INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

UNE APPROCHE POUR LA CONCEPTION DE SYSTEMES DE TRAITEMENT DE TEXTES EXTENSIBLES ET INTERCONNECTABLES

THESE

PRESENTEE POUR L'OBTENTION DU DIPLOME DE

DOCTEUR - INGENIEUR

EN INFORMATIQUE

PAR

MAGDA MOURAD (ép. TANTAWY)

Date de la Soutenance Publique: 8 Juillet 1982

Jury d'Examen:

Président: M. D. COULON

Rapporteur: Mme. M. QUERE

Examinateurs: MM. J. COURTIN

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

UNE APPROCHE POUR LA CONCEPTION DE SYSTEMES DE TRAITEMENT DE TEXTES EXTENSIBLES ET INTERCONNECTABLES

THESE

PRESENTEE POUR L'OBTENTION DU DIPLOME DE

DOCTEUR - INGENIEUR

EN INFORMATIQUE

PAR

MAGDA MOURAD (ép. TANTAWY)

Date de la Soutenance Publique: 8 Juillet 1982

Jury d'Examen:

Président: M. D. COULON

Rapporteur: Mme. M. QUERE

Examinateurs: MM. J. COURTIN

N. NAFFAH

J.P. THOMESSE

Je tiens tout d'abord à exprimer tous mes remerciements et toute ma gratitude à Mme. le Professeur Maryse Quéré qui m'a bien accueillie dans son équipe et m'a entourée de ses conseils fructueux et son aide généreuse, ce qui a été le facteur déterminant qui m'a permis de mener à bien cette thèse.

Je voudrais aussi remercier M. le Professeur Jean Pierre Thomesse pour son accueil inoubliable, son aide précieuse et l'honneur qu'il me fait en participant au Jury.

Mes remerciements sincères vont également à M. le Professeur Daniel Coulon pour avoir accepté de me faire l'honneur de présider le Jury.

Je tiens à remercier vivement M. le Professeur Jacques Courtin de l'Univesité de Grenoble et M. Najah Naffah, Directeur du projet pilote KAYAK à l'INRIA pour le très grand honneur qu'ils me font en acceptant de participer au Jury.

Je voudrais exprimer toutes mes amitiés et ma gratitude à tous ceux qui m'ont fait bénéficier de leurs critiques et de leurs discussions aussi bien au CRIN qu'à l'IMAG.

Magda

RESUME DE LA THESE

UNE APPROCHE POUR LA CONCEPTION DE SYSTEMES DE TRAITEMENT DE TEXTES EXTENSIBLES ET INTERCONNECTABLES

Cette thèse commence par la présentation des systèmes de production de documents. Les phases du cycle de production d'un document sont définies. Les outils informatiques aidant l'utilisateur à mener à bien les tâches de chacune de ces phases sont précisés.

Nous démontrons ensuite qu'il est possible de construire un système extensible pouvant être modelé à tout moment selon les besoins de l'utilisateur.

Les propositions essentielles formulées se développent autour de quelques idées clés: système minimal, extensibilité et flexibilité, et espace adressable. Le système minimal est le noyau autour duquel il est possible de développer d'autres systèmes de puissance et de taille quelconques. Un outil d'extension est donné par un langage modulaire de traitement de textes. L'extension peut ainsi être faite par l'utilisateur lui-même aussi bien que par le constructeur.
Une technique d'adressage est utilisée pour spécifier la partie du texte qui sera affectée par une fonction donnée.

Après avoir décrit l'aspect théorique de notre système extensible, une réalisation prouvant la faisabilité de notre idée est présentée.

Il apparait que le langage de formattage que nous avons développé, étant à la fois minimal et universel, peut être la base d'une solution au problème de l'interconnexion de systèmes hétérogènes de traitement de textes. Cette étude constitue la 2eme. partie de la thèse.

Le problème apparait lorsqu'on veut transférer un texte (document), créé par un système spécifique, pour le diffuser, le ré-éditer ou reformatter sur un (plusieurs) autre(s) système(s).

Trois solutions proposées dans la bibliographie sont présentées. Nous proposons ensuite une solution basée sur notre langage de formattage, et nous la comparons aux autres propositions.

La faisabilité de cette solution est prouvée par une réalisation décrite dans le chapitre-IV- de la thèse.

TABLE DES MATIERES

INTRODUCTION: LA PRODUCTION D'UN I OCUMENT

A. Le recueil d'informations

B. L'élaboration du document

C. L'édition

D. La révision

E. Le formattage du document

F. Les moyens d'illustration

G. L'impression et la reproduction

H. La distribution du document

INTRODUCTION: LA PRODUCTION D'UN DOCUMENT

Dans cette introduction, nous étudions les phases du processus complet de production d'un document. Nous entendons par "document" une information qui peut éventuellement être reproduite sur un support physique (ex. papier, écran) sous forme de texte, de tables, de graphique, d'image, etc... . La longueur du document peut varier d'une à plusieurs pages.

Le processus de production d'un document peut se diviser généralement en plusieurs phases fonctionnellement distinctes. Ces phases sont définies dans la littérature d'une manière plus ou moins complète ‹4, 5, 26, 29, 37, 63›. Nous proposons de définir ces phases comme suit, afin d'avoir un découpage clair, et de pouvoir mieux spécifier pour chacune les outils informatiques utiles:

A. Le recueil d'informations
B. L'élaboration du document
C. L'édition
D. La révision
E. Le formattage du document
F. Les moyens d'illustration
G. L'impression et la reproduction
H. La distribution du document

A- Le recueil des informations de base nécessaires à l'élaboration du document.

Cette phase comprend tout le processus par lequel l'auteur acquiert les informations qu'il va utiliser pour élaborer son document. Elle comporte deux étapes: la recherche de l'information puis l'enregistrement de cette information.

A l'heure actuelle, de nombreuses informations so mémorisées dans des systèmes informatiques. Ceci fait pens que les techniques de documentation ou de gestion de bases données peuvent être utilisées pour faciliter, dans cette phas le travail humain. Les moyens d'aide au recueil d'informati existants dans un système informatisé, peuvent donc être:

1- Un affichage simple sur écran et la possibilité parcourir un document à une vitesse quelconque.

2- La création et la mémorisation d'un catalogue q contiendra, par exemple, le titre de chaque document, s résumé, une liste de mots-clés, etc. Il est ensui possible de rechercher automatiquement des documents répo dant à des critères de choix concernant les éléments catalogue.

3- Un filtrage automatique d'un -ou plusieurs- docum pour n'afficher que les parties répondant aux critères recherche de l'utilisateur.

4- Une constitution automatique de la liste des référenc Les textes qui intéressent l'auteur seront automatiquem ajoutés à la liste des références utiles.

5- La liaison entre documents. Cette possibilité perme l'auteur d'un document de spécifier des parties d'aut documents comme références (statistiques, docume originaux, etc...). Les lecteurs du nouveau docu auront la possibilité de demander un renvoi vers sections référencées. Un retour au document initial fera sur la demande du lecteur.

B- L'élaboration du Document

L'écriture est le moyen de communiquer les idées dans une forme narrative lisible. Ce processus consiste à copier quelques parties d'autres documents, organiser un plan, développer graduellement un texte autour de chacun des éléments de ce plan, et créer des tables et des figures.

L'informatique peut aussi être utile dans cette phase grâce à deux outils:

1- Le matériel de saisie: il comprend l'outil par lequel l'utilisateur envoie la donnée au système informatique sous une forme perceptible par celui-ci. Cet outil peut être un clavier semblable à celui d'une machine à écrire, auquel sont ajoutées des touches spéciales représentant des symboles graphiques spéciaux, ou contrôlant directement le curseur, ou invoquant les différentes fonctions de traitement de textes, etc. D'autres outils d'entrée peuvent aussi être utilisés (ex. un lecteur optique, un système d'entrée vocale, etc.).

Quelquefois le matériel de saisie a la capacité d'afficher les codes entrés. Cet affichage a lieu sur le papier d'une imprimante ou sur un écran de télévision. Les outils d'affichage varient selon leur vitesse, les dimensions de la surface lisible, les types des codes affichés (ex. alphanumérique, graphique), etc. Il est certain que le type des documents produits, le type de l'utilisateur et la destination du document sont les facteurs qui déterminent le choix. Par exemple, l'entrée de texte par une secrétaire nécessite un outil permettant une saisie rapide d'une grande quantité de caractères. Un clavier semblable à celui d'une machine à écrire est plus confortable. Les touches de commandes doivent être claires et simples.

2- Le logiciel de saisie: c'est l'ensemble des program d'application disponibles sur le système informatique destinés à alléger les tâches de saisie. Cela peut al d'un éditeur ayant une liste de commandes à l'a desquelles l'utilisateur peut corriger immédiatement fautes typographiques, à des systèmes plus puissants utilisent des techniques d'intelligence artificielle ‹ ‹61›; par exemple: les logiciels de correction automati des fautes d'orthographe ou de grammaire, les logiciels aident à améliorer le choix de phrases pour exprimer idée, et les logiciels qui aident l'utilisateur à dével per son texte à partir d'un plan d'idées générales. existe d'autres logiciels qui aident à générer autom quement des parties du document construites par répétition des chaines qui existent déjà dans ce docum par exemple: la table des matières, la liste des figu les mots-clé, la liste des références, le lexique, etc.

Il est préférable que le logiciel offre à l'utilisa la sécurité contre la perte ou un changement involont du contenu du document qu'il est en train de créer.

C- L'édition

Nous entendons par édition la révision d'une ver préliminaire d'un document en vue de sa modification éventue Un éditeur peut proposer la réécriture d'une certaine pa pour mieux présenter les idées. Il peut faire recommandations concernant l'organisation et le conten document, sa conformité aux normes d'un langage d'écritur aux normes d'une publication particulière. Le travail n saire à la mise en oeuvre de ces suggestions, y compris la s de nouvelles parties, peut être considéré comme une parti processus de l'édition.

Ce cycle peut être répété, nécessitant ainsi la saisie de quelques parties du document deux ou plusieurs fois.

Nous présentons ici quelques caractéristiques qu'on peut rencontrer dans des éditeurs de systèmes informatisés:

1- Le terminal utilisé: l'éditeur affiche la partie du document à éditer sur ce terminal. Le terminal peut être un téléimprimeur qui imprime cette partie du document et qui n'a donc pas de moyen de repérer visuellement les mouvements d'un pointeur virtuel au milieu du document, ce pointeur étant nécessaire pour désigner l'endroit à éditer. Le terminal peut également être un écran de visualisation, c'est actuellement le genre le plus répandu. La désignation d'un caractère quelconque dans le texte affiché peut se faire soit directement à l'aide d'un crayon optique (ou un appareil similaire), soit indirectement à l'aide des mouvements d'un curseur (touches, souris ou appareil similaire).

2- La méthode de déclenchement des commandes d'édition: elle dépend généralement du type du terminal utilisé. Une commande d'édition est déclenchée à l'aide de deux composantes, la fonction à activer et l'identification de la partie du texte qui sera affectée.
Si le terminal utilisé est un téléimprimeur, la commande est déclenchée en tapant le nom de la fonction (ou son abréviation). Pour localiser la partie du texte qui sera affectée, on donne des commandes de manipulation d'un curseur (virtuel) pour le positionner au début de la chaine à éditer (ex. on donne le numéro de la ligne ou bien on demande la recherche automatique d'une chaine de caractères).

Dans le cas d'un terminal à écran de visualisation, commande est déclenchée en tapant le nom de la fonc (avec ou sans affichage de ce nom). Dans les systèmes évolués, une liste des fonctions est disponible sur l'é (menu). La simple désignation d'une fonction à l'aide curseur ou d'un crayon optique déclenche son exécution. partie du texte qui sera affectée peut être soit déc avec la commande, soit désignée par la sélection dan menu de codes délimiteurs, ou bien un curseur (ou un cr optique) pointera physiquement vers la partie à éditer le texte. Cette partie peut avoir une définition s dard qui sera prise en compte automatiquement à dé d'une autre définition.

3- L'ensemble des fonctions d'édition: les fonct d'édition sont nombreuses et leur puissance varie sel système et selon la puissance de l'ordinateur utilisé Les entités traitées par ces fonctions varient d'une e physique (ex. une ligne physique) aux entités log composant le document (ex. mot, phrase, paragraphe). systèmes puissants donnent la possibilité d'éditio entités représentant la hiérarchisation logique document (ex. chapitre, section, titre, etc.).

4- La désignation d'une partie d'un document: les comm de déplacement à l'intérieur d'un document ainsi q désignation des entités à déplacer sont définies pa codes spéciaux ou par des touches directes ajouté poste de travail ou, encore, par des codes sélectio partir d'un menu affichable. Ces codes spécifie direction du déplacement dans le document (ex. haut, etc..). L'entité à déplacer peut être une entité ph du document, ou une entité qui représente une st hiérarchique dans le document.
Quand les documents sont trop longs, des routin pagination sont nécessaires pour remplacer le texte existant dans la mémoire principale par la partie résidant en mémoire secondaire, en général un disque) que l'utilisateur veut examiner. Ceci peut être transparent pour l'utilisateur.

5- La forme du document édité: l'éditeur peut éditer le document entré par l'opérateur dans sa forme originale, ou dans sa forme de stockage sur les supports du système, ou dans sa forme formattée.

6- L'affichage du résultat de l'édition: dans les systèmes utilisant des téléimprimeurs comme terminaux, les chaines modifiées ne sont imprimées que sur demande de l'utilisateur afin d'économiser le temps d'impression qui est relativement long. Dans quelques systèmes, le chainage de plusieurs commandes d'édition est possible.
Dans les systèmes utilisant des terminaux à écran, il est toujours possible de montrer le résultat de l'édition immédiatement après l'exécution. Quelques commandes nécessitent un traitement assez long renvoient un message indiquant que l'exécution est en cours.

7- La manière dont le document est modifié: deux manières générales existent; (a) le système crée automatiquement une seconde version du document qui contient tous les changements demandés. La version originale, toujours intacte, permet à l'utilisateur de se rattrapper en cas de confusion ou d'erreur. (b) le fichier original est lui-même modifié directement par l'éditeur, après une commande explicite par l'utilisateur, ou automatiquement par le système après chaque fonction d'édition.

8- La possibilité d'édition à partir d'une autre pha

dans les systèmes élémentaires, la phase d'édition
complètement séparée des autres phases de production d
document, sauf pendant la création. Dans ce dernier c
quelques systèmes ne donnent accès qu'à quelques-unes
fonctions principales d'édition. Dans les systèmes p
évolués, l'édition peut se faire pendant quelques
toutes les) phases de production de documents. Le syst
reste dans son environnement initial (ex. formattage) ap
l'exécution des commandes d'édition.

D- LA REVISION:

La révision est le processus consistant à lire
document, l'approuver, le désapprouver, ou faire des pro
sitions de modifications avant l'approbation. Comme l'éditi
la révision peut se répéter plusieurs fois en cyc
consécutifs.

La personne qui fait la révision peut éditer ou demar
l'édition du texte, mais la révision diffère de l'édition pa
position de cette personne, son pouvoir d'approbation ou
désapprobation, son attention sur le contenu du document.

En général, cette phase n'a pas un module spécial
un système de production de documents. En fait, la révision
un cas spécial d'édition.

E. LE FORMATTAGE DU DOCUMENT

Le formattage est la conception de la page imprimée. Il peut être simple, comme dans le cas d'une secrétaire tapant un texte selon des normes prescrites, ou compliqué, comme la mise en page d'une publication par un spécialiste de conception de format.

E.1- LES ETAPES PRINCIPALES DU FORMATTAGE:

(a) La Conception: Le formattage global et le style du document sont déterminés. En ce qui concerne les publications techniques, la conception doit respecter certaines normes. Dans cette étape, le concepteur fait une structuration générale du document en prenant en compte plusieurs éléments comme la typographie, les proportions, les espaces, etc. Ce formattage est proposé au client qui donne son avis. Le cycle peut être recommencé jusqu'à l'accord sur la forme finale. Normalement, un système de traitement de textes permet à l'utilisateur de concevoir une page avec des marges, de la diviser en colonnes ou en forme de tableau, en lui donnant la facilité de définir des en-têtes, renvois, ou demander la numérotation automatique des pages, choisir les espacements, le soulignement, la sélection des polices, etc. Mais il existe aussi une gamme de systèmes utilisant des écrans à haute résolution, avec lesquels il est possible de concevoir un format complexe pour une page (ex. une page d'un manuel de produits mécaniques) sans taper le texte qui y sera écrit.

(b) L'estimation de la taille: C'est une estimation arithmétique de l'espace requis par le texte dans sa forme finale déjà conçue. Cette estimation aura un effet sur la conception du format. Il peut être demandé à l'auteur l'expansion ou la compression du texte selon le cas ‹37›.

(c) La mise en page: le texte est découpé en pages selon la conception initiale et selon les commandes d'impression insérées dans le document. Si des illustrations doivent être insérées, elles le sont dans cette étape.
L'outil le plus utile à ce stade est l'affichage rapide (ex. sur un écran). On peut trouver des systèmes qui donnent la possibilité de voir plusieurs pages à la fois sur l'écran (avec une échelle réduite). Ceci aide à voir l'image finale des pages les unes à côté des autres, ce qui est utile dans la conception de brochures.

(d) La composition: c'est dans cette étape que le document est réalisé matériellement. Le document est composé et préparé pour l'impression. Ceci se fait à l'aide d'une photocomposeuse ou d'un appareil de cette classe qui tiendra compte des types et des tailles des caractères voulus dans chaque partie. Dans plusieurs cas, cette étape est omise car le document est normalement imprimé à l'aide d'une imprimante rapide.

E.2- LE LANGAGE DE FORMATTAGE

L'interface entre l'utilisateur et le système est le "répertoire" de commandes permettant à l'utilisateur de décrire au système le formattage qu'il désire. Chacune de ces **commandes** évoque l'exécution d'une **fonction**. Citons quelques caractéristiques d'un langage de formattage:

a- L'ensemble des fonctions de formattage: les fonctions de formattage sont nombreuses et leur puissance varie selon l'ordinateur utilisé. On peut diviser ces fonctions en deux catégories:
-les fonctions qui décrivent les supports physiques sur lequel le document sort (ex. fonctions spécifiant les dimensions d'une page, ses marges, sa subdivision en deux colonnes)
-les fonctions qui concernent le formattage du texte qui remplit ces espaces définis sur le support (ex. le centrage du texte, la police d'impression de chaque caractère, l'interlignement, l'espace entre les mots, etc.).
Les entités traitées par les fonctions de formattage vont de l'entité physique (ex. une ligne physique, une fenêtre sur papier) aux entités logiques (ex. un caractère, un mot, un paragraphe). Les systèmes puissants donnent la possibilité d'édition des entités représentant la hiérarchisation logique d'un document (ex. NLS donne la possibilité d'avoir le même format pour les entités d'un même niveau hiérarchique ‹4,5›).
Un système qui offre une longue liste de fonctions simples de formattage et qui demande de nombreux efforts pour être compris et utilisé est appelé "Système Procédural" (ex. pour chaque titre dans un document l'utilisateur va définir une commande de saut de ligne, insère les codes de début et fin soulignement, etc.). Par contre un système qui offre une

liste courte de fonctions évoluées de formattage es
appelé "Système Déclaratif" (ex. l'utilisateur peu
simplement marquer "TITRE" et le système saute automa
tiquement une ligne vide et souligne les caractère
composant ce titre).

b- L'affichage du résultat du formattage: il y a deu
types d'affichage possibles
-1- l'utilisateur écrit dans le document les fonction
de formattage, qui seront exécutées plus tard (quan
l'utilisateur fait une demande de formattage), ce qu
implique que l'affichage de leur effet aura lieu dan
cette étape. Un inconvénient de cette technique es
que l'utilisateur ne voit pas immédiatement l'effet d
ses commandes de formattage mais, en même temps, s'i
veut changer le formattage, il n'a qu'à effectuer de
opérations d'édition sur ces commandes qui son
considérées comme du texte au regard de l'éditeur.
-2- Une autre technique consiste à permettre
l'utilisateur de désigner ses commandes de formattage
et l'effet de ces commandes est par contre affich
immédiatement. L'inconvénient de cette technique es
la nécessité d'un ensemble de commandes inverses pou
annuler l'effet d'une commande: un autre inconvénien
est que cette exécution monopolise l'unité centrale
ce qui retarde la réponse aux autres utilisateurs dan
un système multi-postes.

c- L'outil utilisé pour l'affichage du formattage: O
peut trouver des systèmes qui utilisent un écra
simple de 24 lignes pour afficher une partie d'un
page formattée; on trouve aussi des systèmes qu
affichent sur un écran "pleine page", ou plusieur
pages en échelle réduite en même temps.

E.3- L'EDITION EN COURS DE FORMATTAGE

Il existe plusieurs classes de systèmes bornées par les deux suivantes:

-la classe "BATCH": le système commence par la première page d'un document et continue jusqu'à la fin en transformant ce document (suivant les commandes de formattage incorporées dans son contenu) en un autre document formatté et prêt à sortir sur un support physique.

-La classe "INTERACTIVE": le système permet le formattage d'une partie du document, par exemple une page, puis l'affichage de cette partie formattée. A ce moment là l'édition peut avoir lieu. La page éditée est ensuite reformattée et ainsi de suite jusqu'à ce que l'utilisateur passe à la manipulation d'une autre partie du document.

F_ LES MOYENS D'ILLUSTRATION

Les illustrations peuvent être des schémas, des organigrammes, des dessins, ou tout autre moyen de représentation visuelle inclus dans un document. La création d'une illustration peut se faire à n'importe quel moment durant les phases de production d'un document, avant la reproduction de celui-ci.

On peut avoir des systèmes qui ne traitent que les documents contenant uniquement des données textuelles (avec ou sans la possibilité de sortir des textes en polices de caractères différentes). Dans ce cas, les illustrations sont faites séparément et collées à l'endroit vide prévu dans le document imprimé. Un système complet inclut la mémorisation numérique du texte plus les graphiques en noir et blanc ou même les

graphiques en tons de gris. Il peut afficher les deux types d'images sur le poste de travail et/ou l'imprimer sur son imprimante ou photocomposeuse. Ainsi, tout le document est stocké en forme numérique et il peut donc être transféré électroniquemnt d'une station de travail à l'autre, archivé sur support magnétique et sorti -en entier- sur demande sous le contrôle de l'ordinateur.

G- L'IMPRESSION ET LA REPRODUCTION

La reproduction d'un document est la phase dans laquelle plusieurs copies sont faites. Elle comprend l'impression, le tirage, la reliure, etc.

Les systèmes puissants dans le domaine de la reproduction permettent l'utilisation d'outils d'impression de très haute qualité (photocomposeuses, imprimantes xérographiques, etc.). Ils permettent aussi de définir le nombre de copies originales requises (ex. pour les copies de contrats).

Quelques systèmes permettent la sortie du document sur microfilm, ce qui est très pratique pour le stockage et la distribution, quoique très coûteux.

H- LA DISTRIBUTION DU DOCUMENT

La distribution est le processus qui consiste à donner des copies d'un document disponibles aux lecteurs et/ou au système de recueil d'information.

Dans les systèmes manuels actuels, une pile de papier est transmise physiquement à un certain nombre de personnes concernées. Le problème essentiel de cette méthode est son

impossibilité d'intégration avec les systèmes informatisés de recueil d'information.

Dans les systèmes informatisés, la distribution se fait électroniquement en créant ou rendant accessible un fichier contenant le document diffusé aux personnes concernées identifiées par des codes quelconques (messagerie).

Avec la distribution se termine le cycle de base de la production d'un document. Mais ce cycle peut se répéter. Le document distribué peut servir de référence à d'autres documents et ainsi de suite.

Au cours des chapitres suivants, nous abordons les divers problèmes rencontrés lors de l'informatisation de quelques phases essentielles de la production d'un document de type "textuel". Ces phases sont: l'édition et le formattage (traitées dans les chapitres I et II) puis la distribution (traitée dans les chapitres III et IV).

Quelques propositions sont développées et réalisées pour prouver leur faisabilité. Les idées principales portent sur:

1- la définition d'un noyau d'un système extensible de traitement de textes (édition et formattage).

2- la définition d'une représentation des documents transmis sur un réseau informatique reliant des systèmes de traitement de textes hétérogènes, cette définition étant basée sur le langage modulaire de traitement de textes développé dans la première partie.

CHAPITRE I

Extraction Et Conception Du Noyau D'un Système Extensible De Traitement De Textes

I.1. Introduction et présentation du problème

I.2. La modularité
- I.2.1. Les systèmes modulaires
- I.2.2. Les langages modulaires
- I.2.3. Les modules d'un système de traitement de textes
- I.2.4. La structure modulaire proposée

I.3. Les notions de primitives et de noyau

I.4. La démarche suivie pour extraire le noyau

I.5. L'éditeur
- I.5.1. Les fonctions d'édition
- I.5.2. Démarche suivie pour l'extraction des primitives d'édition
- I.5.3. Les primitives d'édition
- I.5.4. Le langage d'édition

I.6. Le formatteur
- I.6.1. Les fonctions de formattage
- I.6.2. Démarche suivie pour l'extraction des primitives de formattage
- I.6.3. Les primitives de formattage
- I.6.4. Le langage de formattage

I.7. Conclusion

CHAPITRE -I-

EXTRACTION ET CONCEPTION DU NOYAU D'UN SYSTEME EXTENSIBLE DE TRAITEMENT DE TEXTES

I.1. INTRODUCTION ET PRESENTATION DU PROBLEME

Nous avons remarqué que la plupart des systèmes de traitement de textes (STT) commercialisés ne sont ni flexibles ni extensibles. Il semble qu'ils aient été développés pour répondre à certains besoins du marché sans qu'on ait donné beaucoup d'importance à la rationalisation du choix des fonctions incorporées selon des critères reposant sur les relations entre ces fonctions.

Le but principal que nous avons recherché est de concevoir un STT de base, c'est-à-dire de bas de gamme, tout en ayant la possibilité d'étendre ce système -sans avoir à le modifier- pour répondre exactement aux besoins des divers utilisateurs. Nous appelons ce système: le "noyau". Pour réaliser ce but, nous avons introduit le concept de "fonction primitive".

L'extension du système doit être très simple et faisable aussi bien par le constructeur du système que par l'utilisateur non-informaticien.

Nous avons pensé qu'une structure modulaire du système et du langage simplifiait énormément la conception et rendait le système réalisable.

Au cours de ce chapitre, nous présentons la philosophie et la méthodologie de conception proposées pour réaliser le noyau de ce système extensible de traitement de textes. Les détails de réalisation du prototype sont donnés au chapitre suivant.

I.2. LA MODULARITE

I.2.1. Les Systèmes Modulaires

Un système modulaire est un système construit à partir d'u nombre de parties logiquement indépendantes nommées "module ‹35,48,49›.

Parmi ces modules, nous pouvons définir des modules "essentiels" et des modules "optionnels": le système pe fonctionner sans l'existence des modules optionnels mais p contre, l'absence d'un module essentiel rend le syst incapable d'exécuter ses fonctions principales.

Les systèmes modulaires ont plusieurs avantages ‹48›:

1. Ils sont simples à concevoir logiquement (désignant un module pour chaque tâche logiqueme indépendante).

2. Ils sont simples à développer car chaque modu peut être conçu indépendamment des autres, chaq module ne se souciant que du "protocole" d'échange d données avec les autres.

3. La documentation est plus simple et, généralemen plus lisible.

4. La maintenance est facilitée.

5. L'extension du système se réduit normalement l'adjonction de nouveaux modules.

6. Ils sont adaptables aux besoins de l'utilisat par adjonction ou suppression de modules optionnel

7. Le temps nécessaire au développement du système réduit par le fait que plusieurs groupes peuv travailler simultanément sur des modules différent

I.2.2. Les Langages Modulaires

Etant en accord avec la phrase de Hammer ‹27›: "Un langage qui convient à une variété de problèmes est également non-pertinent pour tous ces problèmes",nous croyons à la nécessité de disposer d'une variété de langages spécialisés, chacun pour un domaine d'application.

Nous nous sommes donc efforcés de développer un langage de traitement de textes convenant le mieux possible à cette application.

Pour faciliter l'extensibilité du système, nous avons choisi de développer des langages modulaires pour l'édition et le formattage. Nous avons suivi les principes des langages modulaires décrits dans ‹14›: un système informatique est modulaire si le niveau linguistique répond à ces conditions:

1. Une classe d'objets est associée au niveau linguistique

2. Ces objets sont des modules de programmes. Chaque module effectue un changement dans l'état des données (il exécute une certaine fonction).

3. Le niveau linguistique doit fournir un moyen pour combiner des modules et construire ainsi d'autres modules plus sophistiqués sans avoir à modifier aucun des modules utilisés.

4. Le sens d'un module doit être indépendant du contexte dans lequel il est utilisé.

I.2.3. Les Modules d'un Système de Traitement de Textes

Le système est conçu comme un ensemble de modules coopérant pour exécuter une certaine tâche. Les modules essentiels sont:

1. Appel de et retour à l'environnement du système
2. Gestion de fichiers
3. Gestion des entrées/sorties
4. Editeur
5. Formatteur

Notre étude ne concerne que l'éditeur et le formatteur qui sont les modules les plus caractéristiques d'une application de traitement de textes.

Des modules optionnels peuvent être incorporés dans le système pour faciliter des applications spécifiques comme les lettres personnalisées automatiques, le courrier électronique etc.

I.2.4. La Structure Modulaire Proposée

La notion de modularité a été poussée encore plus loin, chacun des modules cités ci-dessus, à savoir l'éditeur et le formatteur, aura à son tour une structure modulaire.

