un document, pour ensuite expliquer certains concepts fondamentaux de ConT_EXt ainsi que certaines questions relatives aux fichiers sources de ConT_EXt

Dans l'ensemble ces chapitres sont destinés aux lecteurs qui ne savaient travailler jusqu'à présent qu'avec des traitements de texte. Un lecteur qui connaît déjà l'utilisation des langages de balisage peut sauter les deux premiers chapitres. Si le lecteur connaît déjà T_EX ou L^AT_EX, il peut également sauter une grande partie du contenu des Chapitres 3 et 4. Mais je vous recommande quand même de lire au moins :

- les informations relatives aux commandes de ConT_EXt (Chapitre ??), et en particulier sur la configuration de son fonctionnement, car c'est là que réside la principale différence dans les conceptions et syntaxes de L^AT_EX et ConT_EXt. Comme cette introduction ne concerne que ce dernier, ces différences ne sont pas expressément mentionnées comme telles, mais quiconque lit ce chapitre et connaît le fonctionnement de L^AT_EX comprendra immédiatement les principales différences dans la syntaxe des deux langages, ainsi que la façon dont ConT_EXt permet de configurer et de personnaliser le fonctionnement de presque toutes ses commandes.
 - Les informations relatives à la gestion des projets ConT_EXt multi-fichiers (Chapitre ??), qui se distingue de celles des autres systèmes basés sur T_EX.
- **La seconde partie**, qui comprend les Chapitres 5 à 9, se concentre sur ce que nous pouvons considérer comme l'aspect général d'un document :

- Les deux aspects qui affectent principalement l'apparence d'un document sont les dimensions et la composition de ses pages ainsi que la police utilisée. Les chapitres ?? et ?? sont consacrés à ces deux questions.
 - ★ Le Chapitre ?? se concentre sur les pages : taille, éléments qui la composent, conception ou composition (c'est-à-dire répartition des différents éléments sur la page), etc. Pour une raison de systématisme, des aspects plus spécifiques sont également traités ici, comme ceux relatifs à la pagination et aux mécanismes qui nous permettent de l'influencer.
 - ★ Le Chapitre ?? explique les commandes relatives aux polices et à leur manipulation. Une explication de base de l'utilisation et de la manipulation des couleurs est également incluse ici, car, bien que les couleurs ne soient pas, à proprement parler, une *caractéristique* de la police, elles influencent de la même manière l'apparence visuelle du document.
- Les chapitres ?? et ?? se concentrent sur la structure du document et les outils que ConT_EXt met à la disposition de l'auteur pour la rédaction de documents bien structurés. Le Chapitre ?? se concentre sur la structure elle-même (divisions structurelles du document) et le Chapitre ?? sur la valorisation de cette structure dans une table des matières.
- Enfin, le Chapitre ?? se concentre sur l'aspect clé de l'utilisation de références par un document, références vers d'autres points du même document (références internes) mais aussi vers des documents externes (références externes). Dans cette dernière situation, nous n'aborderons que le cas de références impliquant un *liens* vers le document externe. Ce chapitre explique l'utilisation de l'outil de ConT_EXtpour la gestion de ces références qui permet des *sauts* entre différents zones d'informations, internes ou externes, et rend ainsi notre document *interactif*.

Ces chapitres n'ont pas besoin d'être lus dans un ordre particulier, sauf peut-être le Chapitre ??, qui est peut-être plus facile à comprendre si vous avez d'abord lu le Chapitre ??. En tout cas, j'ai essayé de faire en sorte que lorsqu'une question se pose dans un chapitre ou une section, alors qu'elle est traitée ailleurs dans ce document, le texte mentionne ce fait et propose un hyperlien vers le point où cette question est traitée. Je ne suis cependant pas en mesure de garantir que ce sera toujours le cas.

- Enfin, **la troisième partie** (Chapitre ?? et suivants) se concentre sur des aspects plus concrets, plus spécialisés. Non seulement les chapitres sont maintenant indépendants les uns des autres, mais également les sections les unes des

autres au sein d'un même chapitre (sauf, peut-être, dans le dernier chapitre). Étant donné la grande quantité de fonctionnalités proposées par ConT_EXt cette troisième partie pourrait être très vaste ; mais comme je devine qu'en arrivant à ce stade le lecteur sera capable de plonger lui-même dans la documentation de ConT_EXt je n'ai considéré que les chapitres suivants :

- Les chapitres ?? et ?? traitent de ce que l'on pourrait appeler les *éléments clés* de tout document texte : Le texte est constitué de caractères, qui forment des mots, qui sont regroupés en lignes, qui à leur tour forment des paragraphes, qui sont séparés les uns des autres par un espace vertical... Il est évident que toutes ces questions auraient pu être incluses dans un seul chapitre. Mais comme cela aurait été trop long, j'ai divisé cette question en deux chapitres, l'un traitant des caractères, des mots et des espaces horizontaux, et l'autre traitant des lignes, des paragraphes et des espaces verticaux.
 - Le Chapitre ?? est une sorte de *pot-pourri* qui traite des éléments et constructions qui sont communs à de nombreux documents, principalement s'ils sont académiques ou scientifico-techniques : notes de bas de page, listes structurées, descriptions, énumérations, etc.
 - Enfin, le Chapitre ?? se concentre sur les objets flottants insérés dans un document et en particulier les deux cas les plus répandus : les images et les tableaux.
- Cette introduction à ConT_EXt se termine par trois **Annexes**. L'une concerne l'installation de ConT_EXt sur un ordinateur, une deuxième annexe présentent plusieurs dizaines de commandes qui permettent de générer divers symboles (principalement, mais pas uniquement, pour un usage mathématique), et une troisième annexe rassemble les commandes ConT_EXt expliquées ou mentionnées tout au long de ce texte sous la forme d'une liste alphabétique.

De nombreux points restent à expliquer : le traitement des citations et des références bibliographiques, la rédaction de textes spéciaux (mathématiques, chimie...), la connexion avec XML, l'interface pour le code Lua, les modes et la compilation basée sur les modes, la collaboration avec MetaPost pour le design graphique, etc. Par conséquent, comme ce document ne contient pas d'explication complète de ConT_EXt et ne prétend pas le faire, j'ai intitulé ce document « une courte (?) introduction à ConT_EXt Mark IV) », et ajouté entre parenthèses l'observation « pas trop courte » car, de toute évidence, elle l'est. Un texte qui laisse tant de choses dans l'encrier et qui dépasse pourtant 300 pages n'est certainement pas une

« courte introduction ». C’est parce que j’essaie de faire comprendre au lecteur la logique de ConT_EXt ; ou du moins la logique telle que je la comprends. Il n’est pas destiné à être un manuel de référence, mais plutôt un guide d’auto-apprentissage qui prépare le lecteur à réaliser des documents de complexité moyenne (ce qui inclut la plupart des documents possibles) et qui, surtout, lui apprend à *envisager et imaginer* ce qui peut être fait avec ce puissant outil et à *repérer* dans la documentation comment le faire. Ce document n’est pas non plus un tutoriel. Les tutoriels sont conçus pour augmenter progressivement la difficulté, afin que ce qui doit être enseigné soit appris pas à pas ; j’ai, en ce sens, préféré, dès la deuxième partie, être plus systématique au lieu d’ordonner le sujet en fonction de sa difficulté. Mais, même si ce n’est pas un tutoriel, j’ai inclus de nombreux exemples.

Il est possible que le titre de ce document rappelle à certains lecteurs celui écrit par OETIKER, PARTL, HYNÄ ET SCHLEGL, en anglais « *The Not So Short Introduction to L^AT_EX 2_ε* » et en français « *Une courte (?) introduction à L^AT_EX 2_ε* ». Ce document, [disponible en 24 langues](#), est l’un des meilleurs documents pour se familiariser avec le monde de L^AT_EX. Ce n’est pas une coïncidence, mais un hommage et un acte de gratitude : grâce au travail généreux de ceux qui écrivent de tels textes, il est possible pour de nombreuses personnes de s’initier à des outils utiles et puissants comme L^AT_EX et ConT_EXt. Ces auteurs m’ont aidé à me lancer dans L^AT_EX ; j’ai l’intention de faire de même pour ceux qui veulent se lancer dans le ConT_EXt, bien que je sois limité au public hispanophone, qui, par ailleurs, manque tellement de documentation dans sa langue. J’espère que ce document répondra à cet objectif.

Joaquín Ataz-López
Eté 2020

I

**Qu'est ce que ConT_EXt ?
Comment travailler avec ?**

Chapitre 1

ConT_EXt : une vue d'ensemble

Table of Contents: 1.1 Qu'est-ce que ConT_EXt ?; 1.2 La composition typographique de document; 1.3 Les langages de balisage; 1.4 T_EX et ses dérivés; 1.4.1 Les moteurs T_EX; 1.4.2 Les formats dérivés de T_EX; 1.5 ConT_EXt; 1.5.1 Une brève histoire de ConT_EXt; 1.5.2 ConT_EXt versus L^AT_EX; 1.5.3 La logique de travail avec ConT_EXt; 1.5.4 Obtenir de l'aide sur ConT_EXt;

1.1 Qu'est-ce que ConT_EXt ?

ConT_EXt est un *système de composition*, c'est à dire un ensemble complet d'outils visant à donner à l'utilisateur un contrôle le plus complet précis possible sur la présentation et l'apparence d'un document électronique destiné à être imprimé sur papier ou affiché à l'écran. Ce chapitre expliquera ce que cela signifie. Mais d'abord, mettons en évidence certains des caractéristiques de ConT_EXt.

- Il existe deux versions de ConT_EXt appelées respectivement Mark II et Mark IV. ConT_EXt Mark II est *gelé*, c'est-à-dire qu'il est considéré comme finalisé, qu'aucun changement ou nouvelle fonctionnalité ne devrait y être introduit. Une nouvelle version ne sera publiée que si un bogue est détecté et doit être corrigé. ConT_EXt Mark IV, en revanche, est toujours en développement, de nouvelles versions intègrent périodiquement des améliorations et fonctionnalité supplémentaires. Mais, bien que toujours en cours de développement, c'est un langage très mature, dans lequel les changements introduits par les nouvelles versions sont très subtils et n'affectent que le fonctionnement de bas niveau du système. Pour l'utilisateur moyen ces changements sont totalement transparents, c'est comme s'ils n'existaient pas. Pour finir, bien que les deux versions aient beaucoup en commun, il y a aussi quelques incompatibilités entre elles. Cette introduction se concentre donc uniquement sur le ConT_EXt Mark IV.
- ConT_EXt est un logiciel libre. Le programme lui-même (c'est-à-dire l'ensemble des outils logiciels qui composent ConT_EXt) est distribué sous la *Licence Publique Générale GNU*. La documentation est fournie sous une licence *Creative Commons* qui vous permet de la copier et de la distribuer librement.

