Mentor rapport.Manipulation de textes structures sous Mentor

B. Melese

INRIA
CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports Techniques

N° 23

MENTOR RAPPORT

MANIPULATION DE TEXTES STRUCTURÉS SOUS MENTOR

Bertrand MÉLÈSE

Avril 1983

Résumé

Mentor-Rapport est une première expérience visant à combiner les avantages de l'édition structurée et ceux de l'édition de texte en plein écran, lorsque, dans le formalisme manipulé coexistent des composantes pour lesquelles l'édition structurée est la plus appropriée et des composantes pour lesquelles l'édition de texte est la plus naturelle.

Rapport est un embryon de langage pour décrire des documents structurés, des rapports techniques par exemple. Dans de tels documents, la structure arborescente constituée par les titres, sections, sous-sections... sera éditée de façon naturelle par un outil tel que Mentor, tandis que le texte même des paragraphes sera édité de façon beaucoup plus naturelle par un éditeur de textes classique.

Mentor-Rapport est donc la première version d'un éditeur structuré de documents techniques.

Abstract

Mentor-Rapport is a first test to use simultaneously structured editing and full screen text editing when some components of the formalism to be manipulated are nicely edited in a structured manner while other components of the same formalism are handled better with a full screen text editor.

Rapport is a first step towards a language to describe structured documents, like technical reports. In such documents, the tree structure of titles, sections, sub-sections,..., is edited in a nice and natural manner by a tool such as Mentor. At the opposite, it is obviously more comfortable to edit the text of paragraphs with an editor like Emacs.

Mentor-Rapport is a first version of a structured editor of technical documents.

Mentor Rapport

Manipulation de textes structurés sous Mentor

Bertrand Mélèse
I.N.R.I.A
Domaine de Voluceau, 78153 Le Chesnay

1 – Introduction

On peut, en premiere approche, classer les composantes de la plupart des formalismes en deux groupes:

- Les composantes qui ont une structure syntaxique forte reflétant de près la sémantique: c'est par exemple le cas des instructions d'un langage de programmation.

- Les composantes dont la structure syntaxique est beaucoup plus libre vis à vis de la sémantique, tel le texte des commentaires d'un programme ou le texte d'un paragraphe dans un document en langage naturel.

Alors qu'il est naturel de manipuler les composantes à structure syntaxique forte avec des systemes dirigés par la syntaxe, ces systèmes présentent des contraintes déraisonnables pour la manipulation des composantes a structure syntaxique plus faible. Pour la manipulation de ces dernières composantes, en attendant une meilleure compréhension des langues naturelles qui permettra peut être un jour leur manipulation directe en fonction du contenu sémantique, il semble raisonnable aujourd'hui de s'en tenir aux méthodes traditionnelles d'édition de textes

non structurés, tout en cherchant à profiter des outils les plus performants pour ce type d'édition.

Mentor-Rapport vise à être un moyen de combiner ces deux approches complémentaires (éditeurs syntaxiques, éditeurs de textes) dans le cas fréquent de formalismes qui possèdent des composantes à structure syntaxique forte et des composantes à structure syntaxique faible.

Dans un rapport technique par exemple, la structure en chapitres, sections, sous sections telle qu'elle apparait dans la table des matières du rapport est raisonnablement et agréablement éditable par un outil tel que Mentor, tandis que la structure interne des paragraphes sera beaucoup plus facilement éditée par des outils comme Emacs ou Ted, c'est à dire par des éditeurs de textes classiques.

Le langage **Rapport** est un embryon de langage pour descrire des documents structurés. En Rapport, les documents sont essentiellement représentés en termes de sections, de sous-sections et de paragraphes. Mentor-Rapport est donc un éditeur structuré de documents qui offre toutes les possibilitées de Mentor dans le domaine de la création, correction, et de la manipulation de documents.

La vocation de Mentor en général étant de représenter les objets sous forme structurée [DON 80, DON 83], et la vocation de Mentor-Rapport étant donc de représenter les documents sous forme structurée, ce processeur est particulièrement intéressant pour manipuler des documents ayant une forte structuration, c'est à dire des documents du type rapport technique, notice d'utilisation, contrat, compte rendu d'assemblée, programme d'une manifestation etc...

Pour créer , corriger et manipuler un document, Mentor-Rapport fournit un certain nombre de primitives. L'utilisation exclusive de celle-ci dispense l'écrivain potentiel d'une connaissance approfondie de Mentor. Parmi ces primitives, sur lesquelles nous reviendrons par la suite, il en existe une, **compose** , qui crée une représentation du document toute prête à être donnée comme entrée au processeur **Compose** ce qui dispense notre écrivain, supposé paresseux, d'apprendre a se servir de ce processeur.

Mentor-Rapport et Mentor multi-langages.

Dans l'état actuel du système Mentor, dans lequel un seul langage peut être manipulé dans une même session, Mentor-Rapport est un éditeur de documents qui reste un peu déconnecté de l'ensemble du système.

Dans un avenir maintenant très proche, Mentor va devenir un système **multi-langages** , c'est à dire un système dans lequel on pourra manipuler simultanéement plusieurs formalismes. Cela signifie en particulier qu'un objet manipulé sous Mentor pourra avoir des composants dans des langages différents, c'est à dire qu'à l'intérieur d'un même arbre pourront co-exister des sous arbres obéissant à des syntaxes abstraites différentes.

Dans ce nouveau cadre, Mentor-Rapport deviendra de toute première importance puisqu'il pourra être utilisé pour rédiger les commentaires ou la documentation d'un programme,

ceux-ci restant liés à l'arbre représentant le programme, ce qui ouvre de nouveaux horizons pour la création et la maintenance de la documentation d'un programme en même temps que la création et la mainenance du programme lui-même.

