LABORATOIRE D'INFORMATIQUE THéORIQUE
& APPLICATIONS DE MARSEILLE
L.I.T.A.M
Département de Mathématiques-Informatique
Luminy
UNIVERSITé AIX-MARSEILLE II

ISSN 0291 - 541

INFORMATIQUE FONDAMENTALE ET APPLICATIONS

SOMMAIRE

P 1... ... éDITORIAL
.Science et affolement.

P 5... ... Notion d'autojectivité, notion de statut.

P 22... VOUZZAVéDIBISAR.

MARS 1988

Adresse postale : FACULTé DES SCIENCES DE LUMINY
Mathématiques-Informatique LITMM BAT TPR 2 9e etage
Case 901 70 route Léon LACHAMP 13 288 MARSEILLE Cedex 9
91 26 90 81

EDITORIAL

E. Blanco

SCIENCE ET AFFOLEMENT.

Quand la chaleur sévit et que vibrent les mouches, une sorte de frénésie s'empare de toute une vie souterraine. Une agitation bizarre saisit tout ce qui peut bouger. Les feuilles tremblent dans un air qui semble sortir de la bouilloire. Des diables de poussière jaillissent de la plaine.

Depuis plus de trente ans les Réformes ne se comptent plus, mais il est pourtant vrai que le **paysage enseignemental** a pour le moins changé. Où sont nos paisibles cinq heures de cours hebdomadaires de propédeutique avec quelques travaux dirigés par des anciens ?

Nos carrosses roulaient au pas. Allait-on moins vite ? sans doute, sans doute.

Et puis l'horizon quatrevingtdouze nous menace. Il nous faut nous préparer à la compétition. Internationale.
La compétition.

Comme le déclarait un ancien ministre au soir des élections :
" Nous n'avons que trop perdu de temps, la France doit vite se remettre au travail " .

Même un ministre peut être drôle. L'humour involontaire n'est pas

toujours le pire.

Après maisoixantehuit Edgar Faure s'était bien rendu compte qu'un étudiant penché sur ses grimoires peut avoir plus de mal à laisser divaguer son imagination.

A cette même époque quelques étudiants croyant s'être libérés des carcans de la société, furent pris des fringales de la Connaissance. Il suffisait d'accorder les violons. Il suffisait de poser la bonne équation : Quantité de connaissances égale quantité d'heures de cours.

On gagnait sur deux plans puisqu'on vidait également les rues d'hôtes qui s'y trouvaient bien indésirables.

Mystères insondables de la sémantique.

Edgar Faure gagnait son pari, quelques ombres fallotes reprenaient son grand œuvre et tout allait très bien. Même le grand chambardement ne put qu'en mettre et en remettre. Poussant l'idée du Maître à la caricature. Le génie de ces successeurs fut d'ordre politique : ce sont leurs héritiers, affreux réactionnaires, qui reçurent les coups.

Mais que l'âme d'Edgar Faure repose en paix, son héritage spirituel est en de bonnes mains.

Après l'Université à péage, un concept s'imposait, celui d'Etudiant à crédit. C'est fait.

La notion de gratuité est-elle raisonnable ? Bien sur que non car elle pousse au gaspillage. Comment supporter que des gens puissent se prélasser pendant des années à passer de vagues diplômes comme nous le faisions en nos temps reculés, alors que le pays croule dessous les charges, soutenir le franc qui s'amollit, soutenir l'honneur au Tchad et à Mururoa, promouvoir l'avion renifleur et le grand trou des halles, sublimer nos Gloires Nationales, et puis la Calédonie.

L'imagination au pouvoir ! quel slogan dangereux. L'imagination au pouvoir eût pu servir à étouffer l'imagination. Heureusement il n'en fut rien.

Les banques déja propriétaires de l'entreprise et du foncier le deviennent de nos futures élites. Que voila une excellente monnaie d'échange pour l'ouverture quatrevingtdouze.

Enfin demeure le plaisir de voir notre jeunesse s'agiter sagement sur les bancs de l'école, un peu comme la frénésie des mouches du début, surchauffées au soleil de la compétition. Absorber du savoir jusqu'à n'en plus pouvoir, saturer ses méninges d'informations sérieuses, remplir ses yeux d'images importantes.