- ConT_EXt n'est pas un programme de traitement de texte ou d'édition de texte, mais un ensemble d'outils conçus pour *transformer* un texte que nous avons précédemment écrit avec notre éditeur de texte préféré. Par conséquent, lorsque nous travaillons avec ConT_EXt :
 - Nous commençons par écrire un ou plusieurs fichiers texte avec n'importe quel éditeur de texte.
 - Dans ces fichiers, en plus du texte qui constitue le contenu réel du document, il y a une série d'instructions, instructions qui indiquent à ConT_EXt à quoi doit ressembler le document final généré à partir des fichiers texte originaux. L'ensemble complet des instructions ConT_EXt constitue en fait un *langage* ; et puisque ce langage permet de *programmer* la transformation typographique d'un texte, on peut dire que ConT_EXt est un *langage de programmation typographique*.
 - Une fois les fichiers sources écrits, ils seront traités par un programme (également appelé « context »¹), qui, à partir de ceux-ci, générera un fichier PDF prêt à être envoyé à une imprimante ou à être affiché à l'écran.
- Dans ConT_EXt, nous devons donc faire la différence entre le document que nous rédigeons et le document que ConT_EXt génère. Pour éviter tout doute, dans cette introduction, j'appellerai *fichier source* le document texte qui contient les instructions de formatage, et *document final* le fichier PDF généré par ConT_EXt à partir du fichier source.

Dans la suite, les points fondamentaux ci-dessus seront développés un peu plus.

1.2 La composition typographique de document

Ecrire un document (livre, article, chapitre, brochure, dépliant, imprimé, affiche...) et le mettre en page, dit également le composer, sont deux activités très différentes.

¹ ConT_EXt désigne donc à la fois un langage et un programme (ainsi qu'un ensemble d'autres outils formant le système complet). Par conséquent, dans un texte comme cette introduction, se pose le problème de devoir parfois faire la distinction entre les deux aspects. J'ai donc adopté la convention typographique consistant à écrire «ConT_EXt» avec son logo (ConT_EXt) lorsque je veux me référer exclusivement à la langue, ou indistinctement à la langue et au programme. Et lorsque je veux me référer exclusivement au programme j'écrirai « context » tout en minuscules et avec une police de caractères à espacement constant, typique des terminaux d'ordinateur et des machines à écrire, que j'utiliserai aussi pour les exemples et pour faire référence aux instructions du langage.

L'écriture du document c'est sa rédaction, qui est effectuée par l'auteur, qui décide de son contenu et de sa structure. Le document créé directement par l'auteur, tel qu'il l'a écrit, s'appelle un *manuscrit*. Le manuscrit, par sa nature même, n'est accessible qu'à l'auteur et aux personnes à qui l'auteur permet de le lire. Sa diffusion au-delà de ce cercle intime nécessite que le manuscrit soit *publié*. De nos jours, publier quelque chose — au sens étymologique de «rendre accessible au public» — est aussi simple que de le mettre sur l'internet, à la disposition de quiconque peut le localiser et souhaite le lire. Mais jusqu'à une date relativement récente, l'édition était un processus qui impliquait des coûts, dépendait de certains professionnels spécialisés dans ce domaine et n'était accessible qu'aux manuscrits qui, en raison de leur contenu ou de leur auteur, étaient considérés comme particulièrement intéressants. Et aujourd'hui encore, nous avons tendance à réserver le mot *publication* au type de *publication professionnelle* par lequel le manuscrit subit une série de transformations de son apparence visant à améliorer la *lisibilité du document*. C'est cette série de transformations qui est appelée *composition* ou encore *composition typographique*.

L'objectif de la composition est — en général, et en laissant de côté les textes publicitaires qui cherchent à attirer l'attention du lecteur — de réaliser des documents avec une *lisibilité* maximale, entendue comme la qualité d'un texte imprimé qui invite à la lecture, ou qui la facilite, et qui fait que le lecteur s'y sente à l'aise. De nombreux aspects y contribuent ; certains, bien sûr, sont liés au *contenu* du document (qualité, clarté, standardisation...), mais d'autres dépendent de questions telles que le type et la taille de la police utilisée, la répartition des espaces blancs sur la page, la séparation visuelle entre les paragraphes, etc., on encore d'autres outils, moins graphiques ou visuels, telles que l'existence ou non dans le document de certaines aides à la lecture comme les en-têtes ou les pieds de page, les index, les glossaires, les caractères gras, les titres dans les marges, etc. On pourrait appeler «art de la composition» ou «art de l'impression» la connaissance et la manipulation correcte de toutes les ressources dont dispose un compositeur, un imprimeur.

Historiquement, et jusqu'à l'avènement des ordinateurs, les tâches et les rôles du rédacteur et du compositeur sont restés clairement différenciés. L'auteur écrivait à la main ou, depuis le milieu du XIX^e siècle, sur une «machine à écrire» dont les ressources typographiques étaient encore plus limitées que celles de l'écriture manuscrite ; puis il remettait ses originaux à l'éditeur ou à l'imprimeur qui se chargeait de les transformer pour en tirer le document imprimé.

De nos jours, grâce à l'informatique, il est plus facile pour l'auteur lui-même de définir la composition jusqu'aux moindres détails. Mais pour autant les qualités

d'un bon auteur ne sont pas les mêmes que celles d'un bon compositeur. L'auteur doit avoir une bonne connaissance du sujet traité, savoir le structurer, l'exposer avec clarté, avec créativité, avec un rythme. etc. Le compositeur typographe doit avoir une bonne connaissance de l'environnement graphique et conceptuel à sa disposition, un goût de l'esthétique pour les utiliser harmonieusement, de façon cohérente avec le sujet traité, avec les tendances du moment.

Avec un bon logiciel de traitement de texte ¹, il est possible d'obtenir une composition raisonnablement bonne. Mais les traitements de texte ne sont généralement pas conçus pour la composition et leurs résultats, même s'ils sont corrects, ne sont pas comparables à ceux obtenus avec d'autres outils spécifiquement conçus pour contrôler la composition des documents. En fait, les traitements de texte sont l'évolution des machines à écrire, et leur utilisation, dans la mesure où ces outils masquent la différence entre la rédaction du texte (la paternité) et sa composition, tend à produire des textes parfois moins structurés et moins bien optimisés typographiquement.

Au contraire, les outils tels que ConT_EXt sont l'évolution de l'imprimerie ; ils offrent beaucoup plus de possibilités de composition et, surtout, il n'est pas possible d'apprendre à les utiliser sans acquérir également, en cours de route, de nombreuses notions liées à la composition, contrairement aux traitements de texte, qui peuvent être utilisés pendant de nombreuses années sans apprendre un seul mot de typographie.

1.3 Les langages de balisage

Avant l'arrivée de l'informatique, comme je l'ai déjà dit, l'auteur préparait son manuscrit à la main ou à la machine à écrire et le remettait à l'éditeur ou à l'imprimeur, qui était chargé de transformer le manuscrit en texte final imprimé. Bien que l'auteur soit relativement peu intervenu dans cette transformation, il l'a fait dans une certaine mesure, par exemple en indiquant que certaines lignes du manuscrit étaient les titres de ses différentes parties (chapitres, sections...) ; ou en indiquant que certains fragments devaient être mis en valeur typographiquement d'une certaine manière. Ces indications étaient faites par l'auteur dans le manuscrit lui-même, parfois expressément, et d'autres fois au moyen de certaines conventions

¹ Par une convention assez ancienne, on fait une distinction entre les logiciels d'édition de texte et les logiciels de *traitement de texte*. Les premiers manipulent des fichiers de texte brut, et les seconds des fichiers de texte au format binaire permettant une plus grande complexité.

qui, avec le temps, se sont développées ; ainsi, par exemple, les chapitres commençaient toujours sur une nouvelle page, en insérant plusieurs lignes vierges avant le titre, en le soulignant, en l'écrivant en majuscules ; ou en encadrant le texte à mettre en valeur entre deux soulignements, en augmentant l'indentation d'un paragraphe, etc.

L'auteur, en somme, *indiquait* dans le texte original quelques éléments concernant la composition typographique du texte. L'éditeur ensuite inscrivait à son tour de nouvelles indication pour l'imprimeur, comme par exemple la police et la taille de caractères.

Aujourd'hui, dans un monde informatisé, nous pouvons continuer à faire de même pour la génération de documents électroniques, au moyen de ce que l'on appelle un *langage de balisage*. Dans ce type de langage, on utilise une série de marques ou d'indications ou encore de *balises* que le programme traitant le fichier qui les contient sait interpréter. Le langage de balisage le plus connu au monde aujourd'hui est sans doute le HTML, car la plupart des pages web sont basées sur ce langage. Un fichier HTML contient le texte d'une page web, ainsi qu'une série de marques qui indiquent au programme de navigation avec lequel la page est chargée, comment elle doit être affichée. L'ensemble des balises HTML compréhensibles par les navigateurs web, ainsi que les instructions sur la manière et l'endroit où les utiliser, est appelé «langage HTML», qui c'est un langage de balisage. Mais en plus du HTML, il existe de nombreux autres langages de balisage ; en fait, ceux-ci sont en plein essor et ainsi, le XML, qui est le langage de balisage par excellence, est aujourd'hui absolument omniprésent et est utilisé pour presque tout : pour la conception de bases de données, pour la création de langages spécifiques, pour la transmission de données structurées, pour les fichiers de configuration d'applications, et ainsi de suite. Il existe également des langages de balisage conçus pour la conception graphique (SVG, TikZ ou MetaPost), les formules mathématiques (MathML), la musique (Lilypond et MusicXML), la finance, la géographie, etc. Il y a aussi, bien sûr, ceux destinés à la transformation typographique des textes, et parmi eux se distinguent T_EX et ses dérivés.

En ce qui concerne les balises *typographiques*, qui indiquent l'apparence que doit avoir un texte, il en existe deux types, que nous pourrions distinguer comme d'un côté les *balises purement typographique* (ou encore graphiques) et de l'autre les *balises sémantiques* (ou encore conceptuelles, logiques). Les balises purement typographiques se limitent à indiquer précisément quelles ressources typographiques doivent être utilisées pour afficher un certain texte ; par exemple, lorsque nous indiquons qu'un certain texte doit être en gras ou en italique, de telle ou telle couleur. Le balisage sémantique, quant à lui, indique la fonction d'un texte donné

dans l'ensemble du document, par exemple lorsque nous indiquons qu'il s'agit d'un titre, d'un sous-titre, d'une citation. En général, les documents qui utilisent de préférence ce deuxième type de balisage sont plus cohérents et plus faciles à composer, car la différence entre la paternité et la composition y est à nouveau claire : l'auteur indique que cette ligne est un titre, ou que ce fragment est un avertissement, ou une citation ; et le compositeur décide comment mettre en valeur typographiquement tous les titres, avertissements ou citations ; ainsi, d'une part, la cohérence est garantie, puisque tous les fragments remplissant la même fonction auront la même apparence, et, d'autre part, on gagne du temps, puisque le format de chaque type de fragment ne doit être indiqué qu'une seule fois.