Un programme pourra alors être vu sous deux angles différents:

- Du code dans un certain langage (Pascal, Ada, Métal, ...) auquel sont accrochés des morçeaux de texte qui suivent la syntaxe abstraite du langage Rapport, ou bien

- Un document en Rapport, que l'on pourra composer pour en avoir une sortie agréable à lire, dans lequel certaines partie sont dans un langage de programmation.

Dans le cadre de Mentor-Rapport, une autre application du multi-langage, et non la moindre, sera simplement l'édition d'un document qui contient des parties dans des formalismes différents. Prenons l'exemple simple du manuel d'un langage de programmation, Pascal pour fixer les idées. Dans un tel document, apparaissent des parties d'explications en langage naturel, des exemples en Pascal, des règles BNF pour décrire la syntaxe concrète du langage, des règles Métal pour décrire sa syntaxe abstraite (dans les bons manuels exclusivement), des diagrammes de syntaxe qui peuvent être décrit en Flip [KAHN 81]. Un tel document sera édité en Mentor multi-langages de façon naturelle, chacune des composantes faisant automatiquement appel au formalisme dans lequel elle est écrite.

Pour la mise en oeuvre de Mentor-Rapport sur le système Multics de l'I.N.R.I.A, l'écrivain impatient se reportera au segment d'information Mentor-Rapport.info qui se trouve dans la bibliothèque d'informations sur Mentor du projet CROAP. Son adresse est donc >udd>Croap>Mentor>info>Mentor-Rapport.info.

2 – Le langage Rapport

Le langage Rapport est un embryon de langage de description de documents. Dans son état actuel, il ne tente pas de définir une structure pour le langage naturel et ne connait donc ni les phrases, ni les symbôles de ponctuations, ni les genres des mots, il ne fait pas l'accord automatique des verbes ou des adjectifs et ne corrige pas automatiquement les fautes d'orthographe. Les atomes de la représentation d'un texte en Rapport sont des lignes et les paragraphes du document édité sont des listes de lignes. La coupure de ces lignes n'a aucune relation avec la structure du texte et ne dépend que de la façon dont l'écrivain a entré le texte en question. Bien entendu, lors de la composition ultérieure du document par la procédure **compose** , la justification à gauche et à droite sera faite, ainsi qu'un certain nombre d'autres adaptations en vue d'obtenir une forme sympathique du document. MENTOR-Rapport ne fait donc pas encore de traduction automatique de Russe en Japonais.

Les écrivains potentiels qui sont néanmoins intéressés par cet outil sont invités à prendre connaissance de la suite du présent document.

En Mentor-Rapport, tous les morceaux de texte sont des atomes du langage et ne pourront donc être modifiés que globalement. Heureusement, cela ne signifie pas que pour modifier un paragraphe, par exemple, vous allez être obligé de le retaper entièrement. Cela signifie que lorsque vous voudrez éditer un morceau de texte en Mentor-Rapport, c'est à dire un titre, un paragraphe, ou une référence, Mentor passera la main à votre éditeur de texte favori: Ted ou Emacs. Ce mécanisme correspond à l'idée de *gates* [DON 83], qui peut se résumer en disant que certains atomes d'un formalisme donné, ici les atomes qui sont des morçeaux de texte, sont en fait des objets structurés dont la structure appartient à un autre formalisme que le formalisme principal, et doivent donc être manipulés par d'autres primitives. En Mentor-Rapport, tous les morçeaux de texte sont structurés comme des suites de caractères et les primitives utilisées pour les manipuler s'appellent Ted et Emacs.

Mentor-Rapport a été conçu de façon à pouvoir être utilisé par des écrivains qui ne sont pas des spécialistes de Mentor. Cependant, il ne faut pas trop réver et quelques connaissances de base sur Mentor faciliteront la tâche et seront de toutes façons un investissement rentable. Il y a au moins une commande Mentor indispensable qui est la commande **p** qui affiche sur l'écran le sous arbre, donc en Mentor-Rapport la partie du document, dont la racine est la position courante.

2.1- La Syntaxe abstraite de Rapport

Nous ne reviendrons pas du tout ici sur le concept de **syntaxe abstraite.** Le lecteur qui ne se sent pas très au point sur ce sujet est invité à se reporter à l'abondante littérature sur MENTOR. Disons simplement que la **syntaxe abstraite** du langage Rapport est un ensemble de règles qui indique quelle sera la représentation arborescente des documents. Nous donnons ci-dessous la syntaxe abstraite de Rapport dans le formalisme Métal [MEL 82, KAHN 83]:

```
*
*  Operateurs
*
*    Atomiques
*
   ident          ->;
   meta           ->;
   ligne          -> implemented as STRING;
   comment        -> implemented as STRING;
   numero         -> implemented as INTEGER;
   francais       -> implemented as SINGLETON;
   english        -> implemented as SINGLETON;
*
*    Unaires
*
   indentpar      -> PARAGRAPHE;
   detailspar     -> PARAGRAPHE;
   telquelpar     -> PARAGRAPHE;
   progpar        -> PARAGRAPHE;
```

```
  titre          -> LIGNE;
  gros_titre     -> LIGNE;
  centre         -> LIGNE;
  fichier        -> IDENT;
*
*   Binaires
*
  biblio         -> ETIQUETTE BIBDESCR;
  etiquette      -> IDENT NUMERO;
  figure         -> NUMERO LIGNE;
*
*   Ternaires
*
  bibdescr       -> NOM_S LIGNE LIGNE;
  section        -> IDENT LIGNE FICHIER_ZONE_S;
  document       -> DOCDESCR_S TITRE_S ZONE_S;
*
*   Listes
*
  zone_s         -> ZONE * ... ;
  titre_s        -> TITRE * ... ;
  bibliographie  -> BIBLIO * ... ;
  paragraphe     -> LIGNE * ... ;
  comment_s      -> LIGNE * ... ;
  nom_s          -> LIGNE * ... ;
  docdescr_s     -> DOCDESCR * ... ;
*
*  Phyla
*
  BIBLIO         ::= biblio;
  NUMERO         ::= numero;
  BIBDESCR       ::= bibdescr;
  ETIQUETTE      ::= etiquette;
  PARAGRAPHE     ::= paragraphe;
  LIGNE          ::= ligne;
  NOM_S          ::= nom_s;
  ZONE_S         ::= zone_s;
  FICHIER_ZONE_S ::= zone_s fichier;
  TITRE_S        ::= titre_s;
  DOCDESCR_S     ::= docdescr_s;
  ZONE           ::= section paragraphe indentpar
                     detailspar telquelpar
                     progpar bibliographie
                     figure;
  TITRE          ::= titre gros_titre centre;
  DOCDESCR       ::= francais english ident;
  IDENT          ::= ident;
```