Plus de pertes de temps en de vaines palabres. Pourquoi rediscuter des faits indiscutables. Laissons les grands esprits réfléchir pour toujours. Si l'on n'apprend pas plus, car sans cela ces jeunes gens en sauraient bientôt plus que leurs maîtres, au moins n'auront-ils plus la désagréable impression d'avoir à s'ennuyer.

Ce sont des décideurs mâtinés de gagneurs dont vivent nos usines. Les excès de savoir, les excès de culture, quelle perte de temps:vive l'activité.

Quelques mots bien exploités pour montrer que l'on sait, suffisent amplement. Il nous faut des fonceurs. Le savoir est un frein, le penseur réfléchit avant de tout gâcher, alors qu'il nous faudra investir le marché. Fonçons d'abord on verra bien après.

Amis affolons-nous la victoire est après l'infarctus.

NOTION d'AUTOJECTIVITE
NOTION de STATUT

E. Bianco

C.R. Subject Classification Informatics: C53 C54 D31 D41

Résumé_

Un système est une sorte d'articulation entre divers niveaux linguistiques. Pour assurer la liaison entre ces divers niveaux il faut disposer de concepts que l'on peut dégager dans un cadre de structures adaptées. L'article détaille la sémantique d'une telle organisation autour de l'autojectivité et de l'insertion fractionnée.

NOTION D'AUTOJECTIVITé
NOTION DE STATUT

INTRODUCTION

La machine formelle est une construction destinée à mettre en évidence les propriétés fondamentales de ce découpage dans l'algorithme et dans la configuration qu'on appelle "la procédure". En même temps cela permet de faire ressortir la nature des services que peut apporter la machine universelle en l'occurence.

Le passage à un langage qui possède un degré d'abstraction supplémentaire, tel que le LAC, oblige à se préoccuper de la notion de compilation. La construction du compilateur met en jeu le traitement des objets des langages de programmation, qui sont, du point de vue de leur organisation proches des langages naturels.

En même temps, la compilation soulève le problème de la gestion du fonctionnement des compilateurs, et par là celui de la gestion complète du fonctionnement de l'ordinateur.

Tous ces problèmes peuvent être résolus par l'aménagement de la Procédure Formelle, comme nous le montrerons plus tard. Mais cela soulève également la question qui s'induit tout naturellement: quelles notions fondamentales peuvent être dégagées qui seraient introduites sous forme synthétique dans un langage de programmation complet destiné à

permettre commodément l'entière description des phénomènes qui se produisent dans un système informatique général.

La réponse en est la construction de la Procédure Formelle Symbolique. Dont le nom rappelle l'origine, complétée par un petit jeu de notions de quelques degrés plus abstraites.

La Procédure Formelle Symbolique, je dirai la PFS, est issue d'un langage machine et est également un langage machine, ce qui implique quelques conséquences sur sa structure. D'abord j'ai voulu conserver la même forme de désignation des opérandes. Pour cela j'ai étendu le calcul d'adresse en conséquence. Et c'est ici qu'intervient la notion de statut qui rend à ce calcul sa forme très simple.

La notion d'autojectivité a été intégrée pour résoudre complètement les problèmes que posent les changements de niveau linguistiques, et en permettre la programmation simple.

je voudrais qu'il n'y ait pas confusion quand j'évoque la simplicité d'une opération, la PFS n'est pas un langage d'amateurs débutants, son intérêt réside dans la compréhension globale de phénomènes fort complèxes, et ce n'est que parvenu à un certain degré de culture qu'on peut percevoir l'efficacité de la programmation en PFS, qui demeurera pour longtemps encore un langage de spécialistes. Plus précisément jusqu'à ce qu'on arrête de confondre enseignement de l'informatique avec enseignement de la programmation.

Il y a plus grave. Je lisais récemment dans une revue "informatique" du CNRS une critique, je devrais dire un dénigrement sur ce vieux langage, l'ATF, à qui on semblait reprocher d'avoir disparu, confondant froidement intérêt scientifique et succès commercial. Il est visiblement plus facile de s'intituler "informaticien" que de réfléchir aux problèmes que soulève la

sémantique des langages. Toutefois l'objet de cet article n'étant pas la polémique, cette remarque doit être considérée du point de vue sémantique comme appartenant à l'ensemble des observations que l'on peut avancer pour que l'informatique ne soit plus d'une part limitée à la programmation, qui se rapproche davantage de l'art, et d'autre part envisagée comme une liste plus ou moins exhaustive d'objets et de propriétés plus ou moins cohérents.