1.4 T_EX et ses dérivés

T_EX a été développé à la fin des années 1970 par DONALD E. KNUTH, professeur de théorie de la programmation à l'université de Stanford, qui l'a utilisé pour composer ses propres publications et ainsi que pour donner un exemple de *programmation littéraire*, une approche de la programmation où le code source du logiciel est systématique commenté et documenté. Avec T_EX, KNUTH a également développé un langage de programmation supplémentaire appelé METAFONT, pour la conception de caractères typographiques, avec lequel il a créé une police qu'il a nommée *Computer Modern*, qui, en plus des caractères habituels de toute police, comprenait également un ensemble complet de «glyphes»¹ pour l'écriture des mathématiques. Il a ajouté à tout cela quelques utilitaires supplémentaires et c'est ainsi qu'est né le système de composition appelé T_EX, qui, pour sa puissance, la qualité de ses résultats, sa flexibilité d'utilisation et ses vastes possibilités, est considéré comme l'un des meilleurs systèmes informatiques pour la composition de textes. Il a été pensé pour des textes dans lesquels il y avait beaucoup de mathématiques, mais on a vite vu que les possibilités du système le rendaient adapté à tous les types de textes.

En interne, il fonctionne comme la machine à écrire d'une presse à imprimer, car tout y est *boîte*. Les lettres sont contenues dans des boîtes, les blancs sont aussi des boîtes. Un mot est une boîte enfermant les boîtes de ses lettres. Une ligne est une boîte enfermant les boîtes de ses mots et des blancs entre ces mots. Un paragraphe est une boîte contenant l'ensemble des boîtes de ses lignes. Et ainsi de suite. Tout

¹ En typographie, un glyphe est une représentation graphique d'un caractère, de plusieurs caractères ou d'une partie d'un caractère et est l'équivalent actuel du type d'impression (la pièce mobile en bois ou en plomb qui portait la gravure de la lettre).

cela avec une précision extraordinaire apportée au traitement des mesures. Il suffit de penser que la plus petite unité que T_EX traite est 65,536 fois plus petite que le point typographique, avec lequel on mesure les caractères et les lignes, qui est généralement la plus petite unité traitée par la plupart des programmes de traitement de texte. Cette plus petite unité traitée par T_EX est d'environ 0,000005356 millimètre.

Le nom T_EX vient de la racine du mot grec τέχνη, écrit en lettres capitales (TÉXNH). Par conséquent, comme la dernière lettre du nom n'est pas un « X » latin, mais le « χ » grec, il faut prononcer « Tec ». Ce mot grec, quant à lui, signifiait à la fois « art » et « technique », c'est pourquoi K_NUTH l'a choisi comme nom pour son système. Le but de ce nom, écrit-il, « est de rappeler qu'il s'occupe principalement de manuscrits techniques de haute qualité. Elle met l'accent sur l'art et la technologie, tout comme le mot grec sous-jacent ». Par convention établie par Knuth, le nom de est à écrire :

- Dans des textes formatés typographiquement, comme le présent texte, en utilisant le logo que j'ai utilisé jusqu'à présent : Les trois lettres sont en majuscules, avec le « E » central légèrement décalé vers le bas pour faciliter un rapprochement entre le « T » et le « X » ; c'est-à-dire : « T_EX » . Pour rendre plus facile l'écriture d'un tel logo, Knuth a inclus dans une instruction qui l'inscrit dans le document final : TeXTeX.

Pour rendre plus facile l'écriture d'un tel logo, Knuth a inclus dans une instruction qui l'inscrit dans le document final : `\TeX`.

- Dans un texte non formaté (tel qu'un e-mail ou un fichier texte), le « T » et le « X » sont en majuscules, et le « e » du milieu est en minuscules ; par exemple : « TeX ».

Cette convention est suivie dans tous les dérivés de T_EX qui l'incluent dans leur propre nom, comme par exemple ConT_EXt, qui lorsqu'il est écrit en mode texte doit être écrit « ConTeXt ».

1.4.1 Les moteurs T_EX

Le programme T_EX est un logiciel libre : son code source est à la disposition du public et chacun peut l'utiliser ou le modifier à sa guise, à la seule condition que, si des modifications sont introduites, le résultat ne puisse être appelé « T_EX ». C'est la raison pour laquelle, au fil du temps, certaines adaptations du programme sont apparues, qui lui ont apporté différentes améliorations, et qui sont généralement appelées *moteurs T_EX* (engine en anglais). En dehors du programme original, les

principaux moteurs T_EX sont, par ordre chronologique d'apparition pdfT_EX, ϵ -T_EX, X_YT_EX et LuaT_EX. Chacun d'entre eux est censé intégrer les améliorations de son prédécesseurs. Ces améliorations, en revanche, jusqu'à l'apparition de LuaT_EX, n'ont pas affecté le langage lui-même, mais seulement les fichiers d'entrée, les fichiers de sortie, la gestion des polices et le fonctionnement de bas niveau des macros.

La question du choix du moteur T_EX à utiliser fait l'objet d'un débat animé dans l'univers T_EX. Je ne m'y attarderai pas ici, car ConT_EXt Mark IV ne fonctionne qu'avec LuaT_EX. En fait, dans le monde de ConT_EXt la discussion sur les moteurs devient une discussion sur l'utilisation de Mark II (qui fonctionne avec pdfT_EX et X_YT_EX) ou Mark IV (qui fonctionne avec LuaT_EX).

1.4.2 Les formats dérivés de T_EX

Le noyau, ou cœur, de T_EX contient seulement un ensemble d'environ 300 instructions, appelées *primitives*, qui conviennent aux opérations de composition et aux fonctions de programmation très basiques. Ces instructions sont pour la plupart de très *bas niveau*, ce qui, en terminologie informatique, signifie qu'elles se rapprochent des opérations élémentaires de l'ordinateur, dans un langage machine peu approprié aux êtres humains, du type « déplacer ce caractère de 0,000725 millimètre vers le haut ».

Pour cette raison, K_NUTH a rendu T_EX extensible, c'est-à-dire disposant d'un mécanisme permettant de définir des instructions de plus haut niveau, dans un langage plus facilement compréhensibles par les êtres humains. Ces instructions, qui au moment de l'exécution sont décomposées en instructions élémentaires, sont appelées *macros*. Par exemple, l'instruction T_EX qui imprime votre logo (`\TeX`), est décomposée lors de son exécution en :

```
T
\kern -.1667em
\lower .5ex
\hbox {E}
\kern -.125em
X
```

On comprend là qu'il est beaucoup plus facile pour un être humain de comprendre et mémoriser la simple commande « `\TeX` » dont l'exécution effectue l'ensemble des opérations typographiques nécessaires à l'impression du logo.

La différence entre les *macros* et les *primitives* n'est vraiment importante que du point de vue du développeur de T_EX. Du point de vue de l'utilisateur, ce sont toutes des

instructions ou, si vous préférez, des *commandes*. Knuth les appelait des *séquences de contrôle*.

Cette possibilité d'étendre le langage par le biais de *macros* est l'une des caractéristiques qui ont fait de T_EX un outil si puissant. En fait, K_NUTH lui-même a conçu environ 600 macros qui, avec les 300 primitives, constituent le format appelé « Plain T_EX ». Il est assez courant de confondre T_EX lui-même avec Plain T_EX et, en fait, presque tout ce qui est dit ou écrit sur T_EX, se réfère en fait à Plain T_EX. Les livres qui prétendent être sur T_EX (y compris le livre fondateur « *The T_EX Book* »), font en fait référence à Plain T_EX ; et ceux qui pensent qu'ils manipulent directement T_EX manipulent en fait Plain T_EX.

Plain T_EX est ce que l'on appelle dans la terminologie T_EX un *format*, constitué d'un vaste ensemble de macros, ainsi que de certaines règles syntaxiques sur la manière et la façon de les utiliser. En plus de Plain T_EX, d'autres formats ont été développés au fil du temps, notamment L^AT_EX un vaste ensemble de macros pour T_EX conçu en 1985 par LESLIE LAMPORT, qui est probablement le dérivé de T_EX le plus utilisé dans le monde universitaire, technologique et mathématique. ConT_EXt est (ou a commencé à être), de même que L^AT_EX un format dérivé de T_EX.

Normalement, ces *formats* sont accompagnés d'un programme qui charge dans la mémoire de l'ordinateur les macros qui les composent avant d'appeler « `tex` » (ou l'un des autres moteurs précédemment listés) pour traiter le fichier source. Mais bien que tous ces formats exécute finalement T_EX, comme chacun possède ses instructions et ses règles syntaxiques spécifiques, du point de vue de l'utilisateur, nous pouvons les considérer comme des *langages différents*. Ils sont tous inspirés de T_EX, mais différents de T_EX, et différents les uns des autres.

1.5 ConT_EXt

En fait, si ConT_EXt a commencé comme un *format* de T_EX, aujourd'hui il est beaucoup plus que cela. ConT_EXt comprend :

1. Un très large ensemble de macros de T_EX. Si Plain T_EX se compose d'environ 900 instructions, il en compte près de 3500 ; et si l'on ajoute les noms des différentes options que ces commandes prennent en charge, on parle d'un vocabulaire d'environ 4000 mots. Le vocabulaire est aussi large car la stratégie de ConT_EXt pour faciliter son apprentissage est d'inclure de nombreux synonymes des commandes et des options.

Ce qui est prévu, pour obtenir un certain effet, c'est de fournir à l'utilisateur l'ensemble des façons dont celui-ci pourrait chercher à appeler cet effet. Par exemple, pour obtenir simultanément un caractère gras (en anglais *bold*) et italique (en anglais *italic*), ConT_EXt propose trois instructions identiques en terme de résultat : `\bi`, `\italicbold` y `\bolditalic`.

2. Un autre ensemble assez complet de macros pour MetaPost, un langage de programmation graphique dérivé de METAFONT, qui, lui-même, est le langage de conception de caractères que K_NUTH a co-développé avec T_EX.
3. Plusieurs *scripts* développés en PERL (les plus anciens), RUBY (certains également anciens et d'autres moins) et LUA (les plus récents).
4. Une interface qui intègre T_EX, MetaPost, LUA et XML, permettant d'écrire et de traiter des documents dans n'importe lequel de ces langages, ou qui mélangent des éléments de certains d'entre eux.

Vous n'avez pas compris grand-chose à l'explication ci-dessus ? Ne vous inquiétez pas. J'ai utilisé beaucoup de jargon informatique et mentionné beaucoup de programmes et de langages. Mais il n'est pas nécessaire de savoir d'où viennent les différents composants pour les utiliser. L'important, à ce stade de l'apprentissage, est de garder à l'esprit qu'il intègre de nombreux outils d'origines diverses qui forment un *système de composition typographique*.

C'est en raison de cette intégration d'outils d'origines différentes que l'on caractérise ConT_EXt de « technologie hybride » dédié à la composition typographique de documents. C'est également ce qui fait de ConT_EXt un système extraordinairement avancé et puissant.

Mais bien que ConT_EXt soit bien plus qu'un ensemble de macros pour T_EX, ses fondamentaux restent basés sur T_EX, et donc ce document, qui se veut n'être qu'une *introduction*, se concentre sur cet aspect.

ConT_EXt en revanche est beaucoup plus moderne que T_EX. Lorsque T_EX a été conçu, l'informatique commençait à peine à émerger, et on était encore loin d'envisager ce que serait (ce qui allait devenir) l'Internet et le monde du multimédia. En ce sens, ConT_EXt intègre naturellement certains éléments qui ont toujours constitué une sorte de corps étranger, tels que l'inclusion de graphiques externes, le traitement des couleurs, les hyperliens dans les documents électroniques, l'hypothèse d'un format de papier adapté d'un affichage sur écran, etc.