Chaque sous-module représente une fonction de traitement de textes et est responsable de son interprétation.

I.3. LES NOTIONS DE PRIMITIVES ET DE NOYAU

Nous appelons "ensemble de fonctions primitives", tout ensemble de fonctions réalisant les deux conditions suivantes:

1. **L'ensemble est minimal:** il ne contient que les fonctions qui ne peuvent pas être réalisées à partir d'une combinaison d'autres fonctions parmi celles qui se trouvent dans ce même ensemble.
2. **Cet ensemble est complet:** n'importe quelle fonction existant sur un système puissant de traitement de textes est réalisable à partir d'une combinaison d'un certain nombre de fonctions choisies dans cet ensemble.

Un tel ensemble constitue le "noyau" du système car on peut construire un système de n'importe quelle taille et puissance à partir des fonctions de ce noyau et des "opérateurs" suivants:

1- la séquence: on a la possibilité de spécifier une séquence linéaire de fonctions (encore appellée "macro") qui pourra être ré-utilisée tout au long du document.

2- la répétition: on a également la possibilité de demander la répétition d'une fonction (ou d'une macro) donnée.

I.4. LA DEMARCHE SUIVIE POUR EXTRAIRE LE NOYAU

Nous avons commencé par étudier plusieurs systèmes de traitement de textes de puissances différentes: IBM ‹26,30›, Vydec ‹65›, DEC ‹16›, Olivetti ‹47›, Xerox, Hermès, Secré, Philips, Wang, Logica, Computext, Siemens, etc.

A partir de cette étude, nous avons pu extraire un ensemble quasi-exhaustif des différentes fonctions d'édition et de formattage. Nous avons ainsi obtenu deux listes. Nous devions ensuite définir le noyau de chacune d'elles. Ceci a été fait en décomposant chaque fonction en sous-fonctions plus élémentaires et ainsi de suite de manière répétitive. A un moment donné, nous avons obtenu des fonctions qui ne pouvaint plus se décomposer sans perdre leur sens de traitement de textes. Il faut aussi préciser qu'il y avait parfois plusieurs décompositions possibles de chaque fonction et qu'il a fallu explorer toutes les possibilités pour élargir le choix des fonctions élémentaires (voir I.5.2). Un choix particulier parmi ces fonctions élémentaires a été fait pour assurer la "mimimalité" et la "complétude" de l'ensemble choisi. Ces fonctions élémentaires sont donc les primitives, et leur ensemble est le noyau.

Ensuite, nous avons développé le noyau en écrivant d modules dont chacun exécute une des primitives. Il est évid que ce noyau occupe moins d'espace mémoire qu'un interpréte réalisant directement toutes les fonctions. Par contre, il fa s'attendre à une exécution plus lente des fonctions comple puisqu'elles ne sont plus exécutées par des modules spécifiq optimisés. Mais nous estimons que la perte de temps envisag est infime et non perceptible par l'utilisateur.

Nous avons enfin pris en compte les consideratio suivantes:

1. Les fonctions doivent affecter une unité sign ficative minimale du texte, afin que le module inte prétant cette fonction puisse être utilisé sa modifications pour exécuter la même fonction sur u partie quelconque du texte par simple répétition.
Par exemple, l'unité minimale pour la fonction centrage horizontal est la ligne physique. Si no voulons centrer "n" lignes consécutives, le syst répètera automatiquement l'exécution du module de ba "n" fois sur les "n" lignes en question.

2. Une technique d'adressage relatif est utilisée po désigner la partie du texte qui sera affectée par u fonction donnée. Nous prenons en considération q l'utilisateur, en commandant quelques fonctions, pourra pas prévoir la position exacte (après forma tage) des chaînes de caractères qui devront ét affectées par ces fonctions.
Par exemple, si nous voulons laisser 4 espaces blan avant le début de la première ligne d'un paragraphe puis laisser 7 espaces blancs avant toutes les lign qui suivent, on ne pourra savoir où se terminera première ligne sur le papier qu'après le découpage d lignes par le formatteur.

I.5. L'EDITEUR

I.5.1. Les Fonctions Classiques D'édition

Nous avons recencé dans les systèmes existants les catégories suivantes de fonctions d'édition:

1. éliminer une chaîne de caractères
2. insérer une chaîne de carctères
3. écrire sur une chaîne déjà existante
4. intervertir deux chaînes de caractères
5. repositionner une chaîne de caractères à un autre endroit dans le texte
6. sauvegarder en mémoire une chaîne de caractères sans l'effacer de sa place dans le texte
7. insérer dans le même texte ou dans un autre texte une chaîne sauvegardée en mémoire
8. créer d'un nouveau texte à partir de parties pré-enregistrées
9. insérer des champs variables dans un texte pré-enregistré
10. changer des majuscules en minuscules ou vice-versa
11. rechercher automatiquement et sélectionner une chaîne de caractères donnée
12. remplacer une chaîne par une autre
13. compacter un texte

En fait, le nombre de fonctions est beaucoup plus grand car, par exemple, dans la catégorie insertion, on trouve les fonctions insertion de caractères, de lignes, de chaînes, etc. De même la substitution peut se faire pour la première occurrence de la chaîne recherchée, pour toutes les occurrences qui suivent ou dans tout le texte.

I.5.2. Extraction des Primitives D'édition

Nous avons suivi la démarche décrite dans -I.4.-, et nous avons trouvé plusieurs décompositions possibles pour quelques fonctions:

FONCTION	DECOMPOSITION
. Eliminer une chaine de caractères	Non-décomposable
. Insérer une chaine	Non-décomposable
. Sauvegarder en mémoire une chaine de caractères (sans l'effacer de sa place)	Non-décomposable
. Insérer une chaine sauvegardée	Non-décomposable
. Rechercher automatiquement une chaine de caractères	Non-décomposable
. Ecrire sur une chaine déjà existante	1. *Sauvegarder (la nouvelle chaine) *Transférer les caractères composant cette chaine, en une seule fois, aux octets de mémoire qui contiennent les caractères de la chaine à remplacer 2. *Eliminer la chaine (à remplacer) *Insérer (la nouvelle chaine)
. Repositionner une chaine de caractères dans un autre endroit dans le texte	1. *La chaine est transférée directement (sans sauvegarde vers sa nouvelle position *Eliminer (la chaine de sa position originale) 2. *Sauvegarder (la chaine à repositionner) *Eliminer (la chaine de sa position originale) *Insérer (la chaine sauvegardée dans la nouvelle position)

.Intervertir deux chaînes de caractères	1. *Repositionner la chaîne "a" à côté de la chaîne "b" (avant ou après). Automatiquement, cette fonction élimine la chaîne "a" de sa place originale (voir la fonction précédente). *Repositionner la chaîne "b" à l'endroit originale de la chaîne "a". "b" sera automatiquement éliminée de sa place originale
	2. *Sauvegarder la chaîne "a" *Eliminer "a" de sa position originale *Insérer la chaîne sauvegardée "a" à côté de la chaîne "b" *Sauvegarder la chaîne "b" *Eliminer "b" de sa position originale *Insérer la chaîne sauvegardée "b" à l'endroit original de l'autre chaîne
. Création d'un nouveau texte à partir de parties pré-enregistrées	1. *Sauvegarder la partie pré-enregistrée *Insérer la chaîne sauvegardée dans sa nouvelle position Ces deux étapes seront répétées pour chaque partie qui va composer le nouveau texte
. Insertion de champs variables dans un texte pré-enregistré	1. *Des codes spéciaux sont insérés dans un texte, représentant les endroits des champs variables dans ce texte *L'utilisateur écrit une chaîne qui contient toutes les variables nécessaires, séparées par des délimiteurs *Le système insère automatiquement chaque élément de la chaîne à sa place dans le texte
	2. *Le système (sur une commande spéciale) cherche la première occurrence d'un champ à remplir dans le texte, et il pointe vers cet endroit *L'utilisateur insère directement la chaîne qu'il veut insérer dans ce champ

	3. *Chercher l'endroit d'insertion *Insérer la variable dan cet endroit Ces deux étapes seront répétées pour chaque insertion d'un champ variable.
. Changement des majuscules en minuscules ou vice versa	1. *Le système effectue le changement automatique-ment sur la chaine délimitée par deux code spéciaux
	2. *Eliminer (par exemple lettre en majuscule) *Insérer (la minuscule de cette lettre)
. Remplacer une chaine par une autre	1. *Transférer directement en mémoire la nouvelle chaine, là où la chaine remplacer est enregistr *Si la nouvelle chaine est plus courte que l'ancienne les derniers caractères de celle-ci sont éliminés.

	Si la nouvelle chaine est plus longue, ses derniers caractères sont insérés dans le texte suivant la partie de la chaine déjà insérée
	2. *Sauvegarder la nouvelle chaine *Eliminer l'ancienne chaine *Insérer la chaine sauvegardée à l'endroit de l'ancienne chaine
. Compacter un texte	Répéter: 1. *Chercher un blanc *Eliminer les blancs non nécessaires qui précèdent chaque mot dans un texte

I.5.3. Les Primitives D'édition

De cette décomposition des fonctions, nous déduisons que cinq fonctions sont non-décomposables et que ces mêmes fonctions peuvent être utilisées pour réaliser toutes les autres (la décomposition utilisant ces fonctions a été citée en dernier lieu pour chaque fonction dans la table précédente).

Les fonctions "primitives" d'édition retenues sont donc:

1. Eliminer
2. Insérer
3. Sauvegarder
4. Insérer une chaine sauvegardée
5. Rechercher automatiquement et sélectionner une chaine de caractères

I.5.4. Le Langage D'édition

Les commandes du langage d'édition sont présentées (détail dans la section.3. de l'Annexe A. Elles se divisent (trois catégories principales:

1- Commandes de gestion de l'écran (voir 3.2.)

2- Commandes de recherche d'une chaine dans un texte (vo 3.3.)

3- Commandes de modification du texte (voir 3.4.)

Les commandes des catégories 2 et 3 sont celles q représentent les cinq fonctions primitives d'édition. Tous l modules exécutant ces cinq fonctions ont besoin de de paramètres

1- Le premier détermine le début de la partie du tex affectée: c'est toujours la dernière position du curs précédant la frappe de la commande. Ce paramètre n'a paraitra donc jamais dans les commandes.

2- Le deuxième paramètre a une signification qui dépend la fonction; il peut être:

* une valeur (une chaine): c'est le cas des command CHERCHER et INSERER
* une limite de l'effet de la fonction: c'est le c des commandes ELIMINER et SAUVEGARDER. Cette limi peut être désignée par:
  - une chaine, ou
  - le nombre de caractères à manipuler
* l'adresse d'une chaine comme dans le cas de commande INSERER UNE CHAINE SAUVEGARDEE. Cette a resse étant toujours fixe, le deuxième paramèt n'apparait donc pas explicitement dans la commande.

Des exemples de commandes:

?E5C = Eliminer 5 caractères

?E.R = Eliminer jusqu'à la fin du paragraphe (.R étant le délimiteur de fin de paragraph voir la liste complète dans 4.2 de l'Annexe

?EXY = Eliminer jusqu'à la première occurrence de chaine "XY" (incluse).

I.6. LE FORMATTEUR

I.6.1. Les Fonctions De Formattage

Les principales classes de fonctions de formattage utilisées fréquemment dans la production de documents sont les suivantes:

1- Décomposition d'un texte en lignes de longueur définie

2- Définition de la marge gauche

3- Définition de la marge droite

4- Changement de ligne

5- Définition du début d'une zone d'ajustement.
La zone d'ajustement est utilisée si le formatteur décompose le texte en lignes au cours de sa frappe. Si une ligne dépasse le début d'une zone définie, l'utilisateur doit choisir une fin de mot ou une coupure de mot.

6- Justification à droite sans justification ordinaire à gauche.
La chaine affectée par cette fonction apparait avec son dernier caractère sur la marge droite. Le premier caractère de la chaine sera quelque part dans la ligne. Cette fonction peut être utilisée, par exemple, dans la numérotation d'équations mathématiques.

7- Justification à droite et à gauche
Alignement d'une ligne sur la marge droite pa
l'insertion d'espaces entre les mots composant cett
ligne.

8- Alignement d'un code donné dans une ligne sur un
certaine colonne.
Exemple: pour un rapport financier on aligne l
virgule sur une certaine colonne.

9- Impression en retrait (décalage de la marge gauche

10- "Wordrapping" d'une ligne
Ex. supposons qu'un texte soit formatté sur des ligne
de 80 caractères mais qu'on le fasse sortir sur de
lignes de 60 caractères seulement. S'il y a "word
wrapping", chaque ligne formattée sortira sur deu
lignes physiques, la deuxième étant décalée à droit
pour indiquer ce "wordwrapping".

11- Définition du point de centrage

12- Centrage horizontal

13- Centrage vertical

14- Soulignement automatique

15- Compression d'une ligne par l'élimination de tou
les espaces non-nécessaires

16- Double espacement (sur un paragraphe, une pag
etc.)

17- Numérotation automatique des pages

18- Répétition automatique de quelques chaines spéciales (entête, date, etc.) dans chaque page

19- Définition d'une zone réservée à l'entête (ou l'en bas) d'une page

20- Insertion de lignes vides avant une ligne donnée

21- Pagination automatique par définition d'un nombre de lignes par page ou l'insertion d'un code changement de page

22- Pagination conditionnelle selon le nombre de lignes restantes dans la page.
Exemple: L'utilisateur met un code spécial associé à une partie du texte (ex. un paragraphe ou un nombre de lignes vides réservées pour une figure). Si cette partie du texte se retrouve, pendant le formattage, en-bas de la page, le système décide automatiquement soit de la mettre dans cette page sont de la transférer entièrement dans la page suivante.

23- Renvois indexés
Ces renvois peuvent être créés en délimitant la chaine de caractères représentant chaque renvoi par les délimiteurs appropriés; en même temps, l'utilisateur doit déclarer le point où cette chaine est référencée dans le texte (par exemple par l'insertion d'un code spécial en ce point)

24- Définition des points de tabulation

25- Tabulation sur colonnes de plusieurs lignes (ex. table de comparaison)

26- Pages à plusieurs colonnes (comme dans le articles des revues)

27- Elimination des "lignes veuves"
Si la coupure d'une page se fait avant un très peti nombre de lignes qui pourraient être insérées en ba de la page précédente, le système supprime la pag insuffisamment remplie, et insère ces quelques ligne en bas de la page précédente.

28- Définition d'espace fixe
Il s'agit d'un espace qui ne peut pas être étend pendant la justification, par exemple celui qui es utilisé dans certaines expressions composées de plu sieurs mots qui ne doivent pas être séparés.

29- Texte Nul
Si une chaine du texte est entrée et délimitée par le codes de cette fonction, cette chaine n'apparait pas l'utilisateur pendant la sortie du texte.
Cette fonction est utile pour délimiter la partie d texte qui représente une entrée dans une autre parti du document (par exemple, une référence dans l bibliographie du document). Le système chainera cett partie du texte dans une liste spéciale et la traiter en temps approprié.

30- Texte Libre
Une chaine de caractères insérée dans un texte affectée par cette fonction sortira sans aucu modification après le formattage du texte. Cet fonction peut être utile pour certaines chaines format spécial (ex. expressions mathématiques chimiques)

I.6.2. Extraction des primitives de formattage

Nous avons pris en compte le fait que quelques fonctions sont très coplexes et peuvent être réalisées d'une manière semi-automatique en utilisant des fonctions simples. Ceci est fait pour améliorer la décomposabilité des fonctions sans avoir à définir des fonctions qui ne sont pas directement liées au traitement de textes.

FONCTION	Décomposition
- Définition de la marge gauche	* Non-décomposable
- Définition de la marge droite	* Non-décomposable
- Définition d'une zone d'entête	* Non-décomposable
- Définition des points de tabulation	*Non-décomposable
- Texte Nul	*Non-décomposable

- Texte libre	*Non-décomposable
- Découpage d'un texte en lignes de longueur fixe	*Non-décomposable
- Insertion de "N" lignes vides	*Répéter N fois: insertion d'une ligne vide
- Centrage horizontal d'une ligne entre deux limites	*Non-décomposable
- Alignement d'un code sur une colonne	*Non- décomposable
- Justification à droite	*Non-décomposable
- Compression d'une chaîne	*Non-décomposable
- Justification à droite sans justification à gauche	*Non-décomposable
- Numérotation automatique des pages	*Non-décomposable
- Pagination automatique	*Non-décomposable
- Soulignement automatique	*Non-décomposable
- changement de ligne	*Non-décomposable . Cette commande peut être associée à la fonction de décomposition du texte en lignes. Un code spécial (ex. R.C.) permet le passage forcé à une nouvelle ligne.

- Définition du début d'une zone d'ajustement ET - Définition d'espaces fixes	* Décomposition d'un text en lignes selon une longu définie * En voyant cette coupure de ligne l'utilisateur pe effectuer le changement qu veut pour obtenir la coup de la ligne comme il veut (par exemple, en utilisan l'éditeur)
- Impression en retrait	* Définition d'une marge gauche décalée pour la zo mise en évidence * Redéfinition de la marg gauche à la fin de la zon mise en évidence
- "Wordwrapping" d'une ligne	* Définir une marge gauc "x" pour la première lign * Définir une marge gauch "x+n" pour les lignes qui suivent la première ligne * Redéfinir la marge gau "x".
- Définition du point de centrage	* Définir une marge gauc et une marge droite, qui ont ce point au centre. * Centrage horizontal de la ligne concernée
- Centrage vertical	Une solution semi-automatique est toujours possible: * Le formatteur décompose le texte en lignes. *le texte formaté est montré à l'utilisateur, qui peut compter le nombre de lignes qu'il doit insérer pour centrer le texte verticalement * L'utilisateur introduit la commande qui insère le nombre voulu de lignes vides avant la première ligne du texte à centrer.
- Double espacement (sur un paragraphe, une page, etc.)	* Insertion automatique de lignes vides (dans ce cas= une ligne) avant chaque ligne de la partie du texte à double espacement.
- Pagination conditionnelle ET - Elimination de "lignes veuves"	Si cette fonction n'existe pas une solution semi-automatique est toujours possible *Le formatteur pagine automatiquement le texte * Quand les pages formattées sont montrées, l'utilisateur introduit la commande qui change le nombre de lignes par page pour arriver à la pagination souhaitée.

- Tabulation sur colonnes de plusieurs lignes	* Définir des points de tabulation * Insérer des codes spécia pour définir le début et l fin de chaque élément du texte représentant l'inter section d'une colonne et d'une rangée * Le système formatte chaq élément comme une partie d texte ordinaire avec toute les fonctions de formattag qui l'accompagnent. Mais l met obligatoirement deux fonctions de formattage pour chaque élément: i- Définition de la marg gauche ii- Définition de la mar droite. Ceci en accord avec la position de cet élément dans la tabl et la définition de points de tabulation * Le système range les élé- ments d'une même rangée de manière qu'ils commencent tous sur la même ligne horizontale.
Page à multi-colonnes	* Définir des points de tabulation * Un code spécial sera mis au début du texte qui doit être formatté sur plu- sieurs colonnes. * Le système formatte le texte avec toutes les fonctions de formattage qui l'accompagnent. Le texte dans chaque colonne a deux fonctions de formattage supplémentaires: i- Définition de la marge gauche ii- Définition de la marge droite. Ceci en accord avec les points de tabula- tion prédéfinis * Le texte, au cours du for- mattage, sera automatique- ment continué dans la colon- ne suivante de la même page, ou dans la première colonne de la page suivante.

- Répétition automatique de quelques chaines spéciales (entête, date, etc.) dans chaque page ET - Renvois indexés	. Texte Nul i- Les chaines de caractères affectées par ces fonctions so insérées dans le tex mais délimitées par des codes spéciaux. ii- Le formatteur manipulera ces chaines en temps approprié (par ex., une entête est mise, au début de chaque page) Il faut note que quand le texte est formatté, ces chaines apparaissent dans de nouveaux endroits et pas dans le texte comm elles ont été créées.

I.6.3. Les primitives de formattage

Nous nous apercevons que les différentes fonctions d formattage sont beaucoup plus distinctes entre elles que le fonctions d'édition. Nous pouvons cependant retenir les primitives de formattage suivantes:

1- Définition de la marge gauche

2_ Définition de la marge droite

3- Centrage horizontal d'une ligne entre deux limites

4_ Insertion automatique d'une ligne vide avant une ligne donnée

5_ Alignement d'un code donné d'une chaine sur une colonne définie

6_ Justification à droite

7_ Compression d'une chaine

8_ Texte justifié à droite sans justification ordinaire à gauche

9_ Numérotation automatique des pages

10_ Pagination automatique par définition d'un nombre de lignes par page ou l'insertion d'un code changement de page

11- Soulignement automatique

12_ Définition d'une zone réservée à l'en-tête (ou l'en-bas) d'une page

13_ Décomposition d'un texte en lignes de longueur définie ou par une commande de passage obligatoire à une nouvelle ligne

14_ Définition des points de tabulation

15_ Texte Nul

16- Texte Libre

I.6.4. Le langage de formattage

Les commandes du langage modulaire de formattage s
divisées en cinq catégories:

1. Les délimiteurs de chaines
   Ce sont les codes qui délimitent certaines chai
   spéciales, par exemple la fin d'une ligne ou d
   paragraphe, ou le début (ou la fin) d'une par
   soulignée, l'insertion d'une note en bas de la pa
   etc. (voir 4.2.1. en Annex A).

2. Les déclarateurs de format physique
   Ils sont utilisés pour donner des valeurs à quelq
   variables comme, par exemple, les marges gauche
   droite, la longueur de la page, etc. (voir 4.1 et 4.
   en Annexe A).

3. Codes pour la manipulation des tables
   Ce sont des codes qui définissent le début et la
   d'une table (et les éléments de cette table), ainsi
   la largeur de chaque colonne de cette table (voir 4.
   en Annexe A).

4. Définition des macros
   Nous utilisons pour le formattage le concept de ma
   utilisé dans l'édition

5. Commandes explicites de formattage
   Ces commandes invoquent les fonctions de formatt
   comme le centrage, la justification, la compressi
   etc. (voir 4.3.2 et 4.4 en Annexe A).

La description complète du langage que nous avons réalisé se trouve en section 4 de l'Annexe A. Voici un exemple de commande:

CENTRER X Y

X: adresse relative de la première ligne qui sera affectée par la commande
   Ce champ peut contenir:
   * un blanc ou zéro = la fonction sera appliquée à la ligne qui suit immédiatement la commande
   OU
   * un entier positif N = la fonction sera appliquée à partir de la (N+1)ème ligne suivante

Y: définit la limite de l'effet de la fonction
   Ce champ peut contenir:
   * un blanc ou "1" = la fonction ne s'appliquera qu'à une seule ligne
   OU
   * un entier positif N = N lignes seront affectées
   OU
   * un code spécial indiquant que la fonction sera appliquée jusqu'à la fin d'un paragraphe ou du texte.

I.7. Conclusion

Dans ce chapitre, nous avons montré la méthode que nous avons suivie pour extraire le noyau d'un système extensible de traitement de textes.

Nous abordons dans le chapitre suivant les aspects de la réalisation expérimentale de ce noyau.

CHAPITRE II

Realisation Du Noyau D'un Système Extensible De Traitement De Textes

II.1. Le matériel utilisé

II.2. Le logiciel utilisé

II.3. La structure de données

II.4. Les modules du système

II.5. Extension du système par le constructeur

II.6. Extension du système par l'utilisateur

II.7. Exemple

CHAPITRE -II-

REALISATION DU NOYAU D'UN SYSTEME EXTENSIBLE DE TRAITEMENT DE TEXTES

Dans ce chapitre, nous décrivons la réalisation du noyau d'un système flexible de traitement de textes. Nous commençons par la description des outils matériels et logiciels desquels nous disposons. Puis nous décrivons brièvement le système que nous avons réalisé. Une description détaillée de ce système, de ses commandes et de son utilisation est donnée dans l'Annexe A.

II.1. LE MATERIEL UTILISE

Le système sur lequel nous avons développé notre logiciel de traitement de textes est le MICRO-1 de Plessey. C'est une machine basée sur le microprocesseur LSI-11. Elle a le répertoire puissant des instructions du mini-ordinateur PDP 11/35 de DEC. Ce répertoire contient plus de 400 instructions.

Il n'y a pas d'instructions séparées pour l'accès à la mémoire, aux portes d'E/S, ou aux accumulateurs. Ceci donne une flexibilité de programmation surtout en ce concerne la manipulation des registres des périphériques.

Nous utilisons comme console l'écran Hazeltine 1500 (sur lequel est basé notre éditeur). Nous avons également utilisé une imprimante à marguerite Diablo 1640. La mémoire externe est composée de 2 unités de disques, dont un est fixe et l'autre amovible. Sa taille globale est d'environ 20 M octets.

La configuration générale du matériel complet est montrée dans la figure II-1-.

II.2. LE LOGICIEL UTILISE

Le système d'exploitation de la machine utilisée est RSX-11M conçu initialement comme un système de multiprogrammation en temps réel pour la gamme PDP-11. Ce système sert plusieurs langages dont MACRO-11 (langage d'assemblage), PASCAL, BASIC, FORTRAN IV, et COBOL.

La taille mémoire du système que nous avons utilisé est de 28K mots de 16 bits chacun. Une option matérielle KT11 peut donner la possibilité d'adressage virtuel des tâches.

Notre logiciel de traitement de textes a été développé en langage d'assemblage MACRO-11M pour optimiser la taille du code et pouvoir utiliser toutes les routines du système. En outre, le langage d'assemblage permet de mieux gérer les ressources matérielles du système. Un inconvénient de notre choix est l'impossibilité de transporter notre logiciel sur une autre machine n'ayant pas un assembleur de MACRO-11M.

Pour le traitement des fichiers, nous utilisons le système logiciel FILES-11. Il permet de manipuler des fichiers stockés sur des unités d'enregistrement magnétique (dans notre cas, disques à tête fixes). On appelle ces unités les "volumes". Les volumes doivent être initialisés pour être reconnus par le système. Des commandes MCR existent pour cet effet.

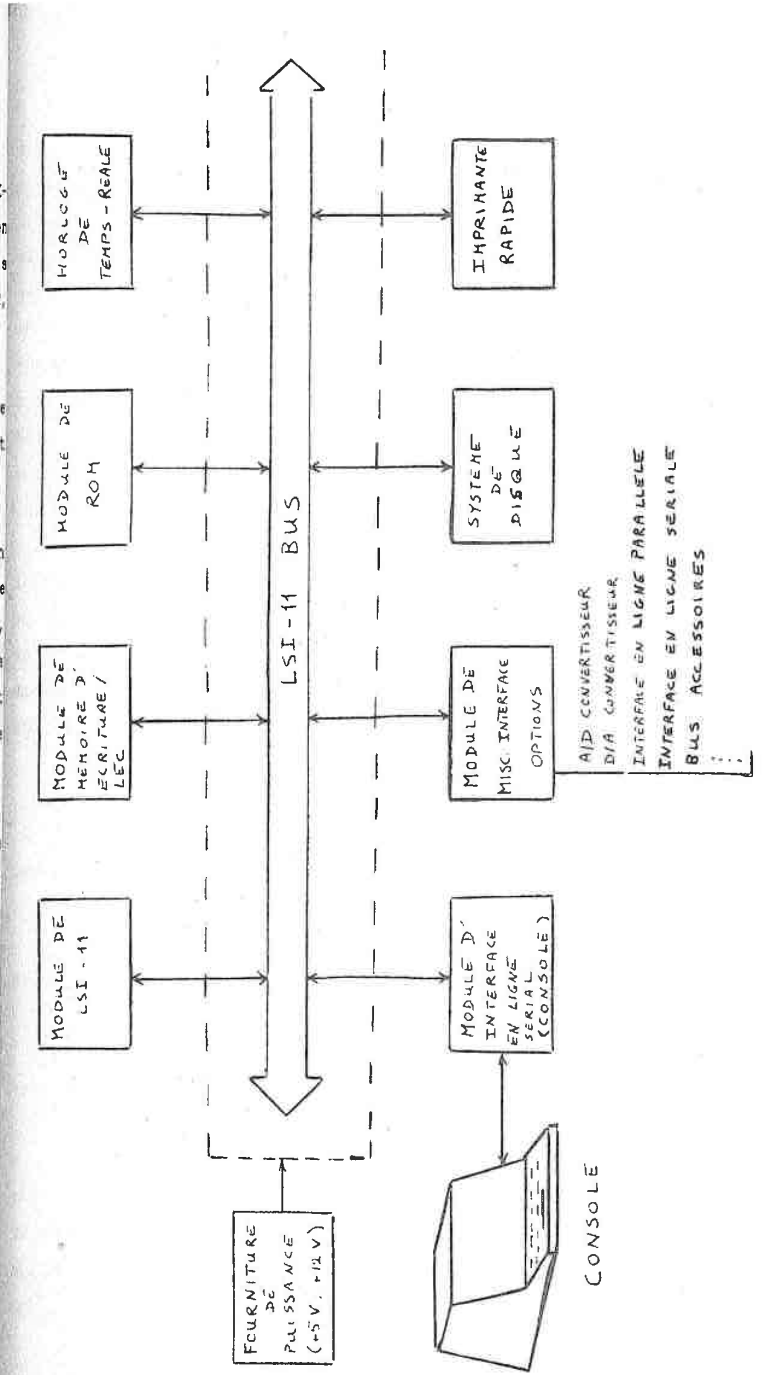


Fig.II.1. Configuration Du Système

II.3. LA STRUCTURE DES DONNEES

II.3.1. L'organisation et le stockage des données communes

Dans un système de traitement de textes, une partie de l mémoire vive doit contenir une partie du fichier texte qui ser traité par les différents modules du système. La taille d cette partie du fichier varie selon la capacité de la mémoir disponible ainsi que le type et la fréquence des opération faites sur ce texte. En général, le fichier texte est divis logiquement en pages et une ou plusieurs pages résident en mêm temps dans la mémoire principale.