La théorie dont je veux amorcer la construction, pour présenter quelque intérêt doit reposer sur des bases pragmatiques. Les ordinateurs étant ce qu'ils sont, j'essaye de dégager la nature des problèmes fondamentaux qui se posent à l'occasion de leur utilisation. Je pars donc de la position du problème.

POSITION DU PROBLEME

La première difficulté qui saute aux yeux quand on essaie de programmer, peut être cernée par la notion de **champ fictif** :tout programme écrit, l'est forcément de façon abstraite, à l'extérieur de l'ordinateur. On construit une sorte d'échaffaudage qui ne pourra prendre un sens qu'à partir du moment où, codé, il est appliqué dans la mémoire. C' est cette opération qui transforme notre champ fictif en **champ réel** .

A ce programme devenu réel, il faut fournir des paramètres qui auront dû être mis en place avant qu'il ne puisse être activé. Or, lors de sa construction, ces **files externes** , appelons-les par leur nom, doivent être connues au moins partiellement du programmeur qui ne peut ignorer leur structure. C'est ainsi qu'une déclaration partielle des files externes est prévue dans le **contexte fictif** avec seulement précisés les statuts

qui lui sont attachés ainsi que les arguments de programmation: **références** et **index** . Volumes et nombres de lignes sont définis à l'extérieur soit dans le contexte où elles sont déclarées, soit calculés au niveau du système.

La notion de **champ réel** implique une opération de changement de niveau, le champ fictif présenté sous forme de code est rendu actif par sa mise en place dans le champ réel où il peut être soumis à un **compilateur dérouleur** . Cette opération qui peut se faire à partir d'un échange, si le code est stocké sur mémoire de masse est réalisée par le système, mais évidemment tout doit être prêt pour l'arrimage.

Il existe donc un contexte spécial **M-contexte** qui dispose des moyens suffisants pour mener à bien ce travail. La structure d'accueil destinée à recevoir tout code est la **file support** et l'élément d'algorithme qui doit concentrer les propriétés d'un arrimage standardisé est l'**insertion fractionnée** .

Le M-contexte assume donc la charge de supporter le traitement du fini-illimité. La compilation se réalise dans un **C-contexte** .

La notion de **système** dès l'instant où elle est générale implique la notion de **sous-système** . Pour définir rigoureusement la notion de système, je mets en œuvre un ensemble de notions qui découlent de la notion fondamentale d'**autojectivité** . C'est grâce à cette notion qu'il m'est possible de séparer complètement les traitements système des traitements compilation. Les seuls échanges qui demeurent alors entre système et compilateurs sont ceux qui sont nécessaires pour le fini-illimité. On va faire le tour, maintenant des moyens indispensables à cette conception.

NOTION DE SOUS_SYSTEME

Je conçois le système comme un organe définitif capable d'accueillir à tout moment de nouveaux jeux de compilateurs, forcément inconnus de lui. Il est clair qu'un certain dialogue doit pouvoir exister entre système et compilateur. Il s'agit de montrer qu'il est possible d'en définir à l'avance les limites sémantiques. Comme on va le voir, les éléments de cet échange se retrouvent en divers points où je m'efforce de les rendre au maximum implicites. J'entends par là que la communication est intégrée dans le PFS, mais ce langage prenant en compte plusieurs niveaux linguistiques à la fois une partie de l'information s'y retrouve codée pour faire intervenir le niveau machine universelle.

Introduire un nouveau langage de programmation signifie ainsi introduire une structure complète de programmation. Cela implique en effet qu'on se munisse des moyens de créer et de récupérer des configurations de données, de créer des programmes qu'on puisse appliquer sur ces configurations et enfin de manipuler des programmes et des configurations considérés comme des objets. En d'autres termes on se donne trois sortes de langages:

LP , un langage de programmation,

LE , un langage de manipulation de données,

LC , un langage de manipulation en bloc de configurations et de programmes.

L'importance relative de ces langages peut varier énormément selon la nature du problème envisagé. Aussi il est impossible au système d'intervenir directement dans l'organisation des opérations. Cela signifie que les compilateurs doivent être auto-organisés. Ils sont conçus pour

décider chacun et en fonction de quelles circonstances, du compilateur qui doit succéder et également des files à lui fournir à titre de paramètres.