1.5.1 Une brève histoire de ConT_EXt

ConT_EXt est né vers 1991. Il a été créé par HANS HAGEN et TON OTTEN au sein d'une société néerlandaise de conception et de composition de documents appelée « *Pragma Advanced Document Engineering* », souvent abrégée en Pragma ADE. Il s'agissait au départ d'un ensemble de macros T_EX en néerlandais, officieusement connu sous le nom de *Prigmatex*, et destiné aux employés non techniques de l'entreprise, qui devaient gérer les nombreux détails de la mise en page des documents à éditer, et qui n'étaient pas habitués à utiliser des langages de balisage et des interfaces dans une autre langue que le néerlandais.

La première version de ConT_EXt a donc été écrite en néerlandais. L'idée était de créer un nombre suffisant de macros avec une interface uniforme et cohérente. Vers 1994, le *paquet* était suffisamment stable pour qu'un manuel d'utilisation soit écrit en néerlandais, et en 1996, à l'initiative de HANS HAGEN, le nom « ConT_EXt » a été utilisé pour s'y référer. Ce nom est censé signifier « Texte avec T_EX » (en utilisant la préposition latine "con" qui a la même signification qu'en espagnol), mais il joue en même temps avec le terme « Contexte », qui en néerlandais (comme en anglais) s'écrit « context ». Derrière ce nom, il y a donc un triple jeu de mots entre « T_EX », « texte » et « contexte ».

Par conséquent, bien que ConT_EXt soit dérivé de T_EX (prononcé « Tec »), il ne devrait pas être prononcé « Conlect » afin de ne pas perdre ce jeu de mots.

L'interface a commencé à être traduite en anglais vers 2005, donnant lieu à la version connue sous le nom de ConT_EXt Mark II, où le « II » s'explique par le fait que dans l'esprit des développeurs, la version « I » est la version précédente en néerlandais, même si elle n'a jamais vraiment été appelée ainsi. Après la traduction de l'interface en anglais, le système a commencé à être utilisé en dehors des Pays-Bas, et l'interface a été traduite dans d'autres langues européennes comme le français, l'allemand, l'italien et le roumain. La documentation « officielle » de ConT_EXt est généralement écrite sur la version anglaise, et c'est donc sur cette version que nous travaillons dans ce document, même si l'auteur de ce document (c'est-à-dire moi) est plus à l'aise en espagnol qu'en anglais.

Dans sa version initiale, ConT_EXt Mark II fonctionnait avec le *moteur* T_EX pdfT_EX. Plus tard, lorsque le nouveau moteur X_YT_EX est apparu, ConT_EXt Mark II a été modifié pour en permettre l'utilisation, qui présentait de nombreux avantages par rapport à pdfT_EX. Des années plus tard encore, lorsque le moteur LuaT_EXa été développé, il a été décidé de reconfigurer le fonctionnement interne de ConT_EXt Mark II pour intégrer toutes les nouvelles possibilités offertes par ce dernier.

moteur. C'est ainsi qu'est né ConT_EXt Mark IV, qui a été présenté en 2007, immédiatement après la présentation de LuaT_EX. La décision d'adapter ConT_EXt à LuaT_EXa très probablement été influencée par le fait que deux des trois principaux développeurs de ConT_EXt, HANS HAGEN et TACO HOEKWATER, font également partie de l'équipe de développement de LuaT_EX. Par conséquent, ConT_EXt Mark IV et LuaT_EX sont nés simultanément et ont été développés à l'unisson. Il existe une synergie entre ConT_EXt et LuaT_EX et qui n'existe avec aucun autre dérivé de T_EX ; et je ne pense pas qu'aucun d'entre eux ne profite des possibilités de LuaT_EX comme ConT_EXt le fait.

Entre Mark II et Mark IV, il existe de nombreuses différences, bien que la plupart d'entre elles soient *internes*, c'est-à-dire qu'elles concernent le fonctionnement de la macro à un bas niveau, de sorte que du point de vue de l'utilisateur, la différence n'est pas perceptible : le nom et les paramètres de la macro sont les mêmes. Il existe cependant quelques différences qui affectent l'interface et vous obligent à faire les choses différemment selon la version avec laquelle vous travaillez. Ces différences sont relativement peu nombreuses, mais elles affectent des aspects très importants comme, par exemple, l'encodage du fichier d'entrée, ou la gestion des polices installées dans le système.



Cependant, il serait apprécié qu'il existe quelque part un document expliquant (ou listant) les différences significatives entre Mark II et Mark IV. Dans le wiki de ConT_EXt, par exemple, il existe parfois *deux syntaxes* (souvent identiques) pour chaque commande. Je suppose que l'une est la version Mark II et l'autre la version Mark IV ; et à deviner, je suppose également que la première version est la version Mark II. Mais en pratique rien n'est indiqué à ce sujet sur le wiki.

Le fait que, pour les utilisateur, les différences au niveau du langage soient relativement peu nombreuses, signifie que dans de nombreux cas, plutôt que de parler de deux versions, nous parlons de deux « saveurs » de ConT_EXt. Mais qu'on les appelle d'une manière ou d'une autre, le fait est qu'un document préparé pour Mark II peut ne pas être compatible d'une compilation avec Mark IV et vice versa ; et si le document mélange les deux versions, il est fort probable qu'il ne se compilera bien avec aucune d'entre elles ; ce qui signifie que l'auteur du fichier source doit commencer par décider s'il l'écrit pour Mark II ou Mark IV.

Si nous avons à travailler avec différentes versions de ConT_EXt, une bonne astuce pour facilement distinguer les versions des fichiers sources consiste à utiliser une extension différente dans le nom des fichiers. Ainsi, par exemple, mes fichiers écrits pour Mark II sont nommés « .mkii » et ceux écrits pour Mark IV sont nommés « .mkiv ». Il est vrai que ConT_EXt s'attend à ce que tous les fichiers sources aient l'extension « .tex », mais vous pouvez changer l'extension tant que lorsque vous invoquez un fichier, vous indiquez explicitement son extension, si elle n'est pas celle par défaut.

La distribution de ConT_EXt que vous installez à partir de leur wiki, « ConT_EXt Standalone », inclut les deux versions, et pour éviter toute confusion —je suppose— propose une commande distincte pour compiler avec chacune d'entre elles. Mark II compile avec la commande « texexec » et Mark IV avec la commande « context ».

En réalité, aussi bien « context » que « texexec » sont des *scripts* qui lancent, avec différentes options, « mtxrun » qui, à son tour, est un *script* Lua.

A ce jour, Mark II est gelé et Mark IV est toujours en cours de développement, ce qui signifie que les nouvelles versions de Mark II ne sont publiées que lorsque des bogues ou des erreurs sont détectés, tandis que les nouvelles versions de Mark IV sont publiées régulièrement ; parfois même deux ou trois par mois ; bien que dans la plupart des cas, ces « nouvelles versions » n'introduisent pas de changements notables dans le langage, et se limitent à améliorer d'une manière ou d'une autre l'implémentation de bas niveau d'une commande, ou à mettre à jour l'un des nombreux manuels qui sont inclus dans la distribution. Néanmoins, si nous avons installé la version de développement — qui est celle que je recommande et celle qui est installée par défaut avec « ConT_EXt Standalone » —, il est approprié de mettre à jour notre installation de temps en temps (voir l'annexe ?? concernant la mise à jour de la version installée de « ConT_EXt Standalone »).

LMTX et autres implémentations alternatives de Mark IV

Les développeurs de ConT_EXt sont soucieux de la qualité du logiciel et n'ont cessé de faire évoluer Mark IV ; de nouvelles versions sont testées et expérimentées. Celles-ci, en général, ne diffèrent de Mark IV que sur très peu de points, et ne présentent pas d'incompatibilité de compilation comme cela existe entre Mark IV et Mark II, ce qui traduit la maturité du langage du point de vue utilisateur.

Ainsi, quelques variantes de Mark IV ont été développées, appelées respectivement Mark VI, Mark IX et Mark XI. Je n'ai pu trouver qu'une petite référence à Mark VI dans le wiki de ConT_EXt où il est indiqué que sa seule différence avec Mark IV est la possibilité de définir des commandes en assignant aux paramètres non pas un nombre, comme c'est traditionnel dans T_EX, mais un nom, comme cela se fait habituellement dans presque tous les langages de programmation.

Plus important que ces petites variantes —je pense— est l'apparition dans l'univers de ConT_EXt d'une nouvelle version, appelée LMTX, nom qui est un acronyme pour luametaT_EX : un nouveau *moteur* de T_EX qui est une version simplifiée et optimisée de LuaT_EX, développé en vue d'économiser les ressources informatiques et d'offrir une solution T_EX aussi minimaliste que possible ; c'est-à-dire que LMTX nécessite moins de place sur disque dur, moins de mémoire et moins de puissance de traitement que ConT_EXt Mark IV.

LMTX a été présenté au printemps 2019 et l'on suppose qu'il n'impliquera aucune altération externe du langage Mark IV. Pour l'auteur du document, il n'y aura aucune différence dans la conception ; mais au moment de la compilation, vous pouvez choisir entre compiler avec LuaT_EX, ou compiler avec lua_{meta}T_EX. Une procédure pour attribuer un nom de commande différent à chacune des installations (section ??) est expliquée dans l'annexe ??, relative à l'installation de ConT_EXt.

1.5.2 ConT_EXt versus L^AT_EX

Comme le format dérivé de T_EX le plus populaire est L^AT_EX la comparaison entre celui-ci et ConT_EXt est inévitable.

Il s'agit bien sûr de langages différents mais, d'une certaine manière, liés par leur origine commune T_EX ; la parenté est donc similaire à celle qui existe entre, par exemple, l'espagnol et le français : des langues qui partagent une origine commune (le latin) qui utilise des syntaxes *similaires* et de nombreux mots se correspondent assez directement. Mais au-delà de cet air de famille, L^AT_EX et ConT_EXt diffèrent dans leur philosophie et leur mise en œuvre, puisque les objectifs initiaux de l'un et de l'autre sont, en quelque sorte, contradictoires.

L^AT_EX vise à faciliter l'utilisation de T_EX, en éloignant l'auteur des détails typographiques spécifiques pour l'inciter à se concentrer sur le contenu, et laisser les détails de la composition entre les mains de L^AT_EX. En d'autres termes, la simplification de l'utilisation de T_EX est obtenue au prix d'une limitation de son immense flexibilité, par la prédéfinition de nombreux formats de base et la réduction du nombre de choix typographiques que l'auteur doit déterminer.

A l'opposé de cette philosophie, ConT_EXt est né au sein d'une entreprise dédiée à la composition de documents. Par conséquent, loin d'essayer d'isoler l'auteur des détails de la composition, ce qu'il tente de faire, c'est de lui donner un contrôle absolu et complet sur ceux-ci. Pour ce faire, ConT_EXt fournit une interface homogène et cohérente qui reste beaucoup plus proche de l'esprit original T_EX que L^AT_EX.