L'opérateur de tête d'un document en Rapport est l'opérateur ternaire **document.** Son premier fils est une liste de descripteurs du document. l'écrivain n'a pas a s'occuper de ces descripteurs qui seront positionnés une fois pour toutes par le système lors de la création du document comme nous le verrons plus loin. Le deuxième fils d'un document est une liste de titres, **titre_s** , et enfin, son troisième fils est une liste de zones, **zone_s.**

Comme vous le voyez dans le phylum **TITRE** , un titre est un **gros_titre** , un **titre** (normal) ou un **centre.**

Dans un premier temps, laissez faire Mentor-Rapport, il décidera pour vous lequel de ces trois opérateurs choisir selon l'emplacement d'un titre donné. Devenu un Mentor-Rapport-Wizard, dans quelques instants, vous pourrez intervenir sur ces choix.

Comme vous le voyez dans le phylum **ZONE** , une zone est l'un de opérateurs suivants: **section** , **paragraphe** , **bibliographie** , **figure** , **indentpar** , **detailspar** , **telquelpar** , **progpar.** Pour l'instant, ne tenez compte que des quatres premiers: Ils vous indiquent les structures fondammentales. Les quatre derniers sont des variantes de paragraphes et vous n'aurez jamais vous en préoccuper. Laissez donc Mentor-Rapport s'en débrouiller à votre place.

2.1.1- Exercice 1

Comprendre d'après les règles de syntaxe abstraite ci-dessus quelle est la structure de la bibliographie d'un document en Mentor-Rapport.

2.1.2- Exercice 2

Dessiner l'arbre de syntaxe abstraite qui represente le rapport que vous êtes entrain de lire en Mentor-Rapport.

2.2- Les points importants de la structure logique d'un document

Nous allons décrire ici les points importants de la structure d'un document en Mentor-Rapport pour les gens qui n'ont pas su faire les 2 exercices précédents. Les autres peuvent sauter cette section.

2.2.1- Structure d'un document

Un document, en Mentor-Rapport, est composé de trois parties: une liste de descripteurs, une liste de titres, un corps.

2.2.1.1- Les descripteurs du document

Les descripteurs sont à usage exclusif du système et sont crées par le système. Dans la version actuelle de Mentor-Rapport un document a deux descripteurs.

Le premier descripteur indique la langue dans laquelle le document est écrit et vaut soit *fr* (document en français), soit *en* (document en anglais). Ce descripteur est créé par les procédures de création de documents **document_français** et **document_anglais.**

Le deuxième descripteur donne le nom du fichier dans lequel le document sera sauvé. Il servira de base au système pour la création des noms des fichiers dans lesquels seront mis les

différentes versions du document créées par la procédure **compose.** Ce descripteur est l'identificateur qui est demandé en premier à l'écrivain lorsque celui-ci appelle l'une des procédures de création de documents citées plus haut.

2.2.1.2- Les titres du document

Les titres du document sont demandés à l'écrivain par les procédures de création de documents. Dans le format standard d'un document en Mentor-Rapport un document a un gros titre et un sous titre. Après ces deux titres, les procédures de création de documents demandent des renseignements sur l'auteur. Ces renseignements sont considérés comme des sous-sous-titres par Mentor-Rapport et sont mis dans la liste des titres du document. Pour intervenir sur les titres, en ajouter ou en enlever, l'écrivain devra attendre de connaître Mentor un peu plus en détails. Cette dernière remarque a d'ailleurs une portée un peu plus générale: Pour intervenir sur le format standard des documents adopté dans Mentor-Rapport notre écrivain sympathique devra patienter jusqu'a être devenu un utilisateur confirmé de Mentor. Cependant, je serai sensible et attentif à toutes les propositions qui me seront faites dans ce domaine.

Attention: un titre doit toujours tenir sur une seule ligne.

2.2.1.3- Le corps d'un document

Le corps d'un document est une liste de zones. Une zone peut être une section, un paragraphe, une figure ou de la bibliographie. Ces zones ne seront créées que par des appels à la procédure de création **créer.**

2.2.2- Structure d'une section

Une section est composée de trois parties: un identificateur, un titre et un corps. l'identificateur et le titre sont demandés à l'écrivain par la procédure **créer** lorsqu'on a demandé à celle-ci la création d'une section. L'identificateur est un simple marqueur de la section qui permettra de la retrouver facilement et rapidement en utilisant la procédure **chercher** (voir la section qui décrit cette procédure). Cet identificateur n'apparaîtra pas dans le document composé.

Le titre a une signification évidente. Il doit tenir sur une seule ligne.