Dans notre système nous avons deux fichiers possibles pou chaque texte. L'un d'eux contient le texte tel qu'il a été tap par l'opérateur avec toutes les commandes de formattag incorporées. L'autre fichier contient le texte formatté comm il sera imprimé sur le papier. Les avantages cette méthod sont:

a) L'espace occupé dans la mémoire externe est en généra minimal puisqu'il n'y aura pas de lignes vides, d'espace réservés pour les figures, de paragraphes en retrait, etc.

b) Les commandes de formattage étant stockées, il ser possible de les modifier à tout moment.

c) Le logiciel est simple puisque le formattage ne se fer pas en même temps de l'édition.

Mais cette méthode de stockage a l'inconvénient d'élimine toute correspondance entre l'image du texte stocké et celle d texte imprimé. Si l'utilisateur veut modifier une parti quelconque du texte ou d'une commande de formattage, il ser obligé de parcourir le texte pour trouver le mot voulu.

Ce problème a été résolu en donnant à l'utilisateur la facilité d'éditer son texte formatté juste après le formattage, page par page, en affichant sur l'écran la partie du texte non-formatté correspondant à la page formattée. Les commandes de formattage seront aussi affichées.

Un autre problème est qu'à chaque impression il faudra reformatter le texte entier. Pour éviter cela, nous avons donné à l'utilisateur la possibilité de stocker la version formattée de son texte. Ceci est recommandé pour les textes révisés et approuvés qui seront ensuite imprimés un bon nombre de fois.

Nous avons divisé le fichier texte non-formatté en pages de très petite taille afin de limiter la partie touchée par une panne quelconque du système.

Le texte est divisé en lignes. Il est stocké d'une façon contiguë. Le texte est donc frappé au kilomètre sans que l'utilisateur se soucie de la coupure des mots en fin de ligne. Le passage à une nouvelle ligne est commandé par un retour-chariot. Dans ce cas, un blanc est inséré immédiatement au début de la nouvelle ligne.

Chaque commande de formattage est insérée sur une ligne spéciale commençant au début de la ligne et se terminant par un retour chariot. La commande commence par le caractère ";".

Au milieu du texte sont insérés tous les codes spéciaux délimitant des chaines soulignées, des coupures obligatoires de pages, de paragraphes, etc. Ces codes commencent par un "." et sont stockés sur une ligne spéciale pour la clarté au moment de l'édition même si cette solution sacrifie un peu de place de la mémoire.

En ce qui concerne le texte formatté, la totalité de la page à imprimer est en mémoire. Ceci facilite la mise en page des textes à plusieurs colonnes ou des tables. De même, l'impression des pages à plusieurs colonnes est possible même avec des imprimantes qui ne peuvent pas reculer automatiquement en arrière.

En outre, un tampon est nécessaire pour la ligne à imprimer. Sa taille est le triple de la taille ordinaire d'une ligne pour permettre d'enregistrer les 3 codes pour un caractère souligné: le caractère, le retour à l'arrière d'un espace, le code du trait de soulignement.

II.3.2. L'espacement proportionnel

Pour tenir compte de la possibilité d'imprimer un texte espacé proportionnellement, une table de 2 colonnes est nécessaire. Dans cette table, le code ASCII d'un caractère sera enregistré dans une colonne tandis que dans l'autre colonne, et sur la même ligne, sera marquée la largeur de ce caractère. Cette largeur peut être soit une largeur réelle en fractions de pouce (ou de centimètre), soit un nombre relatif qui, pour obtenir la largeur réelle, sera multiplié par une constante.

Cette table sera utilisée par le formatteur pour réaliser les coupures de lignes selon la longueur de la ligne physique, de manière à justifier le texte à droite, ou d'éviter les coupures de mots et les débordements. Dans le cas de l'espacement proportionnel, le nombre de caractères n'a aucun sens, car c'est la largeur qui compte.

II.3.3. Les commandes de formattage

Afin de sauvegarder les fonctions de formattage qui sont effectives à un moment donné, 3 blocs de mémorisation existent:

a) STA = 10 fonctions de 3 mots chacune

b) STB = 20 fonctions de 3 mots chacune

c) RSTA = 5 fonctions de 3 mots chacune

La sauvegarde de ces fonctions aide à reformatter une seule page du texte après avoir effectué une opération quelconque d'édition sur cette page. L'influence des pages précédentes sur la page actuelle est résumée dans cette partie de la mémoire.

STA contient les fonctions qui affectent la ligne en cours de formattage. Toute commande de formattage trouvée avant une ligne de texte rentre d'abord dans cette table. Les commandes qui n'affectent que des lignes suivantes sont enregistrées dans STB. On cherche aussi dans STB si une des commandes affecte la ligne en cours pour la transférer à STA.

STB contient les fonctions qui affectent des lignes qui suivent la ligne en cours de formattage.

RSTA contient les commandes de formattage qui ne changent que des paramètres de fonctions déjà existantes dans STA.

Trois autres tables sont utilisées pour les trois tables décrites ci-dessus juste au début de chaque page. Ceci permettra de retrouver l'état concernant la première ligne de la page si on veut la reformatter après avoir utilisé la facilité d'éditer page par page au cours du formattage. Ces tables sont nommées STAD, STBD, et RSTAD.

II.3.4. La ligne en cours de formattage

Deux vecteurs de 80 octets chacun sont utilisés pou formatter la ligne en cours de formattage.

ARN contient la chaine brute de caractères qui constituer la ligne. Elle contiendra un nombre de caractères inférieur o égal à 80. C'est sur cette chaine que les fonctions d formattage agiront pour ajuster, centrer, etc.

ARM contient la version formattée de la chaine ARN. Cett ligne sera ensuite transférée directement vers la place qu correspond à son emplacement dans la page formattée stockée e mémoire centrale.

II.3.5. Autres données

Plusieurs autres variables existent pour mémoriser le typ de l'unité de sortie choisie, le type d'espacement, le nombre d lignes par pages, etc. Il existe aussi un vecteur de 1600 octet pour mémoriser au maximum 20 lignes que l'utilisateur voudr éventuellement sauvegarer de son texte pour les repositionne dans un autre endroit (la fonction insertion d'une chain sauvegardée).

II.4. LES MODULES DU SYSTEME

La fig. II-2- présente les différents modules du systèm ainsi que les chemins d'interaction entre ces modules.

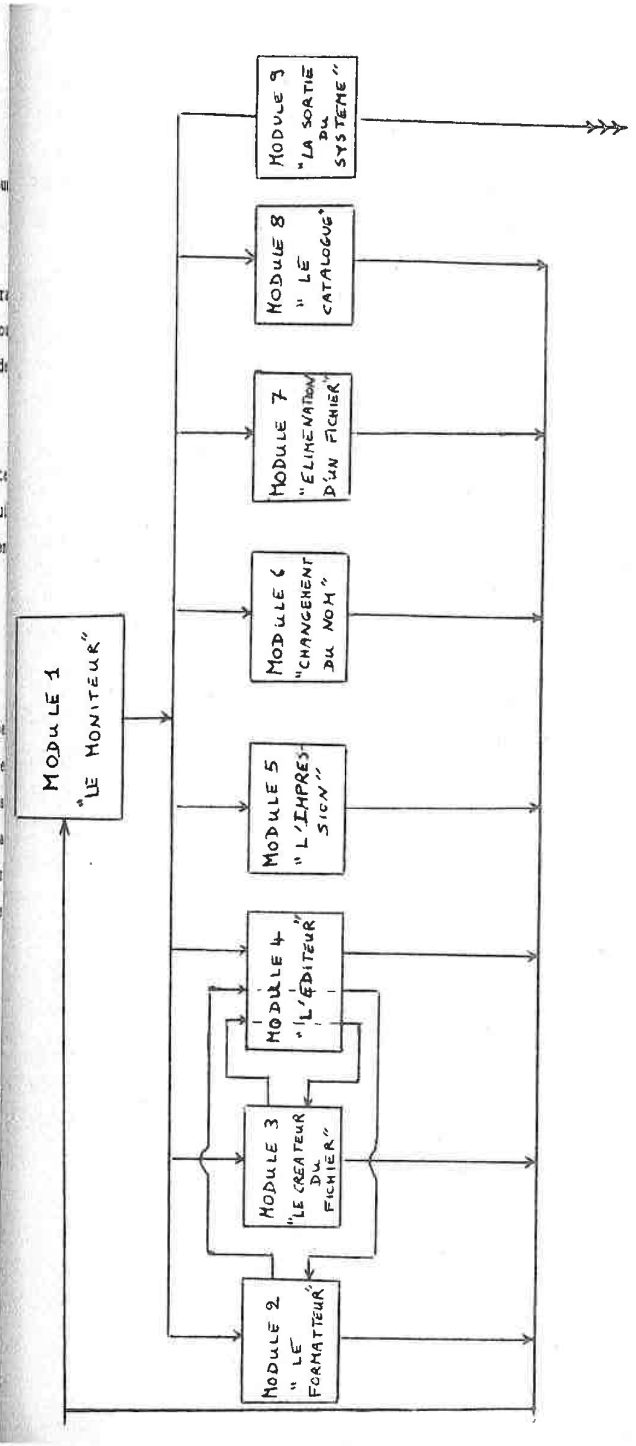


Fig.II.2. Les Modules Du Système et Les Chemins d'Interaction Entre Ces Modules

II.4.1. Module 1: Le moniteur

C'est le module de contrôle du système global. Il a pour fonction de déclarer les capacités du système en affichant un menu des fonctions et des services possibles ainsi que la manière de demander chacun de ces services. Ce menu est affiché sur la console.

En cas de demande d'un service, le moniteur est responsable de lancer l'exécution de la fonction demandée. A la fin de l'exécution, le contrôle retourne automatiquement au moniteur.

En cas de panne quelconque dans l'exécution d'un module, celui-ci rend le contrôle au moniteur qui assure la sortie du système avec indication de mauvais fonctionnement.

Le moniteur est activé immédiatement après l'initialisation du système par l'utilisateur. Il contient, outre les routines de contrôle, plusieurs petites routines utilisées par plus d'un des autres modules. Il lance l'exécution d'un module à l'aide d'un branchement direct vers un point d'entrée de ce module.

II.4.2. Module 2: Le formatteur

Ce module est responsable du formattage d'un texte brut selon les commandes données par l'utilisateur. Le texte formatté page par page est ensuite envoyé vers une unité de sortie: l'imprimante, l'écran, ou le disque. Le choix de l'unité se fait par l'utilisateur au moment de la demande de formattage.

Le formattage se fait page par page. Après chaque page, l'utilisateur a la possibilité de demander l'édition de cette page en utilisant le module 4 qui sera appelé automatiquement. Il pourra donc corriger ou modifier la forme ou le contenu de cette page puis il signalera que l'édition est terminée pour rendre le contrôle au formatteur. Ensuite, il demandera le reformattage de cette page. Le formattage se fera pour la page concernée seulement, grâce à la sauvegarde des commandes de formattage dans TAD, STBD, RSTAD (voir II.3.3.). Ce cycle peut recommencer autant de fois que voulu.

A la fin du processus de formattage, le contrôle sera rendu automatiquement au moniteur.

II.4.3. Module 3: Le créateur de fichier

Ce module est responsable de la création de nouveau fichiers textuels sur les disques du systèmes. Les fichiers créés contiendront des suites de caractères codés en ASCII, ainsi que des fonctions et des codes de formattage. Chaque caractère occupera un octet.

Durant la création d'un texte, il est possible d'éditer ce texte. Si la modification se fait sur la dernière ligne, elle peut se commander directement. Sinon, on sera obligé de faire appel à l'environnement d'édition (module 4) qui exécutera l'édition voulue. En fin d'édition, le retour au créateur se fait en tapant la commande FE (Fin Edition).

A la fin de la création d'un texte, l'utilisateur tape la commande ".X" qui signifie "Fin Texte". Le contrôle sera ainsi rendu au moniteur qui affichera le menu.

II.4.4. Module 4: L'éditeur

Ce module assure deux services principaux à l'utilisateur

1. Il lui donne le moyen de se déplacer à l'intérieur du texte.

2. Il exécute les fonctions d'édition disponibles sur le système.

Nous avons déjà défini les fonctions primitives d'éditi ainsi que le langage d'édition et sa technique d'adressage (voi I.5.). Ce module est la réalisation de cette conception.

Puisque ce système de traitement de textes est conçu pou utiliser un écran comme console, nous avons pu incorporer dan l'éditeur plusieurs fonctions permettant les différents dé placements possibles du curseur à l'intérieur du texte. La modifications qui ont lieu pendant l'édition d'un texte seron immédiatement exécutées sur le texte affiché de manière à donne toujours une image mise à jour du texte. Le curseur sur l'écr correspond directement au pointeur dans la mémoire.

La sur-écriture d'un caractère sur un autre est autorisé et c'est le dernier caractère qui sera pris en compte.

La fin de l'édition se fait par la frappe de la comman "FE". La sortie de ce module se fait comme un retour d'une sou routine pour qu'on puisse utiliser ce module dans le cas de s existence sous le contrôle d'un module quelconque. Le retour d contrôle se fera donc à ce même module.

Pourtant, l'éditeur a deux points d'entrée différents. Le premier lance l'affichage d'un certain nombre de questions concernant l'ouverture du fichier concerné. C'est le point d'entrée utilisé par le moniteur après le choix de la fonction édition du menu.

Le deuxième point d'entrée donne directement accés aux fonctions d'édition disponibles. C'est ce point qui est utilisé par les modules de création et de formattage.

II.4.5. Module 5: L'impression

Ce module est responsable d'imprimer un fichier texte sur l'imprimante ainsi que la gestion de celle-ci.

Il permet à l'utilisateur d'utiliser l'imprimante soit comme unité de sortie soit comme unité d'entrée. Dans ce dernier cas, une secrétaire qui ne veut pas utiliser l'écran parce qu'il est différent de sa machine à écrire, pourra tout de même utiliser le téléimprimeur pour taper son texte. Ce texte entrera ensuite comme un fichier. Ceci peut être utilisé comme technique d'apprentissage au système car on peut également faire appel à quelques fonctions d'édition.

Quand le module d'impression rencontre le code "end of file" il rend le contrôle au moniteur.

II.4.6. Module 6: Changement de nom

Ce module donne à l'utilisateur la possibilité de changer le nom d'un fichier déjà existant sur disque. Il est appelé à partir du moniteur et, à la fin de l'exécution, lui rend le contrôle.

II.4.7. Module 7: Elimination d'un fichier

Ce petit module donne la facilité d'éliminer un fichier existant sur disque. Il est appelé à partir du moniteur et lui rend le contrôle.

II.4.8. Module 8: Le catalogue

Ce module permet à l'utilisateur de savoir les noms, tailles, versions, et date de création de ses fichiers existants sur le disque. La sortie se fait sur l'imprimante pour ne pas perdre les noms des premiers fichiers de la liste si celle-ci est plus longue que le nombre de lignes sur l'écran.

Le retour de contrôle se fait directement au moniteur après la fin de l'impression.

II.4.9. Module 9: La sortie du système global

Ce module permet à l'utilisteur de sortir de l'environnement de notre système de traitement de textes. Le retour du contrôle se fait au module MCR du système RSX-11M (voir II.2.).

L'utilisateur pourra ensuite passer à un autre environnement informatique ou, simplement, démonter son disque et terminer son travail.

II.5. EXTENSION DU SYSTEME PAR LE CONSTRUCTEUR

Le système que nous proposons est extensible grâce à sa structure modulaire ainsi qu'aux langages modulaires d'édition et de formattage.

Le constructeur du système (un programmeur expérimenté) pourra très simplement ajouter des fonctions beaucoup plus complexes que les primitives proposées. Sa tâche est la suivante:

1. Définir la fonction à ajouter

2. Lui donner un code et mettre ce code dans la table des codes (ceci pour que la nouvelle fonction soit reconnaissable par le système).

3. Ecrire la routine exécutant cette fonction en utilisant les modules primitifs que nous avons développés et qui sont suffisants pour définir n'importe quelle fonction complexe par une simple combinaison de quelques uns de ces modules. Ces nouvelles fonctions sont ainsi insérées définitivement dans le système (plus précisément, dans le module d'édition ou de formattage).

Pour concrétiser cette possibilité nous avons réalisé trois fonctions complexes à partir de fonctions primitives. Ces fonctions ont été insérées par le constructeur (nous-mêmes) dans le système que nous avons réalisé sur le LSI-11. Ces fonctions sont:

1. Tabulation sur colonnes de plusieurs lignes (comme dans les tables de comparaison).

2. Mise en colonnes d'un texte (comme dans les articles des revues).

3. Répétition automatique de quelques chaînes constantes en tête (ou en bas) de chaque page.

Nous allons décrire brièvement la réalisation d'une de ce fonctions (tabulation sur colonnes de plusieurs lignes) e utilisant les primitives de formattage(pour comprendre l représentation d'une table voir 4.2.2 en Annexe A):

1. Les primitives "définition de la marge gauche" e "définition de la marge droite" sont utilisées pour défini les marges de chaque élément d'une colonne de la table formatter. Les valeurs des marges sont déduites automa tiquement par le système en utilisant les valeurs de points de tabulation.

2. Chaque élément est considéré comme un texte normal mai sa fin est indiquée par un code spécial. L'élément es formatté entre les marges (définies dans (1) ci-dessus) e utilisant éventuellement les primitives de formattage qu l'affectent

3. A la rencontre d'un code "fin élément" on retourne a début de cette routine pour redéfinir les marges d l'élément suivant et ainsi de suite jusqu'à la fin de l table.

Il faut noter que quelques dispositifs sont prévus dans l formatteur pour permettre l'alignement horizontal des éléments d'une même rangée. Pour ce faire, le numéro de la dernière ligne de chaque élément de la rangée est enregistré (ce numéro est mi à jour automatiquement par le formatteur). La longueur maximal est ainsi déterminée et, par la suite, on définira la ligne d départ de la rangée suivante.

II.6. EXTENSION DU SYSTEME PAR L'UTILISATEUR

Il n'est jamais possible de construire un système "à l carte" pour chaque utilisateur. Il faut aussi noter que le besoins d'un même utilisateur varient d'un jour à l'autre.

Il est donc intéressant que l'utilisateur puisse étendre son jeu de commandes d'une manière simple et pratique pour faciliter son travail.

Pour cela nous donnons à l'utilisateur la possibilité de définir des macro-commandes, c'est-à-dire des commandes complexes regroupant un grand nombre de commandes simples déjà existantes dans le système.

Chaque macro-commande (ou macro) est une suite de commandes simples ou d'autres macros déjà définies dans le système. Les commandes simples sont les commandes utilisées par l'opérateur dans son travail ordinaire. La macro a pour but de lui éviter de réécrire cette séquence de commandes chaque fois qu'il veut l'utiliser. Il lui suffira d'écrire le nom de la macro comme s'il écrivait le nom d'une commande ordinaire. Cette macro s'exécutera en exécutant toutes les commandes qui la forment.

Le fait qu'une macro peut contenir une autre nous paraît très utile. Pour simplifier la tâche du système, nous avons restreint la nomination des macros à une liste que nous définissons (par exemple, M1, M2, ---, M10).

Un utilisateur peut ainsi définir dix macros d'édition et dix de formattage, représentant les fonctions complexes les plus utilisées dans son travail. Ces macros ne sont pas implantées définitivement dans le système. Il peut redéfinir une macro en définissant une autre séquence sous le même nom.

Comme exemple d'édition, supposons qu'en écrivant un texte on a utilisé le mot "ministre" pour plusieurs personnes puis on n'est aperçu que quelques unes d'entre elles ne sont pas ministres mais secrétaires d'état.

On écrit la macro M1:

```
DEF MACRO M1
ELIMINER m (éliminer mot)
INSERER Secrétaire d'Etat
FIN MACRO
```

Puis on écrit:

```
DEF MACRO M2
CHERCHE Ministre
FIN MACRO
```

On a maintenant défini deux macros. On procède ensuite d
la manière suivante: on tape la commande "M2". Le curseur s
positionnera au début du mot Ministre. Si on veut laisser c
mot, on retape "M2" pour chercher le suivant. Sinon, on tape l
commande "M1" qui effectuera le changement nécessaire, puis o
retape "M2" et ainsi de suite.

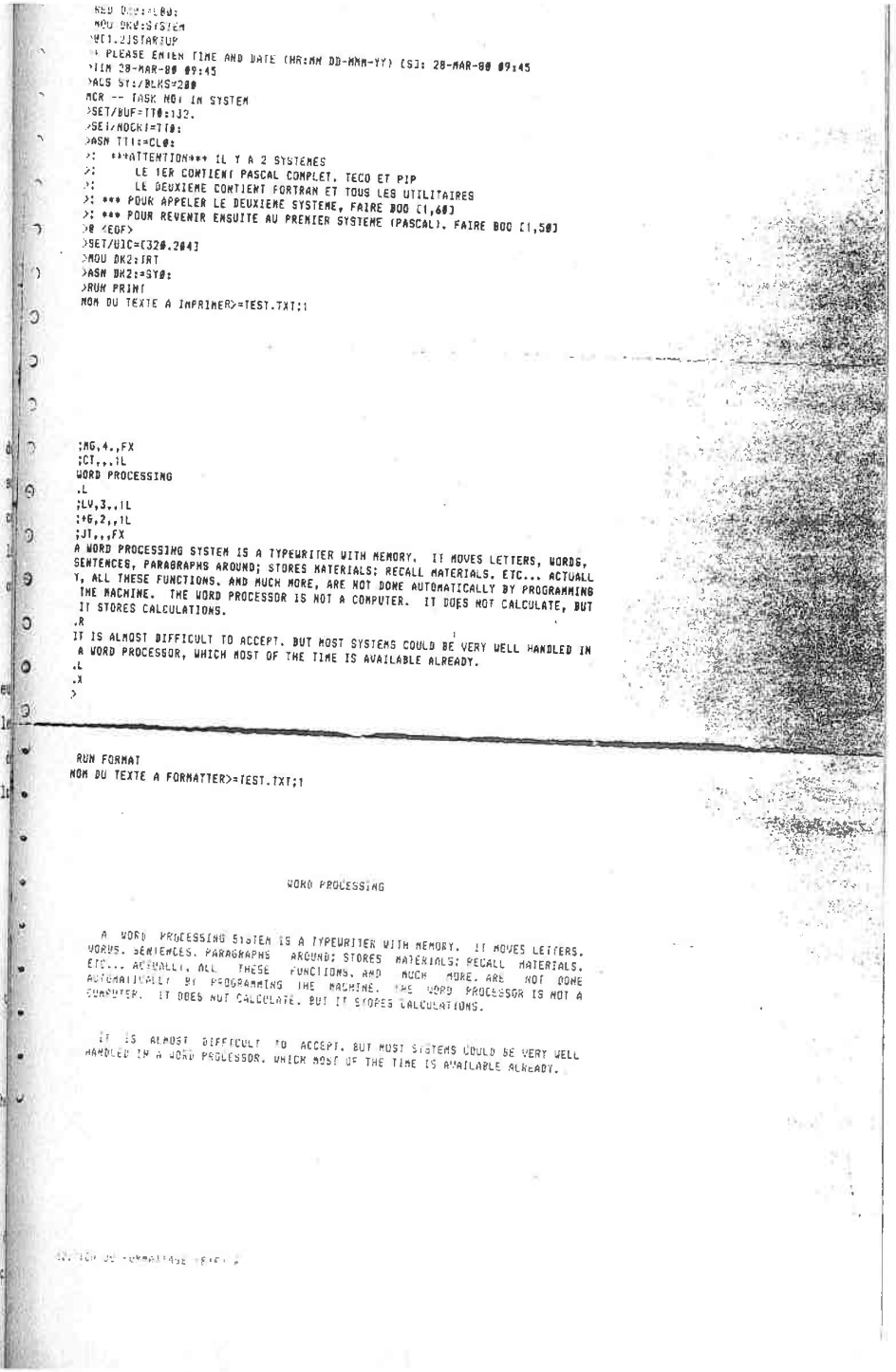
Comme exemple de formattage, supposons qu'un utilisate
veut mettre en évidence quelques paragraphes de son texte en l
décalant de 5 caractères vers la droite. La première ligne d
chaque paragraphe sera en retrait de 10 caractères. Il l
suffit de définir une seule Macro comme suit:

```
DEF MACRO M5
LIGNES VIDES 2
MARGE GAUCHE 10
MARGE GAUCHE 5 1 P (1=après une ligne, P=pour
paragraphe)
FIN MACRO
```

Puis il invoquera la fonction M5 avant tous les paragrap
qu'il veut mettre en évidence.

II.7. Exemple

Un exemple de l'exécution de quelques commandes
formattage affectant un texte est donné dans le listing
dessous.



CHAPITRE -III-

Présentation Des Documents Transmis Sur Un Réseau

III.1. Introduction

III.2. Présentation des problèmes

III.3. Etude bibliographique de solutions existantes

- III.3.1. Proposition A
- III.3.2. Proposition B
- III.3.3. Proposition C

III.4. Solution proposée (Proposition D)

III.5. Comparaisons et critiques

CHAPITRE -III-

Présentation Des Documents Transmis Sur Un Réseau

III.1. INTRODUCTION

Dans l'introduction de cette thèse, nous avons présenté les différentes phases de la production d'un document. Au cours des deux chapitres précédents, nous avons présenté des propositions et une réalisation expérimentale pour résoudre quelques problèmes liés aux phases "édition" et "formattage". Dans ce chapitre, nous abordons l'autre phase très importante qu'est "la distribution d'un document". Nous examinons les problèmes techniques rencontrés au cours de son informatisation et nous proposons une solution dont la réalisation expérimentale est détaillée au chapitre IV.

La distribution "informatisée" d'un document produit par logiciel est devenue possible grâce aux réseaux d'ordinateurs et, en particulier, aux réseaux locaux. C'est donc depuis quelques années seulement que nous pouvons parler de "distribution électronique de documents" ou "messagerie électronique".

Les réseaux locaux permettent l'interconnexion de plusieurs outils informatiques géographiquement répartis dans une zone limitée à quelques kilomètres (Fig. III.1.). Parmi ces outils, on peut avoir:

- des systèmes de traitement de textes (homogènes ou hétérogènes).
- des terminaux et des périphériques de différents types
- des postes de travail intelligents
- éventuellement, des noeuds (gateways) permettant l'accès à d'autres réseaux (locaux ou distants).

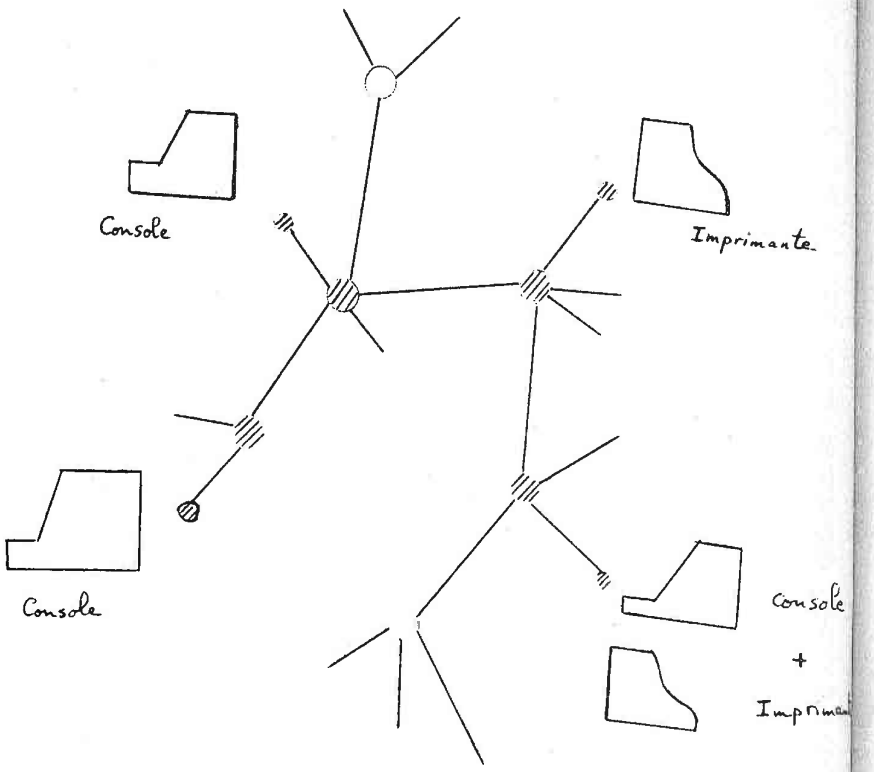


Fig.III.1. Réseau Bureautique

Il faut noter que le réseau proprement dit ne fournit que le service de transport de l'information d'une station émettrice à une station destinatrice, toutes deux connectées, évidemment, au réseau. Nous supposons donc ici que les problèmes de connexion de matériel au réseau, l'accès à celui-ci, et le "transport" de messages (séquences de bits) sont tous résolus, et nous ne prenons en considération que les problèmes de communication liés à l'application même (c'est-à-dire ce qui correspond aux couches 6 et 7 du modèle de référence ISO pour l'interconnexion des Systèmes Ouverts ‹33›).