Cette différence de philosophie et d'objectifs fondamentaux se traduit à son tour par une différence de mise en œuvre. Parce que L^AT_EX, qui tend à simplifier au maximum, n'a pas besoin d'utiliser toutes les ressources de T_EX. Son cœur est, d'une certaine manière, assez simple. Par conséquent, lorsque vous souhaitez étendre ses possibilités, vous devez construire un *paquet*. Cet ensemble de paquets associé à L^AT_EX est à la fois une force et une faiblesse : une force, car l'énorme popularité de L^AT_EX, ainsi que la générosité de ses utilisateurs, impliquent que pratiquement tous les besoins qui se présentent ont déjà été soulevés par quelqu'un, et qu'il existe un paquet qui y réponde ; mais aussi une faiblesse, car ces paquets

sont souvent incompatibles entre eux, et leur syntaxe n'est pas toujours homogène, ce qui signifie que l'utilisation de L^AT_EX exige une plongée continue dans les milliers de paquets existants pour trouver ceux dont nous avons besoin et les faire fonctionner ensemble.

Contrairement à la simplicité du noyau de L^AT_EX et son extensibilité par le biais de paquets, ConT_EXt est conçu pour intégrer et rendre accessibles toutes — ou presque toutes — les possibilités typographiques de T_EX, de sorte que sa conception est beaucoup plus monolithique, mais, en même temps, il est aussi plus modulaire : le noyau ConT_EXt permet de faire presque tout et il est garanti qu'il n'y aura pas d'incompatibilités entre les différentes commandes, il n'y a pas besoin de rechercher les extensions dont vous avez besoin (elles sont déjà présentes), et la syntaxe du langage est homogène entre les différents commande.

Il est vrai que ConT_EXt propose des *modules* d'extension dont on pourrait considérer qu'ils ont une fonction similaire à celle des paquets de L^AT_EX, mais la vérité est que la fonction des deux est très différente : les modules de ConT_EXt sont conçus exclusivement pour accueillir des fonctionnalités supplémentaires qui, parce qu'ils sont en phase expérimentale, n'ont pas encore été incorporés dans le noyau, ou pour permettre à des développeurs en dehors de l'équipe de développement de ConT_EXt de les proposer.

Je ne pense pas que l'une de ces deux *philosophies* puisse être considérée comme préférable à l'autre. La réponse dépend plutôt du profil de l'utilisateur et de ce qu'il souhaite. Si l'utilisateur ne veut pas s'occuper de questions typographiques, mais simplement produire des documents standardisés de très haute qualité, il serait probablement préférable pour lui d'opter pour un système comme L^AT_EX ; au contraire, l'utilisateur qui aime expérimenter, ou qui a besoin de contrôler chaque détail de ses documents, ou qui doit concevoir un design spécial pour un certain document, ferait probablement mieux d'utiliser un système comme ConT_EXt, où l'auteur dispose de tous les contrôle ; avec le risque, bien sûr, qu'il ne sache pas correctement l'utiliser.

1.5.3 La logique de travail avec ConT_EXt

Lorsque nous travaillons avec ConT_EXt, nous commençons toujours par écrire un fichier texte (que nous appellerons *fichier source*), dans lequel nous inclurons, en plus du contenu de notre document final à proprement parler, les instructions (en langage ConT_EXt) qui indiquent exactement comment nous voulons que le document soit composé : quel aspect général nous voulons donner à ses pages et paragraphes, quelles marges nous souhaitons appliquer à certains paragraphes

spéciaux, quelles types de police doit être utilisé, quels fragments souhaitons nous afficher avec une police différente, etc. Une fois que nous avons écrit le fichier source, depuis un terminal, nous exécuterons le programme « context », qui le traitera et, à partir de celui-ci, générera un fichier différent, dans lequel le contenu de notre document aura été formaté selon les instructions qui étaient incluses dans le fichier source. Ce nouveau fichier peut être envoyé à l'imprimante, affiché à l'écran, hébergé sur Internet ou distribué à nos contacts, amis, clients, professeurs, étudiants... bref, à tous ceux pour qui nous avons écrit le document.

C'est-à-dire que lorsqu'il travaille avec ConT_EXt, l'auteur agit sur un fichier dont l'apparence n'a rien à voir avec celle du document final : le fichier avec lequel l'auteur travaille directement est un fichier texte qui n'est pas formaté typographiquement. À cet égard, son fonctionnement est très différent de celui des programmes dits de *traitement de texte*, qui affichent l'aspect final du document édité au fur et à mesure de sa saisie. Pour ceux qui sont habitués aux *traitements de texte*, le fonctionnement de l'application peut sembler étrange au début, et il peut même falloir un certain temps pour s'y habituer. Cependant, une fois que vous vous y serez habitué, vous comprendrez que cette autre façon de travailler, faisant la différence entre le fichier de travail et le résultat final, est en fait un avantage pour de nombreuses raisons, parmi lesquelles je soulignerai, sans ordre particulier, les suivantes :

1. car les fichiers texte sont plus « légers » à manipuler que les fichiers binaires des traitements de texte et que leur édition nécessite moins de ressources informatiques ; ils sont moins sujets à la corruption et ne deviennent pas illibiles si la version du programme avec lequel ils ont été créés change. Ils sont également compatibles avec n'importe quel système d'exploitation et peuvent être édités avec de nombreux éditeurs de texte, de sorte que pour travailler avec eux, il n'est pas nécessaire de disposer d'un logiciel d'édition particulier : n'importe quel autre programme d'édition de texte fera l'affaire, et chaque système d'exploitation informatique propose un voire des programmes d'édition de texte.
2. car la différenciation entre le document de travail et le document final permet de distinguer ce qui est le contenu réel du document de ce qui sera son apparence, permettant à l'auteur de se concentrer sur le contenu dans la phase de création, et sur l'apparence dans la phase de composition.
3. car il vous permet de modifier très rapidement et très précisément l'apparence du document, puisque celle-ci est déterminée par des commandes facilement identifiables.

4. car cette facilité à changer l'apparence, d'autre part, permet de générer facilement plusieurs versions différentes à partir d'un seul contenu : par exemple une version optimisée pour l'impression sur papier, et une autre pour l'affichage sur écran, ajustée à la taille de celui-ci et, peut-être, incluant des hyperliens qui n'ont pas d'utilité dans un document imprimé sur papier.
5. car il est également facile d'éviter les erreurs typographiques courantes dans les traitements de texte comme, par exemple, l'extension de l'italique au-delà du dernier caractère à utiliser, les erreurs d'application de style..
6. car, puisque le fichier de travail ne sera pas distribué et qu'il est « pour nos yeux seulement », il est possible d'incorporer des annotations et des observations, des commentaires et des avertissements pour nous-mêmes, pour des révisions ou des versions futures, avec la tranquillité d'esprit de savoir qu'ils n'apparaîtront pas dans le fichier formaté qui sera distribué.
7. car la qualité que l'on peut obtenir en traitant simultanément l'ensemble du document est bien supérieure à celle que l'on peut obtenir avec un programme qui doit prendre des décisions typographiques à la volée, au fur et à mesure de la rédaction du document.
8. etcétera.

Tout cela signifie que, d'une part, lorsque l'on travaille avec ConT_EXt, une fois que l'on a pris le coup de main, on est plus efficace et productif, et que, d'autre part, la qualité typographique que l'on obtiendra est bien supérieure à celle que l'on obtiendrait avec les *logiciels de traitement de texte*. Et s'il est vrai que, en comparaison, ces derniers sont plus faciles à utiliser, en réalité ils ne le sont *pas beaucoup*. Car s'il est vrai que ConT_EXt se compose, comme je l'ai déjà dit, d'environ 3500 instructions, un utilisateur normal n'a pas à toutes les connaître. Pour faire ce que l'on fait habituellement avec les traitements de texte, il suffira de connaître les instructions qui permettent d'indiquer la structure du document, quelques instructions relatives aux ressources typographiques courantes, comme le gras ou l'italique, et, éventuellement, comment générer une liste, ou une note de bas de page. Au total, pas plus de 15 ou 20 instructions nous permettront de faire presque toutes les choses que l'on fait avec un traitement de texte. Le reste des instructions nous permet de faire différentes choses qui, normalement, sont très difficiles voire impossibles à faire avec un logiciel de traitement de texte. Ainsi, si l'apprentissage de ConT_EXt est plus difficile que celui d'un logiciel de traitement de texte, c'est parce que l'on peut faire beaucoup plus de choses avec.

1.5.4 Obtenir de l'aide sur ConT_EXt

Tant que nous sommes des débutants, le meilleur endroit pour trouver de l'aide sur ConT_EXt est sans aucun doute son [wiki](#), qui regorge d'exemples et dispose d'un bon moteur de recherche, même s'il nécessite bien sûr de bien comprendre l'anglais. Nous pouvons aussi chercher de l'aide sur Internet, mais ici le jeu de mots sur lequel repose ConT_EXt nous jouera un sale tour car une recherche d'informations sur « contexte » renverrait des millions de résultats et la plupart d'entre eux n'auraient aucun rapport avec ce que nous recherchons. Pour rechercher des informations sur ConT_EXt, vous devez ajouter quelque chose au nom « context » ; par exemple, « tex », « luatex », « Mark IV » , « Hans Hagen » (un des créateurs de ConT_EXt), « Pragma ADE », ou quelque chose de similaire (par exemple une autre commande souvent utilisée dans le cas de figure qui vous préoccupe). Il peut également être utile de rechercher des informations par le nom wiki : « context-garden ».

Après en avoir appris un peu plus sur ConT_EXt, et si l'on maîtrise bien l'anglais, on peut consulter l'un des nombreux documents inclus dans « ConT_EXt Standalone » ou demander de l'aide :

- soit sur [TeX – LaTeX Stack Exchange](#) et en particulier [les questions tagguées « ConT_EXt »](#)
- soit sur la liste de diffusion propre à ConT_EXt [NTG-context](#) et son [moteur de recherche](#).

Cette dernière liste diffusion implique les personnes les plus compétents sur ConT_EXt, mais les règles d'une bonne éducation de « cybercitoyen » exigent qu'avant de poser une question, on ait essayé par tous les moyens de trouver la réponse par soi-même dans les documentations déjà existantes.

Chapitre 2

Notre premier fichier source

Table of Contents: 2.1 Préparation de l'expérience outils nécessaires; 2.2 L'expérience elle-même; 2.3 La structure de notre fichier d'exemple; 2.4 Quelques détails supplémentaires sur la façon d'exécuter « context »; 2.5 Traitement des erreurs;

Ce chapitre est consacré à la mise en oeuvre de notre première expérience. Il expliquera la structure de base d'un document ConT_EXt ainsi que les meilleures stratégies pour faire face aux éventuelles erreurs.

2.1 Préparation de l'expérience outils nécessaires

Pour écrire et compiler un premier fichier source, nous devons avoir les outils suivants installés sur notre système.