Le corps d'une section est une liste de zones exactement de la même façon que le corps d'un document. Le nombre de niveau d'emboitement des sections est limité à 9, et le nombre de sections d'un niveau donné est limité à 99. Ces limitations ne proviennent pas de contraintes structurelles mais de contraintes liées à la numérotation des sections qui intervient dans les procédures **plan** , **en_tetes** et **compose.** Ces limitations pourront être élargies si un écrivain en fait la demande bien que je pense que personne n'arrivera à lire un document dans lequel il y a plus de 9 niveaux d'emboitement des sections. Une section de niveau 9 sera numérotée par quelquechose du genre

3.7.1.1.2.12.32.23.45

2.2.3- La structure d'un paragraphe

Un paragraphe est une liste de caractères et est considéré comme un atome par Mentor-Rapport. La création et l'édition d'un paragraphe se fait à travers des éditeurs prévus à cet effet: Emacs ou Ted.

2.2.4- La structure d'une figure

Dans l'état actuel de Mentor-Rapport, une figure est simplement définie par l'espace qu'il est necessaire de lui réserver (en nombre de lignes) et par son titre. Ces deux éléments sont demandés par la procédure **créer** lorsqu'on a demandé à celle ci la création d'une figure.

2.2.5- La structure de la bibliographie

En Mentor-Rapport, la bibliographie est une liste d'éléments, chaque élément étant une référence bibliographique. Un élément de bibliographie est composé de deux parties: une étiquette et un descripteur.

2.2.5.1- L'étiquette d'un élément de bibliographie

L'étiquette d'un élément de bibliographie sert à référencer cet élément dans le texte du document. Il est composé d'un identificateur et d'un numéro. En principe, l'identificateur devrait être le nom, ou une partie du nom, du premier auteur et le numéro devrait être la date de parution de l'article référencé. Ceci ne sera bien entendu pas vérifié par Mentor-Rapport.

2.2.5.2- Les descripteurs d'un élément de bibliographie

Un élément de bibliographie a trois descripteurs: la liste des noms des auteurs, le titre de l'article, l'indication de l'endroit où cet article est paru. Chacun de ces éléments doit tenir sur une seule ligne.

En réalité, notre écrivain inquiet n'aura pas à se préoccuper de la structure de la bibliographie. Celle-ci sera en effet créée par un dialoque avec le système ainsi que cela est expliqué dans la section sur la procédure **créer.**

3 – Les primitives de Mentor–Rapport

Ce chapitre contient la description des procédures de manipulation de documents actuellement disponibles en Mentor-Rapport. Ces procédures sont regroupées en sept grandes familles: Création, Navigation, Edition, Manipulation des fichiers, Utilisation des marques, Table des matières et Composition. A l'intérieur de chacune de ces sections, chaque sous section est consacrée à une ou deux procédures dont les noms apparaissent dans le titre de la sous section correspondante. Dans certaines sous sections, un nom est donné entre parenthèses à

la fin du titre. Ce nom est un deuxième nom pour la même procédure lorsque le nom original de la procédure est trop long pour avoir une chance d'être entré correctement du premier coup ou bien que l'utilisation intensive de la procédure justifie un nom court.

L'appel de l'une quelconque de ces procédures se fait en indiquant son nom précédé d'un point et suivi de ses arguments entre les symboles "<" et ">", selon la méthode standard en Mentor. Pour les procédures qui réclament des arguments, des exemples d'appels sont donnés.

L'ensemble des procédures présenté ici va être amené à grossir au fur et à mesure des désidératas des écrivains fanatiques. N'hésitez pas à me faire part de vos fantasmes les plus fous (en matière de manipulation de texte).

3.1- Les primitives de création de documents

3.1.1- Document_français (df)

Création d'un nouveau document en français. Cette procédure est la première à appeler lorsque l'on attaque la rédaction d'un nouveau document. Elle construit l'en-tête du document en posant des questions. A la sortie de cette procédure, la position courante est sur le corps, encore vide , du document et est donc bien positionnée pour permettre l'appel de la procédure **créer** décrite plus loin.

La première question posée concerne l' **identification** du document. L'identificateur entré en réponse à cette question par notre écrivain courageux sera utilisé par le système pour nommer les différents fichiers crées par les procédures de manipulation de fichiers et de composition.

3.1.2- Document_anglais (de)

Même principe que **Document_français** , à utiliser lorsque l'on veut écrire un document en anglais.

3.1.3- Créer

Procédure de création d'une partie de document: section, paragraphe, figure ou bibliographie. Le choix entre ces quatre possibilitées est déterminé par la réponse à la première question posée par cette procédure. L'emplacement de la nouvelle partie créée dépend de la position au moment de l'appel pour les 3 premières sus-citées. La bibliographie est toujours positionnée à la fin du document, avec concaténation à la bibliographie déjà existante si il y en a une.

Dans le cas de la création d'une section, d'un paragraphe ou d'une figure, le positionnement est effectué de la façon suivante:

- Si la position courante est sur le corps d'un document ou d'une section, (opérateur **zone_s**), la création a lieu à la fin, c'est à dire derrière le dernier élément de cette liste.

- Si la position courante est sur une zone, c'est à dire sur l'un des opérateurs appartenant au phylum **ZONE** , la création a lieu derrière cette position.

- Si la position courante n'est pas dans l'un des 2 cas précedents, elle est remontée sur le corps de la section englobante (ou du document) et on est donc ramené au premier cas. Si il n'y a pas de corps englobant, la création échoue. L'écrivain doit alors se déplacer à l'aide des procédures de navigation jusqu'à un endroit plus logique pour une création.

La demande de création d'un paragraphe a pour effet d'appeler l'éditeur de texte actif (voir les primitives d'édition). Le paragraphe est alors entré sous cet éditeur. Il est integré au document lors du retour de l'éditeur.