III.2. Présentation des problèmes

Généralement, un document est susceptible de contenir des informations de types différents: texte, graphismes, voix, image en fac-similé, vidéo, etc... Nous optons pour la séparation des différents types et l'attribution d'un "environnement" particulier pour la saisie et le traitement de chacun d'eux. Un moyen de commutation entre ces environnements distincts doit exister pour permettre la création et la production de documents de types mixtes. Ceci n'est pas banalisé l'heure actuelle, pour des raisons de coût et de technologie, mais nous pensons que les documents de type mixte deviendront faisables très prochainement ‹34,46›.

Comme nous l'avons déjà signalé, nous ne traiterons ici que les informations purement textuelles, l'application des idées aux autres types d'information pouvant se faire ultérieurement.

D'autre part, pour cette information textuelle, une multitude de systèmes de traitement de textes différents peuvent exister et communiquer à travers le même réseau. C'est actuellement là que les problèmes d'incompatibilité semblent être graves.

1- Au niveau des terminaux et des périphériques:

Aucune normalisation respectée n'existe, aussi bien pour les fonctions de contrôle et de gestion des terminaux (du même type) que pour le codage de ces fonctions. Quant au codage de l'information proprement dite, plusieurs standards existent ‹13,57›.

2- Au niveau des fichiers:

Les systèmes de gestion de fichiers peuvent être incompatibles ainsi que les formats et la présentation des fichiers ‹25›.

3- La forme du document transmis ‹53›:

Le document transmis sur le réseau peut être en forme:

(i) révisable: le texte brut et les instructions explicites de formattage coexistent dans le document en transmission.

(ii) finale: le texte est déjà formatté, il est donc transmis sous forme d'une suite de caractères et de codes spéciaux contrôlant le matériel de sortie déterminé.

(iii) semi-révisable: c'est-à-dire que quelques fonctions de formattage sont exécutées sur le texte brut tandis que le reste des fonctions (surtout les codes) restent incorporées explicitement dans le texte semi-formatté.

4- Les fonctions de formattage:

Aucune normalisation n'existe dans ce domaine. Les fonctions sont différentes d'un système à l'autre et, de plus, si la même fonction existe sur deux systèmes, elle est en général évoquée par des instructions de forme et de syntaxe différentes. Par ailleurs, il est souvent impossible de faire une traduction simple des instructions de formattage d'un système à certains autres.

5- La position des instructions dans le texte:

La position des instructions de formattage incorporées dans le texte (en forme révisable) varie d'un système à l'autre: doivent-elles commencer au début d'une ligne?, occupent-elles une ligne entière?, sont-elles insérées juste avant la ligne affectée?, etc....

6- Les attributs de certaines chaines:

Ce sont les codes qui expriment les qualités logiques (hiérarchiques: titres, sous-titres, etc...) ou physiques (par exemple: en tête, souligné, etc.) d'une chaine de caractères donnée. Ces codes ainsi que leur positionnement ne sont pas du tout standardisés.

7- La séquence de stockage des différentes chaines:

La plupart des documents ont une structure logique. Par exemple, un livre est structuré en chapitres, sections, etc... Les figures, photos, tables, table des matières, et l'index peuvent aussi être considérées comme des parties structurelles d'un document. Les programmes ont aussi une structuration déterminée par la syntaxe du langage de programmation utilisé. Les équations mathématiques ont aussi une structure logique qui est reflétée dans les "conventions" de notation mathématique.

Cette structure logique peut être facilement "aplatie" pour donner la structure linéaire selon laquelle le contenu d'un document est stocké sur les supports magnétiques ou transféré électroniquement d'un site à l'autre. Par exemple, les chaines représentant la table des matières d'un document sont stockées, puis les chaines du chapitre 1, puis 2, et ainsi de suite.

Mais cette linéarisation est quelquefois détruite. Prenons, par exemple, un renvoi indexé sur lequel pointe une chaine de caractères. Le positionnement de ce renvoi dans la structure logique (et donc sa linéarisation par rapport aux autres chaines) n'est pas unique. On peut le stocker à un endroit spécial dans le document, ou juste après la chaine qui pointe vers lui, ou bien encore juste après les

caractères de liaison dans la chaine qui pointe vers lui. La seule relation sûre est qu'il doit apparaître en bas de la page où la chaine qui pointe vers lui est placée. Le même cas se présente pour une figure référencée dans une chaine. Donc, la position de stockage de certaines chaines de données à l'intérieur d'un document varie d'un système à l'autre.

Dans le souci de réduire ces incompatibilités et de simplifier ainsi la connexion et les communications entre les divers systèmes informatiques de traitement de textes, certaines propositions ont été faites et, parfois réalisées.

En ce qui concerne les terminaux, des protocoles d'appareils virtuels (PAV) ont été développés pour les différents réseaux ‹13›, mais il faut noter que, malheureusement, ces PAV ne sont pas "universels"!
Pour gérer le transfert de fichiers, des protocoles existent également (nommés PTF) ‹25›. Récemment, des systèmes d'exploitation ont été développés de manière qu'ils puissent être implantés sur des machines différentes (comme UNIX, CP/M, etc...).

En ce qui concerne le document lui-même (sa forme, ses fonctions de formattage, etc...), quelques travaux ont été entrepris pour définir une forme standard pour la présentation des documents ‹22,41,53›. Une sorte de "Protocole de Document Virtuel" est vraisemblablement dans l'esprit des chercheurs. Ceci veut dire que les solutions proposées tournent autour de la spécification d'une forme unique (ou un nombre minimal de formes) décrivant comment les informations d'un document seront représentées et transmises sur le réseau. Des "protocoles" et des "adaptateurs" permettront ensuite la transmission de ces documents.

III.3. Etude Bibliographique de Solutions Existantes

Dans cette partie, nous présentons une synthèse de travaux publiés concernant la normalisation de la présentation des documents.

De chaque proposition, nous avons extrait les aspects suivants:

1- La structure logique du document: sa décomposition, les objets conceptuels logiques qui le constituent, la hiérarchisation de ces objets.

2- Les types des informations contenues dans le document: quels sont-ils? Peuvent-ils coexister dans un même document?

3- La description du document: en général, un document stocké dans un système informatisé est représenté par une suite linéaire de codes (ou d'octets). A l'intérieur de cette suite, et dans des endroits relatifs particuliers, des codes spéciaux, ou descripteurs, sont insérés pour spécifier les différents aspects de la représentation du document. La syntaxe, la sémantique et l'emplacement de ces descripteurs forment les règles d'un langage de description. Les aspects à décrire sont:

3.1. Le contenu du document et sa structure logique interne

3.2. Le format de la présentation externe du document lors de sa sortie sur un support faisant l'interface avec l'utilisateur humain.

3.3. Bien que la structure interne et le format externe d'un document soient deux aspects distincts, une relation existe entre eux et doit donc être spécifiée. Par exemple, on doit pouvoir spécifier que les chaines désignées comme étant des renvois doivent apparaître au bas de la page dans laquelle elles sont référencées.

4_ Les propriétés d'un document en transmission sur le réseau: sa représentation (langage de description), son type, sa form (révisable, finale, etc...).

5_ Les traitements à effectuer sur un document avant son émissio et après sa réception (par exemple, traduction des descripteurs, ajout de nouveaux descripteurs, transformation de forme d représentation, etc...).

III.3.1. Proposition A

III.3.1.1 Structure Logique

Dans ‹40,41› des protocoles de présentation d'u document sont proposées. La structuration du contenu d'u document est basée sur les 3 objets suivants (voir Fig. III.2.).

- Les "Atomes": ce sont les "cellules" adressables le plus élémentaires d'un document.
- Les "Unités": ce sont des chaines d'atomes.
- Les "Segments": ce sont des ensembles d'unités ou/e d'autres segments.

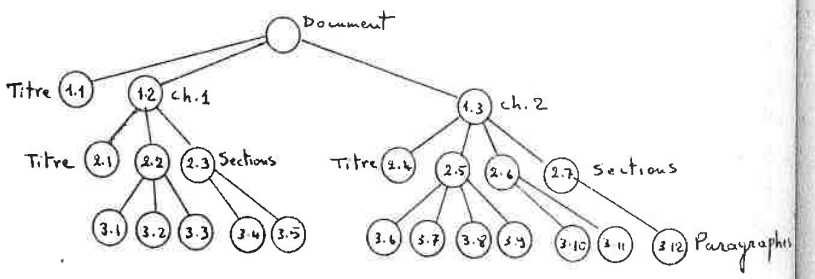


Fig.III.2. Structuration d'un document

Un segment peut être représenté sous la forme d'un arbre structuré avec un nombre variable de niveaux.

Un segment ou une unité a un "Nom Unique", qui indique son emplacement dans le document et sa relation avec le schéma général.

III.3.1.2. Types des données:

Différents types de données sont définis dans un document, à raison d'un type par "unité" de donnée. Le type est déterminé par la nature des "atomes" contenus dans cette "unité".

Ces types sont:

1- **TEXTE:** Les "atomes" qui constituent "l'unité" sont des caractères considérés comme étant dans des boites rectangulaires.

2- **FORMULAIRE:** semblable au type texte mais avec l'existence de champs protégés.

3- **GRAPHIQUE (MATRICE DE POINTS):** Les "atomes" sont des "pels" qui peuvent être groupés et exprimés en formes élémentaires pré-définies.

4- **GRAPHIQUE (VECTEUR):** Les "atomes" sont des segments qui peuvent être groupés et exprimés en fonctions élémentaires graphiques pré-définies.

5- **IMAGES PHOTOGRAPHIQUES:** Les "atomes" sont des "pels" représentés sequentiellement sans aucun groupement de formes élémentaires (ex., image en fac-similé).

6- **VOCAL:** Les "atomes" sont des "phonèmes" (ex. messages vocaux accompagnant un texte ou une image, etc...).

Puisqu'un segment est composé d'une (ou plusieurs) unités, il est normal que le type d'un segment varie selon les types des unités qui le composent. On peut donc avoir des:

1- **Segments à Mode Unique:** qui contiennent une (
plusieurs) unités du même type (ex. message vocal).
2- **Segments Multi-Mode:** qui contiennent des unités d
types différents (ex., un document avec texte+image);

III.3.1.3. Description du Document

Des champs sont définis pour contenir les descripteur
d'un "segment" donné constitué par une seule unité d'un typ
donné. Chaque **"primitive"** est formée par un segment et se
champs descripteurs et elle constitue la plus petite entit
pouvant être transférée. Le format des "primitives" varie
selon le type du segment qu'elle contient:

(a) Description d'un Segment du Type Texte

Le format général d'une primitive "Texte" est:

‹OPEN-SEGMENT›‹SEGMENT-TYPE›‹SEGMENT-NAME›‹ATTRIBUTES›‹TEXT›‹CLOSE-SE
‹POSITION›

où:

**‹OPEN-SEGMENT› : annonce le début de la primitive. So
effet se termine par un ‹CLOSE-SEGMENT›.

**‹SEGMENT-TYPE› : indique le type (texte).

**‹SEGMENT-NAME› : c'est un binôme (numéro du niveau,
numéro d'ordre dans ce niveau) dans l'arbre de hiérar
chisation logique des données d'un document.
L'apparition (ou pas) du nom unique d'un segment sur le
support de sortie dépend d'un attribut spécial, inséré
dans le champ ‹ATTRIBUTES›.

**‹ATTRIBUTES› : Ils peuvent être dynamiques ou statiques. Les attributs statiques sont créés une seule fois par un des partenaires communiquants. Les attributs dynamiques sont modifiables, par des commandes spéciales. Trois classes d'attributs sont définies par l'auteur:

(1) Attributs de Caractères

- Indiquant:
  - *La direction de l'écriture des caractères (gauche à droite ou droite à gauche)
  - *Le jeu de caractères utilisé
  - *Le mode d'affichage des caractères (soulignés, foncés,...)
  - *L'espacement entre les caractères (et les mots) sur le support de visualisation.
  - *‹CHANGE ATTRIBUTE› pour le changement des attributs de caractères. Les attributs des caractères peuvent être modifiés dans un segment par la commande ‹CHANGE-ATTRIBUTE›. Les attributs de caractères du premier caractère du segment s'appliquent sur tous les caractères suivants jusqu'à la rencontre d'une commande ‹CHANGE-ATTRIBUTE›.

(2) Attributs Globaux

Ils indiquent les caractéristiques du rectangle de visualisation dans lequel le contenu de ce segment sera affiché. Ils indiquent:

- *Les dimensions (X, Y) du rectangle
- *Justification (gauche, droite)
- *Affichage du Nom du Segment (oui ou non)
- *Espacement entre les lignes

(3) Attributs de PAGE

Ils indiquent le position du rectangle de visualisation sur la surface du support physique de visualisation. Ils indiquent:

*La marge gauche de la lère ligne du segment
*La marge gauche de la 2ème ligne du segment
*L'espacement entre les différents segments
*La marge de l'en tête de la page
*La marge de l'en bas de la page
*Numérotation des lignes
*Numérotation de la page
*En tête (oui ou non)

Les attributs de page sont dynamiques, et peuvent varier selon le support de visualisation.

Il faut noter que chaque segment doit avoir ces attributs. S'ils ne sont pas indiqués dans le format général de la primitive qui contient un segment, les attributs du segment du niveau précédent seront considérés commes étant les attributs de ce segment.

**La fonction ‹POSITION›
Elle déplace un pointeur dans le texte, et le positionne sur un "atome". Il faut noter que le déplacement du pointeur peut être basé sur une entité logique d'un texte (par exemple: mot, phrase, ...).

(b) Description d'un Segment de Type Formulaire

Le format général de la primitive d'un segment de type formulaire est:

‹OPEN-SEGMENT›‹SEGMENT-TYPE›‹SEGMENT-NAME›‹SEGMENT-ATTRIBUTES›‹FIELDS›‹CLOSE-SEGMENT›

où:
‹OPEN-SEGMENT›, ‹CLOSE-SEGMENT›, ‹SEGMENT-NAME› sont les mêmes que pour le type texte.

**‹SEGMENT-ATTRIBUTES› : Ce sont les "Attributs Globaux" et "Attributs de Page" et ont le même sens que pour le type texte.

**‹FIELDS› : est une sequence ayant le format suivant:
‹FIELD-ID›‹POSITION›‹ATTRIBUTES›‹TEXTE›
où:
‹FIELD-ID› : est un identificateur unique pour le champ à spécifier.
‹POSITION› : donne les coordonnés (X, Y) de la position du ler caractère de ce champ sur la surface physique de visualisation.
‹ATTRIBUTS› : Ils indiquent:
1. Les caractéristiques d'affichage concernant
   * la visibilité des données du champ
   * la mise en évidence des données du champ
   * le jeu de caractères
   * l'espacement entre les caractères (et les mots)
2. Le contrôle d'accès:
   * la protection (ou non-protection) des données du champ
   * les valeurs acceptables pour remplir le champ (par exemple: numérique, entier, positif).

Les primitives des segments d'autres types sont détaillés dans ‹41›.

III.3.1.4. Le Document en Transmission

Le document à transmettre est décomposé en primitives (voir III.3.1.3). qui peuvent être de types différents.

Le document peut être en forme "révisable" (voir III.2.). Dans ce cas, les attributs spécifieront son formattage.

Malheureusement, les détails du langage de spécification de

ces attributs n'ont pas été développés, nous ne pouvons donc pas étudier la facilité de traduction entre ce langage unique et les langages de description spécifiques aux STT connectés au réseau.

III.3.2. Proposition B

C'est la proposition présentée dans ‹22›.

III.3.2.1. Structuration logique

Le document est décomposé en "unités" plus élémentaires composées de "cellules" (cells) et dont chacune représente un code écrit en c-bits. Chaque unité est monotype.

III.3.2.2. Les types des informations

- entiers : (essentiellement, des codes de contrôle non imprimables)
- texte
- audio (vocal)
- graphique
- video
- structure : (c'est le document tout entier, composé d'unités de types hétérogènes)
- array : (c'est le document tout entier, contenant de 1 à n unités et toutes du même type)
- pointeur : (pointant vers une autre entité soit dans le même document ou dans un autre qui le précède). Il est utilisé pour la transmission d'un message à faible redondance destiné à plusieurs destinataires.

III.3.2.3. Description du document

Des champs de descripteurs sont utilisés pour les différents types d'unités. Voici ceux des types textuels:

(a) Texte

Il a le format suivant

‹TYPE›‹LEXICAL ADDRESS›‹SIZE›‹CRYPTO›‹CHARACTER ATTRIBUTES›‹VA

où:

‹TYPE› : spécifie le type de donnée dans cette unité. Dans ce cas il est un "entier' indiquant une information qui encode des informations de contrôle en forme compactée.

‹LEXICAL ADDRESS› : son adresse unique en termes de hiérarchisation logique dans le message. Il se compose de:

(i) niveau lexique dans le message, plus

(ii) index (numéro de séquence de cette unité d'information dans le niveau lexique par rapport au message entier).

‹SIZE› : nombre de cellules incluses dans cette unité d'information trouvée dans le champ "VALUE"

‹CRYPTO› : Ce champ indique si le champ ‹VALUE› des informations est encrypté ou pas, et dans l'affirmative, quelle technique est utilisée pour l'encryptage.

‹CHARACTER ATTRIBUTES› : décrivent

* l'ensemble des caractères (par exemple, ASCII)
* l'apparition des caractères (par exemple, jeu, foncé, couleur, taille, soulignement)
* espacement inter-caractères

‹VALUE› : chaine de caractères qui représente l'information

Dans la suite de cette section nous écrirons ‹X› à la place de :

‹TYPE›‹LEXICAL ADDRESS›‹SIZE›‹CRYPTO›

(b) Array

Le format général est:

‹X›‹DIMENSIONS›‹SIZE DIMENSION 1›..‹SIZE DIMENSION N›

‹NESTED TAG›‹FRAMING ATTRIBUTES›‹VALUE COMPONENTS OF DATA TYPES›

où:

‹X› : voir ci-dessus

‹DIMENSIONS›‹SIZE DIMENSION 1›..‹SIZE DIMENSION N›: représentent les tailles (nombres de cellules) de chaque unité dans ce tableau.

‹NESTED TAG›: représente le type (unique puisque les unités sont homogènes) de unités composant le tableau. Par exemple, si un ‹NESTED TAG› est une chaine textuelle, ceci indique que sa valeur ‹VALUE COMPONENT ...› est de type texte.

‹FRAMING ATTRIBUTES›: décrit le formattage d'une page qui contiendra les informations si le ‹NESTED TAG› est texte, autrement ce champ est vide.

‹VALUE COMPONENTS OF DATA TYPES›: les unités de données composant le tableau.

(c) Pointeur

Le format est:

‹X›‹NESTED TAG›‹REFERENCED LEXICAL ADDRESS›‹COMMAND INDENTI

où:

‹NESTED TAG›: un champ pour aider à la détection des erreurs. Ce champ et le crypto (dans ‹X›) doivent être identiques aux champs correspondants dans l'unité vers laquelle pointe ce type de donnée.

‹REFERENCED LEXICAL ADDRESS›: l'adresse hiérarchique de l'unité vers lequelle ce type de donnée pointe.

‹COMMAND INDENTIFIER›: identificateur d'autres types de commandes concernant le système de messagerie.

III.3.2.4. Caractéristiques du document en transmission

Les documents transmis doivent être en forme "finale" (voir III.2.). Il ne sera donc pas possible de re-éditer un texte reçu par une station destinatrice.

D'autre part, un PAV sera utilisé pour transformer le document dès sa réception de manière qu'il soit compréhensible par les terminaux du destinataire.

III.3.3. PROPOSITION C

Elle est présentée dans ‹53›.

III.3.3.1. Structure logique du document

Pour chacune des formes d'un document: révisable, finale (texte), finale (melange de plusieurs types), une structure spéciale est proposée:

(a) Structuration d'un document en forme révisable:

Le document est décomposé en "entités" disjointes contenant les informations et tous les codes décrivant sa présentation externe. Chaque entité est complètement indépendante des autres en ce qui concerne son traitement et son affichage. Par exemple l'entité peut représenter un chapitre d'un document textuel.

(b) Structuration d'un document en forme finale (texte):

Le document est considéré comme étant une suite de "pages" formées de "lignes" qui sont des suites de caractères".

La page est l'unité minimale transmissible d'une station à l'autre.

(c) Structuration d'un document en forme finale (mélange de plusieurs types):

Le document se compose de "pages" contenant des "objets élémentaires" qui sont formés de "champs" contenant soit des "données" affichables soit des "informations de contrôle".

III.3.3.2. Types de données

Ils ne sont pas définis de manière explicite mais il est clair que la proposition s'applique aux données de type textuel pour le document à circuler en forme révisable.

Mais pour un document en forme finale d'impression la proposition s'applique aux données de type textuel ou mixte (texte + graphiques), quoique les autres types ne soient pas exclus.

III.3.3.3. Description du document

A chaque forme du document correspondent des descripteurs particuliers.

(a) Forme révisable

La syntaxe et la sémantique définies pour les descripteurs sont communes pour tout document en transmission.

Les descripteurs (déclarations de formattage) et les instructions de traitement de textes sont insérées sous forme de champs structurés à l'intérieur du texte proprement dit.

Quelques descripteurs sont insérés avant le début du texte (et après l'indicateur de début de l'entité), ce sont ceux qui affectent le texte de l'entité tout entier. Par exemple, les paramètres de tabulation, les marges, numérotation des lignes ou pages, etc.

D'autres descripteurs sont insérés à l'intérieur du texte et affectent des parties de celui-ci.

Les descripteurs sont codés sur un ou plusieurs octets, ou même dans des champs structurés normalisés.

Exemple d'un champ structuré dans Fig.III.3.:

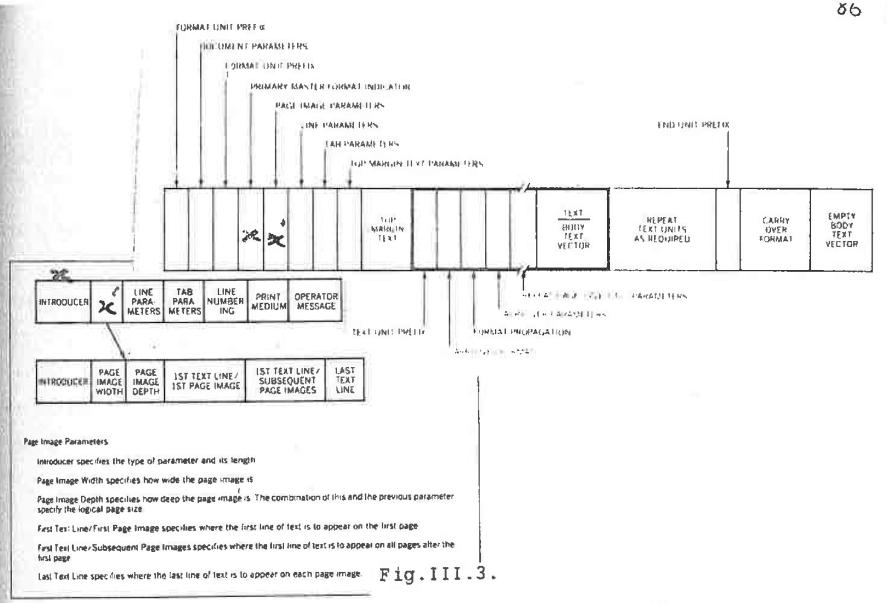


Fig.III.3.

(b) Forme finale (texte)

Des codes de contrôle sont insérés dans des endroits spécifiques à l'intérieur du texte. Ils provoquent l'activation des fonctions de contrôle qui produisent le résultat voulu. Certains codes peuvent activer des processus algorithmiques ou des fonctions matérielles d'un appareil. Tous les codes de contrôle utilisés sont diférents des codes de caractères.

Un exemple de ces codes est donné ici:

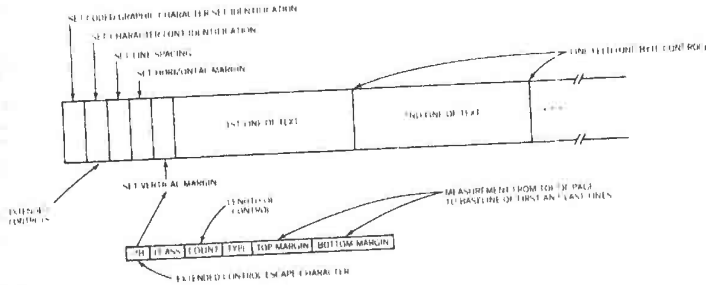


Fig.III.4.

(c) Forme finale (mélange de types de données)

Des indicateurs spéciaux marquent le début et la fin de chaque objet élémentaire et de chaque page.

Les fonctions de contrôle sont hiérarchisées. Celles qui sont applicables à une page sont écrites au début de cette page,de même que celles qui ne s'appliquent qu'à un objet élémentaire se trouvent juste au début celui-ci.

Une fonction n'affecte que l'entité au début de laquelle elle est insérée. Une entité sans fonctions propres n'est donc affectée que par les fonctions de l'entité hiérarchiquement englobante. Un exemple est donnée ci-dessous.

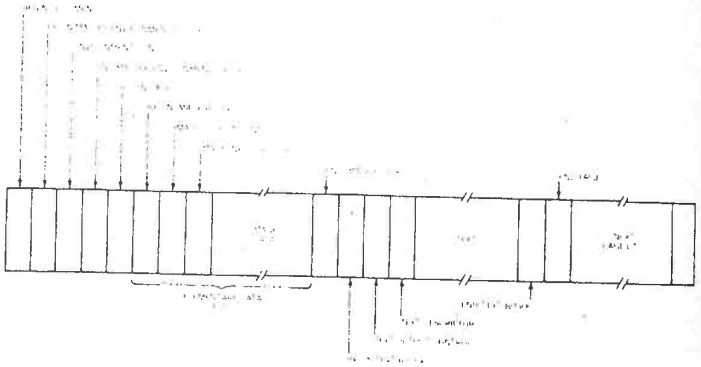


Fig.III.5.

III.3.3.4. Caractéristiques d'un document en transmission

Pour simplifier la manipulation des différentes formes de documents, le choix s'est porté sur la transmission des documents sous une des trois formes présentées ci-dessus. Quelques informations doivent être ajoutées avant chaque partie du document transmis pour identifier cette partie (type, forme, encryptage, etc...) et donner des directives facilitant son traitement et sa distribution.

III.3.3.5. Traitement à effectuer sur le document transmis

Puisque trois formes différentes de documents peuvent exister, on a besoin de "trois canaux de traitements" parallèles pour manipuler les documents transmis.

Avant l'émission d'un document, il y aura besoin de 3 "transformateurs" différents à chaque site pour "exprimer" un document selon le modèle standard correspondant à sa forme.

Il est toutefois possible de transformer un document textuel révisable en un document en forme finale ou semi-révisable dès sa réception par le destinataire selon ses possibilités (par exemple, si c'est une imprimante il faudra qu'elle le reçoive en forme finale). Dans ce cas, la communication doit assurer la bonne interprétation des données, mais pas nécessairement une exécution complète des fonctions impliquées ou exprimées. Si une fonction est exécutée comme spécifiée, ou signalée si non exécutée, l'interprétation est bonne.

Il est également possible de transformer un document en forme finale "texte" en un document en forme finale "mixte". Dans ce cas, un protocole d'appareil virtuel (PAV) sera utilisé pour transformer le document à sa réception de manière qu'il soit compréhensible par les terminaux du destinataire.

III.4. Solution Proposée (Proposition D)

Pour résoudre le problème de la transmission des documents sur un réseau local nous proposons l'approche suivante:

III.4.1. Structure Logique

Le document tel que nous le concevons se compose essentiellement de "paragraphes", ceux-ci étant des entités logiques indivisibles (par exemple; un titre, un paragraphe, etc.). Dans le reste de ce chapitre c'est ce sens large du mot "paragraphe" qui est sous-entendu.

Ces paragraphes sont regroupés en sections, qui sont regroupées, à leur tour, en sections plus globales, et ainsi de suite jusqu'à ce qu'on arrive à regrouper le tout en document entier.

Ceci correspond à une structure hiérarchique d'un document sous forme d'arborescence (Fig. III.6.).

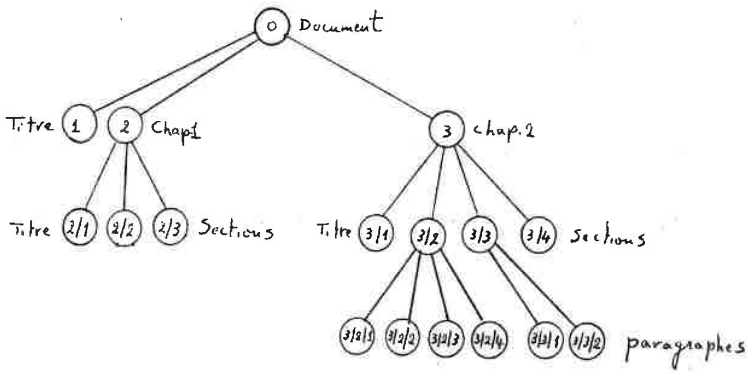


Fig. III.6. Structure arborescente d'un document

Dans cette arborescence, la racine représente l'intégralité du document, les sommets représentent des structures logiques (chapitres, sections, etc...) et les feuilles (qui sont les sommets sans successeurs) représentent à la fois la structure logique la plus élémentaire (qui est le paragraphe) ainsi que son contenu physique (c'est-à-dire les chaînes de caractères qui forment le paragraphe).

Les arcs (ou branches) de l'arborescence représentent la relation d'inclusion. Un sommet inclut (ou contient) tous ses successeurs. Les successeurs d'un sommet quelconque sont tous disjoints et l'union de ces successeurs est représenté par leur sommet père.