D'un autre point de vue ces compilateurs doivent se succéder les uns aux autres, mais l'autojectivité interdit qu'ils s'insèrent l'un dans l'autre comme des procédures. Il faut disposer d'une sorte d'insertion à plat ou encore d'insertion différée, telle que ce puisse être le système qui réalise l'insertion demandée par tel compilateur. C'est le rôle de l'instruction d'**arrêt** qui, placée dans le compilateur fournit l'information de la demande. Bien sûr la relation entre l'instruction d'arrêt qui donne l'indication du compilateur à succéder avec la liste de ses paramètres et l'instruction d'insertion fractionnée du système ne peut être établie que par la machine universelle. Mais il me faut ajouter encore un moyen de véhiculer l'information, c'est le **vecteur information d'insertion fractionnée**.

Si l'on fait intervenir la qualité du service à fournir, sous forme de jeux complets de langages eux-même complets, suffisants pour manipuler de toutes les manières désirables l'information traitée par l'utilisateur,dans un cadre ergonomique convenable et s'il est tenu compte des contraintes que j'impose sur le système, alors on voit apparaître la notion de sous-système. Cette notion circonscrit toutes les propriétés d'entière autonomie apparente que doit posséder une organisation construite autour d'un langage de programmation déterminé pour qu'il soit commodément utilisable.

A partir de là le système n'a plus à connaître d'un sous-système particulier que son nom et les caractéristiques géométriques de son occupation en mémoire. Tout ceci est alors enregistré dans une table des sous-systèmes qui peut être consultée et enrichie ad libitum.

Une autre notion surgit quand on envisage en toute généralité qu'à un

même sous-système correspondent plusieurs tâches différentes, autant qu'on en a besoin. A ce moment il devient nécessaire de disposer en permanence d'une suite d'images de chacune des tâches en déroulement, j'en désignerai le support par **file de distribution** .

Je vais rappeler quelques points de la PFS importants pour les changements de niveau autour de l'insertion fractionnée.

Tableau cartouche

Dans un M-contexte on peut déclarer un tableau de type particulier qui sert à la construction des cartouches d'échanges. Il se déclare comme un tableau normal, sauf qu'il est marqué par la lettre : cartouche . Sa déclaration est spécifique car, comme nous l'avons déjà vu, du point de vue du câblé l'image matérielle en est repérée par une origine particulière, définie par un registre spécial E . Se reporter au Bulletin N° 17.

Par exemple un tel tableau se déclarerait:

40 cartouche (état(1),sens(1),type(1),Nde(2),Np(2),Ns(2),ADR(3),
Vol(2)) : C1 : C2 : C3 ...

Chaque ligne de ce tableau est susceptible de comporter un cartouche et la gestion en est assurée par le système, l'exploitation en est assurée, elle, par la machine de distribution.

Instruction d'échange

Cette instruction fournit les paramètres nécessaires à la constitution d'un cartouche par le système, n'oublions pas que c'est lui qui doit être chargé de la gestion du tableau-cartouche. La machine universelle, qui

rencontre une telle instruction porte les paramètres qu'elle contient dans le VIIF. Dans le modèle de système construit ci-après, on trouve ce traitement à l'aiguillage : aiguillage SYST sur les sorties Initmess et Initech.

La syntaxe de l'échange permet à cette instruction de contenir le minimum d'information nécessaire:

échange <accès>,<volume>

par exemple:

échange $\lceil_{DIS,h3}$. $\lceil_{DIS,h4}$

Ceci signifie que le système décide de tous les autres paramètres du cartouche, tel que le choix de la mémoire de masse, des moyens d'échanges et autres.

Déclaration des files externes

Le principe d'autojectivité implique que toute la place utilisée dans un champ fictif soit méticuleusement déclarée, explicitement ou implicitement. Il faut alors se donner les moyens de fournir des paramètres au futur programme. C'est l'un des problèmes de changement de niveau. En effet le paramètre ne pourra apparaître qu'à l'étape où le champ fictif devient champ réel.

Il faut donc pouvoir préparer la jonction à l'avance. N'oublions toujours pas que c'est le système seul qui peut l'assurer, et qu'il ne peut pénétrer dans le détail local de l'opération. Il faut donc se munir de moyens standards. La déclaration d'un jeu de files externes tient ce rôle. Volumes et localisations sont en principes décidés à l'extérieur du champ fictif.