1. **un éditeur de texte** pour écrire notre fichier de test. Il existe de nombreux éditeurs de texte et il est difficilement concevable qu'un système informatique n'en ait pas déjà un d'installé. Nous pouvons utiliser n'importe lequel d'entre eux : il existe des systèmes simples, d'autres complexes, des puissants, d'autres simples, des payants, des gratuits, des spécialisés pour T_EX, des généralistes, etc. Si nous avons l'habitude d'utiliser un éditeur spécifique, il est préférable de poursuivre avec lui ; si nous n'avons pas l'habitude de travailler avec des éditeurs de texte, mon conseil est, dans un premier temps, de choisir un éditeur simple, afin de ne pas ajouter à la difficulté de l'apprentissage de ConT_EXt la difficulté d'apprendre à utiliser l'éditeur. Bien qu'il soit également vrai que, souvent, les programmes les plus difficiles à maîtriser sont aussi les plus puissants.

J'ai écrit ce texte avec GNU Emacs, qui est l'un des éditeurs généralistes les plus puissants et les plus polyvalents qui existent ; il est vrai qu'il a ses particularités et aussi ses détracteurs, mais en général il y a plus de « *Emacs Lovers* » que de « *Emacs Haters* ». Pour travailler avec des fichiers T_EX ou l'un de ses dérivés, il

existe une extension pour GNU Emacs, appelée AucTeX, qui fournit à l'éditeur quelques fonctionnalités supplémentaires très intéressantes, même si AucTeX est plus développé pour L^AT_EX que pour ConT_EXt. GNU Emacs en combinaison avec AucTeX peut être une bonne option si l'on ne sait pas quel éditeur choisir ; tous deux sont des programmes à code source ouvert, et ils sont disponibles pour tous les systèmes d'exploitation. En fait, dire que GNU Emacs est un *logiciel libre* est un euphémisme, car ce programme incarne mieux que tout autre l'esprit de ce qu'est et signifie le *logiciel libre*. Après tout, son principal développeur était RICHARD STALLMAN, fondateur et idéologue du projet GNU et de la *Free Software Foundation*.

En plus de GNU Emacs + AucTeX, *Scite* et *TexWorks* sont d'autres bonnes options si vous ne savez pas quel éditeur choisir. Le premier, bien qu'il s'agisse d'un éditeur à usage généraliste, non conçu spécifiquement pour travailler avec des fichiers ConT_EXt, est facilement personnalisable et, comme c'est l'éditeur généralement utilisé par les développeurs de ConT_EXt « ConT_EXt Standalone » contient les fichiers de configuration de cet éditeur conçus et utilisés par HANS HAGEN lui-même. *TexWorks* est, quant à lui, un éditeur de texte rapide, spécialisé dans le traitement des fichiers T_EX et de ses langages dérivés. Il est assez facile à configurer pour fonctionner avec ConT_EXt et « ConT_EXt Standalone » prévoit également de fournir des fichiers configurations.

Qu'il s'agisse d'un éditeur ou d'un autre, ce qu'il ne faut pas faire, c'est utiliser, comme éditeur de texte, un *logiciel de traitement de texte* tel que, par exemple, OpenOffice Writer ou Microsoft Word. Ces programmes, qui sont à mon avis trop lents et trop lourds, peuvent certes enregistrer un fichier en « texte pur », mais ils ne sont pas conçus pour cela et nous finirions probablement par enregistrer notre fichier dans un format binaire incompatible avec ConT_EXt.

2. **Une distribution ConT_EXt** pour traiter notre fichier de test. S'il existe déjà une installation T_EX (ou L^AT_EX) sur votre système, il est possible qu'une version de ConT_EXt soit déjà installée. Pour le vérifier, il suffit d'ouvrir un terminal et de taper dans celui-ci

```
1 $ context --version
```

NOTA ceux pour qui l'utilisation du terminal est nouvelle, les deux premiers caractères que j'ai indiqué (« \$ ») n'ont pas à être tapés dans le terminal par l'utilisateur. Je les utilise pour représenter ce qu'on appelle l'*invite* du terminal (le prompt en anglais), qui indique que le terminal attend nos instructions.

Si une version de ConT_EXt est déjà installée, vous devriez obtenir un résultat similaire au suivant :

```
1 $ context --version
2 mtx-context      | ConTEXt Process Management 1.03
3 mtx-context      |
4 mtx-context      | main context file: /usr/share/texmf/tex/context/base/mkiv/context.mkiv
5 mtx-context      | current version: 2020.03.10 14:44
6 mtx-context      | main context file: /usr/share/texmf/tex/context/base/mkiv/context.mxl
7 mtx-context      | current version: 2020.03.10 14:44
```

Dans la dernière ligne, nous sommes informés de la date à laquelle la version installée a été publiée. Si elle est très ancienne, nous devons le mettre à jour ou installer une nouvelle version. Je recommande d'installer la distribution appelée « ConT_EXt Standalone » dont les instructions d'installation se trouvent sur le [wiki de ConT_EXt](#). Les indications sont également incluses dans l'annexe ??.

3. **Un programme de visualisation de fichier PDF** afin de visualiser le résultat de notre expérience à l'écran. Sur les systèmes Windows et Mac OS, la visionneuse omniprésente est Adobe Acrobat Reader. Il n'est pas installé par défaut (ou ne l'était pas lorsque j'ai cessé d'utiliser Microsoft Windows, il y a plus de 15 ans). L'installation se fait la première fois que vous essayez d'ouvrir un fichier PDF, il est donc généralement déjà installé. Sur les systèmes Linux/Unix, il n'y a pas de version mise à jour d'Acrobat Reader, mais il n'est pas nécessaire non plus, car il existe littéralement des dizaines de très bons visualisateurs de PDF gratuits. De plus, dans ces systèmes, il y en a presque toujours un installé par défaut. Mon préféré, pour sa vitesse et sa facilité d'utilisation, est MuPDF ; bien qu'il ait quelques inconvénients comme, par exemple, de ne pas montrer l'index des signets, de ne pas permettre les recherches de texte qui incluent des caractères inexistantes dans l'alphabet anglais (comme les voyelles accentuées ou les eñes) ou de ne pas permettre de sélectionner le texte, d'envoyer le document à l'imprimante ; c'est juste un visualisateur, mais très rapide et très confortable. Lorsque j'ai besoin de certains de ces fonctionnalités absentes de MuPDF, j'utilise généralement Okular ou qPdfView. Mais, encore une fois, la question est une affaire de goût : chacun peut choisir celui qu'il préfère.

Nous pouvons choisir l'éditeur, nous pouvons choisir le visualisateur de PDF, nous pouvons choisir la distribution ConT_EXt... Bienvenue dans le monde du *logiciel libre* !

2.2 L'expérience elle-même

Rédaction du fichier source

Si nous disposons déjà des outils mentionnés dans la section précédente, nous devons ouvrir notre éditeur de texte et créer un fichier appelé « la-maison-sur-le-port.tex » pour notre exemple. Comme contenu du fichier, nous allons écrire ce qui suit :

Durant l'écriture, certains aspects n'ont aucune importance, notamment si vous ajoutez ou supprimez des espaces blancs ou des sauts de ligne. Ce qui est important, c'est que chaque mot suivant le caractère « \ » soit écrit très exactement de la même façon qu'il l'est dans l'exemple, ainsi que le contenu des crochets. Il peut y avoir des variations dans le reste.

Encodage du fichier

Une fois le texte précédent écrit, nous enregistrons le fichier sur le disque. Cela n'est dorénavant qu'une vérification à faire, mais il faut nous assurer que l'encodage du fichier est bien UTF-8. Cet encodage est aujourd'hui la norme et constitue l'encodage par défaut sur la plupart des systèmes Linux/Unix. Néanmoins, je ne sais pas si c'est la même chose sous Mac OS ou Windows et il est encore tout à fait possible que l'encodage ANSI soit utilisé. En tout cas, si nous ne sommes pas sûrs, depuis l'éditeur de texte lui-même, nous pouvons voir avec quel encodage le fichier sera enregistré et, si nécessaire, le modifier. La manière de procéder dépend, bien entendu, de l'éditeur avec lequel nous travaillons. Dans GNU Emacs, par exemple, en appuyant simultanément sur les touches CTRL-X puis Return suivi de « f », dans la dernière ligne de la fenêtre (que GNU Emacs appelle mini-buffer) un message apparaîtra nous demandant un nouvel encodage et nous informant de l'encodage actuel. Dans les autres éditeurs, nous pouvons généralement accéder à l'encodage dans le menu « Enregistrer sous ».

Après avoir vérifié que l'encodage est correct et enregistré le fichier sur le disque, nous fermerons l'éditeur pour nous concentrer sur l'analyse de ce que nous avons écrit.

Regardons le contenu de notre premier fichier source pour ConTeXt

La première ligne commence par le caractère « % ». C'est un caractère réservé qui indique à ConTeXt de ne pas traiter le texte qui le suit et ce jusqu'à la fin de la ligne sur laquelle il se trouve. Cette fonctionnalité est utilisée pour écrire des commentaires dans le fichier source que seul l'auteur pourra lire, car ils ne seront pas incorporés au document final. Dans cet exemple, je l'ai par exemple utilisé pour attirer l'attention sur certaines lignes, en expliquant ce qu'elles font.

Les lignes suivantes commencent par le caractère « \ » qui est un autre des caractères réservés de ConTeXt et indique que ce qui suit est le nom d'une commande. L'exemple comprend plusieurs commandes couramment utilisée dans un fichier source ConTeXt : la langue dans laquelle le document est écrit, le format du papier, la police à utiliser dans le document et la mise en forme appliquée aux titres de chapitres. Plus tard, dans d'autres chapitres, nous détailleront ces commandes, pour le moment je veux juste que le lecteur voit à quoi elles ressemblent : elles commencent toujours par le caractère « \ », suivi du nom de la commande, et ensuite, entre parenthèses ou accolades, selon le cas, les données dont la commande a besoin pour produire ses effets. Entre le nom de la commande et les crochets ou accolades qui l'accompagnent, il peut y avoir des espaces vides ou des sauts de ligne. C'est à l'auteur de choisir la façon dont le code source est le plus clair et lisible.

Sur la 9^{ème} ligne de notre exemple (je ne compte que les lignes qui ont du texte) se trouve la commande importante `\starttext` : elle indique à ConTeXt que le contenu du document commence à partir de cet endroit; et à la dernière ligne de notre exemple, nous voyons la commande `\stoptext` qui indique la fin du contenu. Tout ce qui suit cette dernière commande ne sera pas traité. Ce sont deux commandes très importantes sur lesquelles je reviendrai très bientôt. Entre les deux se trouve donc le contenu à proprement parler de notre document qui, dans notre exemple, consiste en la première strophe de la chanson "« La maison sur le port », dont les paroles sont de AMALIA RODRIGUES, et qui a été reprise notamment par SANSEVERINO. Je l'ai écrit en prose afin de mieux observer le formatage des paragraphes effectué par ConTeXt.

Traitement du document sourcei

Pour l'étape suivante, après s'être assuré que ConTeXt a été correctement installé dans notre système, nous devons ouvrir un terminal dans le répertoire où se trouve notre fichier « la-maison-sur-le-port.tex ».