Pour éclaircir ceci, prenons **l'exemple** suivant (Fig. III.6.): un document contient un titre et deux chapitres. Chapitre-1- contient son titre et deux sections dont l'une contient trois paragraphes et l'autre deux (remarquez que l'un des deux paragraphes peut être le titre de la section). Chapitre-2- contient aussi un titre et trois sections, etc. Notez que "document", "chapitre" et "section" n'ont qu'un sens logique et sont représentés par des sommets. La relation d'inclusion et la disjonction des successeurs d'un sommet sont évidentes dans cet exemple.

Le niveau hiérarchique d'un sommet S est le nombre de sommets parcourus pour arriver à S en partant de la racine (c'est-à-dire la longueur du chemin dont l'extrémité initiale est la racine et l'extrémité terminale est le sommet S).

La profondeur hiérarchique d'un document est le nombre de sommets entre la racine et la feuille la plus éloignée de celle-ci (= le niveau hiérarchique de cette feuille). La profondeur hiérarchique du document représenté en Fig.III.6. est égale à 3.

Il existe une relation parmi les successeurs d'un sommet quelconque (frères): c'est une relation "d'ordre relatif de naissance" basée sur un critère de position (naissance) physique relative d'un des successeurs par rapport aux autres: un successeur doit apparaître (naître) sur le support de sortie avant ceux qui sont représentés à sa droite dans l'arborecence. Dans l'exemple de la figure.III.6., le titre doit apparaître avant chapitre-1-, qui apparaît à son tour avant chapitre-2-. Ceci implique aussi que tous les successeurs de chapitre-1- apparaissent avant ceux de chapitre-2-.

L'adresse hiérarchique

Nous introduisons un concept d'adresse hiérarchique qui permet de donner un nom unique à chaque sommet (et à chaque feuille). Cette adresse est un n-uplet (où n est le niveau hiérarchique du sommet adressé Sa). Chaque élément de cet n-uplet représente donc un sommet Sj du chemin entre la racine et Sa, et il a comme valeur l'ordre de naissance de Sj parmi ses frères. Comme exemple, voir les adresses hiérarchiques associées aux sommets en Fig.III.6.

Le sommet S3/2/1, par exemple, est atteint en parcourant le chemin suivant:

- S3 : le sommet d'ordre 3 parmi les successeurs de la racine "document"
- S3/2 : le sommet d'ordre 2 parmi les successeurs du dernier sommet atteint S3
- S3/2/1 : le sommet d'ordre 1 parmi les successeurs du dernier sommet atteint S3/2

Ce sommet est du 3ème niveau (donc, n=3 et son adresse hiérarchique est le triplet [3/2/1].

On remarque que ce triplet détermine complètement le chemin d'accès à la feuille concernée à partir de la racine. D'autre part, il est évident que chaque adresse de ce genre ne désigne qu'un seul sommet.

Ce système d'adressage reflète toutes les relations hiérarchiques (a le niveau n) et horizontales (par les valeurs des éléments du n-uplet) parmi les feuilles.

Les paragraphes de clarification

Reste un problème à résoudre, c'est celui des renvois, des figures, de numérotation, et des références à des "paragraphes" qui ne font pas partie intégrante du texte d'un document ‹6,8›.

En fait, entre ces **"paragraphes de clarification"** et le reste du document, il n'existe qu'une relation logique de **"subordination"** à un ou plusieurs paragraphes du document qui, de leur part, les utilisent comme moyen de clarification. Cette relation ne doit pas être confondue avec les relations hiérarchiques qui existent parmi les paragraphes essentiels. Nous pensons que les paragraphes "subordonnés" ne doivent pas être inclus dans l'arbre du document au même titre que les autres paragraphes, mais ils seront "attachés" aux sommets (ou feuilles) qui leur font référence.

Le problème d'adressage de ces paragraphes de clarification peut être résolu en donnant à chacun d'eux la **même adresse** hiérarchique que la feuille (=paragraphe essentiel) qui le subordonne. Si plusieurs paragraphes essentiels subordonnent un paragraphe de clarification donné, celui-ci prendra l'adresse de la section contenant ces paragraphes (= adresse du plus grand ancêtre commun). Par conséquent, on peut avoir plus d'un paragraphe ayant la même adresse. Pour pouvoir les différencier, le paragraphe de clarification est marqué par un indicateur de subordination qui apparaît dans ses attributs (voir III.4.3.).

Dans certains cas, un paragraphe de clarification peut apparaître physiquement plusieurs fois en relation avec une section donnée (par exemple, une en tête répétée sur toutes les pages d'un chapitre).

Ce rapport entre subordination et position est, en principe, une des caractéristiques d'un document donné et doit figurer dans les attributs de sa représentation.

III.4.2. Le type des données d'un document

Un document peut généralement contenir des paragraphes de types différents choisis parmi les suivants:

1. "Texte Simple"
2. "Expression": une chaîne de caractères générée automatiquement après le traitement d'une expression donnée (par exemple, numéro de page auto-incrémenté) ‹8›.
3. "Pointeur": une chaîne représentant une section du texte et signifiant qu'elle doit être "recopiée" à la place de ce pointeur (pour éviter la redondance: par exemple, pour l'écriture d'une table des matières) ‹22›.

Nous n'avons approfondi que les documents dont le type principal est "textuel". Par contre, nous n'excluons pas les autres types, surtout en ce qui concerne les paragraphes de clarification qui peuvent être "graphiques" ou autre. Mais une étude des représentations spéciales doit être menée pour la généralisation des concepts que nous développons ici.

III.4.3. Les descripteurs du document

Puisque le langage de formattage que nous avons développé (voir chapitre.1.) est complet et capable de décrire une fonction de formattage quelconque, il nous a paru adéquat de l'utiliser comme un langage standard de description de formats de documents.

D'autre part, nous avons besoin de décrire des propriétés autres que le format d'un document, par exemple, le type de données, le traitement des paragraphes de clarification, l'encryptage, etc... Les descripteurs correspondants doivent donc accompagner le document transmis.

Dans cette section, nous développons le format de représentation d'un document transmis. Les divers champs de description sont définis formellement en détail. Pour clarifier ces formules (écrites en pseudo-BNF) nous avons traité en détail un exemple complet (voir Annexe B).

La structure et le formattage d'un document sont représentés par des descripteurs insérés dans le texte lui-même dans les endroits suivants selon le type du descripteur:

- Au début du document ou d'une section (c'est-à-dire avant le premier paragraphe de cette section)
- Au début d'un paragraphe
- Au début et à la fin d'une chaîne de caractères (à l'intérieur d'un paragraphe).

En général, notre idée est de ne mettre des descripteurs que pour déclarer le changement d'un attribut donné. Par conséquent, les descripteurs n'apparaissent qu'un minimum de fois et les champs de description ne sont pas régulièrement insérés avant tout paragraphe.

Le format général d'un document avec ses descripteurs est le suivant:

‹document› = {‹bloc section›}‹fin texte› (où "{ }" indiquent la répétition)

où:

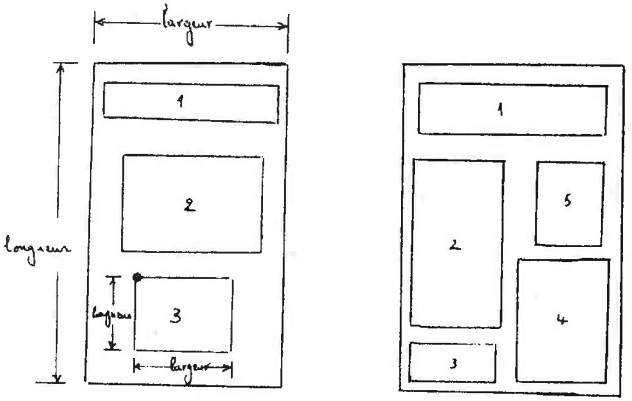
‹fin texte› = ".X "

‹bloc section› = ‹descripteur format›{‹bloc paragraphe›}

Les différents champs peuvent être définis formellement comme suit:

** ‹descripteurs format›

D'un point de vue matériel, le document se compose de pages divisées en cadres rectangulaires destinés à contenir les différentes parties du document ‹28› (Fig.III.7.). Des descripteurs de format sont introduits au début du document, d'une page, d'un paragraphe ou d'un caractère.



Page 1

Page 2
(et suivantes)

Fig.III.7. Format des pages et des cadres

Le champ ‹descripteur format› définit des formats standard des pages
et des cadres dans chaque page.

‹descripteur format› = ‹indicateur début›‹type de support›
{‹format page i›{‹format cadre j de la page i›}$_{j=1 \text{ à } nj}$}$_{i=1 \text{ à } ni}$

‹indicateur début› = "descripteur format" ‹nombre de formats de page, i›
‹type support› = "écran" ou "papier"
‹format page› = ‹numéro du format de page›‹longueur›‹largeur› ‹nombre de cadres dans la page›
‹format cadre› = ‹numéro cadre›‹coordonnés du coin gauche› ‹longueur cadre›‹largeur cadre›

**** ‹bloc paragraphe›**

Le ‹bloc paragraphe› se compose principalement de champs descripteurs suivis du texte même du paragraphe. Ce texte peut contenir des codes de formattage indiquant la qualité d'impression d'une chaine (exemple: soulignée) ou un découpage en chaines particulières (exemple: éléments d'une table). La fin d'un bloc paragraphe est indiquée par un code qui signifie en même temps le début éventuel du bloc suivant. Les définitions formelles du ‹bloc paragraphe› et de ses composants sont les suivantes:

‹bloc paragraphe› = **‹indicateur début paragraphe›**
‹descripteur interne du paragraphe›
[‹descripteur format du paragraphe›]
‹valeur› ‹code fin paragraphe›

"[]" indique la possibilité d'absence d'un champ
où:
* **‹indicateur début parag.›** = ‹indicateur encryptage› ‹indicateur de subordination›‹type›‹longueur d'adresse›

‹indicateur encryptage› = un bit mis à 1 s'il y a encryptage, indiquant qu'un champ ‹technique d'encryptage› existe
‹indicateur subordination› = un bit mis à 1 pour les paragraphes de clarification. Ceci indique en même temps l'existance du champ ‹code affichage d'un paragraphe subordonné›
‹type› = un bit mis à 1 si le type des données incluses est différent de celles du paragraphe précédent. Ceci indique en même temps qu'un champ ‹type de données› existe

‹**longueur d'adresse**› = 4 bits représentant le nombre d'éléments de l'adresse hiérarchique (= le niveau hiérarchique de la feuille). La profondeur hiérarchique d'un document ne peut donc pas dépasser 15 (ceci nous semble largement suffisant, mais il est possible d'étendre ce champ si nécessaire).

* ‹**descripteur interne du paragraphe**› = [‹technique d'encryptage›] [‹type de données›]‹adresse hiérarchique›

* ‹**descripteur format du paragraphe**› = ‹indicateur début format paragraphe›[‹séquence des cadres à remplir›][‹code affichage d'un paragraphe subordonné›][[‹commandes de formattage›]]

  ‹**indicateur début format paragraphe**› = "format paragraphe"‹nP› (où nP = nombre de pages standards utilisées dans la séquence des cadres à remplir)

  ‹**séquence des cadres à remplir**› = {‹numéro du format de page› ‹nombre des cadres utilisés dans cette page›{‹numéro de cadre›}|

  Exemple:

  la ‹séquence des cadres› peut être définie par la suite:

  ‹page1›‹2›‹cadre5›‹cadre6›‹page2›‹1›‹cadre1›

  Ceci signifie qu'à partir du paragraphe courant, on remplira l'espace correspondant aux cadres 5, et 6 d'une page ayant le format numéro 1 puis le cadre 1 d'une page ayant le format numéro 2. Si ces cadres sont insuffisants, le reste du texte sera écrit dans le format de la dernière page utilisée (dans ce cas ‹page 2›‹1›‹cadre 1›).

  ‹**code affichage d'un paragraphe subordonné**› ="code" désignant les propriétés et les conditions d'affichage (physique) d'un paragraphe subordonné. Surtout en ce qui concerne sa répétition en relation logique avec une section donnée et sa liaison physique avec une page, cadre ou ligne.

  ‹**commandes de formattage**› = une des commandes décrites dans les sections 4. de l'Annexe A.

* ‹**valeur**› = {‹chaine›}

  ‹**chaine**› = [‹code de formattage›]‹texte›

  ‹**code de formattage**› = Dans le champ ‹code de formattage› on peut insérer des descripteurs de qualité d'affichage (souligné, double intensité, etc...) ou de contrôle d'accès (protection des données, types des valeurs acceptables dans un paragraphe vide, etc...) ou un code de liaison avec un subordonné (ex. renvois). Nous terminons l'effet de chaque descripteur par un code ‹fin effet› particulier à chaque qualité (voir 4.2. en Annexe A).

  ‹**texte**› = ‹chaine de caractères›/‹adresse hiérarchique›(si le type est "Pointeur")/‹valeur initiale›‹pas d'incrémentation›(si le type est "Expression")

* ‹**fin paragraphe**› = ".R" (voir 4.2.1. en Annexe A)

III.4.4. Caractéristiques d'un document en Transmission

Nous préconisons une seule forme standard du document circulant sur le réseau, forme uniquement révisable.
Nous avons choisi cette forme pour les raisons suivantes:

1- Elle permet de représenter des documents existants sous une forme quelconque (en utilisant un déformatteur si le document existe en forme finale (voir III.4.5).

2- Elle simplifie la modification d'un document ou son reformattage dans une station quelconque du réseau (puisqu'il est révisable).
Notons qu'il serait souhaitable d'avoir un moyen d'accès à une section quelconque d'un document en utilisant son adresse hiérarchique.

3- La taille du document transmis est optimisée (puisqu'il ne contient pas d'espaces ou lignes vides inutiles).

4- Le nombre de traducteurs nécessaires pour N stations différentes est 2xN (et pas Nx(N-1) dans le cas de traducteurs ad-hoc).

5- Un inconvénient: un document en forme finale sera traduit inutilement dans les deux sens. Ceci est toutefois compensé par les avantages cités ci-dessus.

III.4.5. Traitement d'un document

Avant qu'un document soit transmis, il est traité par un traducteur qui le met en forme standard (décrite dans III.4.3.) en insérant les descripteurs appropriés. Deux cas se présentent:

1- Le document original est en forme révisable:
   Chaque commande de formattage explicite ou implicite est traduite en une (ou plusieurs) commande(s) standard(s) mise(s) à l'endroit approprié.

2- Le document est en forme finale:
   Un "déformatteur" se charge de la représentation du document formatté en termes d'instructions de formattage affectant des données non-formattées. Une réalisation d'un déformatteur est décrite au cours du chapitre .IV. (prouvant ainsi la faisabilité de cette idée). Elle utilise le langage de formattage standard décrit au chapitre .I.

D'autre part, un document reçu par une station est toujours en forme révisable standard. Il doit être "traduit" en termes du langage de formattage de cette station réceptrice afin qu'elle puisse éventuellement le traiter ultérieurement. La station doit également changer les codes de contrôle en des codes compris par les unités d'entrée-sortie de cette station.

III.5. COMPARAISONS ET CRITIQUES

Dans cette section, nous présentons une comparaison générale des divers aspects des propositions A,B,C et D.

III.5.1. La structure logique

a) Les propositions A et B représentent la hiérarchie par deux paramètres qui sont le niveau et le rang dans le niveau (Fig.III.2.).
Ceci est criticable car:

1. La hiérarchie logique n'est pas représentée.
2. La relation de subordination n'existe pas et n'est pas représentable.
3. Il n'existe pas de relation directe entre cette numérotation et les positions physiques relatives de deux entités appartenant à des niveaux différents.

b) La proposition C ne se préoccupe pas beaucoup de la définition de structures logiques claires.

c) Dans la proposition D, la structure logique d'un document est clairement définie par les relations diverses représentées par l'adresse hiérarchique (n-uplet) et le bit de subordination.
L'inconvénient de cette méthode est la longueur de l'adresse (n octets) et son association à chaque "paragraphe" mais, généralement, le surplus d'octets ne dépassera pas 1% de la longueur du document et permettra, par contre, toutes les opérations d'accès à une entité quelconque (Fig.III.6.). En plus, une relation directe existe entre l'adresse hiérarchique d'un paragraphe et sa position relative sur le support de sortie.

III.5.2. Les types de données

a) Les propositions A et B présentent la notion de type d'une manière détaillée (surtout B qui introduit de nouveaux types).

b) La proposition C ne tolère l'hétérogénéité des types que dans la forme finale d'un document.

c) La proposition D permet la co-existence de plusieurs types mais ne traite en détail que les types textuels (elle en définit trois).

III.5.3. Descripteurs d'un document

Dans toutes les propositions, les descripteurs apparaissent avant l'unité minimale du document. Par contre, la proposition D permet la suppression de quelques champs en signalant leur absence dans l'‹indicateur du début›.

Les techniques de description du formattage sont légèrement différentes mais en gros équivalentes. Toutefois, D a l'avantage d'utiliser un langage formé d'un minimum de commandes et utilisant une technique d'adressage puissante.

Les propositions A et D permettent de définir des cadres ou boîtes à l'intérieur d'une page. En plus, D permet la définition de la séquence de remplissage des cadres.

III.5.4. Le document en transmission

Contrairement à la proposition C, les propositions B et D ne transmettent des documents qu'en une seule forme (finale pour B et révisable pour D). Ceci a ses avantages (surtout l'optimisation du nombre des traducteurs) et ses inconvénients (surtout la nécessité de tout transformer en cette forme choisie). Le choix de B est plus utile dans les systèmes de messagerie où les documents ne sont pas ré-éditables, et celui de D est meilleur si on prévoit des modifications des textes dans les différentes stations.

La proposition D est la seule à avoir traité le problème du "déformattage" d'un document, ce qui permet la transmission et la ré-édition de documents créés sur des systèmes n'ayant pas de logiciel de traitement de textes ou créés par des personnes ne sachant pas utiliser ce logiciel s'il existe.

Il faut ajouter que le choix des commandes de formattage dans la proposition D facilite largement la traduction des commandes des différents systèmes de traitement de textes ainsi que le "déformattage" d'un document.

CHAPITRE .IV.

REALISATION D'UN TRANSFERT DE DOCUMENT ENTRE DEUX SYSTEMES HETEROGENES

IV.1. Introduction

IV.2. Le matériel utilisé

IV.3. Le déformatteur

IV.4. La transformation de la forme standard en forme locale

CHAPITRE .IV.

REALISATION D'UN TRANSFERT DE DOCUMENT ENTRE DEUX SYSTEMES HETEROGENES

IV.1. INTRODUCTION

Au cours du chapitre précédent, nous avons examiné les problèmes posés par le transfert d'un document entre deux systèmes de traitement de textes hétérogènes, et nous avons proposé une solution consistant en une forme standard des documents.

Le présent chapitre décrit une partie des outils nécessaires à la mise en oeuvre de cette solution.

Par ces réalisations, nous avons voulu prouver la faisabilité de notre proposition et, en même temps, résoudre le problème concret posé par le projet d'informatisation du bulletin de liaison du Centre de Recherche en Informatique de Nancy ‹51›. Il s'est avéré souhaitable que les auteurs des différents articles puissent éditer et/ou formatter leurs textes sur des postes de travail différents, puis transférer ces textes à travers un réseau vers le poste du rédacteur- en-chef (REC).

Le REC est souvent amené à re-éditer un texte, modifier son format pour l'inclure dans le bulletin, ou, tout simplement, l'insérer entre deux articles pour créer un fichier texte contenant l'ensemble des articles d'un numéro du bulletin.

Normalement, la modification d'un texte entraine la nécessité de le reformatter. Deux classes de textes peuvent être reçues par le REC:

1. Texte formatté prêt à l'impression.
2. Texte compacté accompagné des commandes de formattage appropriées.

Si le texte est reçu sans commandes de formattage, il est impossible de le reformatter sans la réinsertion de celles-ci.

Si le texte est reçu avec des commandes de formattage non-connues du poste du REC, il faut les traduire avant de pouvoir les interpréter.

Il apparait nécessaire de disposer d'un "transformateur" qui traduit le document créé en une forme standard de représentation d'un document transmis. Et de l'autre côté, un autre "transformateur" transforme le document transmis (en forme standard) en un document dont la forme est spécifique du système récepteur.

Deux tels "transformateurs" sont présentés après une brève description du matériel destiné à les supporter et sur lesquels ils ont été réalisés.

IV.2. Le Matériel Utilisé

Les deux systèmes hétérogènes utilisés dans cette expérience sont:

(a) Un micro-ordinateur MICRAL 80/30 construit autour du microprocesseur Z-80 et exploité sous le système multi tâches PROLOGUE. Cette machine dispose de:

1. 64K RAM.
2. Deux unités de disques (l'une fixe et l'autre amovible) d'une capacité de 10M octets chacune.
3. Deux unités de minidisquettes (5 pouces) de 130K octets chacune.
4. Une imprimante à aiguille.
5. Un écran-clavier non-intelligent, géré par la machine elle-même.

Le Micral 80/30 est également le site central d'un réseau homogène étoilé (SOR) comprenant deux machines Micral 80/21.

Le logiciel existant sur le 80/30 comporte un assembleur AZM-80, un éditeur de textes et de programmes (Word Star), un système de mise au point de programmes, un éditeur de liens, et un système de gestion de fichiers. Un certain nombre d'interpréteurs et de compilateurs de langages de haut niveau existent également mais ne nous intéressent pas dans cette réalisation.

(b) Un micro-ordinateur North-Star Horizon construit autour du micro processeur Z-80 et exploité sous le système CP/M (multi-utilisateurs). Cette machine dispose de:

1. 64K RAM.
2. Deux unités de mini disquettes (5 pouces) de 360K chacune.
3. Une imprimante SANDERS à 10 polices de caractères simultanées.
4. Un terminal semi-graphique FALCO TS1 permettant d'afficher simultanément à l'écran deux jeux de caractères complets.

Le logiciel existant sur cette machine comporte les utilitaires CP/M, et un logiciel "Scientexte" adapté au traitement de textes scientifiques.

IV.3. Le Déformatteur

Un "déformatteur" est un programme qui transforme un texte formatté en un autre texte non formatté (donc, plus compact) accompagné des fonctions de formattage décrivant le format d'origine.

IV.3.1. Idée générale du déformatteur

Un texte, dans sa forme finale de distribution, se compose d'un ensemble de chaines alphanumériques, qui expriment les idées de l'auteur.

Un langage naturel est utilisé pour construire ces chaines. Pour chaque langage, des règles particulières définissent la ponctuation de ces chaines: c'est-à-dire à la fois leur sémantique et leur syntaxe.

Lors du déformattage (ou compactage) d'un texte, il faut absolument garder la trace des passages obligatoires à une nouvelle ligne car ce découpage -à la fois physique et logique- du texte en paragraphes (et donc, en idées distinctes) est très important du point de vue sémantique.

Notons aussi que les commandes de formattage ne pouvant être insérées qu'au début d'une nouvelle ligne, on peut donc décrire -au moyen de ces commandes- l'état des nouvelles chaines qui suivent jusqu'au prochain point de coupure obligatoire.

Il est essentiel que le déformatteur distingue entre deux types de Retour Chariot "R.C." qu'il peut rencontrer dans un texte imprimable. Le premier concerne la coupure à l'intérieur d'un paragraphe ou d'une phrase à cause du manque d'espace dans la ligne physique actuelle; on est donc obligé de "continuer" sur la ligne suivante (voir Fig.IV.1.).

Par contre, l'autre type de "R.C." est utilisé, justement, pour marquer la "discontinuité" entre deux phrases (ou en fait deux paragraphes, ou deux alinéas dans le même paragraphe).

Il faut donc que le déformatteur cherche une "caractéristique" associée au "R.C." pour distinguer entre le R.C. qui ne coupe pas une ligne obligatoirement et les autres types de "R.C." qui coupent une ligne obligatoirement pour avoir un nouveau paragraphe ou un nouvelle alinéa dans le même paragraphe. Et, après cette étape, le déformatteur insère les fonctions de formattage décrivant chaque paragraphe rencontré.

Terminons enfin par les langages de programmation les plus couramment utilisés : LISP (ou INTERLISP) , APL , PL/1, PROLOG. Le premier présente le danger de mal séparer le modèle de représentation de sa programmation, mais offre des facilités. Le dernier constitue à lui seul un résolveur de problèmes, sans offrir cependant de traitement des échecs ni de détection des boucles. Il faut là encore se méfier et essayer de faire des mesures, comme on l'a déjà dit pour les formalismes de représentation des connaissances.

Les nœuds sont reliés entre eux par un réseau fortement maillé de canaux rapides : entre deux nœuds reliés, il y a au moins deux liaisons à 72000 bauds [ADM-78 *f*]. Enfin une surveillance permanente du réseau est assurée au niveau des centres de gestion nationaux (gestion du réseau) et locaux (contrôle des nœuds).

b) Accès au réseau
Les utilisateurs peuvent être raccordés directement aux nœuds du réseau TRANSPAC ou y accéder dans le cadre d'un abonnement au réseau téléphonique ou télex (*fig.* 10.6).

FIG.IV.1. Coupure des lignes dans un texte

IV.3.2. Description Générale du Programme

Ce programme est réalisé sur le système Micral en Assembleur Z-80. Le schéma général du traitement est donné en Fig.IV.2.

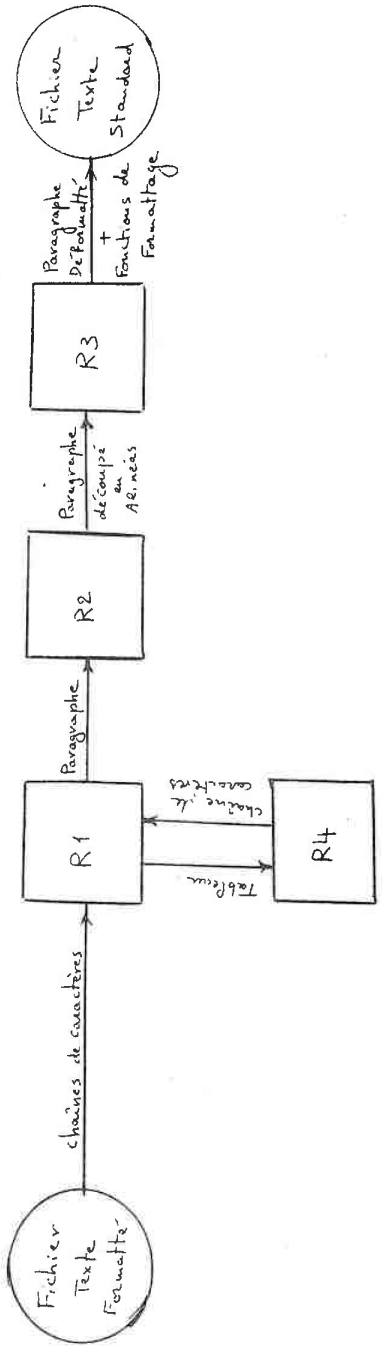


Fig.IV.2. Flux Des Données

Nous décrivons ci-dessous les fonctions assurées par les différents modules.

Module R1

- Détecte la fin du paragraphe commencé et passe au module R2 quand cette fin est détectée.
- Indique si un code "fin-texte" est rencontré.
- Elimine les lignes vides précédant un paragraphe et les remplace par la commande primitive adéquate.
- Indique si un tableau commence. Dans ce cas il termine le paragraphe actuel et utilise le module R4 pour transformer un tableau entier en une suite de colonnes contiguës. Ces colonnes sont découpées en paragraphes (éléments) par les procédures normales de R1.

Module R4

- Est appellé par R1, si celui-ci rencontre un tableau.
- Lit un tableau complet et le mémorise sous forme d'une suite de colonnes consécutives.
- Retourne à R1 pour découper chaque colonne en paragraphes

Module R2

- Détecte les points de passage obligatoire à une nouvelle ligne dans un paragraphe donné et insère les codes "nouvelles lignes".
- Compacte le texte par élimination des blancs non nécessaires.

Module R3

- Insère les commandes primitives de formattage (voir organnigramme en Fig.IV.3.).
- Compacte le texte (élimine les lignes vides à l'intérieur du paragraphe courant).
- Enregistre -dans un nouveau fichier texte standard- le paragraphe traité.