Dans le contexte on déclare donc la partie structure de la file, avec la liste des index et des références locales. Pour la commodité de la

programmation des instructions d'arrêt les files externes sont repérées par un nom sous forme de numéro.

exemple:

externe 1 dynamique (p,q) 1 (3(2),n1(3)), 2 (4(1),2(2),f(2)) : P : P1 : ...

externe 2 tableau (un(1),deux(3),trois(2)) : H : R : ...

Tableau des files des sous-systèmes

Le système doit disposer d'une liste extensible des sous-systèmes. Il y accède par le nom du sous-système et on y trouve les descriptifs de toutes les files de chaque compilateur, également accessible par son nom dans la liste des compilateurs qui constituent le sous-système.

Les files y sont notées avec leur type, tableau, dynamique, leurs caractéristiques, volume, nombre de lignes, nombres d'index et de références. Si la file est externe c'est également indiqué.

Les files doivent toujours être déclarées avec leur volume, dans leur déclaration originale, mais ce volume peut être modifié en cours de traitement pour chaque cas spécifique de tâche. Cela peut être fait par l'intermédiaire de l'instruction d'arrêt.

File de distribution

Un même sous-système doit pouvoir servir à créer autant de tâches qu'il est nécessaire. Cela implique qu'on doit conserver une image de chacune des tâches en cours de traitement. C'est le rôle attribué à la file de distribution que de contenir ces images.

Très généralement l'autojectivité est conçue pour assurer au système le contrôle complet sur le temps et sur l'espace. C'est donc un moyen puissant pour le multitâches, c'est dans cet esprit un peu particulier que

je présente le système bien que tout ceci soit généralisable à une structure de système quelconque, à partir du moment où il apparaît au moins deux tâches à traiter dans un même intervalle de temps, et dont une peut être asynchrone.

Chaque image de tâche est donc un VIIF, vecteur information de l'insertion fractionnée, qui se trouve ainsi défini par sa fonction comme je vais l'indiquer en détail.

Instruction d'arrêt

C'est par son intermédiaire que se décide du sort des files résultat. Lorsque le compilateur en cours de déroulement constate que le travail est achevé, en fonction des conditions rencontrées, il décide du compilateur à insérer à sa suite et du jeu de files paramètres à lui indiquer. Ce sont les files résultat. Parvenue en ce point du déroulement du compilateur, la machine universelle rencontre le code d'une instruction d'arrêt dont elle va pouvoir extraire le nom du compilateur successeur, les noms des diverses files à transférer, prises dans la liste des files externes du compilateur, bien évidemment.

Le jeu des files à transférer ne recouvre pas forcément l'ensemble de files externes du compilateur actuel, mais doit, et dans l'ordre, recouvrir obligatoirement l'ensemble des files externes du compilateur successeur. S'il n'était pas nécessaire de transférer toutes les files on laisse des blancs dans la liste des paramètres effectifs de l'instruction d'arrêt. Je n'ai pas traité ce cas dans l'exemple de système que je donne.

Dans cette instruction, on peut noter pour chaque file un mode d'utilisation:

Transférer simplement le contenu,

Transférer et conditionner ce contenu,

Supprimer.

L'instruction libération alors employée par le système dans un M-programme, permet à la machine universelle de trafiquer la directory de la file support correspondante pour que le contenu de la file soit désormais ignoré.

Vecteur information de l'insertion fractionnée

Le vecteur information sert de point de communication entre plusieurs niveaux linguistiques différents. Il sert à transférer de l'information entre le système, la machine universelle et le compilateur.

C'est la conséquence de plusieurs conditions. Deux compilateurs ne peuvent communiquer que par l'intermédiaire du système à cause de l'autojectivité. Comme par ailleurs le système construit une fois pour toutes ne pourrait communiquer directement avec chaque compilateur, car je tiens à éliminer les contraintes que cela imposerait à leur implémentation, le contact ne peut alors se réaliser que par l'intermédiaire de la machine universelle.

Cela me parait la condition de l'indépendance maximale entre tout compilateur et le système, dans la mesure où celui-ci est complet au sens du problème posé. Ainsi le maximum des calculs nécessaires se réalisent de manière implicite. Le langage doit alors fournir un minimum de moyens de programmation pour que les acteurs en présence puissent communiquer.

Une image de tâche, considérée comme un élément de la file de distribution, comporte deux parties:

Des indications de structure de l'image du sous-système.

Le vecteur information de l'insertion fractionnée pour chacun des

compilateurs du sous-système.