De nombreux éditeurs de texte vous permettent de compiler le document sur lequel vous travaillez sans ouvrir un terminal. Cependant, la procédure *canonique* pour traiter un document avec ConT_EXt implique de le faire à partir d'un terminal, en exécutant directement le programme. Je vais procéder de cette manière (ou supposer qu'il en est ainsi) tout au long de ce document pour plusieurs raisons ; la première est que je n'ai aucun moyen de savoir avec quel éditeur chaque lecteur travaille. Mais le plus important est que, depuis le terminal, nous aurons accès à la *sortie* de « context », c'est à dire les messages émis par le programme.

Si la distribution ConT_EXt que nous avons installée est « ConT_EXt Standalone », nous devons tout d'abord exécuter le *script* qui indique au terminal les chemins et l'emplacement des fichiers dont ConT_EXt a besoin pour fonctionner. Sur les systèmes Linux/Unix, cela se fait en tapant la commande suivante :

```
1 $ source ~/context/tex/setuptex
```

en supposant que nous avons installé ConT_EXt dans un répertoire appelé « context ».

En ce qui concerne l'exécution du *script* qui vient d'être mentionné, voir ce qui est dit dans l'annexe ?? concernant l'installation de « ConT_EXt Standalone ».

Une fois que les variables nécessaires à l'exécution de « context » ont été chargées en mémoire, nous pouvons l'exécuter. Cela se fait en tapant dans le terminal :

```
1 $ context la-maison-sur-le-port
```

Notez que bien que le fichier source s'appelle « la-maison-sur-le-port.tex » dans l'appel à « context » nous avons omis l'extension du fichier. Si nous avons appelé au fichier source, par exemple, « la-maison-sur-le-port.mkiv » (ce que je fais habituellement pour savoir que ce fichier est écrit pour Mark IV), il aurait fallu indiquer expressément l'extension du fichier à compiler en tapant « context la-maison-sur-le-port.mkiv ».

Après avoir exécuté « context » dans le terminal, plusieurs dizaines de lignes s'affichent à l'écran, informant de ce que fait ConT_EXt. Les informations s'affichent à une vitesse impossible à suivre par un être humain, mais ne vous inquiétez pas, car en plus de l'écran, ces informations sont également stockées dans un fichier auxiliaire, avec l'extension « .log » qui est généré avec la compilation et que nous pourrions consulter tranquillement plus tard si nécessaire.

Après quelques secondes, si nous avons bien écrit le texte de notre fichier source, sans faire d'erreur grave, l'émission de messages vers le terminal se terminera. Le

dernier des messages nous informera du temps nécessaire à la compilation. La première fois qu'un document est compilé, cela prend toujours un peu plus de temps, car ConT_EXt doit construire à partir de zéro certains fichiers contenant les informations de notre document. Par la suite ils seront juste réutilisés et complétés pour les compilations suivantes qui iront plus vite. Le message du temps passé indique que la compilation est terminée. Si tout s'est bien passé, trois fichiers supplémentaires apparaîtront dans le répertoire où nous avons exécuté « context » :

- la-maison-sur-le-port.pdf
- la-maison-sur-le-port.log
- la-maison-sur-le-port.tuc

Le premier est le résultat de notre traitement, c'est-à-dire : le fichier PDF déjà formaté. Le deuxième est le fichier dans lequel sont stockées toutes les informations qui ont été affichées à l'écran pendant la compilation ; le troisième est un fichier auxiliaire que ConT_EXt génère pendant la compilation et qui est utilisé pour construire les index et les références croisées. Pour le moment, si tout a fonctionné comme prévu, nous pouvons supprimer les deux fichiers (« la-maison-sur-le-port.log » et « la-maison-sur-le-port.tuc »). S'il y a eu un problème, les informations contenues dans ces fichiers peuvent nous aider à localiser la source du problème et à déterminer comment le résoudre.

Si nous n'avons pas obtenu ces résultats, c'est probablement dû à un ou plusieurs de points qui suivent :

- soit nous n'avons pas installé correctement notre distribution ConT_EXt, auquel cas en tapant la commande « context » dans le terminal, un message « commande inconnue » sera apparu.
- soit notre fichier n'a pas été encodé en UTF-8 et cela a généré une erreur de compilation.
- soit peut-être que la version de ConT_EXt installée sur notre système est Mark II. Dans cette version, vous ne pouvez pas utiliser l'encodage UTF-8 sans l'indiquer explicitement dans le fichier source lui-même. Nous pourrions corriger le fichier source pour qu'il compile bien, mais, puisque cette introduction se réfère à Mark IV, cela n'a pas de sens de continuer à travailler avec Mark II : la meilleure chose à faire est d'installer « ConT_EXt Standalone ».
- Soit nous avons fait une erreur en écrivant dans le fichier source le nom de certaines commandes ou leurs données associées.

Si après l'exécution de « context », le terminal a commencé à émettre des messages, mais s'est ensuite arrêté sans que le *prompt* ne réapparaisse, avant de continuer, il faut

appuyer sur CTRL-X pour interrompre l'exécution de ConT_EXt qui a été interrompue par l'erreur.

En cas de problème, nous devons donc vérifier chacune de ces possibilités, et les corriger, jusqu'à ce que la compilation se déroule correctement.

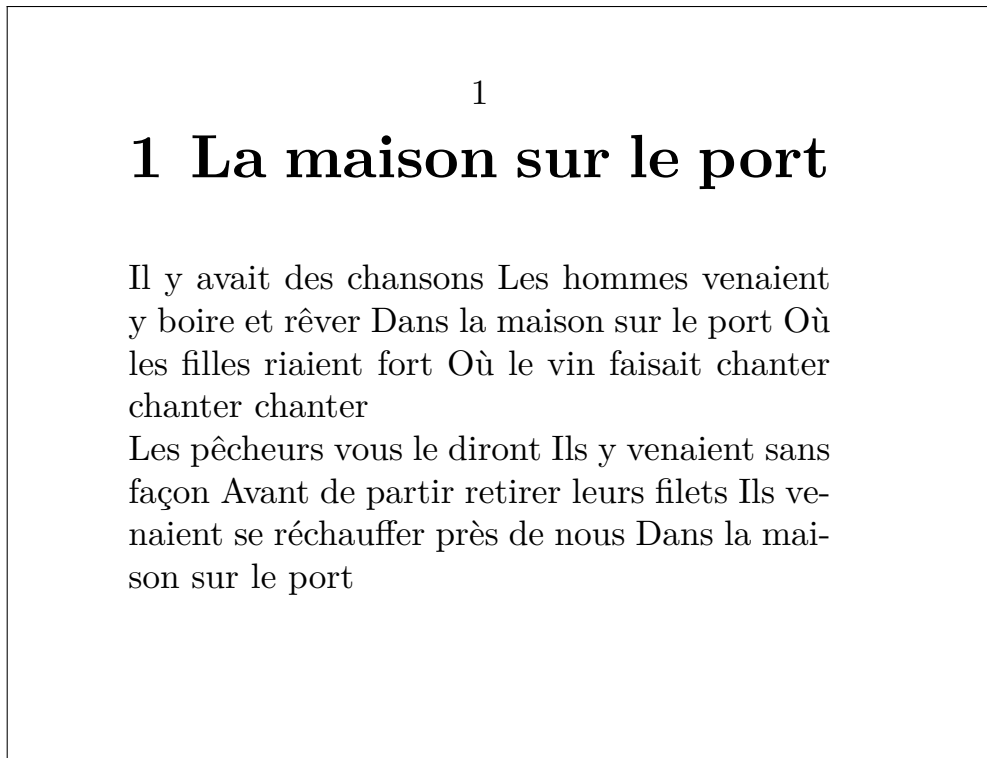


Figure 2.1 La maison sur le port

La [figure 2.1](#) montre le contenu de « la-maison-sur-le-port.pdf ». Nous pouvons voir que ConT_EXt a numéroté la page, numéroté le chapitre et écrit le texte dans la police indiquée. Il a également réparti le mot « venaient » entre la sixième et la septième ligne, ainsi que le mot « maison » la septième et la huitième ligne. ConT_EXt, par défaut, active la césure (division syllabique) des mots afin de répartir les blancs (les espaces vides entre les mots) de façon la plus homogène possible. C'est pourquoi il est si important d'informer ConT_EXt de la langue du document, car les modèles de césure varient selon la langue. Dans notre exemple, c'est l'objectif de la première commande du fichier source (`\mainlanguage[fr]`).

En bref : ConT_EXt a transformé le fichier source et a généré un fichier dans lequel nous avons un document formaté selon les instructions qui étaient incluses dans le fichier source. Les commentaires en ont disparu et, en ce qui concerne les commandes, ce que nous avons maintenant n'est pas leur nom, mais le résultat de leur application par ConT_EXt.

2.3 La structure de notre fichier d'exemple

Dans un projet aussi simple que notre exemple, développé dans un seul fichier source, la structure de celui-ci est très simple et est marquée par les commandes `\starttext` ... `\stoptext`. Tout ce qui se trouve entre la première ligne du fichier et la commande `\starttext` constitue le *préambule*. Le contenu du document lui-même est inséré entre les commandes `\starttext` et `\stoptext`. Dans notre exemple, le préambule comprend quatre commandes de configuration globale : une pour indiquer la langue de notre document (`\mainlanguage`), une autre pour indiquer la taille des pages (`\setuppapersize`) qui dans notre cas est « S5 », représentant les proportions d'un écran d'ordinateur, une troisième commande (`\setupbodyfont`) qui nous permet d'indiquer la police de caractère et sa taille, et une quatrième (`\setuphead`) qui nous permet de configurer l'apparence des titres des chapitres.

Le corps du document est encadré par les commandes `\starttext` et `\stoptext`. Ces commandes indiquent, respectivement, le point de départ et le point final du texte à traiter : entre elles, nous devons inclure tout le texte que nous voulons que ConTeXt traite, ainsi que les commandes qui ne doivent pas affecter le document entier mais seulement des fragments de celui-ci. Pour le moment, nous devons supposer que les commandes `\starttext` et `\stoptext` sont obligatoires dans tout document ConTeXt, bien que plus tard, en parlant des projets multifichiers (section ??) nous verrons qu'il y a quelques exceptions.

2.4 Quelques détails supplémentaires sur la façon d'exécuter « context »

La commande « context » avec laquelle nous avons procédé au traitement de notre premier fichier source est, en fait, un *script* LUA, c'est-à-dire : un petit programme LUA qui, après avoir effectué quelques vérifications et opérations, appelle LuaTeX pour traiter le fichier source.