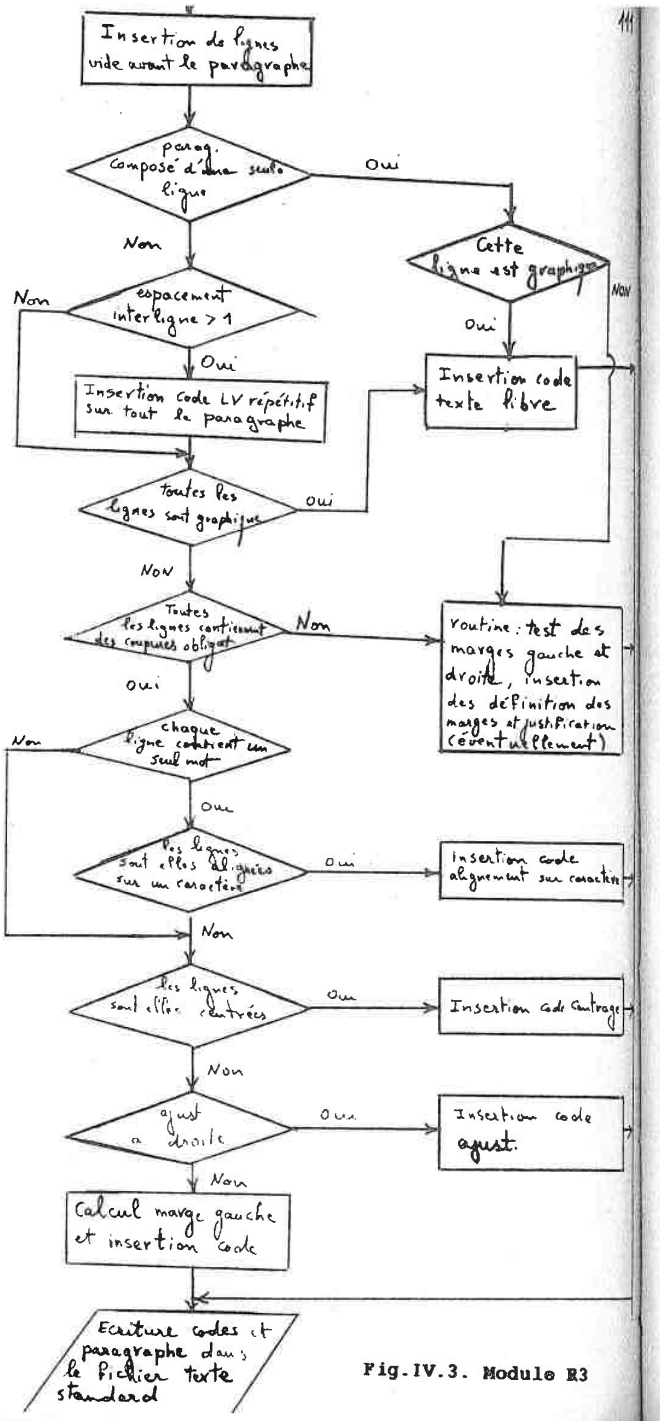


Fig.IV.3. Module R3

IV.3.3. Détails des traitements effectués

IV.3.3.1. Découpage du texte en paragraphe

Le lecteur humain a la possibilité de reconnaitre les différents paragraphes, nouvelles lignes, etc....

Mais pour une machine, il est nécessaire de définir des indicateurs explicites (reconnaissance de syntaxe) permettant au programme de détecter les coupures obligatoires et mettre le code nécessaire.

Nous avons remarqué que la reconnaissance d'un début de paragraphe peut être associée à:

(a) Une indication implicite par l'insertion d'un code de coupure obligatoire d'une page.

(b) La rencontre d'une partie du texte arrangée sur les lignes physiques sous une nouvelle forme (tableau, ou changement des points de tabulation). Un exemple est donné dans la Fig.IV.4.

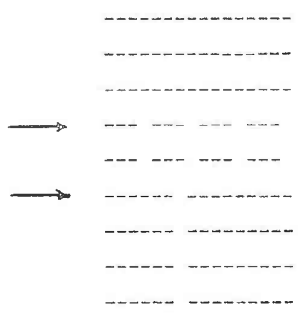


Fig.IV.4.

(c) Une désignation spéciale au début de la première ligne d'un paragraphe indiquant le début d'une nouvelle entité logique. Cette désignation peut être un (ou plusieurs) caractère(s) special(ux), précédé(s) éventuellement par un(des) caractère(s) indiquant une numérotation de ces entités. Un exemple est donné dans la Fig.IV.5.

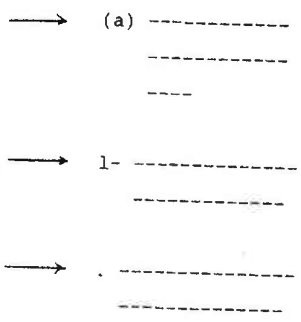


Fig.IV.5.

(d) Un changement de la marge gauche. Un exemple est donné dans Fig.IV.6.

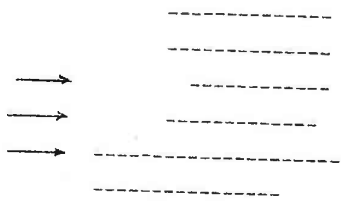


Fig.IV.6.

Les divers cas possibles si un changement de marge gauche est détecté sont schématisés dans l'arbre de décisions suivant (voir Fig.IV.7.) :

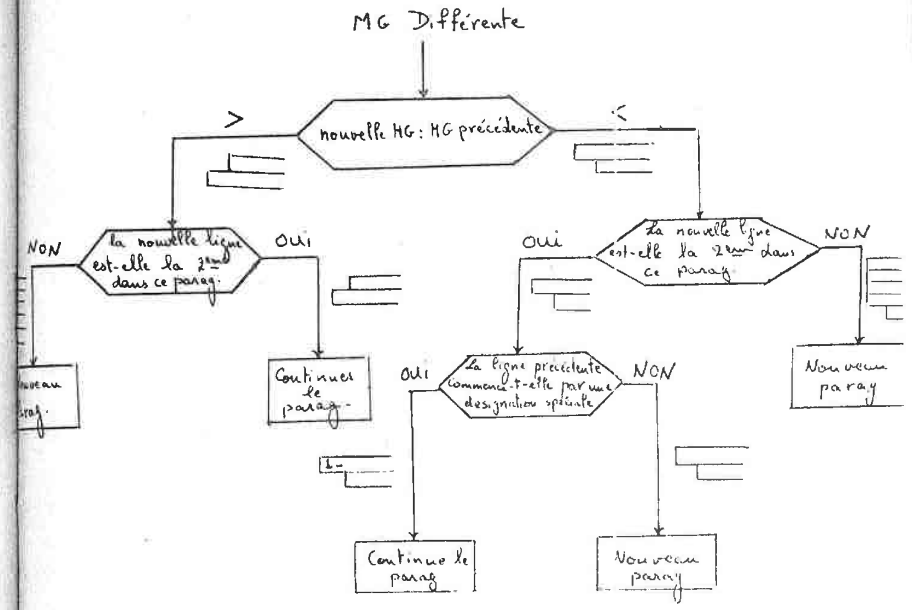


Fig.IV.7.

(e) L'insertion d'un nombre de lignes vides avant le nouveau paragraphe. Supposons que "n2" est le nombre de lignes vides avant la ligne actuelle, et "n1" est le pas précédent (en lignes vides). Deux cas peuvent se présenter (voir Fig.IV.8.)

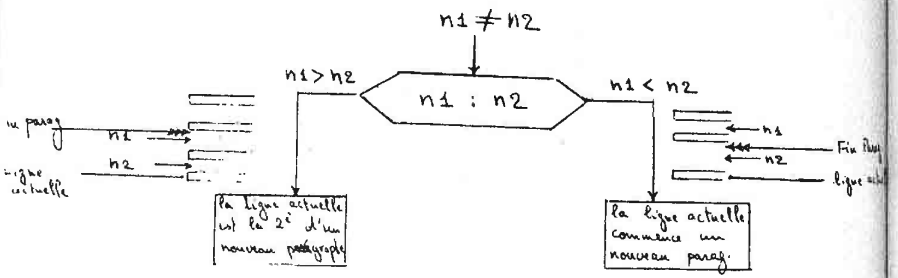


Fig.IV.8.

IV.3.3.2. Découpage d'un paragraphe en alinéas

Il faut noter qu'il m'est apparu difficle de déterminer si une ligne se terminant par un point implique un passage à une nouvelle ligne après celle-ci, ce point pouvant être mis par exemple à cause d'une abréviation.

Le système examine donc chaque ligne dans le paragraphe actuel, et il met un code de passage obligatoire à une nouvelle ligne dans les cas suivants:

(a) Après une ligne contenant un seul mot (voir exemple dans la Fig.IV.9.)

● Le réseau TRANSPAC

a) Architecture
TRANSPAC est un réseau public de transmission de données par paquets ouvert au public en janvier 1979 (*fig.* 10.5). Ce réseau est composé de nœuds assurant les fonctions de concentration et de commutation des données (chaque nœud comporte 2 types de matériel : un ou plusieurs modules de commutation CP 50 et une unité de commande MITRA 125). Un nœud peut recevoir jusqu'à 16 000 lignes d'abonnés. Afin de garantir un haut degré de disponibilité, chaque matériel existe en 2 exemplaires.
Le tableau suivant indique le nombre de nœuds prévus pour TRANSPAC.

FIG.IV.9.

(b) Après une ligne ne contenant pas des lettres minuscules, par exemple, un titre ou les exemples donnés dans Fig.IV.10.

b) Définition du site de stockage d'informations
Tout langage de commande local possède une commande particulière permettant de décrire un ensemble de données. L'accès à des informations non locales nécessite également l'introduction du paramètre SITE dans cette commande.
Exemple : 1. utilisation en *A* d'un fichier stocké en *B* dans un réseau de systèmes OS-360 sous JCL
// DD ... (SITE = *B*)
La carte DD (Data Definition) contient alors l'indication du site.
2. même utilisation mais dans un réseau de systèmes SIRIS 7-8
! ASSIGN... (SITE = *B*).

Fig.IV.10.

(c) Après une ligne ne contenant pas des mots commençant par une lettre minuscule, par exemple, un titre ou des lignes d'adresse. Voir exemple dans Fig.IV.11.

Direction : De l'ETD
Les signaux transmis sur ce circuit indiquent s'il y a une probabilité raisonnable d'erreur dans les données reçues sur la voie de données. La qualité de signal indiqué est conforme aux spécifications appropriées de l'Avis relatif à l'ETD.
L'état FERMÉ indique qu'il n'y a pas de raison de croire qu'une erreur s'est produite.
L'état OUVERT indique qu'il y a une probabilité raisonnable d'erreur.

Fig.IV.11.

(d) Avant une ligne ne commençant pas par une lettre minuscule et dont le premier mot pouvait être inséré à la fin de la ligne précédente sans débordement de celle-ci (découpage volontaire par l'auteur). Voir exemple dans la Fig.IV.12.

Pour les chiffres les positions de zone sont à 1 et les positions numériques constituent le code pondéré 8 - 4 - 2 - 1 du chiffre considéré.
Il faut noter que, comme le code ASCII, le code EBCDIC permet le codage des majuscules et des minuscules et contient également les fonctions de commande pour la transmission de données. Beaucoup de configurations restent inutilisées.

Fig.IV.12.

En plus de ces conditions, dans le cas particulier d'un paragraphe ne commençant pas par une désignation spéciale d'une entité logique, on passe à une nouvelle ligne après chaque ligne du paragraphe actuel si:

(a) Chaque ligne commence par une lettre majuscule (éventuellement des lignes d'adresse). Voir un exemple dans Fig.IV.13.

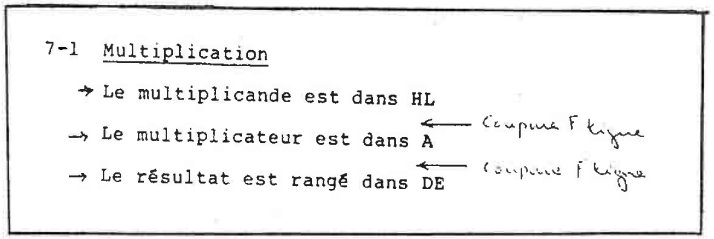


Fig.IV.13.

(b) Le premier caractère du paragraphe n'est pas une lettre majuscule (paragraphe anormal). Voir exemple dans la Fig.IV.14.

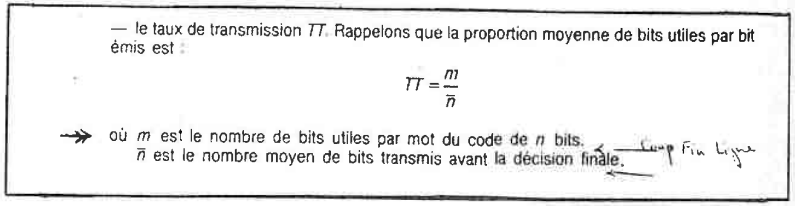


Fig.IV.14.

IV.3.3.3. Insertion des commandes de formattage

Cette tâche est généralement simple car les formats impliquant l'utilisation des primitives de formattage sont facilement détectables. Deux cas sont ambigüs:

1. Si un groupe de lignes a des marges gauche et droite égales, on ne peut pas dire si elles sont centrées, ajustées à droite, ou tout simplement si elles forment un groupe normal justifié par hasard. Dans ce cas, ce groupe est traité comme étant un groupe normal. Voir exemple dans la Fig.IV.15.

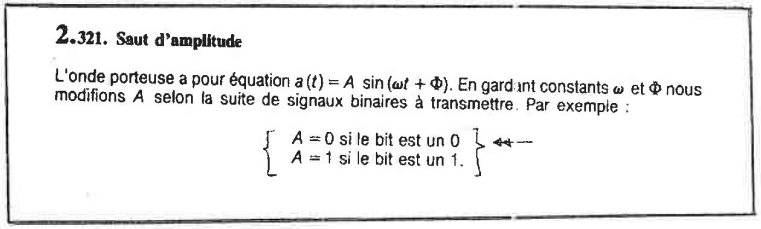


Fig.IV.15.

2. Si un paragraphe contient une ligne formée entièrement de caractères spéciaux (impliquant une certaine forme graphique), tout le paragraphe est pris comme un texte libre pour ne pas détruire une figure éventuelle. Voir un exemple dans la Fig.IV.16.

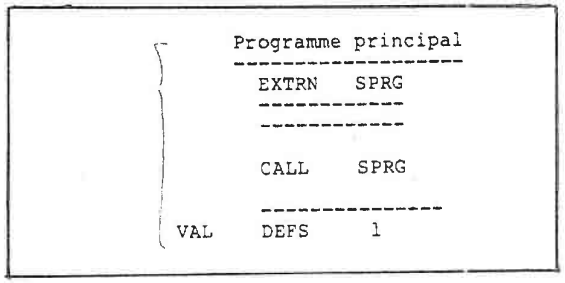


Fig.IV.16.

IV.3.3.4. Traitement des tableaux

Le module R4 est appelé pour décomposer un tableau en une séquence de colonnes entières de celui-ci. Cette décomposition se fait de la manière suivante:

R4 lit tout le tableau (c'est-à-dire toutes les lignes ayant les mêmes points de tabulation) et le range dans un tampon en mémoire.

Ensuite, il demande à R1 de lire dans ce tampon en considérant que les marges sont les limites de la première colonne, puis la deuxième et ainsi de suite. R1 fera appel à R2 puis R3 après la détection de chaque paragraphe.

R4 se charge également de l'insertion des codes indiquant les débuts et fins des colonnes pour les éléments composant le tableau.

Un exemple est donné dans la Fig.IV.17.

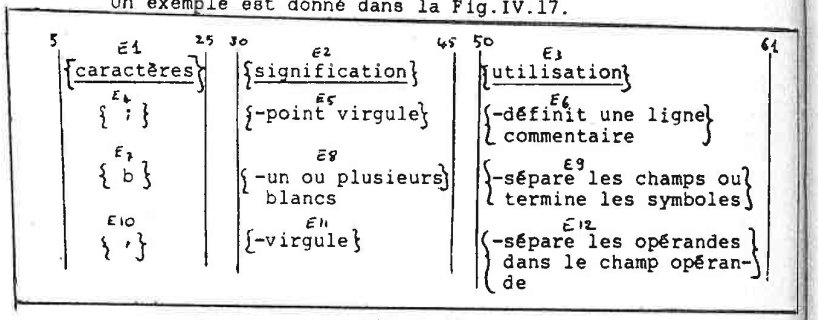


Fig.IV.17.

Le tableau précédent composé de 3 colonnes de 4 éléments chacune.

5	25	30	45	50	61
E1		E2		E3	
E4		E5		E6	
E7		E8		E9	
E10		E11		E12	

Sera décomposé en une suite:

```
;PT,5,25,30,45,50,61      = Points de tabulation
.C                        = Début colonne
E1 E4 E7 E10              = Transformé en forme standard
.FC                       = Fin colonne
.C
E2 E5 E8 E11
.FC
.C
E3 E6 E9 E12
.FC
```

IV.3.4. Extension Possible

Quelques extensions peuvent éventuellement être ajoutées au programme pour traiter des cas plus compliqués. Par exemple:

1. Détection et traitement des en têtes et des notes en bas d'une page.
2. Détection et traitement des numéros de page.

IV.3.5. Exemple d'un texte déformatté

TRANSFERT DE DOCUMENTS ENTRE S.T.T.
HETEROGENES

INTRODUCTION AU PROBLEME

Je m'intéresse plus spécialement au problème du transfert entre systèmes de traitement de textes (STT) hétérogènes de textes formattés (ou non-formattés accompagnés de fonctions de formattase). Les STT peuvent évidemment être connectés par un même réseau local ou à travers un réseau distant: ceci n'a aucune influence sur notre étude.

Le problème apparaît lorsqu'on veut re-éditer ou re-formatter un texte déjà formatté sur une autre machine. Un exemple se présente dans l'édition du bulletin de liaison périodique du laboratoire CRIN à NANCY (1),où les auteurs pourront écrire leurs articles sur des postes de travail différents puis transférer ces textes à travers un réseau vers le poste du rédacteur en-chef (REC).

```
;LV,6, ,
;MG,8, ,FR
;MD,80, ,FR
TRANSFERT DE DOCUMENTS ENTRE S.T.T.
.R
;MG,17, ,FR
;MD,80, ,FR
HETEROGENES
.R
;LV,2, ,
;MG,0, ,FR
;MD,80, ,FR
INTRODUCTION AU PROBLEME
.R
;LV,1, ,
;MG,3, ,1L
;MG,0,1,FR
;MD,54, ,FR
Je m'intéresse plus spécialement au problème du transfert entre systèmes de tra
itement de textes (STT) hétérogènes de textes formattés (ou non-formattés accom
pagnés de fonctions de formattage). Les STT peuvent évidemment être connectés
par un même réseau local ou à travers un réseau distant: ceci n'a aucune influe
nce sur notre étude.
.R
;LV,1, ,
;MG,3, ,1L
;MG,0,1,FR
;MD,80, ,FR
Le problème apparaît lorsqu'on veut re-éditer ou re-formatter un texte déjà for
matté sur une autre machine. Un exemple se présente dans l'édition du bulletin
de liaison périodique du laboratoire CRIN à NANCY (1),où les auteurs pourront
écrire leurs articles sur des postes de travail différents puis transférer ces
textes à travers un réseau vers le poste du rédacteur en-chef (REC).
.R
.F
```


IV.4. LE TRAITEMENT DE LA FORME STANDARD EN FORME LOCALE

Ce programme écrit en Assembleur Z-80 effectue la transformation d'un document à sa réception par la machine NORTH-STAR. Le document reçu est en forme standard et est transformé en forme compatible avec le système de traitement de texte Scientexte.

L'organigramme du traducteur est donné en Fig.IV.18.

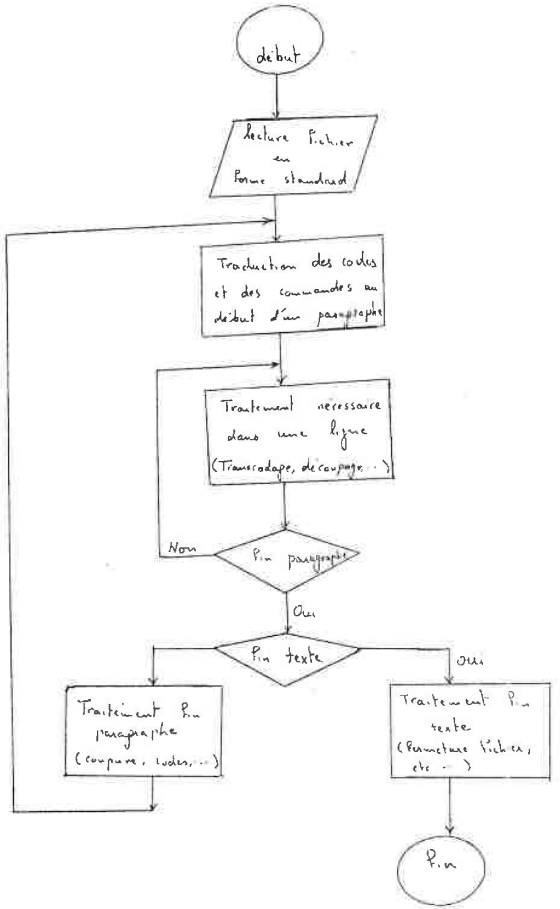


Fig.IV.18. Organigramme du traducteur

Présentation des données

Nous considérons que le document reçu se trouve dans un fichier "Document Reçu" (DR). Il a le format standard tel qu'il est décrit dans II.3 et section 4 de l'Annexe A. Les blocs de ce fichier sont appelés en RAM l'un après l'autre pour être transformés et réécrits consécutivement dans un nouveau fichier "Document Transformé" (DT).

Opérations effectuées

1. Au début du DT, on insère les descripteurs de page (longueur, largeur).
2. Au début de chaque paragraphe de DR, on détecte les commandes de formattage, et on les traduit en une (ou plusieurs) commandes de Scientexte qui sont insérées avec le texte du document et le tout est enregistré dans DT.
3. Les codes délimitant des chaînes spéciales (par exemple, début d'en tête) sont également traduits en codes Scientexte.
4. Découpage du texte en lignes (car ceci n'existe pas dans la forme standard).
5. Execution du transcodage des caractères de contrôle inclus dans le texte.

Un problème: La traduction des commandes ayant des champs d'adressage

Le système Scientexte ne reconnaissant pas l'adressage logique d'un texte, quelques décisions ont été prises pour pallier cela:

1. Une commande dont l'effet ne commence pas à la ligne qui la suit immédiatement est insérée juste avant la première ligne qu'elle affecte. (après le découpage en lignes).

Exemple:

En langage standard:

```
;MG,3,,1L
;MG,0,1,FR
----------------------------
----------------------------
----------
.R
```

En Scientexte ce sera:

```
.Y , , ,..,12, ,...,
------------------ 1ère ligne coupée selon la longueur
                   des lignes

.Y , , ,..,9, ,...,
---------------------------- à partir de la 2ème ligne
----------------------------
```

2. L'adressage de la limite logique de l'effet d'une commande se traduit par la répétition de l'insertion de la commande correspondante avant les lignes contenant l'entité logique définie.

Exemple:

En langage standard:

```
;CT,,,FR
----------------
----------------
----------------
.R
```

En Scientexte ce sera:

```
‹.c---------------›
‹.c---------------›
‹.c---------------›
```

3. Les fonctions qui n'existent pas dans Scientexte (comme la tabulation) sont exécutées sur le texte qui sera enregistré en forme finale (chaine de caractères plus codes de contrôle de points de tabulation).

Exemple

En langage standard:

```
;AL,.,20,FR
33.555
555.876
40999.33
.R
```

En Scientexte ce sera:

```
.Y , , ..  ,.. = pas de justification
  ▼▼▼33.555
   ▼▼555.876
    40999.33
```

Les trois lignes sont précédées par les blancs nécessaires pour faire le justification

CONCLUSION

Dans ce travail nous avons essayé de résoudre quelques problèmes rencontrés lors de l'informatisation de l'édition, du formattage et de la distribution d'un document quelconque.

Plus concrètement, nous avons effectué les travaux suivants:

1- Extraction du noyau d'un système extensible d'édition et de formattage de textes.

2- Conception d'un langage de traitement de textes simple et modulaire basé sur les fonctions du noyau extrait.

3- Réalisation du noyau et de l'interpréteur du langage.

4- Développement d'une forme de représentation commune pour le transfert de textes sur un réseau bureautique. Cette forme est basée sur le langage modulaire conçu et présente quelques avantages certains.

5- Réalisation d'un "déformatteur" de textes.

6- Réalisation de la transformation d'un texte en forme "standard" en un texte equivalent mais en forme "locale" révisable.

Néanmoins, nous pensons que l'effort doit se poursuivre pour la généralisation de notre forme de représentation de documents afin qu'elle soit applicable aux documents non-textuels.

BIBLIOGRAPHIE

‹1› Adams, R.C. & Cleave, G.A. "System Design for Text Processing" Proc. of the Eighth Australian Computer Conference, Vol.1, Août-Sept. 1978

‹2› Andrews, D.I.; Boli, B.R. & Poggio, A.A. "AN Introduction to the NSW Frontend" SRI-ARC Report No. 28743, 1977

‹3› idem "A Guide to the Command Meta Language and the Command Language Interpreter" SRI-ARC Report No. 28744, 1977

‹4› Boli, B.R.; Lehtman H.; Michael, E.; Rom, R.; Van Nouhuys, D. & Zolotow, N. "A Model Production System: Document Production and Control System Design Study" SRI-ARC, Report No. 29000, Juil. 1977

‹5› Boli, B.R.; Michael, E.; Rom, R.; van Nouhuys, D. & Weinberg, A. "Document Production and Control System Design Survey" SRI-ARC, Report No. 37730, Dec 1976

‹6› Borkin, S.A. & Prager, J.M. "Some Issues in the Design of an Editor Formatter for Structured Documents" IBM Cambridge Scientific Center, Sept. 1981

‹7› Boughlam, A. et al. "Le Projet Burimag" Actes de Congrès Bureautique'78, Mars 1978

‹8› Chamberlin, D.D.; King, J.C.; Slutz, D.R.; Todd, S.J.P. & Wade, B.W. "Janus: An Interactive System For Document Composition" ACM, SIGOA Newsletter, Vol. 2, Number 1 & 2, pp. 82-91, Spring/Summer 1981

‹9› Cherry, L. "Computer Aids for Writers" ACM, Sigplan Sigoa Symp. on Text Manipulation, pp. 61-67, Juin 1981

‹10› Cohen, H.A. & Francis, R.S. "Macro-assemblers and Macro-based Languages in Microprocessors Software Developement" Computer, Vol. 12, No. 2, pp. 53-64, Fev. 1979

‹11› van Dam, A. & Rice, D.E. "On-line Texte Editing: A Survey" Comp. Surv., Vol. 3, No. 3, pp.93-114, Sept. 1971

‹12› Datapro Research Corp. "Automatic & Standard Typewriters" 1974

‹13› Day, J.D. "Terminal Protocols" IEEE Trans. on commun., Vol COM-28, No. 4, pp.585-593, Avril 1980

‹14› Dennis, J.B. "Modularity" Notes de Cours donné à l'Université Technique de Munich, 1972

‹15› Deutsch, D.; Ulmer, F. & Walker, J. "Service Specification of a Message Transfer Protocol" NBS Report No. ICST/cbos-82-3, Février 1982

‹16› Digital Equipement Corp. "WPS-8 User's Manual"

‹17› idem "RSX-11M/RSX-11S Documentation Directory" 1977

‹18› Ellis, C.A. & Nutt, G.J. "Office Information Systems and Computer Science" ACM Comp. Surv., Vol. 12, No. 1, pp.27-60, Mars 1980

‹19› Engelbart, D.C. "Intellectual Implications of Multi-access Computer Networks" Proc. of the Interdisciplinary Conf. on Multi-access Computer Networks, 1970

‹20› Engelbart, D.C.; Watson, R.W. & Norton, J.C."The Augmented Knowledge Workshop" AFIPS Proc. of NCC'73, Juin 1973

‹21› Foley, J.D. "An Approach to the Optimum Design of Computer Graphics Systems" Comm. of the ACM, Vol. 14, No. 6, pp.380-390, Juin 1971

‹22› Garcia-Luna-Aceves, J. & Kuo, F.F. "Design Issues of Protocols for Computer Mail" Proc. of the 7th. Data Comm. Symp., pp.28-36, Mexico, Oct. 81

‹23› Gardner, Jr. "A System for the automated office environment" IBM Syst. J., Vol. 20, No. 3, pp.321-344, 1981

‹24› Gehani N. "The Potential of Forms in Office utomation" IEEE Trans. on Commun., Vol. COM-30, No. 1, pp. 120-125, Jan. 1982

‹25› Gien, M. "Protocole de Transfet de Fichiers" Doc. IRIA, No. IFR512, Juin 1977

‹26› Gruhn, A.M. & Hohl, A.C. "A Research Perspective on Computer Assisted Office Work" IBM Syst. J., Vol. 18, No. 3, pp.432-455, 1979

‹27› Hammer, M. "The Design of Usable Programming Languages" MIT Report.