La création de la bibliographie est faite à travers un dialogue. Essayez, vous verrez, c'est facile il suffit de répondre aux questions posées.

3.1.4- Trier_la_Bibliographie (tribib)

Cette procédure trie les éléments de la bibliographie sur les champs *identification* et *date* de ces éléments.

3.2- Les primitives de navigation

3.2.1- Début

Positionnement au début du document en cours d'édition.

3.2.2- Titre

Va sur le titre de la construction courante si celle-ci est une section, un document ou un élément de bibliographie. Echoue et ne bouge pas si la construction courante n'a pas de titre.

3.2.3- Corps

Va sur le corps de l'unité courante. Echoue et ne bouge pas si l'unité courante n'a pas de corps. Les unités qui ont un corps sont **section** et **document.** Si la position courante n'est pas une section ou un document, la procédure **corps** cherche si il y a une section englobante et dans ce cas, va sur le corps de celle-ci.

3.2.4- Section (s)

Remonte à la section englobante si il en existe une. Ne bouge pas et échoue si il n'y en a pas.

3.2.5- Paragraphe

Va sur le premier paragraphe de la structure courante lorsque celle-ci est une section, une liste de zones ou un document. Lorsque la structure courante est un paragraphe détails, telquel, indenté ou programme (c'est à dire l'un des opérateurs **détailspar** , **telquelpar** , **indentpar** ou **progpar** respectivement), va sur le paragraphe normal correspondant. Ne bouge pas si la structure courante est un paragraphe normal ou si un paragraphe répondant aux critères ci-dessus n'existe pas.

3.2.6- Psuiv et Psp

La procédure **psuiv** va au paragraphe suivant du document. Echoue et ne bouge pas si il n'y en a pas. A la différence de la procédure **paragraphe** , la procédure **psuiv** peut, si nécessaire, changer de section pour trouver le paragraphe suivant. La procédure **Psp** est une toute petite variante: elle fait **psuiv** puis imprime le paragraphe. Ce nom, **psp** , est bien sûr une hérésie linguistique: c'est une contraction de *psuiv; p* qui est la commande Mentor faite par cette procédure.

3.2.7- Pder

Va sur le dernier paragraphe du document courant. Echoue et remonte au début si il n'y a aucun paragraphe dans ce document.

3.2.8- Ppre

Va sur le paragraphe précédent. Echoue et ne bouge pas si il n'y en a pas.

Attention, cette procédure refuse de chercher dans le cas où la position courante n'est pas un paragraphe ET qu'il n'existe pas de paragraphe suivant de la position courante. Ce comportement bizzare (mais heureusement assez rare) provient d'un problème assez délicat à avouer: je ne sais pas l'implémenter dans ce cas là.

3.2.9- Paragraphes_courts (pc), Paragraphes_longs (pl)

Les procédures **pc** et **pl** sont des procédures de changement d'état. Cela signifie qu'elles n'exécutent pas une action mais qu'elles influent sur le comportement ultérieur du système.

La procédure **pc** met le système dans l'état *Paragraphes en forme courte* ce qui signifie que les paragraphes de plus de 4 lignes seront abrégés par leurs 2 premières lignes suivies de points

de suspension. Cet état est assez sympathique car, à 1200 bauds, l'affichage des gros paragraphes peut être long et, en général, leurs 2 premières lignes suffisent à les identifier.

La procédure **pl** met le système dans l'état *Paragraphes en forme longue.* Dans cet état les paragraphes qui sont affichés sont toujours affichés en entier.

3.2.10- Chercher

Demande l'identification d'une section et va sur la section correspondante. Echoue et ne bouge pas si cette section n'existe pas. Notez que la procédure **chercher** explore l'arbre en pré-ordre à partir de la position courante, ce qui signifie qu'elle cherche uniquement **APRES** la position courante. Si la section cherchée est située avant la position courante il faut se repositionner plus haut avant de chercher, par exemple au début du document en utilisant la procédure **début** décrite plus haut.

Rappelons que l'identification d'une section est l'identificateur qui est demandé en premier par la procédure **créer** lorsqu'on lui a demandé de créer une section. Cet identificateur est décompilé entre deux tirets "-" devant le titre de la section et n'est qu'une marque destinée à permettre la recherche rapide d'une section: il serait en effet difficile et peu agréable de repérer une section par son titre, celui-ci pouvant être long à taper. Les identificateurs de section n'apparaitront pas lors de la composition du document.

3.2.11- Bib

Va sur la bibliographie du document courant. Echoue et ne bouge pas si le document courant n'a pas de bibliographie.

3.3- Les primitives d'édition

3.3.1- Edit (e)

Appelle l'éditeur de texte actif sur l'unité courante du document si celle-ci est éditable par un éditeur de texte. Une unité éditable par un éditeur de texte est soit un paragraphe soit une unité de type **ligne** c'est à dire un titre, un nom d'auteur dans la bibliographie ou une référence bibliographique par exemple.

L'écrivain a le choix entre les éditeurs de texte Emacs et Ted. L'activation de l'un ou l'autre de ces éditeurs est faite par les procédures **emacs** et **ted** décrites ci-dessous. Au retour de l'éditeur de texte, l'unité modifiée remplace l'ancienne dans le document.

Pendant l'édition en Emacs ou en Ted, notre écrivain courageux dispose de TOUTES les commandes de ces éditeurs. Il peut par exemple inserer des fichiers crées préalablement ou même, en Emacs, en profiter pour lire son courrier.

Les procédures **Emacs** , **Ted** et **Mentor** décrites ci-dessous sont des procédures de changement d'état.