Nous pouvons appeler « context » avec plusieurs options. Les options sont saisies immédiatement après le nom de la commande, précédées de deux traits d'union. Si nous voulons saisir plus d'une option, nous les séparons par un espace blanc. L'option « help » nous donne une liste de toutes les options, avec une brève explication de chacune d'elles :


```
1 $ context --help
```

Parmi les options les plus intéressantes, citons les suivantes :

Interface Comme je l’ai dit dans le chapitre d’introduction, l’interface de ConT_EXt est traduite en plusieurs langues. Par défaut, c’est l’interface anglaise qui est utilisée, mais cette option nous permet de lui demander d’utiliser la version néerlandaise (nl), française (fr), italienne (it), allemande (de) ou roumaine (ro).

purge, purgeall Supprime les fichiers auxiliaires générés pendant le traitement.

result=Name indique le nom que doit porter le fichier PDF résultant. Par défaut, ce sera le même que le fichier source à traiter, avec l’extension « .pdf ».

usemodule=list Charge les modules qui sont indiqués avant d’exécuter ConT_EXt (un module est une extension de ConT_EXt, qui ne fait pas partie de son noyau, et qui lui fournit une utilité supplémentaire).

useenvironment=list Charge les fichiers d’environnement qui sont spécifiés avant de lancer ConT_EXt (un fichier d’environnement est un fichier contenant des instructions de configuration).

version indique la version de ConT_EXt.

help affiche des informations d’aide sur les options du programme.

noconsole Supprime l’envoi de messages à l’écran pendant la compilation. Toutefois, ces messages seront toujours enregistrés dans le fichier « .log ».

nonstopmode Exécute la compilation sans s’arrêter sur les erreurs. Cela ne signifie pas que l’erreur ne se produira pas, mais que lorsque ConT_EXt rencontre une erreur, même si elle est récupérable, il continuera la compilation jusqu’à ce qu’elle se termine ou jusqu’à ce qu’il rencontre une erreur irrécupérable.

batchmode Il s’agit d’une combinaison des deux options précédentes. Il fonctionne sans interruption et omet les messages à l’écran.

Pour les premières utilisation et pour l’apprentissage de ConT_EXt, je ne pense pas que ce soit une bonne idée d’utiliser les trois dernières options, car lorsqu’une erreur se produit, nous n’aurons aucune idée de l’endroit où elle se trouve ou

de ce qui l’a produite. Et, croyez-moi chers lecteurs, tôt ou tard, une erreur de compilation se produira.

2.5 Traitement des erreurs

En travaillant avec ConT_EXt, il est inévitable que, tôt ou tard, des erreurs se produisent dans la compilation. En gros, nous pouvons regrouper les erreurs dans l’une des quatre catégories suivantes :

1. **Erreurs de frappe.** Elles se produisent lorsque nous orthographions mal le nom d’une commande. Dans ce cas, nous envoyons au compilateur une commande qu’il ne comprend pas. Par exemple, lorsque, au lieu d’écrire la commande `\TeX`, nous écrivons `\Tex` avec le « X » final en minuscule, puisque ConT_EXt fait la différence entre les majuscules et les minuscules et considère donc que « TeX » et « Tex » sont des mots différents ; ou si les options utilisées pour une commande, au lieu de les mettre entre crochets, sont mises entre accolades, ou si nous essayons d’utiliser un des caractères réservés comme s’il s’agissait d’un caractère normal, etc.
2. **Erreurs par omission.** Dans ConT_EXt il y a des instructions qui démarrent une tâche, dont il faut indiquer explicitement quand la fermer ; comme le caractère réservé « \$ » qui active le mode mathématique, qui est maintenu jusqu’à ce qu’on le désactive, et si on oublie de le désactiver, une erreur sera générée dès qu’on trouvera un texte ou une instruction qui n’a pas de sens dans le mode mathématique. Il en va de même si nous commençons un bloc de texte au moyen du caractère réservé « { » ou d’une commande `\startUnTruc` et que, par la suite, la fermeture explicite n’est pas trouvée (« } » ou `\stopUnTruc`).
3. **Erreurs de conception.** J’appelle ainsi les erreurs qui se produisent lorsque vous appelez une commande qui nécessite certains arguments, sans les fournir, ou lorsque la syntaxe d’appel de la commande n’est pas correcte.
4. **Erreurs situationnelles.** Certaines commandes sont destinées à ne fonctionner que dans certains contextes ou environnements, et sont donc inconnues en dehors de ceux-ci. Cela se produit, en particulier, avec le mode mathématique : certaines commandes ConT_EXt ne fonctionnent que lors de l’écriture de formules mathématiques et si elles sont appelées dans d’autres contextes, elles génèrent une erreur.

Que faire lorsque « context » nous avertit, pendant la compilation, qu’une erreur s’est produite ? La première chose est, évidemment, d’identifier quelle est l’erreur.

Pour ce faire, nous devrons parfois analyser le fichier « .log » généré pendant la compilation ; mais encore plus souvent il suffira de remonter dans les messages produits par « context » dans le terminal où il est exécuté.

```
tex error      > tex error on line 14 in file la-maison-sur-le-port_bug.tex: ! Undefined
control sequence

1.14 \starttext
           % Début du contenu du document

4
5  \setuppapersize[S5] % Format du papier
6
7  \setupbodyfont      % Police = Latin Modern, 12 points
8  [modern,18pt]
9
10 \setuphead          % Format des titres de chapitre
11 [chapter]
12 [style=\bfc]
13
14 >> \starttext      % Début du contenu du document
15
16 \startchapter
17 [title=La maison sur le port]
18
19 Il y avait des      chansons
20 Les hommes          venaient y boire et rêver
21 Dans la maison      sur le port
22 Où les filles        riaient fort
23 Où le vin faisait chanter chanter chanter
24

mtx-context    | fatal error: return code: 256
```

Figure 2.2 Sortie d’affichage en cas d’erreur de compilation

Par exemple, si dans notre fichier de test, « la-maison-sur-le-port.tex », par erreur, au lieu de `\startttext` nous avons écrit `\starttext` (avec un seul « t »), ce qui, par ailleurs, est une erreur très courante, lors de l’exécution de « context la-maison-sur-le-port », lorsque la compilation était arrêtée, dans l’écran du terminal nous pouvions voir l’information montrée dans la [figure 2.2](#).

Nous pouvons y voir les lignes de notre fichier source numérotées, et à l’une d’entre elles, dans notre cas la ligne 14, entre le numéro et le texte de la ligne le compilateur a ajouté « >> » pour indiquer que c’est dans cette ligne qu’il a trouvé l’erreur. Le numéro de la ligne est également indiqué plus haut, avant l’affichage des lignes, dans une ligne commençant par « tex error ». Le fichier « la-maison-sur-le-port.log » nous donnera plus d’indices. Dans notre exemple, il ne s’agit pas d’un très gros fichier, car la source que nous compilons est très petite ; dans d’autres cas, il peut contenir une quantité écrasante d’informations. Mais

nous devons nous y plonger. Si nous ouvrons « `la-maison-sur-le-port.log` » avec un éditeur de texte, nous verrons que ce fichier enregistre tout ce que fait ConT_EXt. Nous devons y chercher une ligne qui commence par un avertissement d'erreur « `tex error` », pour cela nous pouvons utiliser la fonction de recherche de texte de l'éditeur. Nous trouverons les lignes d'erreur suivantes :

```
tex error          > tex error on line 14 in file la-maison-sur-le-port_bug.tex: ! Undefined
control sequence

1.14 \starttext
      % Début du contenu du document
```

Note : La première ligne informant de l'erreur, dans le fichier « `la-maison-sur-le-port.log` » est très longue. Pour que cela soit présentable ici, en tenant compte de la largeur de la page, j'ai supprimé une partie du chemin indiquant l'emplacement du fichier.

Si nous prêtons attention aux trois lignes du message d'erreur, nous voyons que la première nous indique à quel numéro de ligne l'erreur s'est produite (ligne 14) et de quel type d'erreur il s'agit : « `Undefined control sequence` », ou, ce qui revient au même : « `Unknown control sequence` », c'est-à-dire une commande inconnue. Les deux lignes suivantes du fichier journal nous montrent la ligne 14, qui commence à l'endroit où l'erreur s'est produite. Donc il n'y a pas de doute, l'erreur est dans `\starttext`. Nous le lirons attentivement et, avec de l'attention et de l'expérience, nous nous rendrons compte que nous avons écrit « `starttext` » et non « `startttext` » (avec un double « `t` »).

Pensez que les ordinateurs sont très bons et très rapides pour exécuter des instructions, mais très maladroits pour lire nos pensées, et que le mot « `starttext` » n'est pas le même que « `startttext` ». Dans le second cas, le programme sait comment l'exécuter ; dans le premier cas, il ne sait pas quoi faire.

D'autres fois, la localisation de l'erreur ne sera pas aussi facile. En particulier lorsque l'erreur est qu'une tâche a été lancée et que sa fin n'a pas été expressément spécifiée. Parfois, au lieu de chercher l'expression « `tex error` » dans le fichier « `.log` », vous devez chercher un astérisque. Ce caractère au début d'une ligne du fichier journal représente, non pas une erreur fatale, mais un avertissement. Et les avertissements peuvent être utiles pour localiser l'erreur.

Et si les informations du fichier « `.log` » ne sont pas suffisantes, il faudra aller, petit à petit, localiser l'endroit de l'erreur. Une bonne stratégie pour cela consiste à changer l'emplacement de la commande `\stoptext` dans le fichier source. Rappelez-vous que ConT_EXt arrête de traiter le texte dès qu'il trouve cette commande. Par

conséquent, si, dans mon fichier source, j'écris, plus ou moins à la hauteur du milieu, un `\stoptext` et que je compile, seule la première moitié sera traitée ; si l'erreur se répète, je saurai qu'elle se trouve dans la première moitié du fichier source, si elle ne se répète pas, cela signifie que l'erreur se trouve dans la deuxième moitié... et ainsi, petit à petit, en changeant l'emplacement de la commande `\stoptext`, nous pouvons localiser l'emplacement de l'erreur.

Une autre astuce consiste à mettre en commentaires le paquets de lignes dou-teuses avec le caractère « % » (certain éditeur de texte propose une fonction pour commenter et décommenter tout un paquet de ligne automatiquement).

Une fois que nous l'avons localisée, nous pouvons essayer de la comprendre et de la corriger ou, si nous ne pouvons pas comprendre pourquoi l'erreur se produit, au moins, ayant localisé le point où elle se trouve, nous pouvons essayer d'écrire les choses d'une manière différente pour éviter que l'erreur se reproduise.ⁱ

TODO Garulfo : la remarque me semble plus source de complication dans l'esprit du lecteur que source d'éclaircissement.

Ce dernier point, bien sûr, uniquement si nous sommes les auteurs ; si nous nous limitons à composer le texte de quelqu'un d'autre, nous ne pourrons pas le modifier et nous devrons continuer à enquêter jusqu'à ce que nous découvriions les raisons de l'erreur et sa possible solution.

Dans la pratique, lorsqu'on crée un document relativement volumineux avec Con-TeXt, ce que l'on fait habituellement, c'est de le compiler de temps en temps, au fur et à mesure de la rédaction du document, de sorte que si une erreur se produit, nous savons plus ou moins clairement quelle est la partie du document qui vient d'être introduite depuis la précédente compilation et qui engendre la nouvelle erreur.

II

Global aspects of the document

III

Particular issues

Appendices

a

aide et ressources [31](#)

c

compilation [36](#)

composition [15](#)

l

langages de balisage

balises [17](#)

markup [17](#)

m

Mark II [14](#)

Mark IV [14](#)

moteurs T_EX [19](#)

p

préambule [40](#)

r

rédaction [15](#)