‹28› Hammer, M.; Ilson, R.; Anderson, T.; Gilbert, E.; Good, M.; Niamir, B.; Rosenstein, L. & Schoichet, S. "The Implementation of Etude, An Integrated & Interactive Document Production System" Proc. of ACM Sigoa Symp. on Text Manipulation, pp.137-146, Juin 1981

‹29› Hulburt, A."Publication Design" Van Nostrand Reinhold Co., New York, 1976

‹30› IBM Corp. "Manuel d'utilisation des Systèmes IBM 430, 440, 450" 1977

‹31› Irby, C.H. "Display Techniques for Interactive Text Manipulation" AFIPS Proc. of NCC'74, pp.247-255, 1974

‹32› Irons, E. & Pramanik, S. "A data-handling mechanics of on-line text editing system with efficient secondary storage access" NCC'79, pp.273-277, 1979

‹33› ISO/TC97/SC16 "Reference Model of Open System Interconnection" Dec.1980

‹34› Limb, J.O. "Communications in the Automated Office - Guest Editor's Prologue" IEEE Trans. on Commun., Vol. COM-30 No. 1, pp.1-5, Jan. 1982

‹35› Liskov, B. "An Introduction to CLU" Report for IFIP-WG.2.1 1975

‹36› Martin, J. "Design of Man-Computer Dialogue" Prentice-Hall 1973

‹37› Michael, E.K. et al. "Document Production and Control System Design Study" SRI-ARC Report No. 37730, 1977

‹38› Mulvenna, G.F. "Development of a CBMS Message Transfer Protocol" à paraitre dans: Proc. of ACM-SIGOA Conf. on Office Information Systems, Juin 1982

‹39› Naffah, N. "Les réseaux Locaux en Burotique" Doc. Interne INRIA-KAYAK, REL.2.508, Juin 1979

‹40› idem "Principes de description de l'information écrite en Burotique" Doc. Interne INRIA, Kayak, NEV.2.551, Avril 1980

‹41› idem "Design Issues of Presentation Protocols in office systems" NTC 80, Houston, Texas, Dec. 80

‹42› idem "Communication Protocols for Integrated Office Systems" Computer Networks, No. 5, pp.445-454, 1981

‹43› idem "Protocoles de Communication pour les Sytèmes Intégrés de Bureautique" L'Informatique Professionnelle, No. 1, pp.62-81, Mars 1982

‹44› National Bureau of Standards "Features of a Message Transfer Protocol" Report No. ICST/CBOS-82-1, Nov. 1981

‹45› Norton, J.C.; Bair, J.H. & Engelbert, D.C. "AKW System Capabilities and Features, An Overview" SRI-ARC Report No. 38769, Jan. 1977

‹46› Niznik, C.A. "Cost-Benefit Analysis for Local Integrated Facsimile/Data/Voice Packet Communication Networks" IEEE Trans. on Commun., Vol. COM-28, No. 1, pp.19-27, Jan. 1982

‹47› Olivetti "TES 501: Manuel d'Utilisation" 1978

‹48› Parnas, D.L. "A Technique for Software Module Specification with examples" Commun. of the ACM, Vol. 15, No. 5, Mai 1972

‹49› idem "On the Criteria to be used in Decomposing Systems into Modules" Commun. of the ACM, Vol. 15, No. 12, Dec. 1972

‹50› Prager, J.M. & Brokin, S.A. "Polite Project Progress Report" Technical Report, IBM Cambridge Scientific Center, April 1982

‹51› Quéré, M. "Utilisation d'un réseau local de Micro Ordinateurs dans le Contexte Universitaire" Rapport de recherche CRIN No. 81-R-041, 1981

‹52› Schuman, S.A. "Toward Modularity in High Level Programming Languages" R.R. Centre Sc. IBM de Grenoble, 1972

‹53› de Sousa, M.R. "Electronic Information Interchange in an Office Environment" IBM System Journal, Vol. 20, No. 1, pp.4-21, 1981

‹54› Tantawi, A. & Mourad, M. "A Modular Office Automation System" Proc. of the ISMM 8-th Intl. Symp. on Mini and Micro Computers, Zurich, Mai 1979

‹55› idem "Solving Some Problems of Ofice Automation Systems" Proc. of the IEEE Micro and Mini Computer Conf., Houston, Texas, Nov.1979

‹56› idem "A Universal Modular Language and Software Core for a Flexible Office System" MIMI'80, Asilomar, California, Jan. 1980

‹57› idem "Workstations in the Electronic Office" Viewdata'80 Conf. Record, Londres, Mars 1980

‹58› idem "A New Concept for the Design of Flexible Office Systems" IERE Proc. of the Intl. Conf. on the Electronic Office, Londres, Avr. 1980

‹59› Tektronix, Inc. "Tektronix 4027 Programmer's Reference Guide" 1978

‹60› Tsichritzis, D.C. & Lochovsky, F.H. "Office Information Systems: Challenge for the 80's" IEEE Spectrum, Vol. 68, No. 9, pp.1054-1059, Sept. 1980

‹61› Turba, T.N. "Checking for Spelling and Typographical Errors in Computer-Based Text" Proc. of ACM Sigplan Sigoa Symp. on Text Manipulation, pp.51-60, Juin 1981

‹62› Turoff, M. "Meeting Through Your Computer" IEEE Spectrum, pp.58-64, Mai 1977

‹63› Uhlig, R.P.; Farber, D.J. & Bair J.H. "The Office of the Future" North Holland Publishing Co., 1979

‹64› Vallee, J. "The FORUM Project: Networks Conferencing and its Future Applications" Computer Networks, Vol. 1, No. 1, pp.39-52, Juin 1976

‹65› Vydec corp. "Vydec 1200 Users Manual" 1977

‹66› Wirth, N. "On the Design of Programming Languages" Information Processing'74, North Holland Publ., 1974

‹67› Wordplex Europe "Wordplex Comparison Grid" 1977

ANNEXE A

Manuel D'utilisation
Du Système Extensible de Traitement de Textes Réalisé

1. Le Démarrage du système

2. La création de fichiers

3. L'édition

4. Le formattage

5. L'impression

6. Liste des noms de fichiers existants

7. Effacer un fichier

8. Renommer un fichier

ANNEXE A

1. Le Démarrage du système

Etant donné que nous avons développé notre système sur une machine hôte non-spécialisée (Le MICRO-1 de Plessey), on est obligé de suivre les commandes nécessaires pour le démarrage de cette machine avant d'entrer dans notre système lui-même.

La séquence d'initialisation et de démarrage du MICRO-1 se trouve dans les manuels de cette machine. Nous la résumons en ce qui suit:

1. Démarrer le MICRO-1 et les terminaux en appuyant sur les boutons ON/OFF.

2. Mettre le disque de l'utilisateur dans son emplacement spécial en le manipulant avec attention.

3. Utiliser le bouton "boot" pour charger le système d'exploitation dans la mémoire vive.

4. Le système RSX-11M affiche une étoile (*) sur la console. Ceci marque le début d'une séquence interactive entre l'utilisateur et le système comme suit (les caractères soulignés sont entrés par l'utilisateur, et le code "Retour Chariot" est symbolisé par "<").

```
* RK
     RSX-11M V3.1 BL22 28K
› RED DK0: = SY0:
› RED DK0: = LB0:
› MOV DK0: = SYSTEM
› à (1,2) START UP
```

› * PLEASE ENTER TIME AND DATE (HR:MN DD-MMM-YY) (S): 15-MAY-80 14:50 ‹
› TIM 15-MAY-80 14:50
› ASC SY: /BLKS = 300
› à ‹EOF›
› MOU DK2 ou 3:‹
› SET/UIC = (,)‹
› ASN DK2 ou 3:=sy0:‹
› SET/BUF=TT0:120‹
› SET/CRT=TT0:‹
› TIM 14:50:05‹ optionnel
› RUN SYSTEM‹

5. La commande "RUN SYSTEM" appelle notre système de traitement de textes. Le menu principal est affiché et l'utilisateur a à taper le numéro de la fonction choisie:

SYSTEME DE T.T.

1- CREATION
2- EDITION
3- FORMATAGE
4- IMPRESSION
5- LISTE DES NOMS DES TEXTES CREES
6- EFFACEMENT D'UN FICHIER
7- RENOMMER UN FICHIER
8- SORTIE DU SYSTEME

TAPER LE NUMERO DU TRAVAIL CHOISI ›x‹

6. Si l'utilisateur tape le caractère "8", le contrôle retourne au système RSX-11M (plus précisément au sous-système MCR) qui affiche le caractère ">". L'utilisateur peut écrire:

› DMO DK2 ou 3:‹

pour terminer le travail.

Si un autre chiffre est tapé (1 à 7), le module correspondant à la fonction choisie est évoqué. Dans le reste de ce manuel, nous décrivons l'utilisation des différents modules.

2. La Création de Fichiers

(Fonction numéro 1 du menu principal)
L'écran est effacé et seul le message:

ENTREZ LE NOM DE TEXTE A CREER›xxxx‹

apparait. L'utilisateur répondra par la frappe du nom du texte à créer (en lettres majuscules). S'il fait une erreur de frappe, il peut se rattrapper en utilisant une des 2 possibilités suivantes:

i- effacer autant de caractères qu'il veut en tapant une fois pour chaque caractère sur la touche "DEL" du clavier.

ii- enfoncer simultanément les touches "Contrôle" et "U". Ceci efface la chaîne entière.

Un code "Retour Chariot" marque la fin du nom du nouveau fichier.

Les caractéristiques des noms de fichier sont les suivantes:

1- Le nom doit être formé de 1 à 9 caractères alphanumériques.

2- Un type doit être associé au nom du fichier. Par défaut le type est ".TXT".

3- La version est supposée être 1 par défaut si le nom du fichier à créer n'existe pas dans le catalogue de la face du disque utilisée. Par contre, si un fichier existant porte le nom proposé pour le nouveau fichier, la version doit être spécifiée par l'utilisateur et doit être différente de la version associée au fichier portant le même nom. Par défaut de spécification de version, une valeur n+1 est donnée, n étant la version du fichier portant ce même nom.

Une erreur de syntaxe est signalée par le message:

ER SYNTAX

et le contrôle retourne à l'affichage du menu principal.

En cas de syntaxe correcte, le message suivant est affiché:

ENTREZ LE TEXTE A CREER

Le curseur se positionne au début d'une nouvelle ligne, le système est prêt à recevoir le texte et l'affiche immédiatement sur l'écran. Le curseur se positionne toujours à l'endroit où le caractère suivant sera affiché.

La frappe au kilomètre est possible et l'utilisateur ne se soucie pas des fins de lignes. Il n'a à taper "Retour Chariot" que s'il veut un passage obligatoire à une nouvelle ligne.

Le dernier caractère tapé avant le code "Retour Chariot" déterminera le nombre d'espaces blancs laissés avant le premier caractère de la nouvelle ligne:

i- Un point "." = 2 espaces
ii- Un trait d'union "-" = aucun espace
iii- Tout autre code = un espace

La seule exception à cette règle est quand un code de formattage est entré dans la nouvelle ligne. Il est toujours enregistré dans la première position de cette ligne.

Après l'écriture de 20 lignes sur l'écran, le système les transfère sur le disque et efface l'écran pour recevoir une nouvelle "page" de 20 lignes. Le message "ENTREZ LE TEXTE A CREER" est toujours affiché. Le découpage du texte en petites "pages" est fait pour prévenir contre la perte de grandes parties du texte en cas de panne quelconque.

Les commandes et les codes de formattage sont mémorisés avec le texte mais toujours chacun sur une ligne séparée. Ces lignes commencent par le code ";" pour les commandes et "." pour les codes. A la fin de la commande, il faut toujours taper "Retour Chariot".

Deux méthodes existent pour éditer un texte au cours de sa création.

La première méthode est "l'édition de la dernière ligne". Au cas où l'on veut modifier la ligne en cours de frappe, on peut utiliser la touche "DEL" pour éliminer le dernier caractère ou les touches "Contrôle" et "U" simultanément pour effacer toute cette ligne.

La deuxième méthode consiste à utiliser les commandes d'édition (voir la section "A.3." de cette annexe) pour modifier une partie quelconque du texte déjà créé. Pour ce faire, l'utilisateur doit taper "$" puit "Retour chariot" pour sortir du mode "saisie". L'image de l'écran est changée pour l'affichage des lignes de la page en cours de création à partir de la première ligne de l'écran et le message suivant:

"EDITION DISPONIBLE"

est affiché sur la 23ème ligne de l'écran. L'utilisateur peut utiliser, à partir de ce moment, toutes les fonctions d'édition (voir section "A.3."). Les messages de l'éditeur apparaissent dans les 4 dernières lignes de l'écran.

A la fin de l'édition (commande: "FE"), le contrôle retourne au début d'une nouvelle ligne pour permettre à l'utilisateur de continuer la frappe du reste de son texte.

On quitte le module création quand on frappe le code de formattage ".X" (Fin Texte). A ce moment-là, le menu principal est réaffiché sur l'écran et l'utilisateur choisit sa prochaine tâche.

3. L'EDITION

(Fonction 2 dans le menu principal)

3.1. Appel de l'éditeur

L'entrée dans ce module entraîne l'effacement de l'écran et l'affichage de la requête:

ENTREZ LE NOM DU TEXTE A EDITER >

L'utilisateur doit alors taper le nom du fichier (comme dans le cas de la création, mais le type ".txt" doit être fourni explicitement).

Si une erreur de syntaxe est faite, un message est affiché et un retour s'effectue vers l'affichage du menu principal.

Si le nom fourni n'existe pas dans le catalogue, le message **"FICHIER INEXISTANT"** est affiché avant le retour au menu principal.

Si deux fichiers ont le même nom, et la version n'est pas spécifiée par l'utilisateur, c'est la dernière version qui sera considérée par défaut.

Dans le cas de commande correcte, les 20 premières lignes du fichier sont affichées sur l'écran.

L'édition se fait à l'aide des fonctions qu'on peut diviser en trois catégories:

1- Commandes de gestion de l'écran.

2- Commandes de recherche dans un texte.

3- Commandes de modification du texte.

Avant chacune de ces commandes on tape d'abord "Contrôle" et "G" en même temps. Le système affiche "*>" sur la 23ème ligne de l'écran et attend un mnémonique pour déterminer la commande à exécuter. Chaque commande se termine par un "retour chariot".

Nous présentons les commandes en détail dans le reste de cette section

3.2. Commandes de gestion de l'écran

*› H = (HOME) le curseur se positionne dans le coin supérieur-gauche de l'écran

*› U = (UP) le curseur se déplace verticalement vers la ligne précédant la ligne courante

*› B = (BAS) le curseur se déplace verticalement vers la ligne suivant la ligne courante

*› Ax,y = (x= 1 à 80 , y= 1 à 20) le curseur se positionne dans la colonne x de la ligne y

*› Z = Effacement de l'écran

*› L = Effacement à partir de la position du curseur jusqu'à la fin de la ligne courante

*› F = Effacement à partir de la position du curseur jusqu'à la fin de l'écran

*› E = La ligne sur laquelle le curseur est positionné est éliminée et les lignes suivantes défileront vers le haut, laissant une ligne vide à la fin de la page

*› I = Une ligne vide est insérée avant la ligne sur laquelle le curseur est positionné. Cette ligne ainsi que toutes les lignes qui la suivent défileront vers le bas pour laisser une ligne vide. Le curseur se positionne au début de cette ligne vide. La 20ème ligne disparaîtra de l'écran

Quant au déplacement du curseur vers la gauche et la droite sans insertion de blancs, il se fait à l'aide des deux codes directs suivants (qui ne sont pas des commandes):

"Back Space" : Le curseur se déplace une position vers la gauche. Si le curseur est dans la première postion de la première ligne de l'écran, il ne bouge pas.

"Shift + Back Space" : Le curseur se déplace une position vers la droite. S'il est dans la colonne 80 de la 20ème ligne de l'écran il ne bouge pas.

3.3. Commandes de recherche dans un texte

*› ?C‹chaine› : Le système cherche et affiche la première page (de 20 lignes) du texte contenant la chaine spécifiée. Le curseur se positionne au début de cette chaine. Si la chaine n'est pas trouvée jusqu'à la fin du fichier, le message: "CHAINE NON TROUVEE" est affiché au dessous de la page du texte où l'utilisateur a demandé la recherche.

Note: La commande suivante ne correspond pas à une primitive. Elle est optionnelle et est ajoutée pour simplification.

*› ?+(ou-)‹n›B : (n est un entier › ou = 1)
La n-ième page, de 20 lignes, suivant (si le signe est +) ou précédant (si le signe est -) la page actuelle est trouvée et affichée.
Si n n'est pas spécifiée, la valeur 1 lui est donnée par défaut. Le signe + est aussi donné par défaut.
Si la valeur de n est trop grande, la première page est affichée (signe -) avec le message "PREMIERE PAGE" apparaissant au bas de l'écran. Dans le cas d'un signe +, la dernière page est affichée avec le message "DERNIERE PAGE" apparaissant au bas de l'écran.

3.4. Commandes de modification du texte

*› ?I‹CHAINE›

: Tout le texte affiché à partir de la position du curseur est décalé pour permettre l'insertion de la chaine spécifiée avant la position du curseur. Si la chaine insérée se termine par "$" ceci signifie que ce qui suit doit commencer sur une nouvelle ligne. Si la chaine insérée est trop longue, plusieurs lignes sont défilées vers le bas. Les dernières qui disparaissent de l'écran réapparaitront au début de la page suivante.

*› ?E ‹n›C
ou
*› ?E‹CHAINE›

: Cette commande efface une partie du texte. La position du curseur avant la frappe de cette commande détermine le premier caractère de la partie à effacer. La fin de cette partie est déterminée par une des deux méthodes suivantes:
i) nC: n est un entier (› ou = 1) qui représente le nombre de caractères à effacer. Par défaut, sa valeur sera 1.
ii) Une chaine d'au moins 2 caractères qui sont les 2 premiers caractères de la ligne jusqu'à laquelle l'effacement aura lieu. Cette ligne sera elle aussi effacée A défaut d'une chaine, le texte sera effacé jusqu'à la fin du fichier.

*› ?S‹n›C
ou
*› ?S‹CHAINE›

:Cette commande est utilisée pour sauvegarder une chaine de caractères dans un endroit prédéfini dans la mémoire. Elle n'entraine aucun changement dans l'image de l'écran. La technique d'identification de la chaine à sauvegarder est indentique à celle utilisée pour (?E). La longueur maximale d'une chaine sauvegardée est une page de 20 lignes de 80 caractères. Il faut noter que la commande "liste des fichiers dans le catalogue" (existant dans le menu prinicipal) détruit l'espace de sauvegarde et, donc, il ne faut pas l'utiliser avant de remettre cette chaine dans le texte actuel ou un autre texte en utilisant la commande d'insertion de cette chaine (?P).

*› ?P

: Cette commande cause l'insertion d'une chaine sauvegardée dans un endroit qui précède immédiatement la position du curseur avant la frappe de cette commande. La chaine sauvegardée se trouve dans un endroit pré-spécifié dans la mémoire. Le texte suivant la chaine insérée est décalé et les lignes suivantes sont éventuellement défilées vers le bas.

3.5. La sortie de l'éditeur

Elle se fait par la commande "fin édition":

* En tapant "Contrôle" et "G" par l'utilisateur
* Le système affiche "*›" en bas de l'écran. L'utilisateur doit ensuite taper:
* "FE‹" = FIN EDITION

Le contrôle retourne ainsi au point auquel l'appel de l'éditeur a été fait. Ceci peut être:

1. Pendant la création d'un fichier.

ou

2. Pendant le formattage.

ou

3. Pendant le choix du menu principal (dans ce cas, le menu est réaffiché).

4. Le Formattage

(Fonction 3 dans le menu principal)

4.1. Procédure d'initialisation et de déclaration du format physique

L'écran est effacé et le message suivant est affiché:

NOM DU TEXTE A FORMATTER ›

L'utilisateur a donc à spécifier ce nom en suivant les mêmes règles utilisées dans l'édition.

Le système commencera par afficher consécutivement quelques questions pour spécifier quelques paramètres:

NUMEROTATION DE PAGE? (O/N) ›

L'utilisateur doit donc taper O ou N suivie d'un "Retour Chariot" ("R.C."). S'il ne tape que "R.C.", la réponse est considérée comme étant "OUI".

Si la réponse est "OUI", les 3 messages suivants sont affichés, si non ils ne le sont pas:

NUMERO DE LA PREMIERE PAGE ›

Un entier doit être spécifié, sinon la valeur "1" est considérée par défaut.

NUMERO DE LIGNE SUR LAQUELLE LE NUM. DE LA PAGE SERA MIS›

Un entier doit être entré. Par défaut, la valeur est 1.

MARGE GAUCHE POUR LE NUMERO DE PAGE›

Un entier doit être entré. Sa valeur sera 76 par défaut. Ensuite le message suivant est affiché:

NUMERO DES PAGES A FORMATTER›

Il faut taper ces numéros, par ordre ascendant, séparés par des virgules. La liste se termine par un "R.C.". Si seul la touche "R.C." est tapée, tout le texte est formatté.

Après cela, le formatteur cherche ces pages une à une, et exécute toutes les commandes de formattage puis imprime la page formattée. Ensuite, la question suivante apparait au bas de l'écran:

EDITION OU FORMATTAGE (E/F)›

La lettre "E" évoque l'appel de l'éditeur et affiche les 20 premières lignes (non-formattées) de la page actuelle. A la fin de l'édition, le formatteur reprend le contrôle et reformatte la page. Ce cycle peut être recommencé plusieurs fois.
Si l'utilisateur tape "F", le système formatte la page suivante.

Quand le texte se termine, le menu principal est affiché.

Si un nombre de lignes vides est demandé mais ne peut pas tenir sur le reste de la page en cours de formattage, les lignes restantes sont mises au début de la page suivante. Pour signaler ceci à l'utilisateur, le message suivant est affiché au bas de l'écran:

COUPURE EN FIN DE PAGE

Le formatteur réserve automatiquement 6 lignes au début et à la fin de chaque page pour la numérotation des pages, les chaînes apparaissant sur toutes les pages, c'est-à-dire les en-tête et les renvois.

4.2. Les Codes de Formattage:

Il faut noter que chaque code de formattage doit être écrit sur une ligne séparée et suivie par "R.C.". Ces codes sont:

4.2.1. Les délimiteurs de chaînes

Ils sont insérés pour désigner les limites d'une chaîne (ligne, en tête, etc.). Les codes qui indiquent la fin d'une ligne, paragraphe, ou page impliquent une coupure physique obligatoire dans le texte.

.L : Fin de ligne obligatoire

.R : Fin de paragraphe

.G : Fin de page

.X : Fin de texte

.S : Début d'un soulignement d'une chaîne de caractères

.FS : Fin de soulignement d'une chaîne de caractères

.T : Début d'une "en-tête" constante

.FT : Fin d'une "en-tête" constante

.B : Début d'un "en-bas" constant

.FB : Fin d'un "en-bas" constant

4.2.2. Les codes pour la manipulation des tables

.C : Début d'un élément d'une table

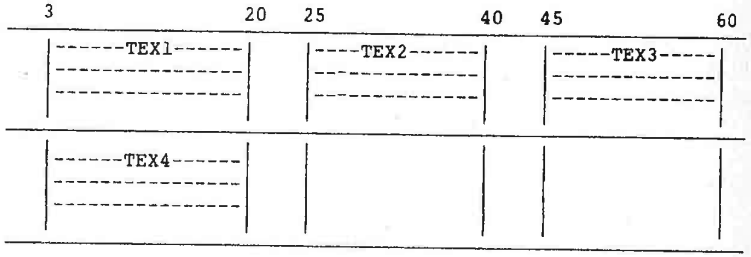
.FC : Fin d'un élément d'une table

.PT : La ligne qui suit indique la marge gauche et la marge droite pour les colonnes d'une table

Exemple: Le texte suivant est entré (non-formatté) par un utilisateur ("‹" = "R.C."):

```
.PT‹
3,20/25,40/45,60‹
.C‹
------TEX1-------‹
.FC‹
.C‹
-----TEX2--------‹
.FC‹
.C‹
-----TEX3---------‹
.FC‹
.C‹
-----TEX4--------‹
.FC‹
```

Le formatteur traduit ce texte comme suit



4.3. Les Commandes Primitives De Formattage

La **forme générale d'une commande** de formattage est:

;‹F›,‹m›,‹n›,‹Y›"R.C."

où:

‹F› = Code de la fonction

‹m› = est le premier paramètre. Sa signification varie selon la commande. Il peut prendre les valeurs suivantes:

‹m› = Entier supérieur ou égal à 0 (1èr paramètre)

ou

= Vide

‹n› = est le deuxième paramètre. Il signifie l'adresse relative de la ligne à partir de laquelle la fonction prendra effet. ‹n› peut prendre les valeurs suivantes:

‹n›= Entier supérieur ou égal à 0. (la fonction sera effective après n lignes; c'est à dire à partir de la n+1 ème ligne suivant cette fonction).

ou

= Vide. (la première ligne suivant cette commande sera affectée).

ou

= Code alphanumérique. (en cas d'utilisation de ‹n› comme code d'alignement: commande 3 dans 4.3.2 ci-dessous)

<Y> = définit la limite de la partie du texte affectée par la commande. <Y> peut avoir les valeurs suivantes:

<Y> = <k>L, k=entier positif (k: est le nombre de lignes affectées).

ou

= Vide (une seule ligne est affectée)

ou

= <k>G k: entier positif (k est le nombre de pages affectées)

ou

= FR (La commande affecte le texte jusqu'à la première détection d'un code fin paragraphe)

ou

= FX (la commande affecte le texte jusqu'à sa fin)

Il faut noter que chaque fonction de formattage doit être écrite sur une ligne séparée.

4.3.1. Les commandes de déclaration de format physique

1) NOMBRE DE LIGNES PAR PAGE (60 PAR DEFAUT)

;NL,<m>,, "R.C."

m est un entier =1 ou >1 qui définit le nombre de lignes par page.

2) MARGE DROITE

;MD,<m>,<n>,<Y> "R.C."

m = le numéro de la colonne à considérer comme la marge droite

3) MARGE GAUCHE

;MG,<m>,<n>,<Y> "R.C."

m = le numéro de la colonne considérée comme la marge gauche.

4.3.2. Les commandes de formattage

1) TEXTE LIBRE

;NT,,,<Y> "R.C."

2) INSERTION DE LIGNES VIDES

;LV,<m>,<n>,<Y> "R.C."

m = nombre de lignes à insérer avant chaque ligne du texte

3) ALIGNEMENT SUR UNE COLONNE DONNEE D'UN CODE CONTENU DANS UNE CHAINE

;LA,<m>,<n>,<Y> "R.C."

m = numéro de la colonne d'alignement

n = <code>, c'est-à-dire le code alphanumérique sur lequel se fait l'alignement

4) CENTRAGE

;CT,,<n>,<Y> "R.C."

5) COMPRESSION

;CD,,<n>,<Y> "R.C."

6) AJUSTEMENT A DROITE

(exemple: les numéros des équations)

;AJ,,‹n›,‹Y› "R.C."

7) JUSTIFICATION

;JT,,‹n›,‹Y› "R.C."

4.4. Commandes Auxiliaires de Formattage

Les commandes suivantes ne correspondent pas à des fonctions primitives mais elles sont incorporées dans le système pour faciliter le travail de l'utilisateur. Elles lui permettent de changer certains paramètres (marges, etc...) pour une partie donnée du texte sans avoir à connaître les valeurs de ces paramètres en dehors de la partie du texte concernée. Par exemple, s'il veut mettre en évidence une ligne en décalant sa marge gauche de 5 espaces vers la droite, il n'a pas à chercher la valeur actuelle de la marge gauche, puis à la modifier, mais il peut simplement écrire la commande suivante avant la ligne concernée:

;+G,5,,1L "R.C."

Dans cette commande, 5 est la valeur du décalage et 1L signifie la ligne suivante.

La forme générale de ces commandes est identique à celle des commandes primitives de formattage (voir 4.3.) sauf le fait que le champ ‹n› est toujours vide pour signaler un effet immédiat de la commande.

;‹F›,‹m›,,‹Y› "R.C."

Ici ‹m› est la différence (positive ou négative) entre la valeur voulue et la valeur actuelle du paramètre concerné

1) CHANGEMENT DU NOMBRE DE LIGNES PAR PAGE

;+L,‹m›,, "R.C."
;-L,‹m›,, "R.C."

2) CHANGEMENT DE LA MARGE DROITE

;+D,‹m›,,‹Y› "R.C."
;-D,‹m›,,‹Y› "R.C."

3) CHANGEMENT DE LA MARGE GAUCHE

;+G,‹m›,,‹Y› "R.C."
;-G,‹m›,,‹Y› "R.C."

4) CHANGEMENT DE NOMBRE DE LIGNES VIDES INSEREES AVANT UNE LIGNE

;+V,‹m›,,‹Y› "R.C."
;-V,‹m›,,‹Y› "R.C."

5. L'IMPRESSION D'UN FICHIER

(Commande 4 dans le menu prinipal)

Quand l'utilisateur choisit cette fonction en tapant le chiffre 4, l'écran est effacé et le message suivant apparait:

NOM DU TEXTE A IMPRIMER›

L'utilisateur doit donc écrire le nom du texte ainsi que son type (.TXT) et la version à imprimer. Si la version est omise, la dernière version est imprimée.

6. LISTE DES NOMS DES FICHIERS EXISTANTS

(Commande 5 dans le menu principal)

Quand l'utilisateur tape le chiffre 5, l'écran est effacé et le message suivant apparaît:

A L'APPARITION DE "›" TAPEZ "à INDEX R.C."
LA LISTE SORTIRA SUR L'IMPRIMANTE

L'utilisateur a donc à taper les 6 caractères àINDEX puis retour chariot "R.C.". La liste des noms des fichiers existants sur la face utilisée du disque est affichée sur l'écran et imprimée sur l'imprimante. Cette liste est ensuite effacée de l'écran pour l'affichage du menu principal.

7. EFFACEMENT D'UN FICHIER

(Commande 6 dans le menu principal)

Le message suivant est affiché sur l'écran:

NOM DU FICHIER A EFFACER ›

L'utilisateur doit donc écrire le nom, le type, et la version à effacer. Le retour se fait au menu principal.

8. RENOMMER UN FICHIER

(Commande 7 dans le menu principal)

Le message suivant est affiché sur l'écran:

NOM DU FICHIER A RENOMMER ›

Après la spécifiction du nom, type (.TXT), et version du fichier, le message suivant est affiché:

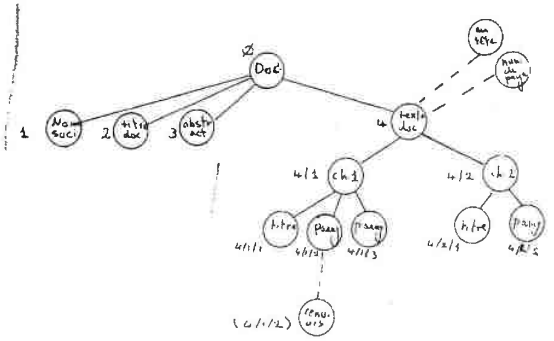
NOUVEAU NOM DU FICHIER ›

L'utilisateur doit donc écrire le nouveau nom et le type (.TXT). La version est spécifiée automatiquement.