Le vecteur information est subdivisé en cinq parties:

La liste des accès,

L'image du cartouche,

Les caractéristiques des files externes,

Les caractéristiques des files locales,

Les caractéristiques des files résultats.

Dans la liste des accès on trouve les informations suivantes:

L'accès au code du compilateur,

L'accès à la configuration des caractéristiques des files qui lui sont locales.

L'état de déroulement de la tâche: "demandé", "en cours", "terminé", "attente".

L'accès à la configuration des caractéristiques des files externes.

Le nom (numéro) du compilateur successeur.

L'accès à la configuration des caractéristiques des files résultat.

L'état d'un échange de masse: "demandé", "en cours", "terminé", "panne".

L'état d'un échange conversationnel.

Le système qui va délivrer un quantum de travail, examine le VIIF, et en toute priorité il scrute les états des échanges. Ce n'est que dans le cas où ils sont achevés qu'il passe à l'état du déroulement de la tâche.

La liste des files externes comporte un nombre de files, suivi de triplets composés d'un type de file, d'un accès à la file, et d'un volume en nombre de lignes.

La liste des files locales a la même organisation. Quand à la liste des files résultats, elle commence par un nombre de files suivi d'une liste de

quadruplets,qui contiennent chacun: un mode, un numéro de file pris dans la liste des files externes, un accès à la file, un volume.

Le mode est celui qui a été défini dans l'instruction d'arrêt.

UN SYSTèME

Je donne d'un système de temps partagé la partie qui correspond à la gestion des insertions fractionnées des compilateurs d'un sous-système , pour chacune des tâches, dans le cas où il y a un nombre quelconque de tâches. Dans la file de distribution, l'image d'une tâche, pour un sous-système à k compilateurs se présente sous la forme:

[(ηtâche),(k),(N°C.act)] , [VIIF 1] , [VIIF 2] , [VIIF 3] , ... [VIIF k]

Ce sont des lignes qui sont portées entre crochets.
Le (N°C. act) désigne le numéro du compilateur en cours de déroulement.
L'un parmi les k VIIF a la structure de détail suivante:

(pi),(di),(ηi),(Di),(Csucc),(FRi),(ηi),(ηi),(----------),
1 2 3 4 5 6 7 8 image du cartouche
(Nb f ext),(te1),(ae1),(ve1),(te2),(ae2),(ve2),... ...,(ten1),(aen1),(ven1),
Di k1 k2 k3
(Nb f loc),(tl1),(bl1),(vl1),(tl2),(bl2),(vl2), ... ,(tln2),(bln2),(vln2),
di
(Nb f res),(m1),(j1),(α1),(v1),(m2),(j2),(α2),(v2), ..,(mn3),(jn3), (αn3), (vn3)
FRi,h h1 h2 h3 h4

M-contexte

10 000 , 200 dynamique (h,h1,h2,h3,k1,k2,k3) 1(16(2), n1(2), n2(2)) :
TSS : DIS ;

1 000 000 , 50 000 support (.....) : Files : T ;

40 cartouche (état(1), sens(1), type(1), Nde(2), Np(2), Ns(2), ADR(3),
Vol(2)) : Cart ;

M-texte

$\lceil_{TSS,3}$:= 1 ;

Rotation : itère ;

TSS := initial ;

DIS := → TSS ;

Répartition : itère DIS de initial à DIS1 ;

DIS := $\lceil_{TSS,3}$ → TSS ;

Inserfract DIS ;

aiguillage SYST ($\lceil_{DIS,7}$ = "en cours" : Evit1 , $\lceil_{DIS,6}$ = "en cours : Evit2 ,
$\lceil_{DIS,7}$ = "terminé" : Suitmess , $\lceil_{DIS,6}$ = "terminé" : Suitech ,
$\lceil_{DIS,7}$ = "demandé" : Initmess , $\lceil_{DIS,3}$ = "en cours" : suit ,
$\lceil_{DIS,3}$ = "term" : Success) ;

Suitmess : $\lceil_{DIS,7}$:= "nul" ;
TSS := $\lceil_{TSS,2}$ → TSS ;
TSS := → TSS ;
sortie SYST ;