3.3.2- Emacs

Active l'éditeur de texte Emacs. Après cette commande, la procédure **édit** appellera Emacs jusqu'à ce que celui-ci soit désactivé par un appel à l'une des procédures **ted** ou **mentor** décrites ci-dessous.

3.3.3- Ted

Active Ted. Il restera actif jusqu'à un appel ultérieur de l'une des procédures **emacs** ou **mentor.**

3.3.4- Mentor

Active Mentor comme éditeur de texte. Utile uniquement pour les utilisateurs confirmés de Mentor.

3.3.5- Indent, Détails, Telquel, Programme

Ces quatres procédures modifient le statut d'un paragraphe, c'est à dire la façon dont le paragraphe sera décompilé sous Mentor et la façon dont il sera traité lors de la composition du document. Elles ne doivent donc être invoquées que lorsque la position courante est un paragraphe.

- La procédure **indent** transforme le paragraphe en un paragraphe indenté, c'est à dire en un paragraphe qui sera indenté lors de la décompilation du document sous Mentor et lors de sa composition ultérieure.

- La procédure **détails** transforme le paragraphe en un paragraphe qui sera traité comme un élément d'une énumération. Il sera simplement indenté lors de la décompilation du document sous Mentor, mais il sera précédé d'un tiret lors de la composition ultérieure du document.

- La procédure **telquel** indique que le paragraphe devra sortir sans modifications à la composition.

- La procédure **programme** indique que le paragraphe est un morçeau de programme dans un langage quelconque. Lors de la composition, ce paragraphe sera dans une fonte différente du texte et dans un corps plus petit, sans modification de formattage.

Les quatre paragraphes précédents sont des paragraphes *détails*.

3.3.6- Casser <n>

La procédure **casser** permet de casser un paragraphe en deux paragraphes consécutifs. Lors de son invocation, l'écrivain doit lui passer en argument le numéro de la ligne DERRIERE laquelle il désire couper le paragraphe. Après l'exécution de cette procédure, la position courante est sur le deuxième des deux paragraphes résultant de la coupure.

Exemple d'appel: .casser<8>

Cette procédure peut, entre autres choses être utilisées pour diminuer le nombre des appels à l'éditeur de texte: lorsque l'on désire entrer plusieurs petits paragraphes, on peut les entrer comme un seul paragraphe au cours du même appel à l'éditeur et les casser après être revenu sous Mentor.

3.3.7- Substitution globale (sg), Substitution locale (sl)

Ces deux procédures ont été introduites pour permettre à notre écrivain fatigué de faire simplement, mais pas à faible coût, des substitutions d'une chaine de caractères par une autre dans tout une partie du document en une seule commande.

Ces deux procédures demandent la chaine originale puis la nouvelle chaine de caractères et se chargent de faire les substitutions. En première approche on peut dire qu'elles simulent des appels à l'éditeur de texte sur tous les paragraphes concernés en appliquant la substitution sur chacun d'entre eux.

La procédure **sg** applique la transformation sur toute la partie du document qui suit la position courante.

La procédure **sl** applique la substitution uniquement sur la partie du document qui est à l'intérieur du sous arbre dont la position courante est la racine.

3.4- Les primitives de manipulation de fichiers et de sauvegarde

3.4.1- Séparer, Regrouper

Lorsqu'un document devient gros, il peut être pénible, inutile et cher de le manipuler systématiquement dans son ensemble. La procédure **séparer** envoie une partie du document dans un autre fichier dont elle demande le nom à notre écrivain courageux. Dans le document principal, la partie qui a été séparée est remplacée par l'indication du fichier dans lequel elle a été mise. Si cette séparation n'apparait pas satisfaisante, la partie qui a été séparée peut être ramenée à sa place légitime dans le document principal avec la procédure **regrouper.**

Une partie qui a été séparée peut ensuite être manipulée sous Mentor indépendamment du document principal. Seules les sections peuvent être séparées.

3.4.2- Document_complet

Ecrit dans un fichier le document complet avec le même format que celui sous lequel il est présenté à l'écran sous Mentor. Sur Multics, le nom du fichier est composé de l'identificateur du document en majuscules avec l'extension *rapport* en minuscules (par exemple DOC.rapport).

Cette procédure se charge de faire tous les regroupements nécessaires des parties séparées. Le document sorti est donc bien complet. Cette procédure sera donc utilisée pour faire une sortie lisible sur papier du document sans le composer.

3.4.3- Sauver

Sauve le document sous sa forme arborescente dans un fichier dont le nom serait DOC.polish si l'identificateur du document était doc. Ce sont ces fichiers *polish* qui seront rechargés par la procédure **load** dans une session ultérieure de Mentor-Rapport.

3.4.4- Load

Chargement d'un fichier *polish* dont le nom est demandé à l'écrivain. La procédure **load** est une primitive standard de Mentor, ce qui explique que son nom soit en mauvais français. L'écrivain persévérant constatera que certains messages du système sont également en mauvais français. Cela permet de distinguer les messages qui sont standards en Mentor de ceux qui sont particuliers à Mentor-Rapport.

3.4.5- Ecrire

Comme **document_complet** mais sans regroupement des parties séparées.

3.4.6- Sortir

Sortie de Mentor-Rapport avec sauvegarde du document sous forme polish.

3.4.7- S_automatique

Passe en mode de sauvegarde automatique. Dans ce mode, le document est sauvé sous forme *polish* à chaque retour de l'éditeur de texte quelqu'il soit.

3.4.8- Pas_de_sauvegarde_automatique (pdsa)

Supprime les sauvegardes automatiques. Ne pas oublier de sauver son document avant de sortir de Mentor-Rapport. Une bonne habitude à prendre est de toujours sortir de Mentor-Rapport par la procédure **sortir** décrite plus haut.