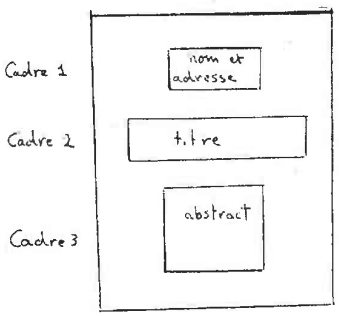
ANNEXE B

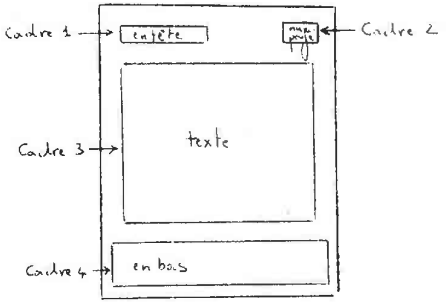
Exemple de Représentation d'un Fichier en Forme Proposée dans la Solution D

Soit un document dont la structure logique est la suivante:



et dont les formats des pages sont les suivants:





A remarquer que le titre doit apparaitre en tête de toutes les pages à partir de la deuxième.

Le numéro de page est à calculer et insérer automatiquement à partir de la deuxième page.

Notre but essentiel, étant de montrer comment les descripteurs sont insérés, nous avons préféré ne pas écrire le texte lui même.

Descripteurs du Document

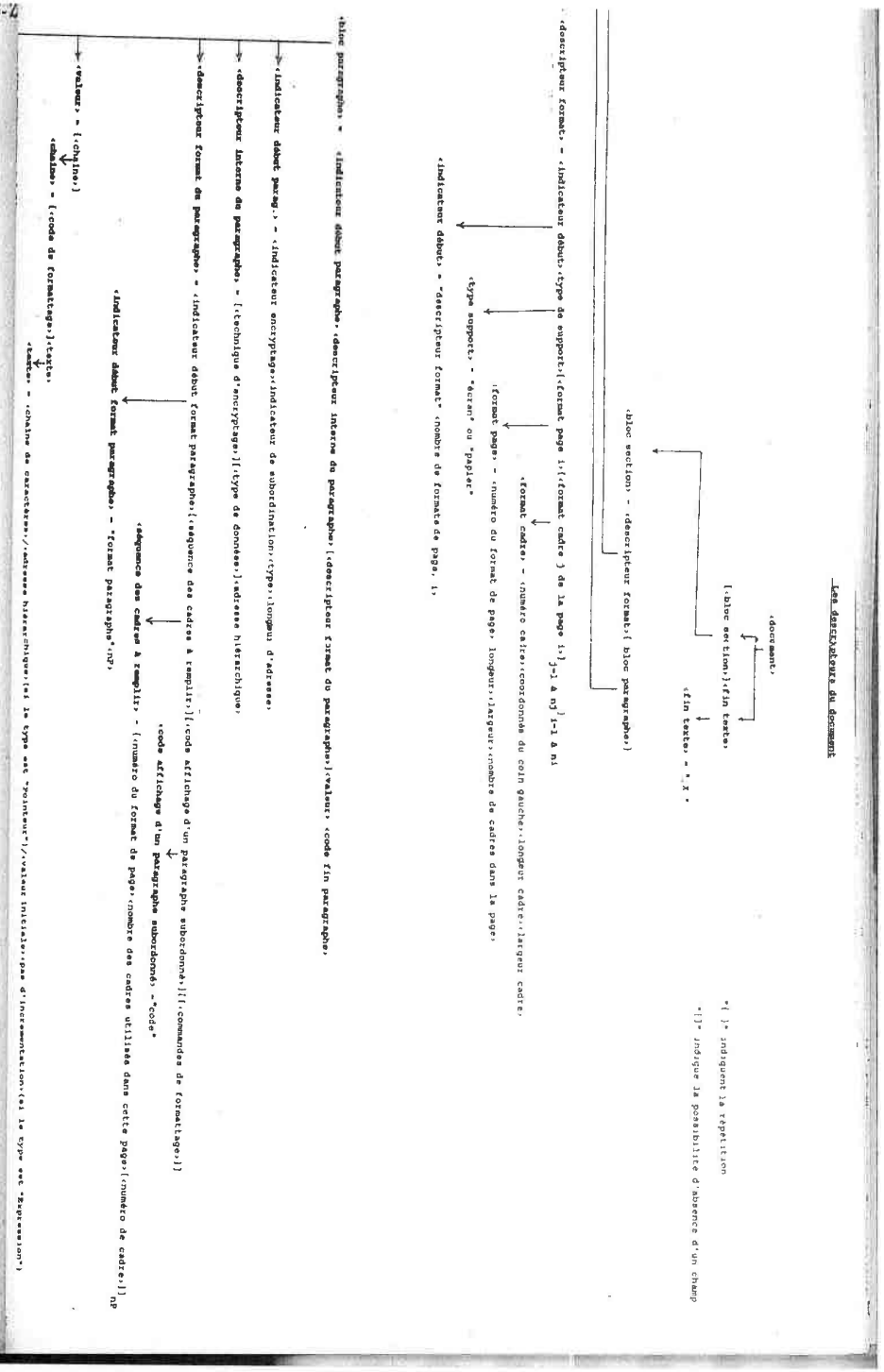
‹document› = {‹bloc section›} répété 1 fois ‹fin texte›

‹bloc section› = ‹descripteur format›{‹bloc paragaphe›} répété x fois

où " {} " = indique répéte
" [] " = indique la possibilité d'absence d'un champ

En Détail

‹descripteur format›/‹bloc parag.1› x=1 ‹bloc parag.2› x=2
‹bloc parag.3› x=3 /‹bloc parag.4› x=4 ‹bloc parag.5› x=5
‹bloc parag.6› x=6 ‹bloc parag.7› x=7 ‹bloc parag.8› x=8
‹bloc parag.9› x=9 /‹bloc parag.10› x=10 ‹bloc parag.11› x=11 /



‹descripteur format›

‹indicateur début› = "descripteur format",2

‹type de support› = "papier"

Page 1

‹format page 1› = 1,60,80,3

numéro de page = 1

longueur = 60

largeur = 80

nombre de cadres dans la page = 3

‹format cadre 1 page 1› = 1,5,20,4,40

numéro cadre = 1

coordination: y=5 et x=20

longueur = 4

largeur = 40

‹format cadre 2 page 1› = 2,10,10,3,60

‹format cadre 3 page 1› = 3,20,10,25,60

Page 2›

‹format page 2› = 2,60,80,4

‹format cadre 1 page 2› = 1,3,10,2,40

‹format cadre 2 page 2› = 2,3,70,1,5

‹format cadre 3 page 2› = 3,7,10,42,60

‹format cadre 4 page 2› = 4,50,10,5,60

** **‹bloc paragraphe›**

* **‹bloc parag.1›** (nom et adresse de l'auteur)

‹indicateur début› = "X0010001

‹indicateur encryptage› = 0

‹indicateur subordination› = 0

‹type› = 1

‹longueur d'adresse hiérarchique› = 0001

‹descripteur interne› = "chaine de carac.",1

‹type de donnée› ="chaine de caractères

‹adresse hiérarchique› = 1

‹descripteur format du parag.› =

‹indicateur début format› = "format parag",1

‹séquence de cadres à remplir› = 1,1,1

‹commandes de formattage› = ;CT,,,FR

‹valeur› =

‹code de formattage› = .S (début soulignement)

‹texte› = texte du nom et adresse

‹code de formattage› = .FS (fin soulignement)

‹code fin paragraphe› = .R

* **‹bloc parag.2›** (titre du document)

‹indicateur début parag.› ="X000001"

‹descripteur interne› = 2 (= adresse hiérarchique)

‹descripteur format du parag.› =

‹Indicateur début format› = "format parag",1

‹séquence de cadres à remplir› = 1,1,2

‹commandes de fromattage› = ;---- ;-----

‹valeur› = (texte titre)

‹code fin parag.› = .R

* **‹bloc parag.3›** (abstract)
  ‹indicateur début parag.› = "X0000001"
  ‹descripteur interne› = 3 (adresse hiérarchique)
  ‹descripteur format de parag.› =
    ‹indicateur début format› = "fromat parag",1
    ‹séquence de cadres à remplir› = 1,1,3
    ‹commandes de formattage› absentes
  ‹valeur› = (texte d'abstract)
  ‹code fin parag.› = .R

* **‹bloc parag.4›** (entête constante = le titre du doc.)
  ‹indicateur début parag.› = "X011001"
    ‹indicateur encryptage› = 0
    ‹indicateur subordination› = 1
    ‹type› = 1 (changé)
    ‹longueur d'adresse hiérarchique› = 1
  ‹descripteur interne› = "Pointeur" , 4 (adresse hiér.)
  ‹descripteur format de paragraphe› =
    ‹indicateur début format› = "format parag.", 1
    ‹séquence de cardes à remplir› = 2,1,1
    ‹code affichage d'un parag. subordonné› = code "C1"
      (C1 = répéte + chaque page)
    ‹valeur› = 2 (adresse hiérarchique du titre du
    document, qui est le paragraphe désigné par le "Pointeur"
    ‹fin parag› = .R

* **‹bloc parag.5›** (numéro de page)
  ‹indicateur début parag.› = "X0110001"
  ‹descripteur interne› =
    ‹type de donnée › = "expression"
    ‹adresse hiérarchique› = 4
  ‹descripteur format parag› =
    ‹indicateur début› = "forme parag.",1
    ‹séquence de remplir des cadres› = 2,1,2
    ‹code affichage du parag. subordonné› = "C1"
  ‹valeur› =
    ‹valeur initiale› = 2
    ‹pas d'incrémentation› = +1
  ‹code fin parag.› = .R

* **‹bloc parag.6›** (renvoi)
  ‹indicateur début parag.› = "X0110011"
  ‹descripteur interne du parag.› =
    ‹type de donnée› = "chaine de caractères"
    ‹adresse hiérarchique› = 4,1,2
  ‹descripteur format› =
    ‹indicteur début format parag.› = "format parag.",1
    ‹séquence de cadres à remplir› = 2,1,4
    ‹code affichage d'un parag. subordonné› = "C2"
      C2 = à afficher **une seul fois** sur la même
      page qui contient le paragraphe essentiel (4,1,2),
      et plus précisément la ligne contenant le
      code de liaison du renvoi
    ‹commande de formattage› = ;JT,,,FR
  ‹valeur› = (texte du renvoi)
  ‹fin parag.› = .R

* **‹bloc parag.7›** (titre du chapitre 1)
  ‹indicateur début parag.› = "X0000011"
  ‹descripteur interne › =
    ‹adresse hiérarchique› = 4,1,1
  ‹descripteur format parag› =
    ‹indicateur début› = "format parag",1
    ‹séquence des cadres› = 2,1,3
    ‹commande de formattage› = .G (changement de page)
                              ;MG,12,,FX
  ‹valeur› = (texte du titre)
  ‹fin paragraphe› = .R

* **‹bloc parag.8›** (texte)
  ‹indicateur début parag.› = "X0000011"
  ‹descripteur interne d'un parag.› =
    ‹adresse hiérarchique› = 4,1,2
  ‹descripteur format parag.› =
    ‹indicateur début› = "format parag.",0
      0 = ce parag. est continué après le précédent
    ‹commande de formattage› = ;+G,3,,1L
  ‹valeur›=(texte du parag.)(code de liaison avec renvoi)(text
  ‹code fin parag.› = .R

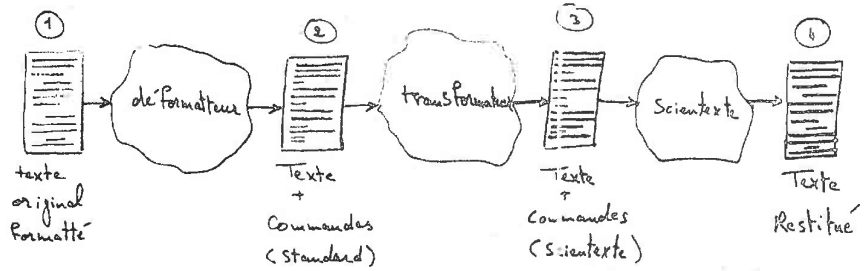
* **‹bloc parag.9›** (texte)
  ‹indicateur début parag.› = "X0000011"
  ‹descripteur interne du parag.› =
    ‹adresse hiérarchique› = 4,1,3
  ‹valeur› = (texte du paragraphe)
  ‹code fin parag.› = .R

* **‹bloc parag.10›** (titre du chapitre 2)
  ‹indicateur début parag› = "X0000011"
  ‹descripteur interne du parag› =
    ‹adresse hiérarchique› = 4,2,1
  ‹descripteur format du parag› =
    ‹indicateur début du parag.› = "format parag",1
    ‹séquence des cadres à remplir› = 2,1,3
    ‹commandes de formattage› = .G (changement de page)
                              ;CT,,,FR
  ‹valeur› = texte du titre
  ‹code fin parag› = .R

* **‹bloc parag.11›** (texte chap2)
  ‹indicateur début parag› = "X0000011"
  ‹descripteur interne du parag.› =
    ‹adresse hiérarchique› = 4,2,2
  ‹descripteur format du parag.› =
    ‹indicateur début› = "format parag",0
    ‹commande de formattage› = ;+G,3,,1L
  ‹valeur› = (texte)
  ‹fin parag› = .R

ANNEXE-C

ns cette annexe, nous présentons sur 2 exemples différents formes du texte après les diverses étapes de
ns formation :



- Le premier exemple est celui présenté en p. 121 dans la thèse

- Le deuxième est une collection des cas particuliers présentés au cours du chapitre IV.

EXEMPLE 1

Texte original

TRANSFERT DE DOCUMENTS ENTRE S.T.T.
HETEROGENES

NTRODUCTION AU PROBLEME

Je m'intéresse plus spécialement au problème du
ransfert entre systèmes de traitement de textes (STT)
étérogènes de textes formattés (ou non-formattés
ccompagnés de fonctions de formattage). Les STT peu-
ent évidemment être connectés par un même réseau
ocal ou à travers un réseau distant" ceci n'a aucune
nfluence sur notre étude.

Le problème apparaîtra lorsqu'on veut ré-éditer ou
e-formatter un texte déjà formatté sur une autre machine.
n exemple se présente dans l'édition du bulletin de liaison
ériodique du laboratoire CRIN à NANCY (1),où les auteurs
ourront écrire leurs articles sur des postes de travail
ifférents puis transférer ces textes à travers un réseau
ers le poste du rédacteur en-chef (REC).

Texte + Commandes (standard)

)6, ,
)8, ,FR
)80, ,FR
NSFERT DE DOCUMENTS ENTRE S.T.T.

)17, ,FR
)80, ,FR
EROGENES

)2, ,
)0, ,FR
)80, ,FR
ODUCTION AU PROBLEME

)1, ,
)3, ,1L
)0,1,FR
)54, ,FR
s'intéresse plus spécialement au problème du transfert entre systèmes de tr
ent de textes (STT) hétérogènes de textes formattés (ou non-formattés accoi
és de fonctions de formattage). Les STT peuvent évidemment être connectés
un même réseau local ou à travers un réseau distant: ceci n'a aucune influ
sur notre étude.

)1, ,
)3, ,1L
)0,1,FR
)60, ,FR
roblème apparaîtra lorsqu'on veut ré-éditer ou re-formatter un texte déjà
atté sur une autre machine. Un exemple se présente dans l'édition du bulle
de liaison périodique du laboratoire CRIN à NANCY (1),où les auteurs pourroi
crire leurs articles sur des postes de travail différents puis transférer c
extes à travers un réseau vers le poste du rédacteur en-chef (REC).

Texte + Commandes (Scientexte)

1 120 6 48 1 255 8 1 0 72 46 0 11 255 80
NSFERT DE DOCUMENTS ENTRE S.T.T.
1 120 6 48 1 255 17 1 0 63 46 0 11 255 80
EROGENES

1 120 6 48 1 255 0 1 0 80 46 0 11 255 80
RODUCTION AU PROBLEME

1 120 6 48 1 255 3 1 0 51 46 0 11 255 80
m'intéresse plus spécialement au problème du
1 120 6 48 1 255 0 1 0 54 46 0 11 255 80
msfert entre systèmes de traitement de textes (STT) hétérogènes de textesfo
ttés (ou non-formattés accompagnés de fonctions de formattage). Les STTpeu
t évidemment eìztre connectés par un meìzme réseau local ou à travers unrés
u distant: ceci n'a aucune influence sur notre étude.

1 120 6 48 1 255 3 1 0 57 46 0 11 255 80
problème apparaìztra lorsqu'on veut ré-éditer ou
1 120 6 48 1 255 0 1 0 60 46 0 11 255 80
-formatter un texte déjà formatté sur une autre machine. Un exemple seprése
e dans l'édition du bulletin de liaison périodique du laboratoire CRIN ANANC
(1),où les auteurs pourront écrire leurs articles sur des postes detravail c
iérents puis transférer ces textes à travers un reseau vers le postedu rédac
ur en-chef (REC).

Texte Restitué

TRANSFERT DE DOCUMENTS ENTRE S.T.T. HETEROGENES

NTRODUCTION AU PROBLEME

Je m'intéresse plus spécialement au problème du ransfert entre systèmes de traitement de textes (STT) étérogènes de textes formatés (ou non-formattés ccompagnés de fonctions de formattage). Les STT euvent évidemment être connectés par un même réseau ocal ou à travers un réseau distant: ceci n'a aucune nfluence sur notre étude.

Le problème apparaitra lorsqu'on veut ré-éditer ou e-formatter un texte déjà formatté sur une autre machine. n exemple se présente dans l'édition du bulletin de liaison ériodique du laboratoire CRIN à NANCY ‹1›,où les auteurs ourront écrire leurs articles sur des postes de travail ifférents puis transférer ces textes à travers un réseau ers le poste du rédacteur en-chef (REC).

EXEMPLE 2

Texte original

rchitecture
SPAC est un réseau public de transmission de données par
ets ouvert au public en Janvier 1979. Ce réseau est
osé de noeuds assurant les fonctions de concentration et
ommutation des données. Un noeud peut recevoir jusqu'à
00 lignes d'abonnés. Afin de garantir un haut degré
isponibilité, chaque matériel existe en 2 exemplaires.
ableau suivant indique le nombre de noeuds prévus pour
SPAC.

NNNNNN
SSSSSSS

éfinition du site de stockage d'informations
langage de commande local possède une commande particu-
re permettant de décrire un ensemble de données.
cès à des informations non locales nécessite également
troduction du paramètre SITE dans cette commande.
ple: 1. utilisation en A d'un fichier stocké en B dans un
eau de systèmes OS-360 sous JCL
D ... (SITE=B)
carte DD (Data Definition) contient alors l'indication du
e.
ême utilisation mais dans un réseau de systèmes
IS 7-8
SSIGN... (SITE=B).

A=0 ANN ANN
nnn nnn nnnn
A=1 Ann Annn

ection: De l'ETD
signaux transmis sur ce circuit indiquent s'il y a une
abilité raisonnable d'erreur dans les données reçues sur
voie de données. La qualité de signal indiqué est
forme aux spécifications appropriées de l'Avis relatif à
TD. L'état FERME indique qu'il n'y a pas de raison de
re qu'une erreur s'est produite.
tat OUVERT indique qu'il y a une probabilité raisonnable
reur.

les chiffres les positions de zone sont à 1 et les posi-
s numériques constituent le code pondéré 8-4-2-1 du
ffre considéré.
faut noter que, comme le code ASCII, le code EBCDIC permet
odage des majuscules et des minuscules et contient égale-
t les fonctions de commande pour la transmission de donnée.
ucoup de configurations restent inutilisées.

ultiplicande est dans HL

ultiplicateur est dans A

résultat est rangé dans LE

où m est le nombre de bits utiles par mot du code de n bits
n est le nombre moyen de bits transmis avant la décision

A=0 si le bit est un 0
A=1 si le bit est un 1.

Programme principal

```
EXTRN SPRG
-----------
-----------
CALL SPRG
---------------
VAL  DEFS    1
```

Texte + Commandes (standard)

,0, ,FR
,60, ,FR
Architecture

NSPAC est un réseau public de transmission de données par paquets ouvert au
lic en janvier 1979. Ce réseau est composé de noeuds assurant les fonctions
concentration et de commutation des données. Un noeud peut recevoir jusqu'
6 000 lignes d'abonnés. Afin de garantir un haut degré de disponibilité, ch
e matériel existe en 2 exemplaires. Le tableau suivant indique le nombre de
euds prévus pour TRANSPAC.

,1, ,
,0, ,FR
,11, ,FR
NNNNNN

SSSSSSS

,1, ,
,0, ,FR
,60, ,FR
Définition du site de stockage d'informations

t langage de commande local possède une commande particulière permettant de
rire un ensemble de données.

ccès à des informations non locales nécessite également l'introduction du pa
ètre SITE dans cette commande.

,0, ,FR
,61, ,FR
mple: 1. utilisation en A d'un fichier stocké en B dans un réseau de système
S-360 sous JCL

,0, ,FR
,60, ,FR
DD ... (SITE=B)

carte DD (Data Definition) contient alors l'indication du site.

,0, ,FR
,51, ,FR
même utilisation mais dans un réseau de systèmes SIRIS 7-8

,0, ,FR
,80, ,FR
SSIGN... (SITE=B).

,1, ,

6, ,FR
18, ,FR
ANN ANN

nnn nnnn A=1 Ann Annn

1, ,
0, ,FR
59, ,FR
ction: De l'ETD

signaux transmis sur ce circuit indiquent s'il y a une probablité raisonnab
'erreur dans les données reçues sur la voie de données. La qualité de sign
ndiqué est conforme aux spécifications appropriées de l'Avis relatif à l'El
L'état FERME indique qu'il n'y a pas de raison de croire qu'une erreur s'es
oduite.

at OUVERT indique qu'il y a une probabilité raisonnable d'erreur.

1, ,
0, ,FR
62, ,FR
les chiffres les positions de zone sont à 1 et les positions numériques co
tuent le code pondéré 8-4-2-1 du chiffre considéré.

aut noter que, comme le code ASCII, le code EBCDIC permet le codage des maj
les et des minuscules et contient également les fonctions de commande pour
ransmission de donnée. Beaucoup de configurations restent inutilisées.

1, ,
1,1,FR
0, ,FR
29, ,FR
ultiplicande est dans HL

ultiplicateur est dans A

résultat est rangé dans DE

2, ,
0, ,FR
80, ,FR
m est le nombre de bits utiles par mot du code de n bits

3, ,FR
80, ,FR
st le nombre moyen de bits transmis avant la décision

1, ,
7, ,FR
30, ,FR
si le bit est un 0

si le bit est un 1.

```
1, ,
, ,FR
Programme principal
EXTRN SPRG
CALL SPRG
0, ,FR
80, ,FR
 DEFS    1
```

Texte + Commandes (Scientexte)

1 120 6 48 1 255 0 1 0 60 46 0 11 255 80
Architecture
NSPAC est un réseau public de transmission de données par paquets ouvert au
lic en janvier 1979. Ce réseau est composé de noeuds assurant les fonctions
concentration et de commutation des données. Un noeud peut recevoir jusqu'
000 lignes d'abonnés. Afin de garantir un haut desré de disponibilité,ch
e matériel existe en 2 exemplaires. Le tableau suivant indique le nombrede
euds prévus pour TRANSPAC.

1 120 6 48 1 255 0 1 0 11 46 0 11 255 80
NNNNNN
SSSSSSS

1 120 6 48 1 255 0 1 0 60 46 0 11 255 80
Définition du site de stockage d'informations
t langage de commande local possède une commande particulière permettant de
rire un ensemble de données.
ccès à des informations non locales nécessite également l'introduction dupa
ètre SITE dans cette commande.
1 120 6 48 1 255 0 1 0 61 46 0 11 255 80
emple: 1. utilisation en A d'un fichier stocké en B dans un réseau desystème
OS-360 sous JCL
1 120 6 48 1 255 0 1 0 60 46 0 11 255 80
DD ... (SITE=B)
carte DD (Data Definition) contient alors l'indication du site.
1 120 6 48 1 255 0 1 0 51 46 0 11 255 80
me!zme utilisation mais dans un réseau de systèmes SIRIS 7-8
1 120 6 48 1 255 0 1 0 80 46 0 11 255 80
ASSIGN... (SITE=B).

1 120 6 48 1 255 6 1 0 12 46 0 11 255 80
0 ANN ANN
n nnn nnnn A=1 Ann Annn

1 120 6 48 1 255 0 1 0 59 46 0 11 255 80
rection: De l'ETD
s signaux transmis sur ce circuit indiquent s'il y a une probabilitéraisonnat
d'erreur dans les données reçues sur la voie de données. Laqualité de sign
indiqué est conforme aux spécifications appropriées de l'Avisrelatif à l'ET
L'état FERME indique qu'il n'y a pas de raison de croirequ'une erreur s'es
produite.
'état OUVERT indique qu'il y a une probabilité raisonnable d'erreur.

1 120 6 48 1 255 0 1 0 62 46 0 11 255 80
our les chiffres les positions de zone sont à 1 et les positions numériquesc
stituent le code pondéré 8-4-2-1 du chiffre considéré.
l faut noter que, comme le code ASCII, le code EBCDIC permet le codage desma
scules et des minuscules et contient également les fonctions de commandepour
a transmission de donnée. Beaucoup de confisurations restentinutilisées.

```
1 120 6 48 1 255 0 1 0 29 64 0 11 255 80
multiplicande est dans HL                                        C -
multiplicateur est dans A
résultat est ranse dans DE


1 120 6 48 1 255 0 1 0 80 46 0 11 255 80
m est le nombre de bits utiles par mot du code de n bits
1 120 6 48 1 255 3 1 0 77 46 0 11 255 80
est le nombre moyen de bits transmis avant la décision

1 120 6 48 1 255 7 1 0 23 46 0 11 255 80
0 si le bit est un 0
1 si le bit est un 1.

1 120 6 48 1 255 0 1 0 80 32 0 11 255 80
  Programme principal
  ------------------------
  EXTRN SPRG
  ------------
 ------------
  CALL SPRG
  ------------------
1 120 6 48 1 255 0 1 0 80 46 0 11 255 80
L  DEFS    1
```

) Architecture
RANSPAC est un réseau public de transmission de données par
aquets ouvert au public en Janvier 1979. Ce réseau est
omposé de noeuds assurant les fonctions de concentration et
e commutation des données. Un noeud peut recevoir jusqu'à
6 000 lignes d'abonnés. Afin de garantir un haut degré de
isponibilité, chaque matériel existe en 2 exemplaires. Le
ableau suivant indique le nombre de noeuds prévus pour
RANSPAC.

=1 NNNNNN
=2 SSSSSSS

) Définition du site de stockage d'information
out langage de commande local possède une commande
articulière permettant de décrire un ensemble de données.
'accès à des informations non locales nécessite également
'introduction du paramètre SITE dans cette commande.
xemple: 1. utilisation en A d'un fichier stocké en B dans un
éseau de systèmes OS-360 sous JCL
/ DD ... (SITE=B)
a carte DD (Data Definition) contient alors l'indication du
ite.
- même utilisation mais dans un réseau de systèmes
IRIS 7-8
ASSIGN... (SITE=B).

A=0 ANN ANN
nnn nnn nnnn
A=1 Ann Annn

Direction: De l'ETD
Les signaux transmis sur ce circuit indiquent s'il y a une probabilité raisonnable d'erreur dans les données reçues sur la voie de données. La qualité de signal indiqué est conforme aux spécifications appropriées de l'Avis relatif à l'ETD. L'état FERME indique qu'il n'y a pas de raison de croire qu'une erreur s'est produite.
L'état OUVERT indique qu'il y a une probabilité raisonnable d'erreur.

Pour les chiffres les positions de zone sont à 1 et les positions numériques constituent le code pondéré 8-4-2-1 du chiffre considéré.
Il faut noter que, comme le code ASCII, le code EBCDIC permet le codage des majuscules et des minuscules et contient également les fonctions de commande pour la transmission de donnée. Beaucoup de configurations restent inutilisées.

```
; multiplicande est dans HL
; multiplicateur est dans A
; résultat est rangé dans DE

; m est le nombre de bits utiles par mot du code de n bits
  n est le nombre moyen de bits transmis avant la décision

     A=0 si le bit est un 0
     A=1 si le bit est un 1.

   Programme principal
   -------------------
   EXTRN SPRG
   ----------
   ----------
   CALL SPRG
   ---------------
L  DEFS    1
```

institut
national
polytechnique
de lorraine

Le Président,

N/Réf. : Scol.

AUTORISATION DE SOUTENANCE DE THESE DE DOCTORAT-D'INGENIEUR

-:-:-:-:-:-:-:-:-:-:-:-:-:-:-

VU LE RAPPORT ETABLI PAR :

Madame le Professeur QUERE Maryse

le Président de l'Institut National Polytechnique de Lorraine autorise :

Madame TANTAWI Magda née MOURAD

à soutenir, devant l'I.N.P.L., une thèse intitulée :

"UNE APPROCHE POUR LA CONCEPTION DE SYSTEMES DE TRAITEMENT DE TEXTES EXTENSIBLES ET INTERCONNECTABLES."

en vue de l'obtention du titre de DOCTEUR-INGENIEUR

Spécialité "INFORMATIQUE"

Fait à NANCY, le *30 Juin 1982*

Le Président de l'I.N.P.L.

M. LUCIUS