Initmess : $\lceil_{Cart,état}$:= $\lceil_{DIS,9}$;

```
                    [Cart,sens := [DIS,10 ;
                    [Cart,type := [DIS,11 ;
                    [Cart,Nde := [DIS,12 ;
                    [Cart,Np := [DIS,13 ;
                    [Cart,Ns := [DIS,14 ;
                    [Cart,ADR := [DIS,15 ;
                    [Cart,Vol := [DIS,16 ;
                    sortie SYST ;
Initech : [Cart,état := [DIS,9 ;
                    [Cart,sens := [DIS,10 ;
                    ----------------
                    ----------------
                    sortie SYST ;
Evit2:Evit1:Suit : TSS := [TSS,2 → TSS ;
                    TSS := → TSS ;
                    DIS1 := → TSS ;
                    sortie SYST ;
Suitech : [DIS,8 := "nul" ;
                    TSS := [TSS,2 → TSS ;
                    TSS := → TSS ;
                    DIS1 := → TSS ;
                    sortie SYST ;
```

Success : k1 := $\lceil_{DIS,4}$ + 1 ;

k2 := k1 + 1 ;

k3 := k2 + 1 ;

h := $\lceil_{DIS,6}$;

h1 := h + 1 ;

h2 := h1 + 1 ;

h3 := h2 + 1 ;

h4 := h3 + 1 ;

Parom : itère de 1 à $\lceil_{DIS,h}$;

Aiguillage Filepar ($\lceil_{DIS,h1}$ = ("transf" : Trans , "transcond" : condens , "suppress" : Suppr) ;

Trans : $\lceil_{DIS,k2}$:= $\lceil_{DIS,h3}$;

+3 sur k1,k2,k3 ;

+4 sur h1,h2,h3,h4 ;

sortie Filepar ;

Condens : Aiguillage Vol ($\lceil_{DIS,h4}$ > $\lceil_{DIS1,k3}$: ouvert , sinon Suit) ;

Suit : $\lceil_{DIS,k2}$:= $\lceil_{DIS,h3}$;

$\lceil_{DIS2,k3}$:= $\lceil_{DIS,h4}$;

+3 sur k1,k2,k3 ;

+4 sur h1,h2,h3,h4 ;

sortie Vol ;

ouvert : F1 := $\lceil_{DIS,k2}$ → Files ;

libérer F1 ;

file T , $\lceil_{DIS,h4}$;

```
                        T := → T ;
                        +3 sur k1,k2,k3 ;
                        +' sur h1,h2,h3,h4 ;
                        sortie Vol ;
                        noeud Vol ;
Suppr : F1 := [DIS,h3 → Files ;
                        libérer F1 ;
                        +3 sur k1,k2,k3 ;
                        +4 sur h1,h2,h3,h4 ;
                        sortie Filepar ;
                        noeud Filepar ;
répétition Param ;
sortie SYST ;
                        noeud SYST ;
répétition Répartition ;
répétition Rotation ;
```

T est la référence borne qui permet d'accéder à la mémoire disponible pour les tâches.

DIS est la référence borne qui permet d'accéder à la ligne libre en file de distribution.

Dans le système, la gestion de ces deux files est assurée sur l'exploitation du langage hors-texte. C'est à la demande de l'utilisateur que le système explore sa table des sous-systèmes, et ayant trouvé le sous-système demandé, crée une image supplémentaire de tâche en file de distribution, et occupe de la mémoire disponible sur la file support en créant les VIIF qui correspondent.

VOUZZAVEDIBISAR

FORCES DE CORIOLIS

Les forces de Coriolis sont ces forces qui, dans un univers courbe, font qu'il n'y a pas de droit chemin d'un point à un autre. Et plus l'univers est tordu, et plus on tourne autour du pot.

Ainsi en politique. Univers particulièrement distort. Imaginons un centre très attractif: de la soupe par exemple. L'expérience montre qu'un politicien est fortement attiré par un tel centre, et d'autant plus qu'il en est plus proche.

Il s'y précipiterait dessus s'il n'était également animé par une tendance. Ainsi il entame une spirale d'approche d'autant plus lente vers la droite que sa tendance droitière est plus forte . Il en est de même pour les tendances de gauche, mais dans l'autre sens. On pourrait penser que plus la tendance est faible, plus l'approche est rapide, oui, mais théoriquement elle n'en demeure pas moins infinie. Heureusement les frictions accélèrent la chute.

Seuls dans tout l'horizon les centristes sont armés pour foncer droit sur la soupe au point que c'en est une merveille. Oui mais gare à la moindre attraction déviante, elle suffirait pour qu'on frôle le but sans pouvoir s'arrêter ... Ah Misère !