3.5- Utilisation des marques

Remarque préliminaire:

> les procédures de manipulation des marques sont un moyen permettant à l'écrivain débutant en Mentor d'utiliser les variables Mentol sans les voir. Elles procurent en plus une sécurité supplémentaire dans la manipulation de ces variables en interdisant de détruire des marques existantes ou d'utiliser des marques non encore définies.

Nous appelons *marque* , un repère positionné à un endroit quelconque du document. Une *marque* désigne donc un sous arbre de l'arbre représentant le document. Elle permettra d'aller sur ce sous arbre en adressage direct, de transporter ce sous arbre ailleurs dans le document et bien d'autres choses encore.

3.5.1- Marquer

Demande le nom de la marque. Refuse si la marque existe déjà. Marque la position courante, c'est à dire la partie du texte constituée par le sous arbre dont la racine est la position courante.

3.5.2- Voir_les_marques_existantes (vm)

Indique les marques déjà enregistrées, c'est à dire les marques qui ont été créées par un appel à la procédure **marquer** depuis le début de la session Mentor-Rapport. Les marques positionnées dans le documents ne sont pas sauvegardées dans les fichiers *polish* et ne sont donc pas rémanentes d'une session Mentor-Rapport à une autre.

3.5.3- Aller <@nom>

Déplace la position courante sur la position de la marque passée en argument. Ne bouge pas si la marque indiquée n'existe pas. Pour passer une marque en argument, il faut indiquer son nom précédé du symbole "@".

Exemple d'appel : .aller<@ici>

3.5.4- Echanger <@nom1,@nom2>

Echange les positions de sous arbres désignés par les marques passées en arguments si ces marques sont indépendantes, c'est à dire si les sous arbres désignés sont disjoints. Refuse l'échange dans le cas contraire.

Exemple d'appel: .echanger<@ici,@labas>

3.5.5- Substituer <@nom>, Remettre

La procédure **substituer** remplace le sous arbre courant par le sous arbre désigné par la marque passée en argument. Le sous arbre qui était à la position courante est conservé dans une pile de sous arbres.

La procédure **remettre** remplace le sous arbre courant par le sous arbre qui est au sommet de la pile sus-mentionnée.

3.6- Table des matières

3.6.1- Plan

La procédure **plan** présente à l'écran la table des matières du document avec la numérotation des sections. Celle-ci pourra être integrée au document lors de sa composition.

3.6.2- En_têtes

Comme **plan** mais indique aussi les identifications des sections. Très utile pour se rappeler ces identifications pour pouvoir ensuite foncer droit sur une section particulière avec la procédure **chercher.**

3.6.3- Numéroter, Ne_pas_numéroter

La procédure **numéroter** passe dans un mode ou les sections sont numérotées systématiquement à chaque décompilation du texte. La procédure **ne_pas_numeroter** sort de ce mode. Lorsque la numérotation est active, le numéro d'une section donnée dépend de la position à partir de laquelle la décompilation à lieu. Les numéros indiqués ne sont alors significatifs que relativement à cette position.

La seule façon d'obtenir la numérotation absolue des sections est d'utiliser l'une des procédures **plan** ou **en_tête** sus-citées.

3.7- Composition d'un document

3.7.1- Compose

Cette procédure compile le document dans un format accepté par le processeur *compose* (sur Multics) et près à être traité par ce processeur. L'écrivain à alors la possibilité envoyer son document pour photo-composition, si il a accès à la VIP de l'I.N.R.I.A., en utilisant les commandes *vipfcomp* et *vipprint* qui ont étés mises au point par les membres du projet Typo de l'I.N.R.I.A. [TYPC 81].

Bien entendu, cette procédure, tout comme la procédure **document_complet** se charge de regrouper les parties séparées du document. La procédure **compose** crée deux fichiers dont les noms seraient DOCCOMP.compin et DOCPLAN.compin si l'identificateur du document

était doc et qui contient les données pour le processeur **Compose.** Le fichier DOCCOMP.compin contient le code *compose* pour le document tandis que le fichier DOCPLAN.compin contient le code *compose* pour la table des matières de ce document.

Le texte est composé sous une forme standard dont je suis actuellement le seul responsable. Toutes les critiques et propositions sur ce format seront acceptées et eventuellement prises en compte. Les spécialistes de *compose* trouveront tout seuls comment adapter ce format à leurs désidératas.

Pour finir, il faut quand même signaler à l'écrivain enthousiaste que la procédure **compose** accompli un gros travail et prend donc un peu de temps. Il faut, par exemple, 35 secondes de temps UC à cette procédure pour composer le document que vous avez sous les yeux.

3.7.2- Règles de composition à respecter

Lorsque l'on écrit un document en français, un certain nombre de conventions doivent être respectées:

- les caractères accentués doivent être entrés sous la forme du caractère voulu suivi de l'accent que l'on veut lui mettre. Ne pas mettre de caractère *back-space* entre les deux.

- Le ç est entré par un c suivi du caractère "/ ".

- Le oe est entré par un o suivi d'un e

Les procédures **vipfcomp** et **vipprint** se chargent de la transformation lors de la sortie sur la VIP. Une procédure, **ajfcomp** , permet de sortir son document sur un terminal de type Anderson-Jacobson avec les caractères français. Si vous n'avez accès à aucun de ces 2 types de périphériques, pas de panique, il est possible de sortir le document sur n'importe quel terminal imprimant en faisant quelques modifications triviales sur le fichier *compin* créé par la procédure de composition.

Si, maintenant, vous désirez écrire un document en anglais, il n'y a aucuns problèmes de ce type.

3.7.3- Les changements de police de caractères

Passons maintenant à des questions plus attractives: les changements de police de caractères à l'intérieur d'un paragraphe.

Du point de vue de Mentor, les paragraphes sont des atomes de la structure. Il n'y a donc, dans l'arbre, aucune indication de composition interne au paragraphe. Si notre écrivain pointilleux veut améliorer l'ordinaire il devra rajouter lui même des ordres de composition à l'intérieur de ses paragraphes au moment ou il les rentre avec son éditeur de texte favori. Actuellement, les seuls ordres de composition autorisés à l'intérieur des paragraphes sont des ordres de changement de la police de caractères.

Les changements de police sont indiqués par des ordres mis dans le texte . Ces ordres commencent et finissent par le caractère "|". Pour mettre dans son texte le caractère "|" il faut en mettre deux à la suite. Un ordre est simplement le nom de la police désirée. Actuellement, Mentor-Rapport connait 3 polices: normale (norm), gras (gras), italique (ital). Les indications entre parenthèses sont les noms qui devront être utilisés pour ces polices dans le texte. Un changement de police commence au caractère qui suit l'ordre de changement de police et se termine soit à la fin de la ligne courante, soit au prochain changement de police. Chaque fin de ligne provoque un retour à la police par défaut qui est la police normale.

Exemple:

Cette partie sera en police normale |gras|celle-ci sera en gras
et ici on repasse en normale|ital|et là en italique
|ital|ici l'italique continue|norm|ici on est en normale.
Ce paragraphe est bien entendu un paragraphe telquel.

Restrictions:

- Il ne doit pas y avoir de changement de police dans les titres.

- Les changements de police ne sont pas pris en compte dans les paragraphes de type **telquel** et **programme.**

- Le processeur *compose* pour lequel la procédure **compose** de Mentor-Rapport crée du code sur Multics n'aime pas du tout les caractères point (".") en début de ligne et interprète les blancs (" ") en début de ligne comme des ordres de non formattage de la ligne correspondante, grosso-modo. Aussi, Mentor-Rapport est obligé de vérifier que ces caractères n'apparaîssent pas en début de ligne et de les supprimer si c'est le cas. Dans le code créé pour *compose* , chaque portion de texte qui suit un changement de fonte se retrouve en début de ligne, ce qui interdit donc de commencer ces portions de texte par les blancs ou des points. En fait vous pouvez les commencer par ces caractères mais la procédure **compose** les supprimera. Pour les blancs tout ceci n'est pas grave car, de toutes façons un changement de police est considéré comme un séparateur et provoquera la mise d'un blanc à l'endroit du changement de police.

- On ne peut donc pas changer de police à l'intérieur d'un mot.

Je suis désolé si ces restrictions chagrinent notre écrivain attristé mais je n'en suis pas vraiment responsable. Révons ensemble d'un système de composition de texte plus malin que *compose.*

Pour les aventuriers confirmés, je signale tout de même l'existence d'un *flag* dans Mentor-Rapport qui s'appelle **wizard** et qui supprime les vérifications citées ci-dessus. Quand ce **flag** est actif,

la procédure **compose** ne supprime pas les blancs et les points en début de ligne: votre responsabilité est alors entière pour les ennuis qui peuvent découler de cette utilisation abusive, mais prévue, de Mentor-Rapport.

4 – Conclusion

Comme cela est annoncé dans l'introduction, l'écrivain tenace a pu constater que Mentor-Rapport est un système très peu ambitieux qui cherche simplement à utiliser au mieux les capacitées de Mentor et des éditeurs de texte Emacs et Ted.

Son principal défaut actuel réside certainement dans le mode d' interaction entre Mentor et les éditeurs de texte. En effet, la communication se fait par l'intermédiaire d'un fichier temporaire créé par Mentor et relu par l'éditeur de texte concerné. A la sortie de l'éditeur de texte, ce fichier temporaire modifié sera relu par Mentor qui lui appliquera les traitements necessaires à son insertion correcte dans le reste du document. Une interaction beaucoup plus ergonomique consisterait, bien entendu, à remplacer les passages explicites de Mentor aux éditeurs de texte par des passages implicites. Le seul fait de demander l'édition d'un paragraphe rendrait alors actives les commandes de l'éditeur de texte courant sur la représentation textuelle du paragraphe courant.

J'espère que, dans le cadre d'un poste de travail personnel (basé sur la SM90 ?), il sera possible d'améliorer considérablement l'interface utilisateur de Mentor, et donc de Mentor-Rapport, par une intégration beaucoup plus étroite des outils d'édition en plein écran dans Mentor, et, pourquoi pas, de réaliser cette interface utilisateur sur un écran à haute résolution.

Je pense qu'aujourd'hui il faut considérer Mentor-Rapport comme une preuve de faisabilité et d'utilité d'un tel système, et garder en mémoire qu'il ne prendra toute son ampleur que dans le cadre du futur Mentor multi-langages.

Bibliographie

Don 80

V. Donzeau-Gouge, G. Huet, G. Kahn, B. Lang, *Programming environments based on structured editors: The Mentor experience* I.N.R.I.A., Rapport de Recherche no 26, Juillet 1980

Don 83

V. Donzeau-Gouge, G. Kahn, B. Lang, B. Mélèse, *Outline of a tool for document manipulation* Papier soumis à l'IFIP 83

Kahn 81

G. Kahn, *Flip: manuel de référence* I.N.R.I.A., Rapport Technique no 2, Juin 1981

Kahn 83

G. Kahn, B. Lang, B. Mélèse, *Metal: a formalim to specify formalisms* A paraître, disponible auprès des auteurs

Mel 82

B. Mélèse, *Métal, un langage de spécification pour le système Mentor* Technique et Science Informatique (AFCET), Vol. 1 No 4, Juillet-Aout 1982

Typo 81

Projet Typo, *Utilisation de Compose* Rapport Technique No 2.1, Juin 1981

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique